

## **Part I : Filtering and Data Association**

## **Part II : Crowd-scene Analysis**

VLPR 2012, Shanghai, China

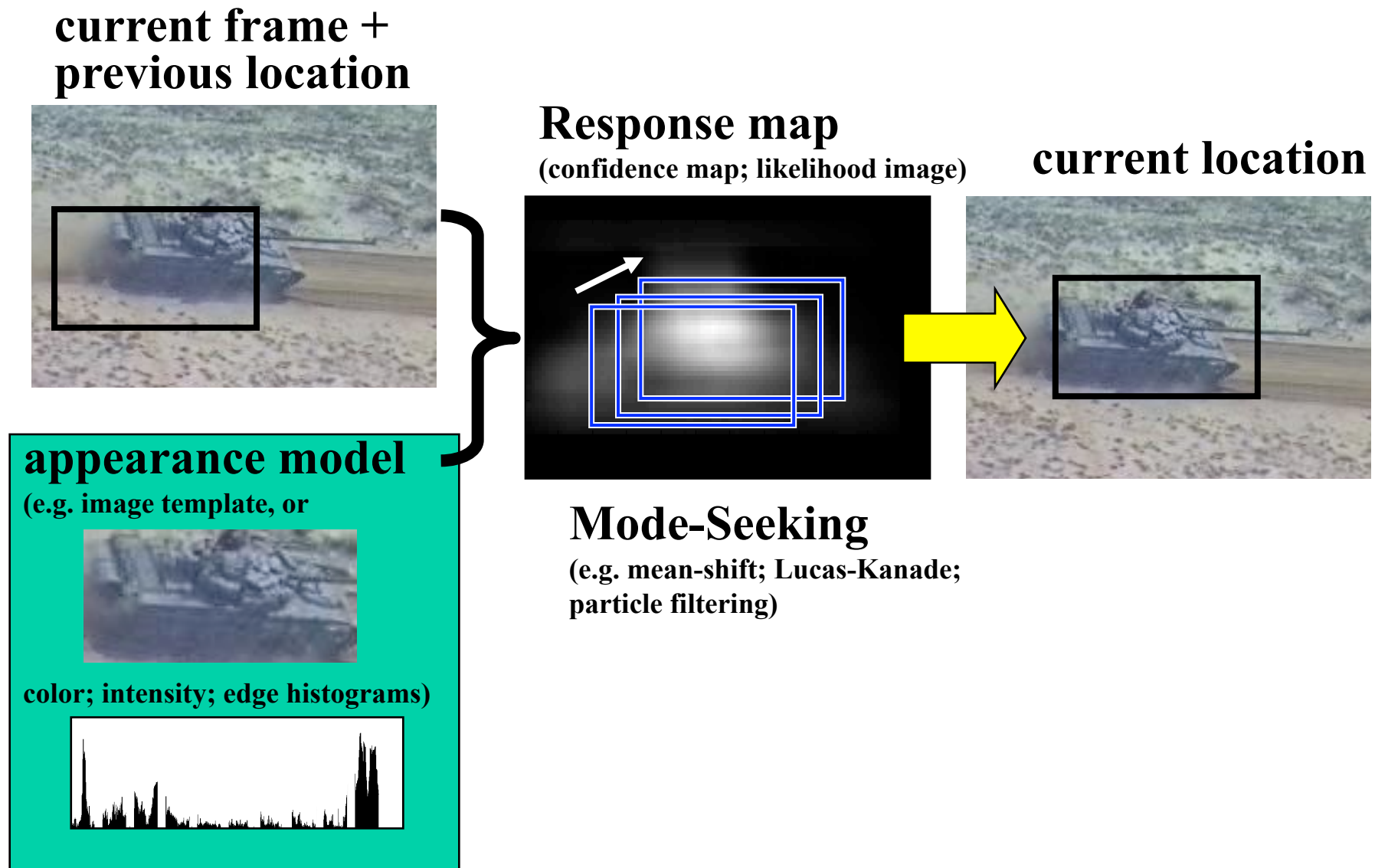
Bob Collins, July 2012

# What is Tracking?



typical idea: tracking a single target in isolation.

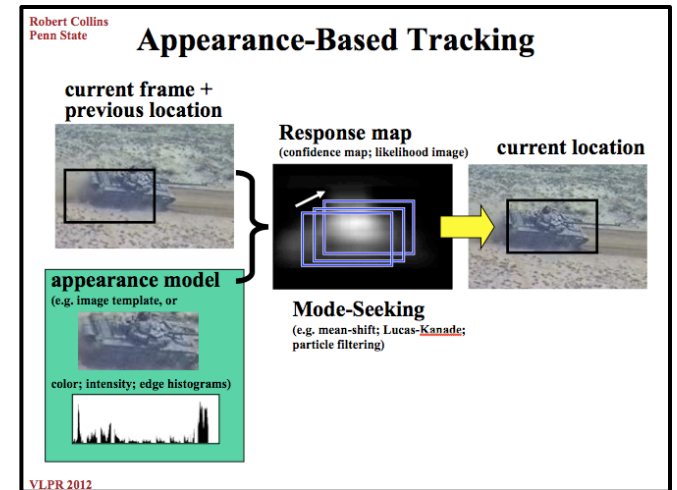
# Appearance-Based Tracking



# Appearance-based Tracking

Research tends to focus on:

- finding discriminative features
- model adaptation



Search for best match tends to be simple gradient ascent (hill-climbing).

Motion prediction tends to be simplified to constant position + noise (assumes previous bounding box significantly overlaps object in the new frame).

Previous VLPR lectures with more detail:

<http://www.cse.psu.edu/~rcollins/CollinsVLPR2009Lecture.pdf>

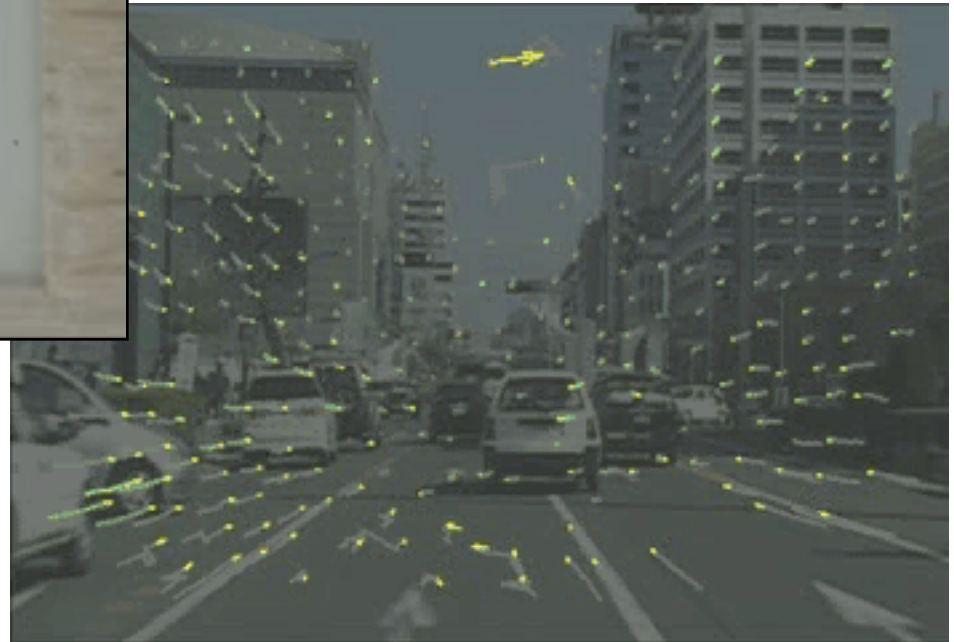
<http://www.cse.psu.edu/~rcollins/CollinsVLPR2010Lecture.pdf>

# What is Tracking?

## Multi-target tracking....



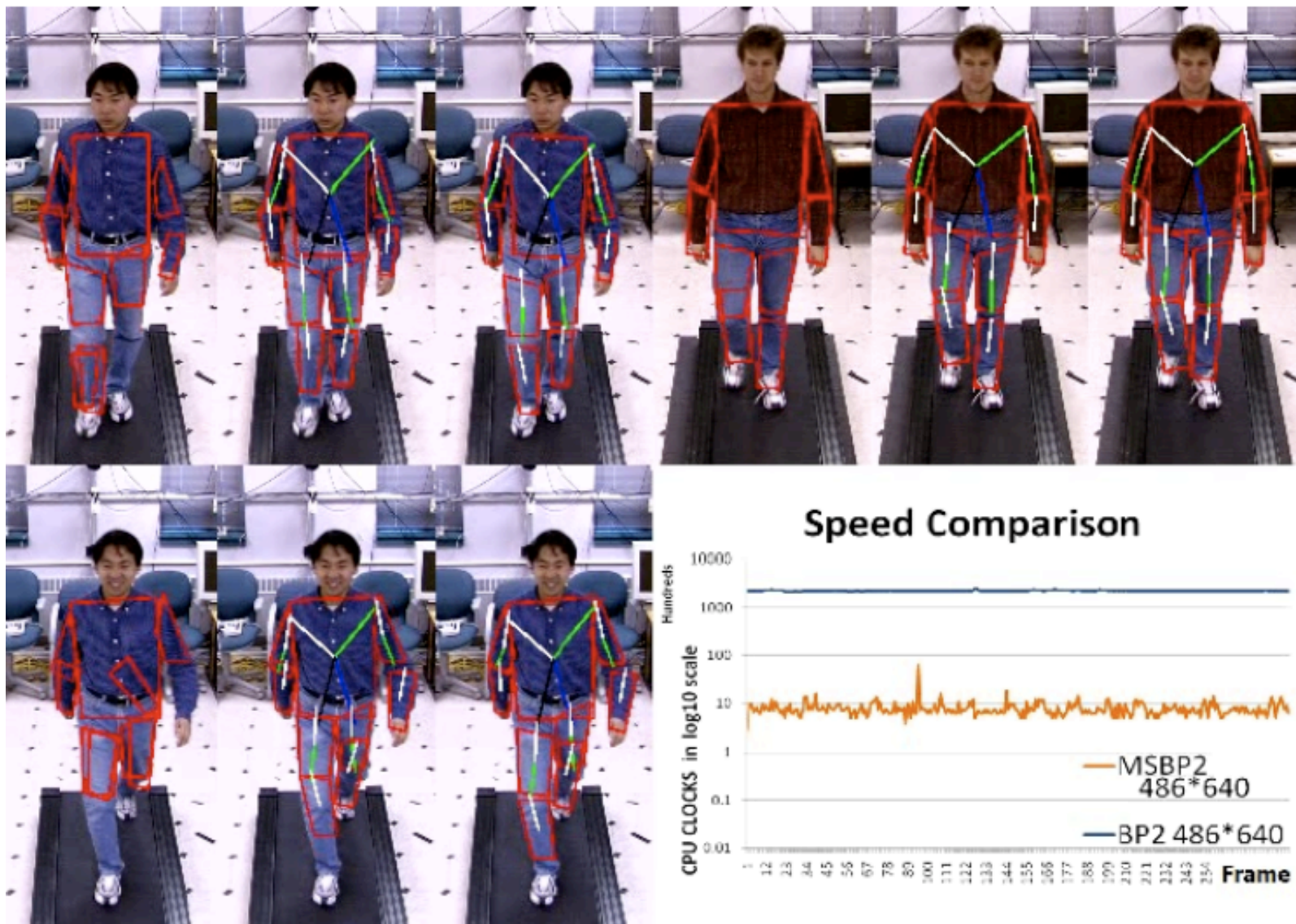
ant behavior, courtesy of  
Georgia Tech biotracking



“targets” can be corners, and  
tracking gives us optic flow.

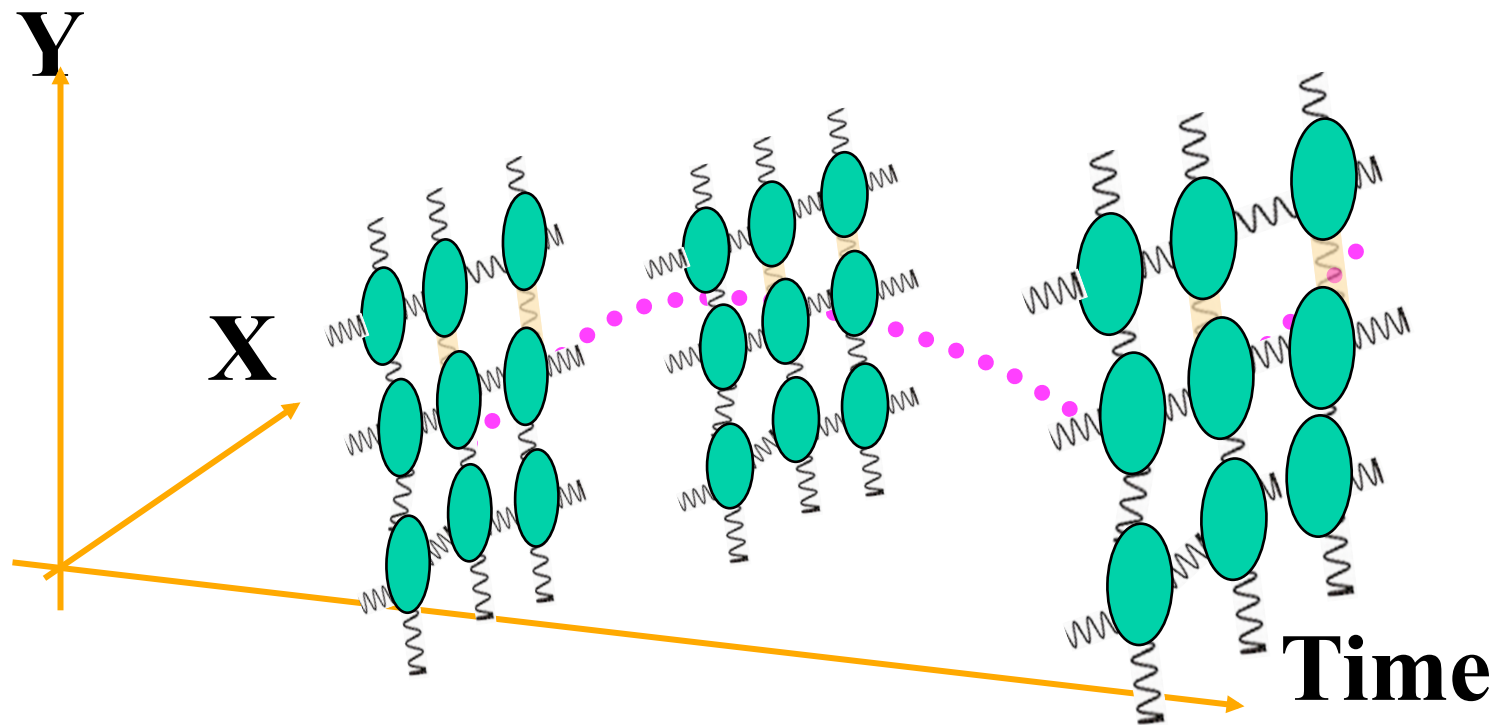
# What is Tracking?

articulated objects having  
multiple, coordinated parts



# Multi-Target Tracking

- The previous two slides are examples of tracking multiple targets over time. Such work tends to focus on constraints between individuals, either strong geometric constraints, or weak exclusion constraints.



# What is Tracking?

**Active tracking involves moving the sensor in response to motion of the target. Needs to be real-time!**





# Why are there so many papers on tracking?

Because what kind of tracking “works”  
depends on problem-specific factors.

# Factors: Discriminability

How easy is it to discriminate one object from another.



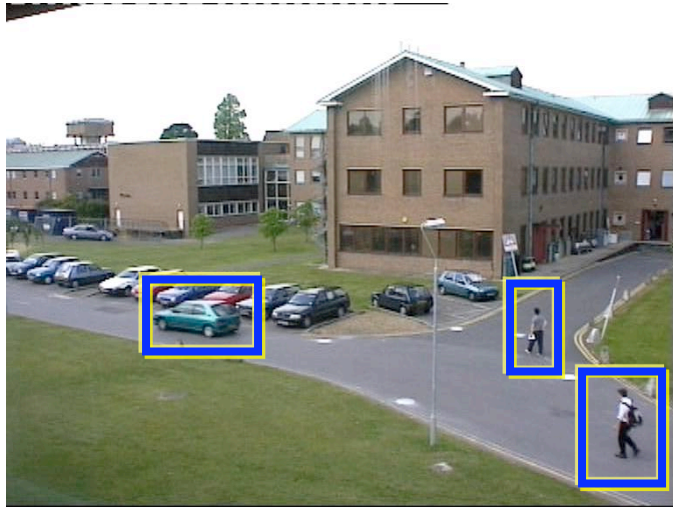
appearance models can  
do all the work



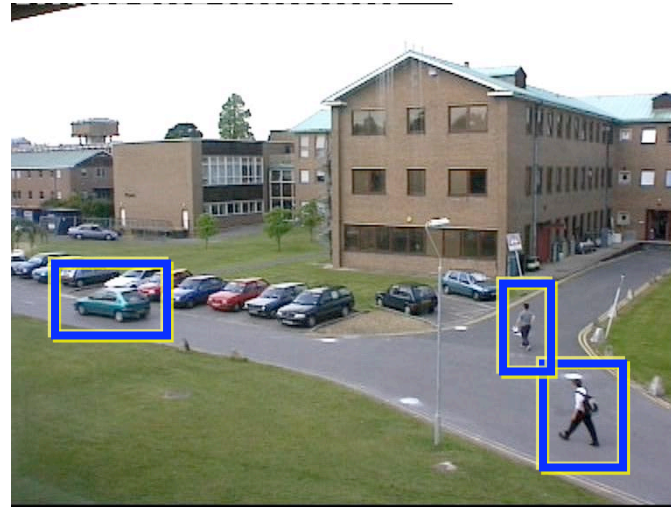
constraints on geometry  
and motion become crucial

# Factors: Frame Rate

frame n



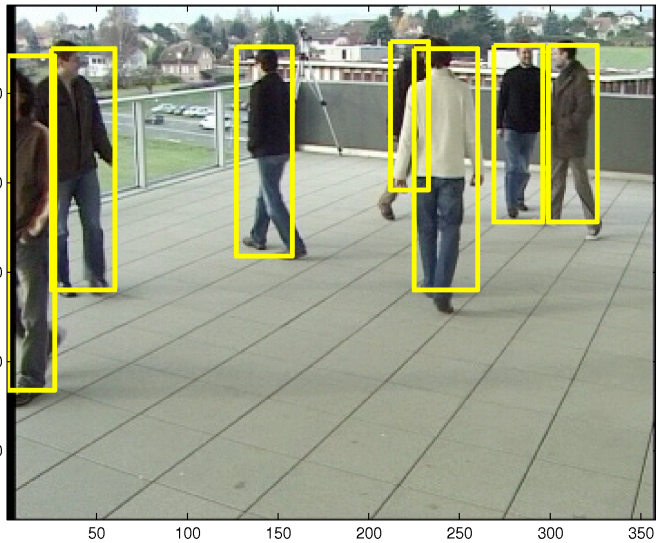
frame n+1



gradient ascent  
(e.g. mean-shift)  
works OK

H  
I  
G  
H

frame 2325: nmatch 7 nmissed 0 nfalse 0



frame 2375: nmatch 6 nmissed 0 nfalse 0



much harder  
search problem.  
data association

L  
O  
W

# Other Factors

**single target vs multiple targets**

**single camera vs multiple cameras**

**on-line vs batch-mode**

**do we have a good generic detector?  
(e.g. faces; pedestrians)**

**does object have multiple parts?**



# Filtering and Data Association

- Filtering
  - Recursive Bayesian estimation
  - (continuous) Probability Theory
- Data Association
  - Assignment problems
  - (discrete) Combinatorics

# State Space Approach

Two vectors of interest:

1) State vector: vector of variables  $x_k$  representing what we want to know about the target

*examples:  $[x,y]$ ;  $[x,y,dx,dy]$ ;  $[x,y,\theta,scale]$*

2) Measurement vector: noisy observations  $z_k$  related to the state vector.

*examples: image intensity/color; motion blobs*

# Foundation: Bayesian Filtering

Rigorous general framework for tracking. Estimates the values of an unknown state vector given a time series of uncertain observations.

Key idea: use a recursive estimator to incrementally update the posterior probability density function (pdf) of the state vector based on most recent data.

Bayesian hypothesis: All quantities of interest, such as MAP or marginal estimates, can be computed from the posterior pdf.

# Bayes Rule

Bayes rule can be derived by a simple manipulation of the rules of probability. But it has far-reaching consequences.

$$p(x,y) = p(x|y)p(y) = p(y|x)p(x)$$

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)} = \frac{p(y|x)p(x)}{\int_x p(y|x)p(x)}$$

interpretation:

posterior  $\propto$  likelihood \* prior



**Thomas Bayes**  
1702-1761



# Posterior Distribution

For a Bayesian, the posterior distribution is the starting point for answering all well-posed statistical questions about the state.

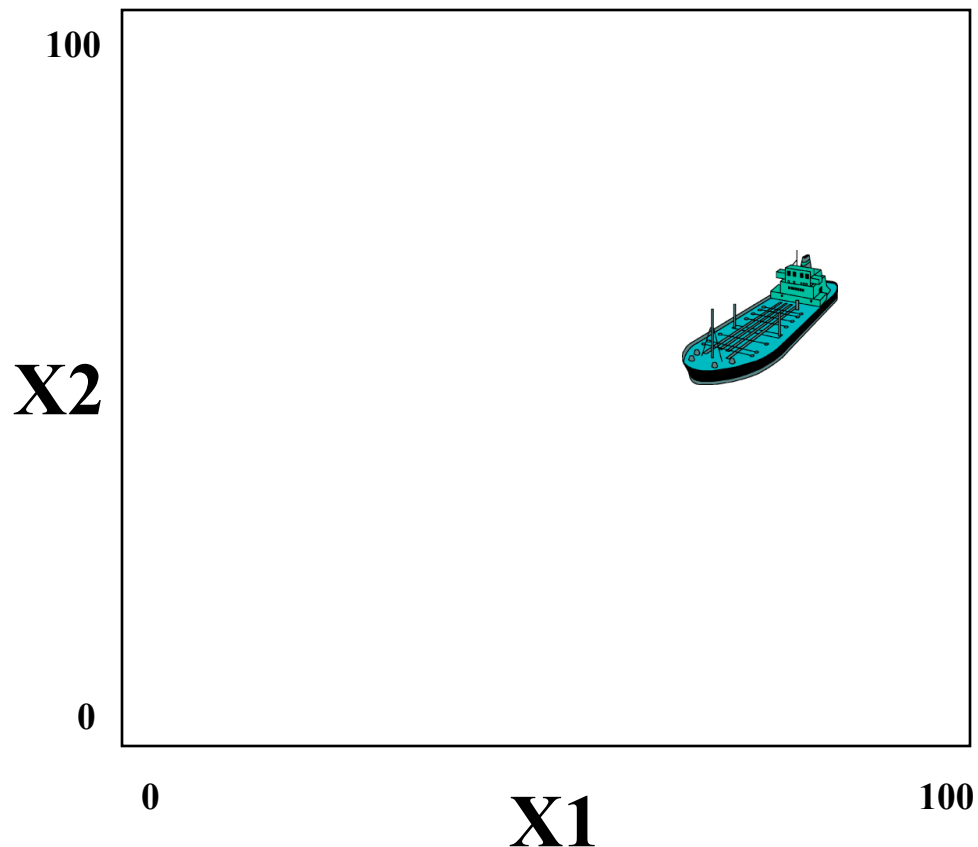
e.g.

- What is the most likely location of this object?  
the mode of the posterior (MAP estimate)
- With what certainty do I know that location?  
spread of probability mass around the MAP estimate
- Are there other likely locations? If so, how many?  
analysis of multi-modality of the posterior

Important point: output of Bayesian approach is not a single point estimator, but a whole probability distribution over state values.

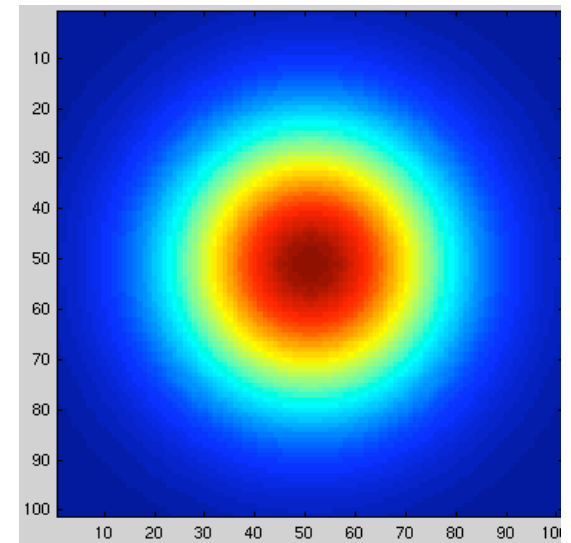
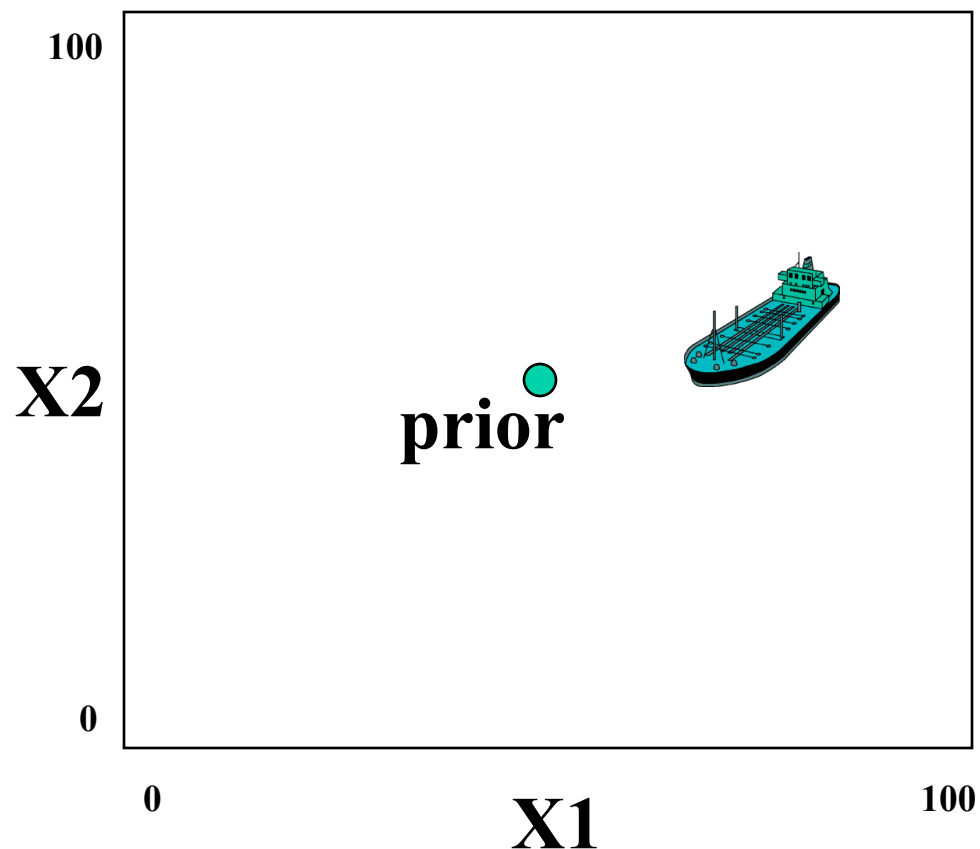
# A Simple “Cartoon” Example

Let’s assume our target state  $x$  is a 2D location  $(x_1, x_2)$ , and that our target is within some bounded region of interest, say  
 $0 < x_1 < 100$  and  $0 < x_2 < 100$



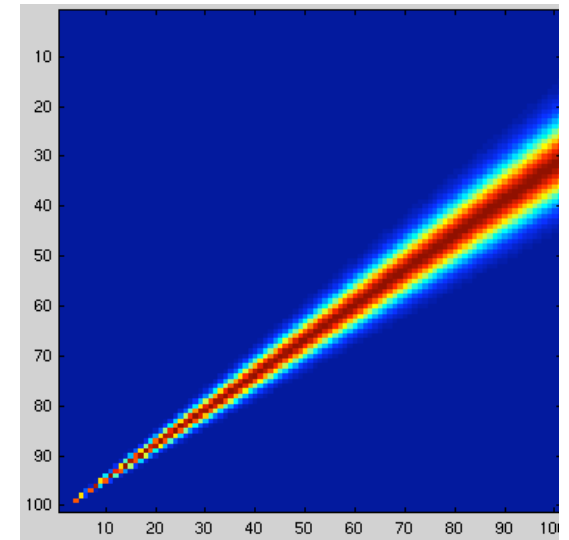
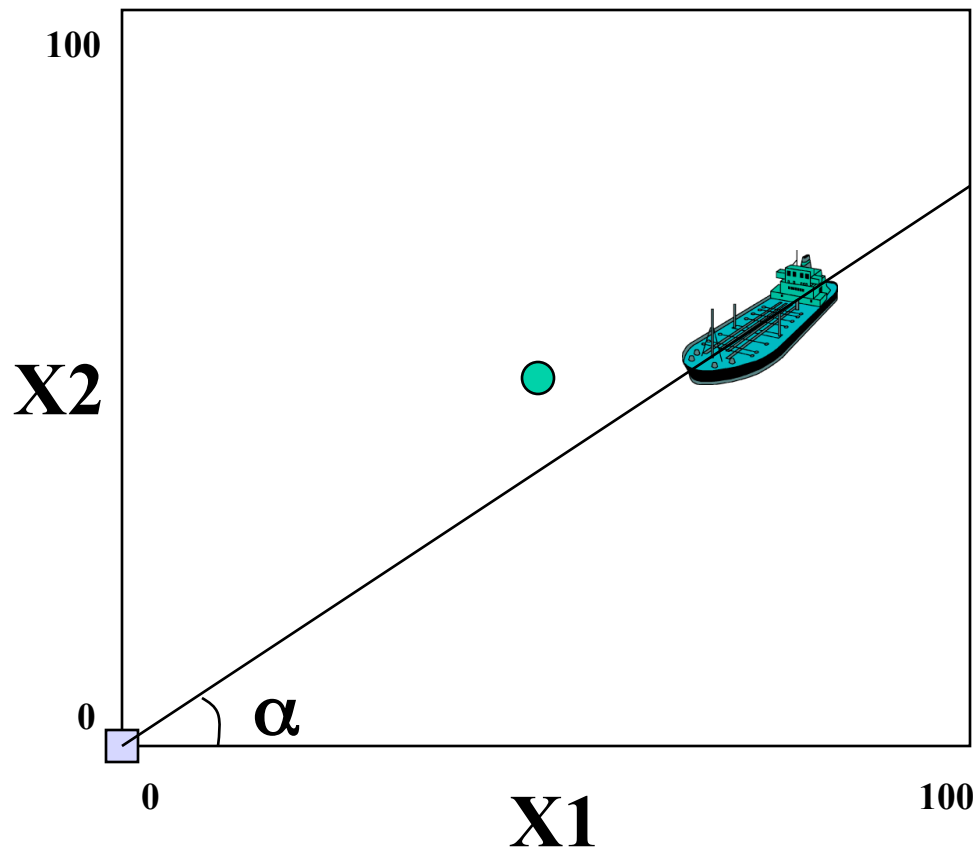
# Bearings-Only Example

Initially we don't know the target is, so we guess it is at (50,50). We are uncertain, so model that uncertainty as a Gaussian with high variance (truncated and normalized so all mass lies within the region of interest)

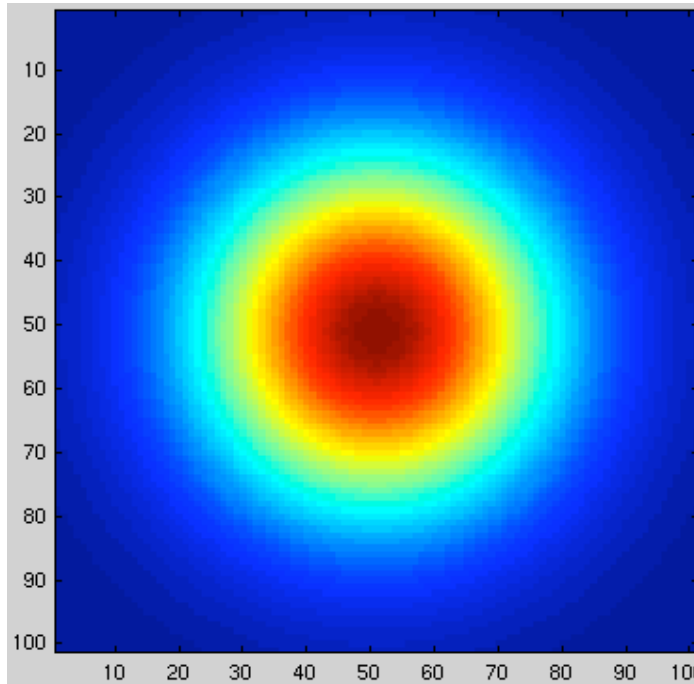


# Bearings-Only Example

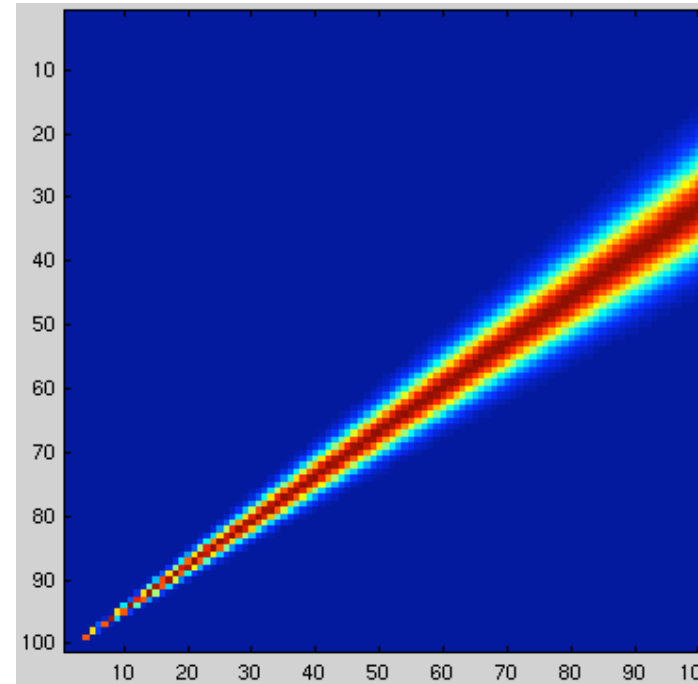
A bearings-only sensor located at  $(0,0)$  takes a noisy reading of the angle (alpha) towards the target. We will model the difference between the measured angle and actual angle as a zero-mean, 1D Gaussian.



# Bearings-Only Example



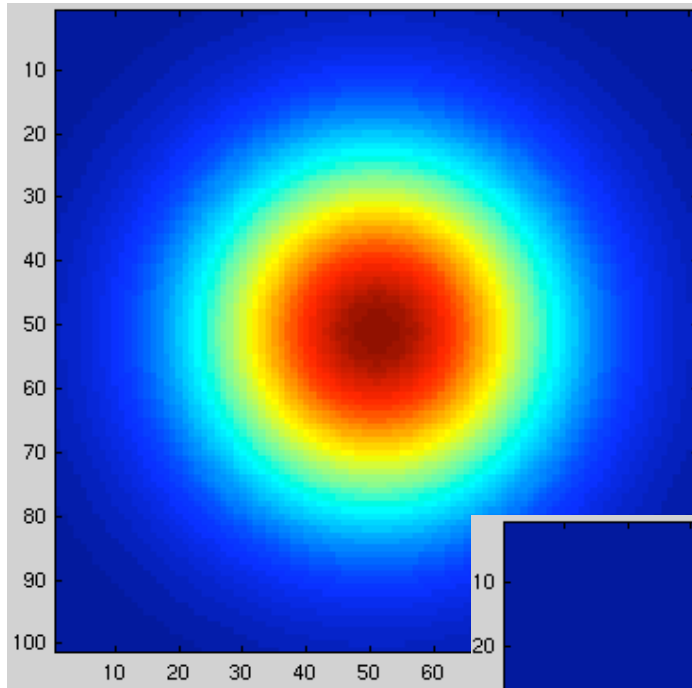
prior



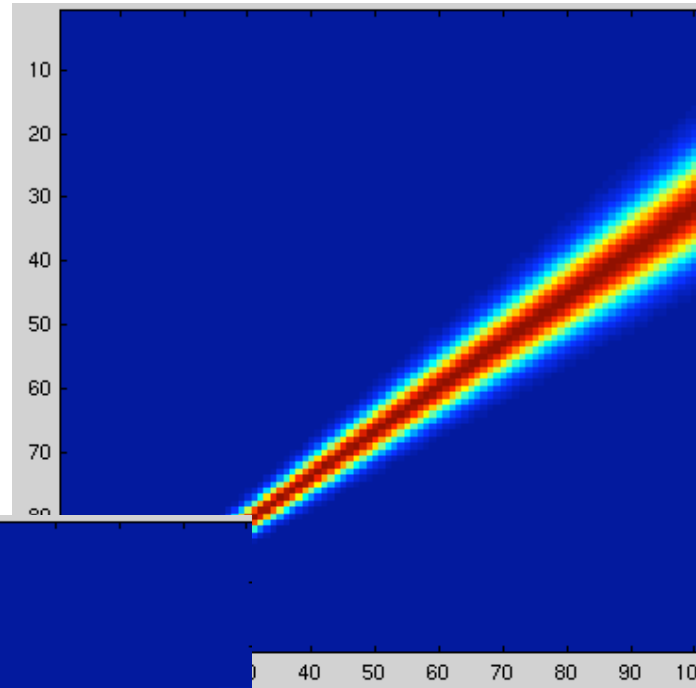
likelihood

to combine using Bayes rule: point-wise multiply the prior times the likelihood, then renormalized the result so that total mass sums up to 1.

# Bearings-Only Example

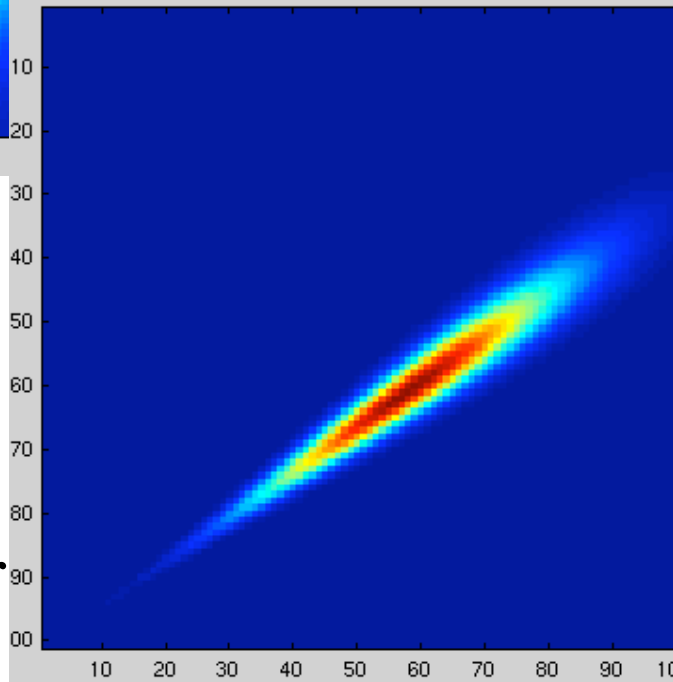


prior



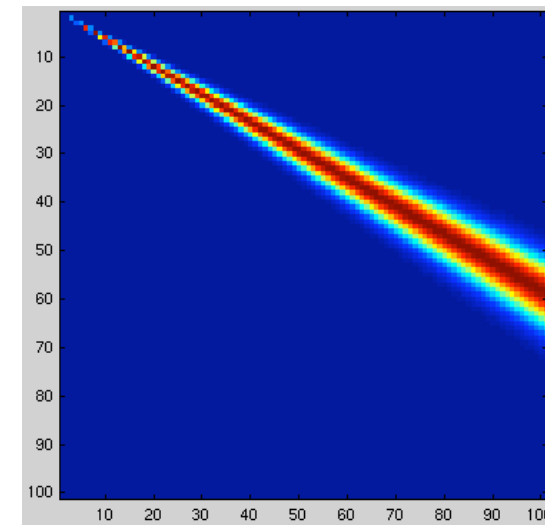
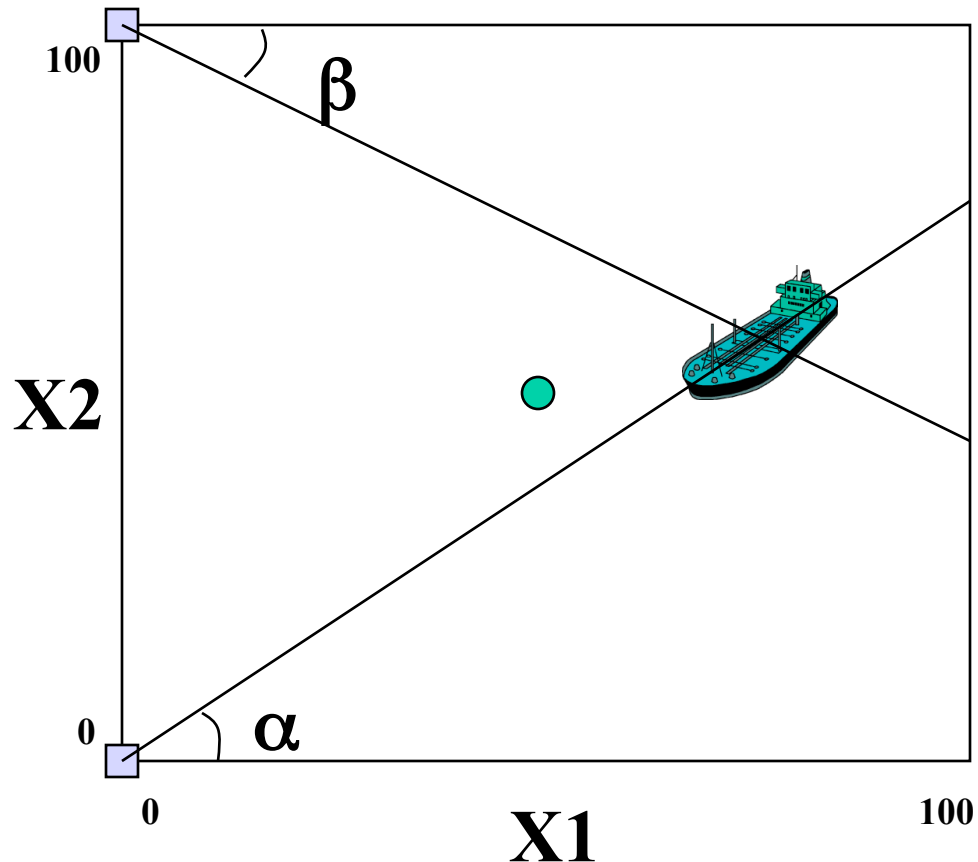
likelihood

posterior

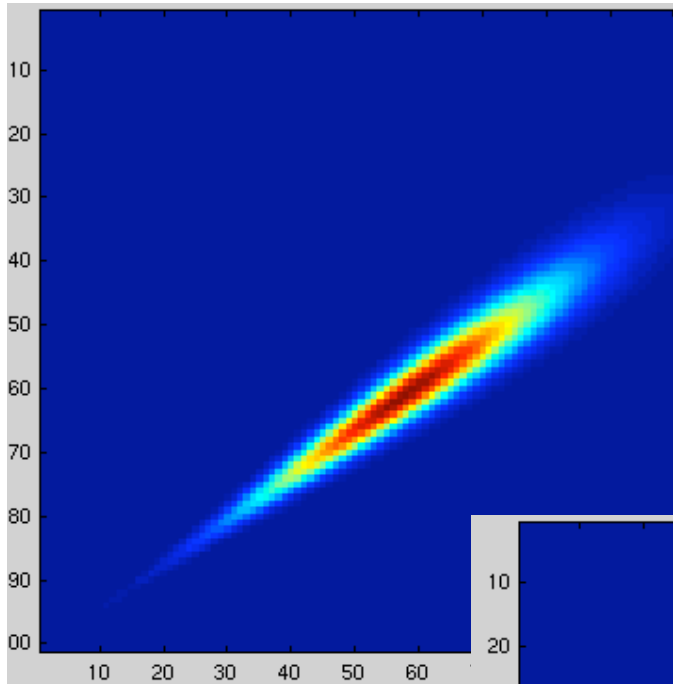


# Bearings-Only Example

Say a second sensor at  $(0,100)$  also takes a second noisy measurement of the target. We this have another likelihood function to represent this second observation.

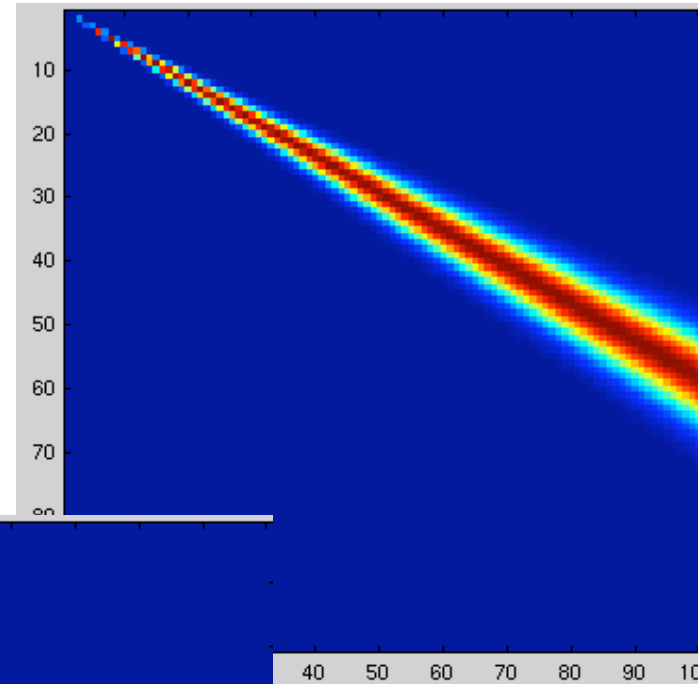


# Bearings-Only Example

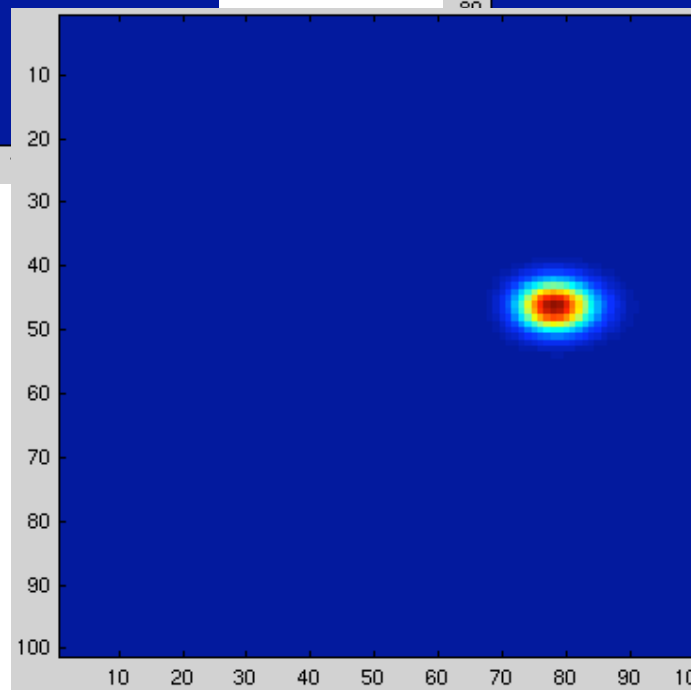


prior = our  
old posterior

new posterior



likelihood



becomes prior for  
new observations

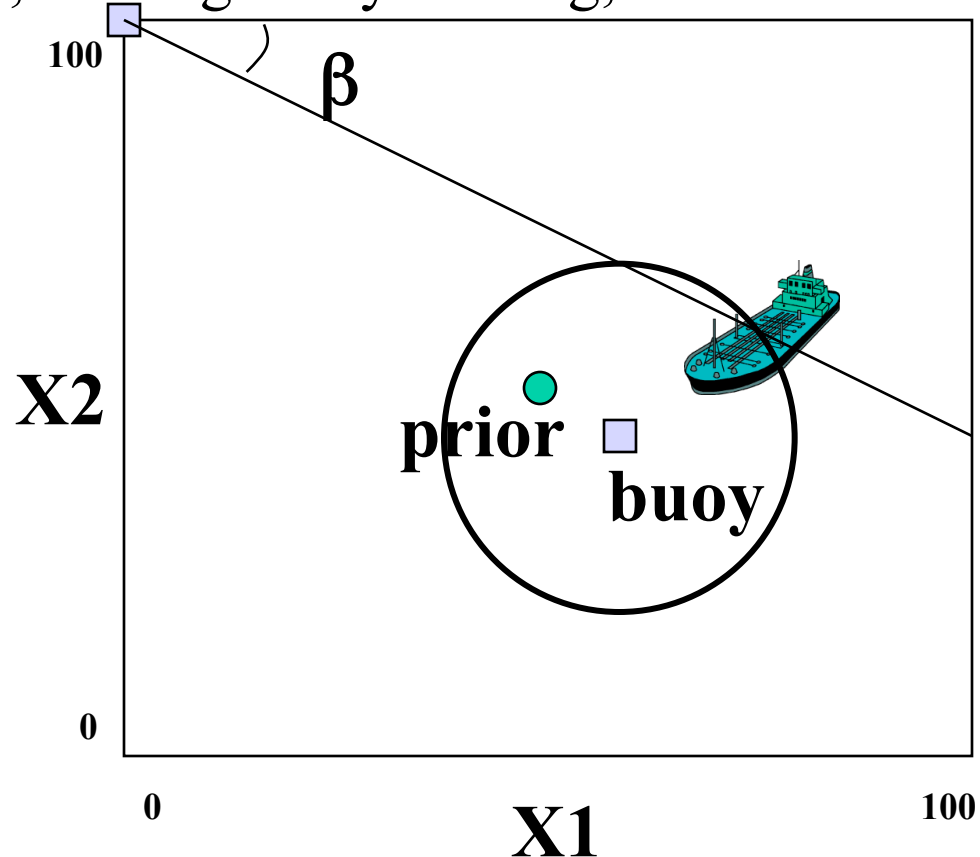




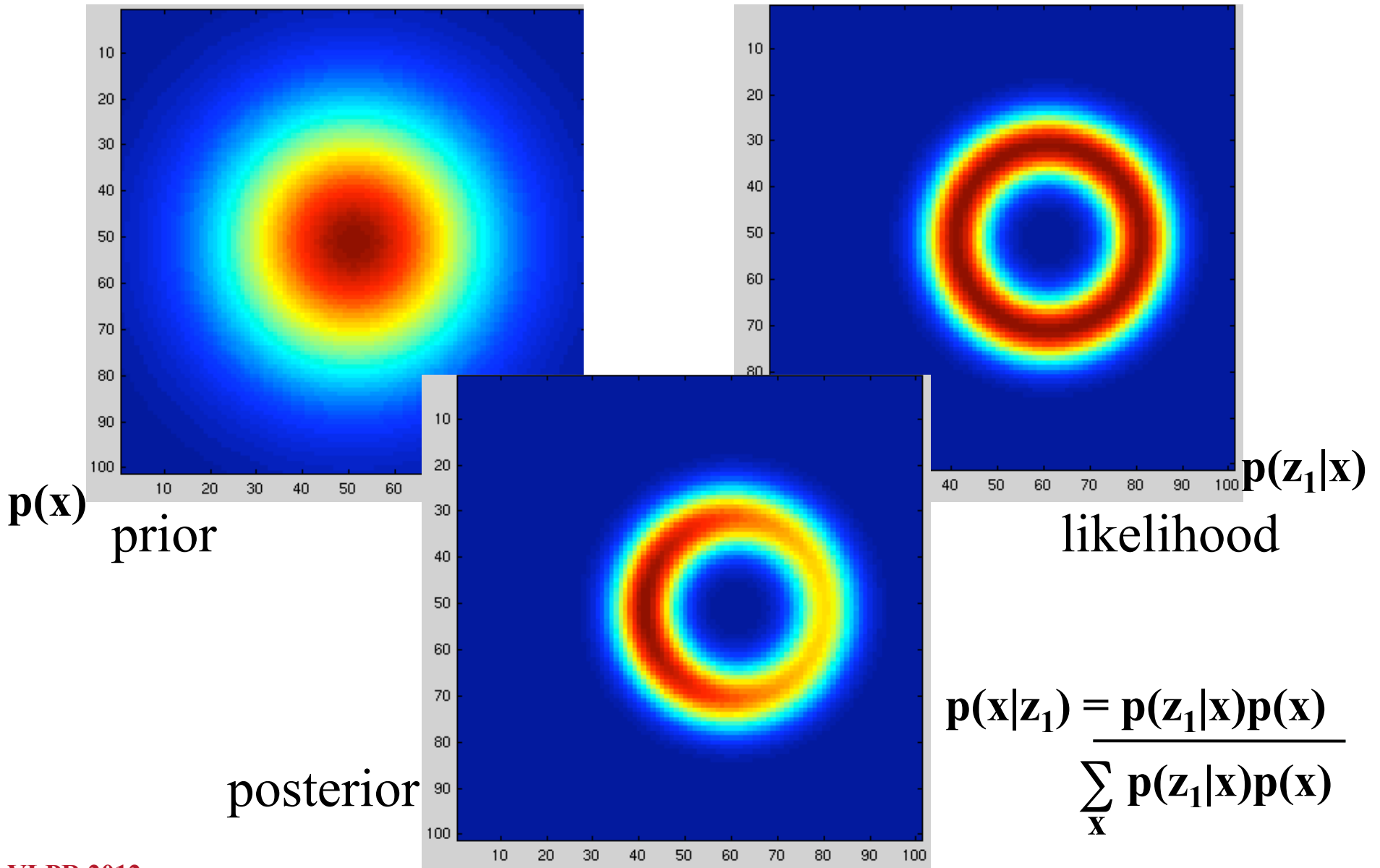
# A More Interesting Example

Let's say our ship wants to be found, and is broadcasting a radio signal, picked up by a transmitter on a buoy. That gives us a known distance to the ship.

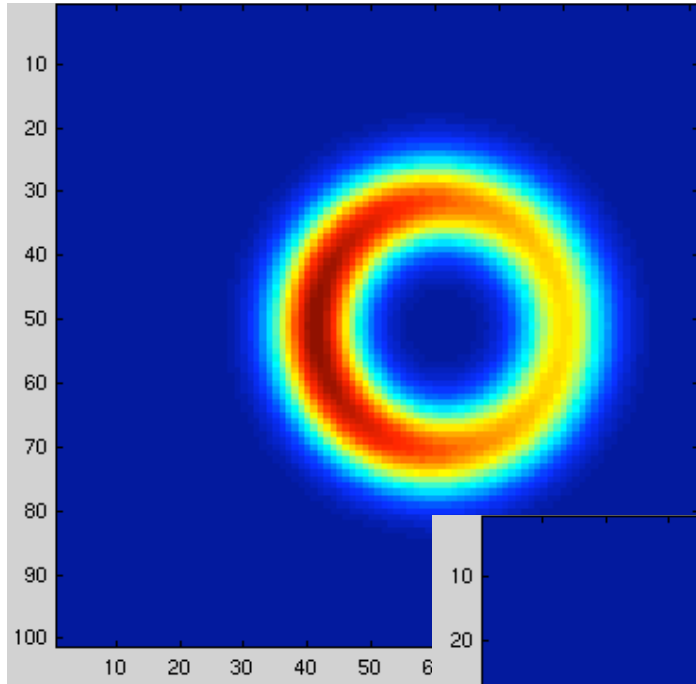
A second, bearings-only reading, is also taken...



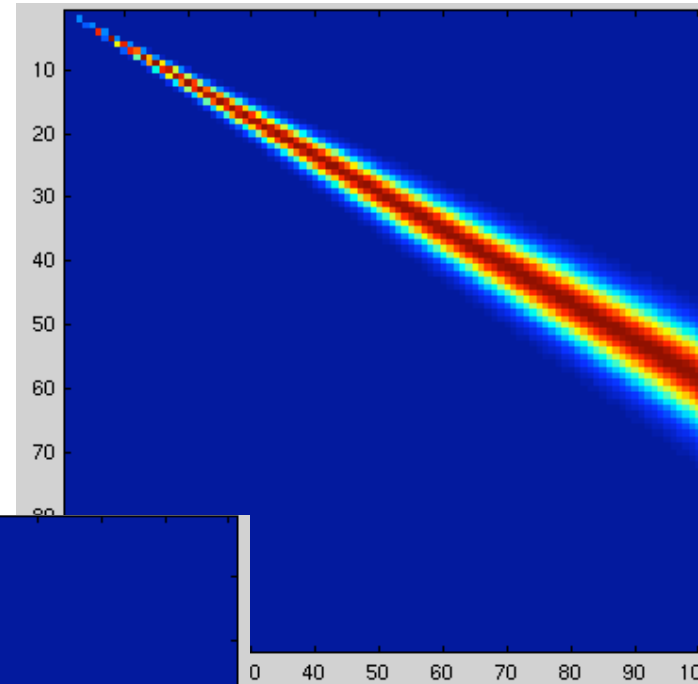
# Range+Bearings Example



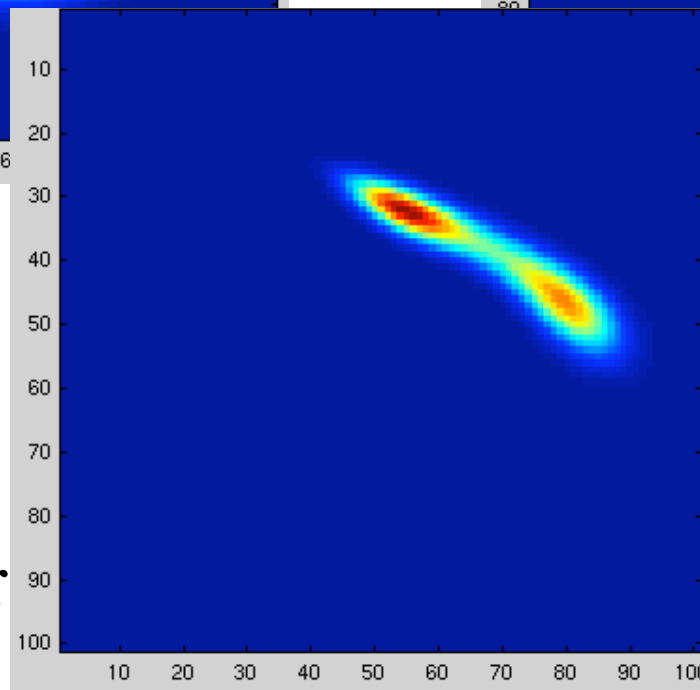
# Range+Bearings Example



prior  
 $p(\mathbf{x}|\mathbf{z}_1)$



likelihood  
 $p(\mathbf{z}_2|\mathbf{x})$



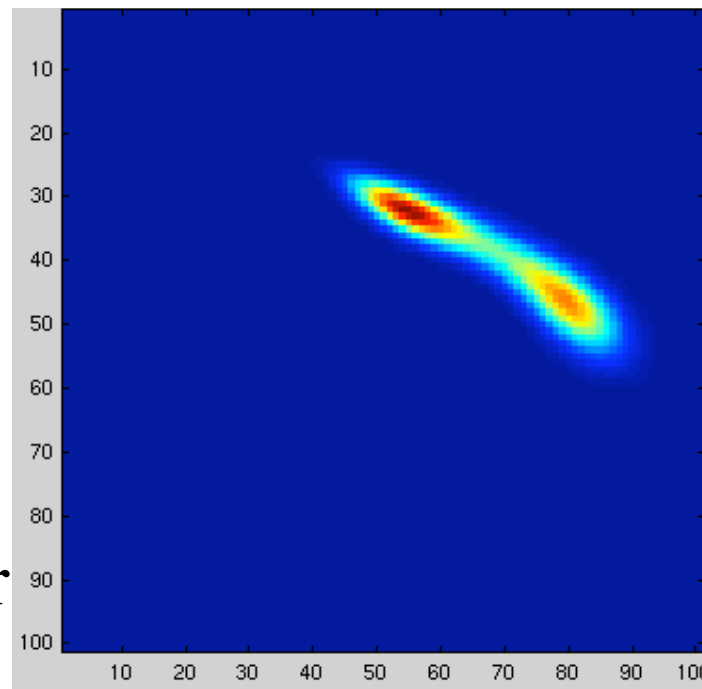
posterior

$$p(\mathbf{x}|\mathbf{z}_1, \mathbf{z}_2) = \frac{p(\mathbf{z}_2|\mathbf{x})p(\mathbf{x}|\mathbf{z}_1)}{\sum_{\mathbf{x}} p(\mathbf{z}_2|\mathbf{x})p(\mathbf{x}|\mathbf{z}_1)}$$

# Range+Bearings Example

Note the posterior distribution is multimodal. Presumably a second bearings reading from a sensor at the lower left of the region would disambiguate the location. But it is important that the multimodality be preserved, in order for that to happen! If we only kept the highest peak (in this example), we would get the wrong answer.

posterior



# Adding Motion Prediction

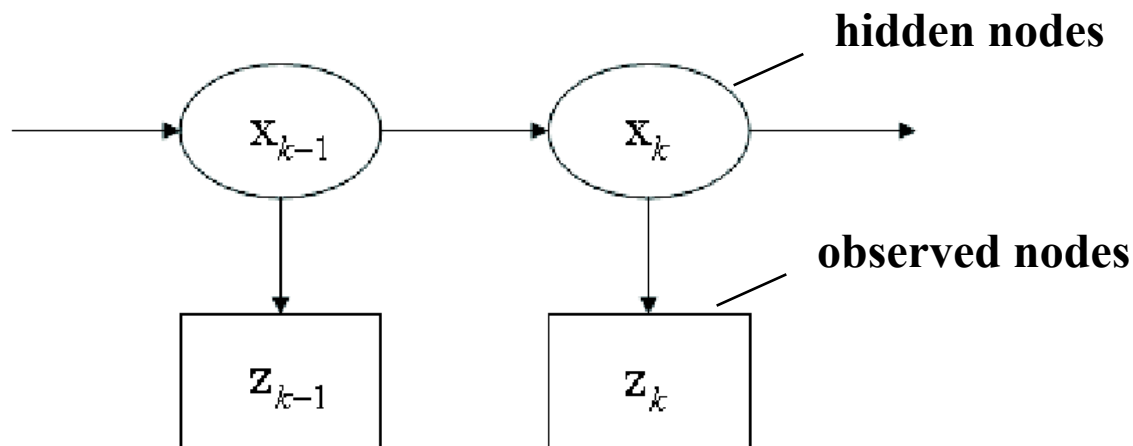
Our previous examples left out something important...

We didn't take into account that the target could be moving!

To do this we also need to model  $p(\mathbf{x}_t | \mathbf{x}_{t-1})$

# Tracking as Bayesian Estimation

Graphical Model:



## Markov assumptions

$$p(\mathbf{x}_k | \mathbf{x}_0, \dots, \mathbf{x}_{k-1}) = p(\mathbf{x}_k | \mathbf{x}_{k-1})$$

$$p(\mathbf{z}_k | \mathbf{x}_0, \dots, \mathbf{x}_k) = p(\mathbf{z}_k | \mathbf{x}_k)$$

## Factored joint probability distribution

$$p(\mathbf{x}_0, \dots, \mathbf{x}_k, \mathbf{z}_1, \dots, \mathbf{z}_k) = p(\mathbf{x}_0) \prod_{i=1}^k p(\mathbf{z}_i | \mathbf{x}_i) p(\mathbf{x}_i | \mathbf{x}_{i-1})$$

# Recursive Bayes Filter

Motion Prediction Step:

**predicted current state**                      **state transition**                      **previous estimated state**

$$\underline{p(\mathbf{x}_k | \mathbf{z}_{1:k-1})} = \int \underline{p(\mathbf{x}_k | \mathbf{x}_{k-1})} \underline{p(\mathbf{x}_{k-1} | \mathbf{z}_{1:k-1})} d\mathbf{x}_{k-1}.$$

Data Correction Step (Bayes rule):

**estimated current state**                      **measurement**                      **predicted current state**

$$\underline{p(\mathbf{x}_k | \mathbf{z}_{1:k})} = \frac{\underline{p(\mathbf{z}_k | \mathbf{x}_k)} \underline{p(\mathbf{x}_k | \mathbf{z}_{1:k-1})}}{\underline{p(\mathbf{z}_k | \mathbf{z}_{1:k-1})}}$$

↓ **normalization term**

$$p(\mathbf{z}_k | \mathbf{z}_{1:k-1}) = \int p(\mathbf{z}_k | \mathbf{x}_k) p(\mathbf{x}_k | \mathbf{z}_{1:k-1}) d\mathbf{x}_k$$

# Problem

**Except in special cases, these integrals are intractable.**

Motion Prediction Step:

$$p(\mathbf{x}_k | \mathbf{z}_{1:k-1}) = \int p(\mathbf{x}_k | \mathbf{x}_{k-1}) p(\mathbf{x}_{k-1} | \mathbf{z}_{1:k-1}) d\mathbf{x}_{k-1}.$$

Data Correction Step (Bayes rule):

$$p(\mathbf{x}_k | \mathbf{z}_{1:k}) = \frac{p(\mathbf{z}_k | \mathbf{x}_k) p(\mathbf{x}_k | \mathbf{z}_{1:k-1})}{p(\mathbf{z}_k | \mathbf{z}_{1:k-1})}$$

$$p(\mathbf{z}_k | \mathbf{z}_{1:k-1}) = \int p(\mathbf{z}_k | \mathbf{x}_k) p(\mathbf{x}_k | \mathbf{z}_{1:k-1}) d\mathbf{x}_k$$



# Such Special Cases

All pdfs are Gaussian → Kalman Filtering

Monte Carlo Integration → Particle Filtering; MCMC

Hill-climbing on posterior → Mean-Shift; Lucas-Kanade

and of course there are lots  
of papers on all of these

# Filtering and Data Association

- Filtering
  - Recursive Bayesian estimation
  - (continuous) Probability Theory

- Data Association
  - Assignment problems
  - (discrete) Combinatorics

# Intro to Data Association

Let's consider a different tracking approach

- Detect objects in each frame.
- Determine interframe correspondences between them.

Actually, in “ancient” times when tracking meant looking at blips on a radar screen, this was the natural approach.

It is popular again due to successes in object detection.

# Remainder of this Talk

- Data Association
  - associate two sets of objects
  - e.g. matching detections across frames
- Two frames: linear assignment problem
- Generalize to three or more frames

– Greedy Method

– Mincost Network Flow

– Multidimensional Assignment ← **NP-hard**

**Polynomial time**



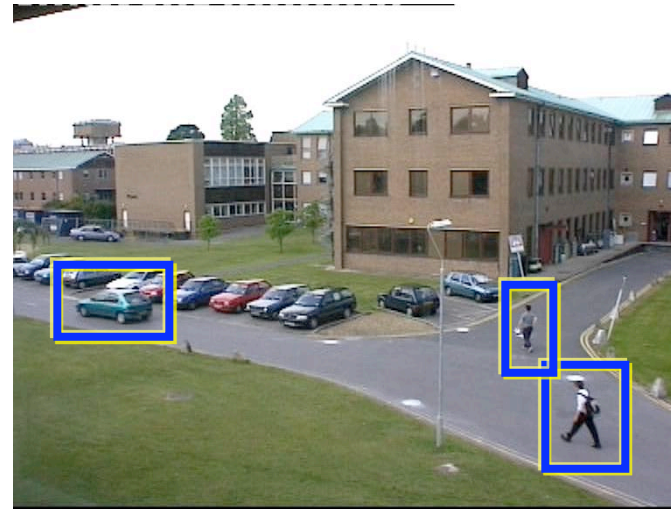
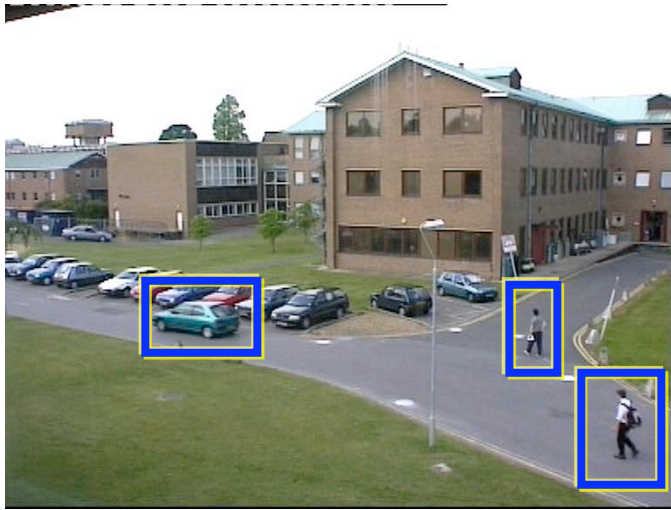
**increasing  
solution  
quality**

↓ approximate solution

R.Collins, "Multitarget Data Association with Higher-Order Motion Models," *CVPR 2012*.

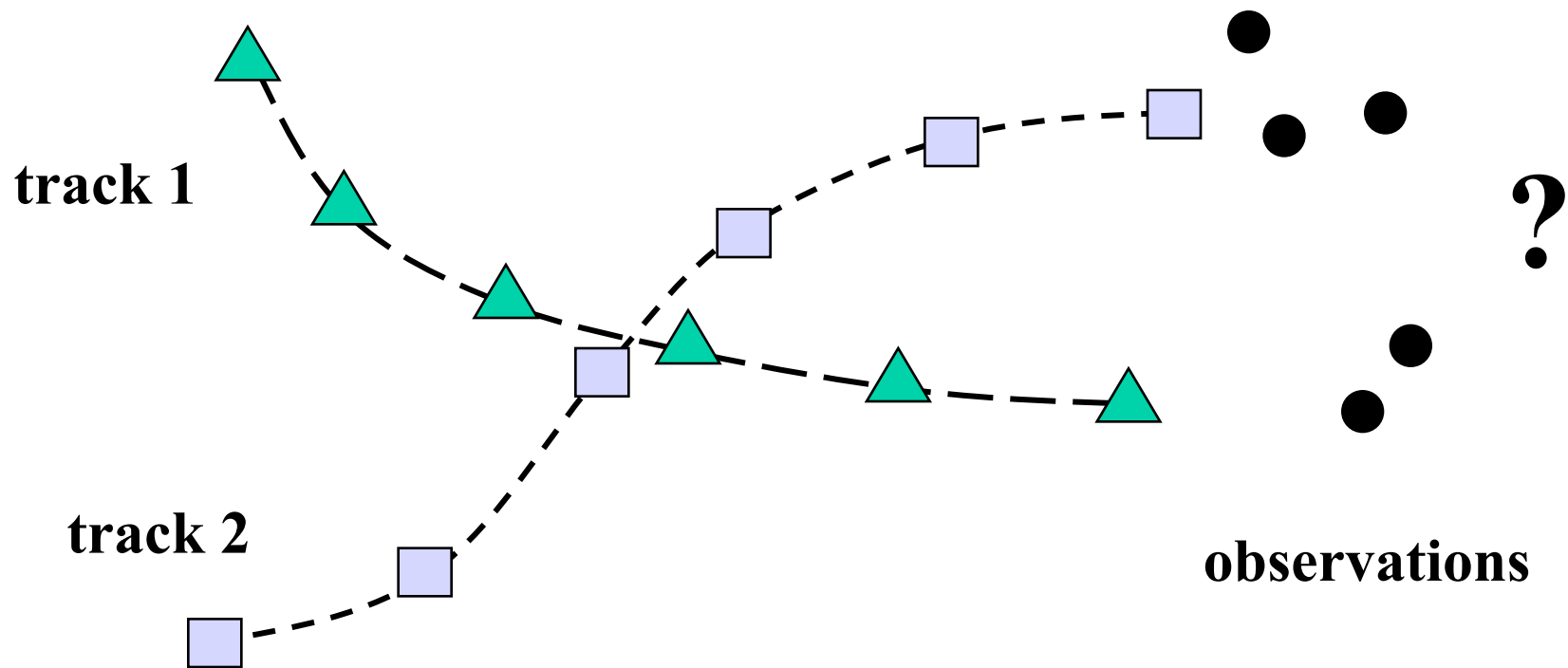
# Data Association Scenarios

Match object detections across frames



# Data Association Scenarios







Match a current set of trajectories to object detections in the next frame



How to determine which observations to add to which track?

# Linear Assignment Formulation

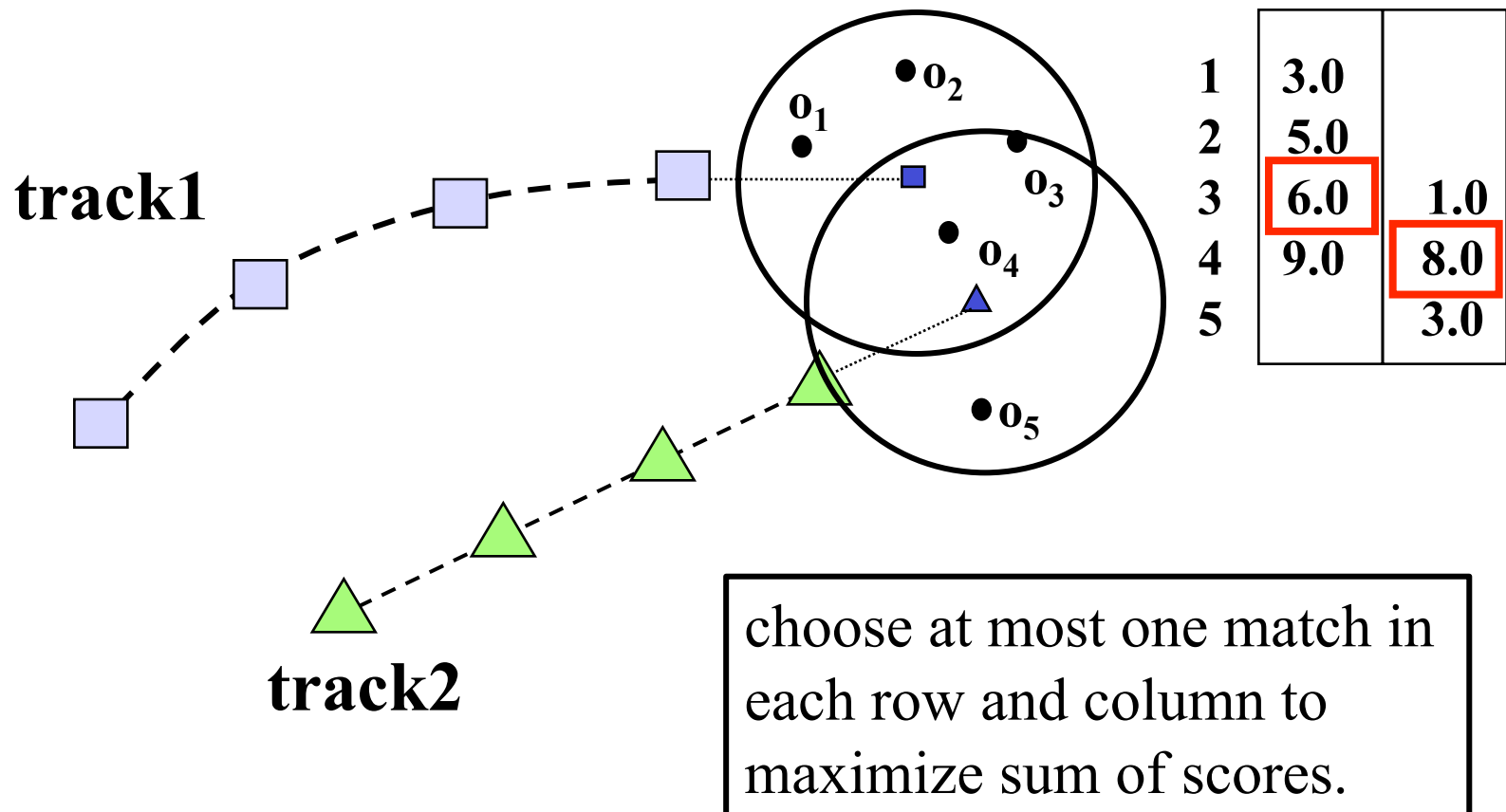
Form a matrix of  
pairwise similarity  
scores

		Frame T+1		
				
Frame T		.11	<b>.95</b>	.23
		.85	.25	<b>.89</b>
		<b>.90</b>	.12	.81

We want to choose one match from each  
row and column to maximize sum of scores

# Linear Assignment Formulation

Extend each trajectory (motion prediction) and assign scores to each observation based on inverse distance such that closer observations get higher numbers.





# Linear Assignment Problem

## Mathematical Definition

maximize:  $\sum_{i=1}^n \sum_{j=1}^n w_{ij} x_{ij}$

subject to:  $\sum_j x_{ij} = 1; \quad i = 1, 2, \dots, n$   
 $\sum_i x_{ij} = 1; \quad j = 1, 2, \dots, n$   
 $x_{ij} \in \{0, 1\}$

constraints that say  
X is a permutation matrix

The permutation matrix ensures that we only match up one object from each row and from each column.

note: alternately, we can minimize costs rather than maximize weights

minimize:  $\sum_{i=1}^n \sum_{j=1}^n c_{ij} x_{ij}$

# Hungarian Algorithm

There is an algorithm called Kuhn-Munkres or “Hungarian” algorithm specifically developed to efficiently solve the linear assignment problem.

If you need to solve LAP, you should use it.

However, we are going to look at other algorithms, because they generalize more readily to multi-frame ( $> 2$  frames) problems.

# Greedy Solution to LAP

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	0.95	<del>0.76</del>	<del>0.62</del>	<del>0.41</del>	<del>0.06</del>
2	<del>0.23</del>	<del>0.46</del>	<del>0.79</del>	0.94	<del>0.35</del>
3	<del>0.61</del>	<del>0.02</del>	0.92	<del>0.92</del>	<del>0.81</del>
4	<del>0.49</del>	0.82	<del>0.74</del>	<del>0.41</del>	<del>0.01</del>
5	<del>0.89</del>	<del>0.44</del>	<del>0.18</del>	<del>0.89</del>	0.14

$$\text{Score} = .95 + .94 + .92 + .82 + .14 = 3.77$$

Is this the best we can do?

# Greedy Solution to LAP

No!

	1	2	3	4	5
1	0.95	0.76	0.62	0.41	0.06
2	0.23	0.46	0.79	0.94	0.35
3	0.61	0.02	0.92	0.92	0.81
4	0.49	0.82	0.74	0.41	0.01
5	0.89	0.44	0.18	0.89	0.14

**Greedy Solution**

**Score=3.77**

	1	2	3	4	5
1	0.95	0.76	0.62	0.41	0.06
2	0.23	0.46	0.79	0.94	0.35
3	0.61	0.02	0.92	0.92	0.81
4	0.49	0.82	0.74	0.41	0.01
5	0.89	0.44	0.18	0.89	0.14

**Optimal Solution**

**Score=4.26**

Greedy method is easy to program; quick to run; and yields “pretty good” solutions in practice.

But it often does not yield the optimal solution.

Robert Collins  
Penn State

Small (3x3) example

	b1	b2	b3
a1	<b>3</b>	<b>2</b>	<b>3</b>
a2	<b>2</b>	<b>1</b>	<b>3</b>
a3	<b>4</b>	<b>5</b>	<b>1</b>

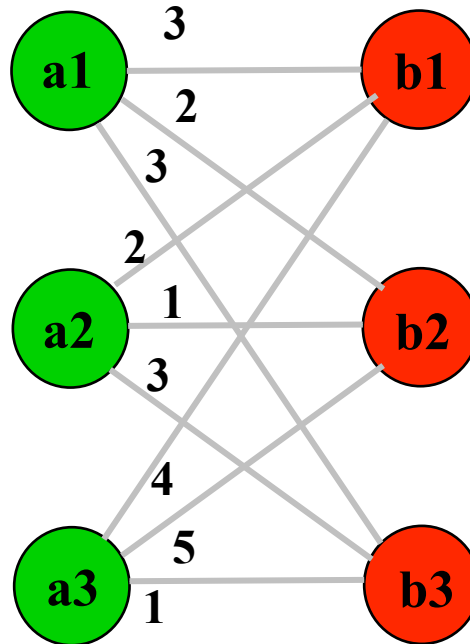
# Min-Cost Flow

Robert Collins  
Penn State

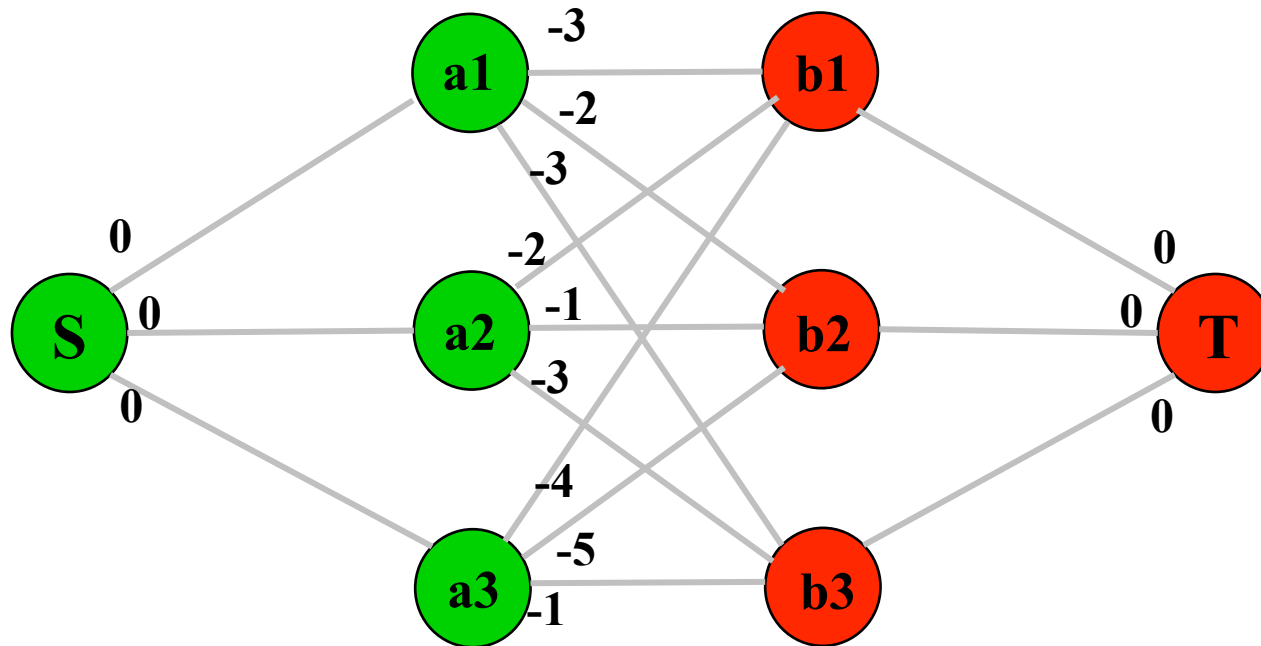
Small (3x3) example

	b1	b2	b3
a1	<b>3</b>	<b>2</b>	<b>3</b>
a2	<b>2</b>	<b>1</b>	<b>3</b>
a3	<b>4</b>	<b>5</b>	<b>1</b>

# Min-Cost Flow



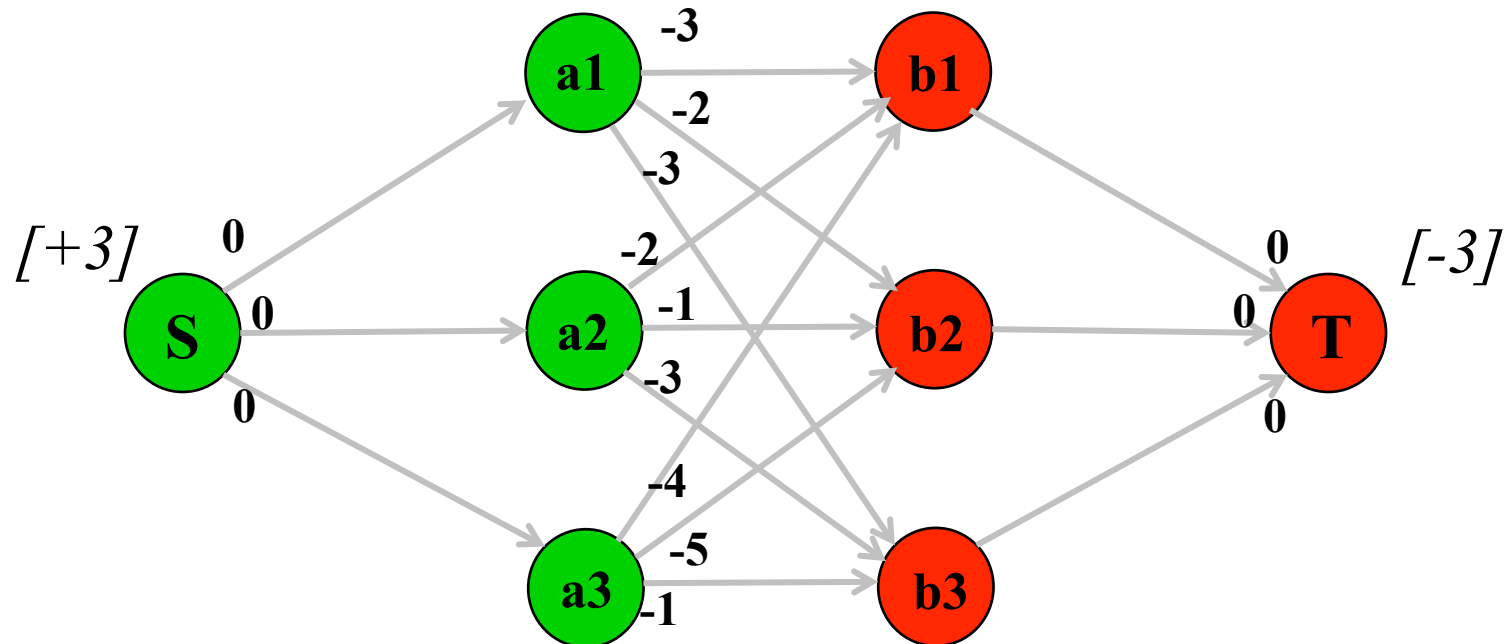
# Min-Cost Flow



transform weights into costs  $c_{ij} = \alpha - w_{ij}$

add source/sink nodes with 0 cost

# Min-Cost Flow



transform weights into costs  $c_{ij} = \alpha - w_{ij}$

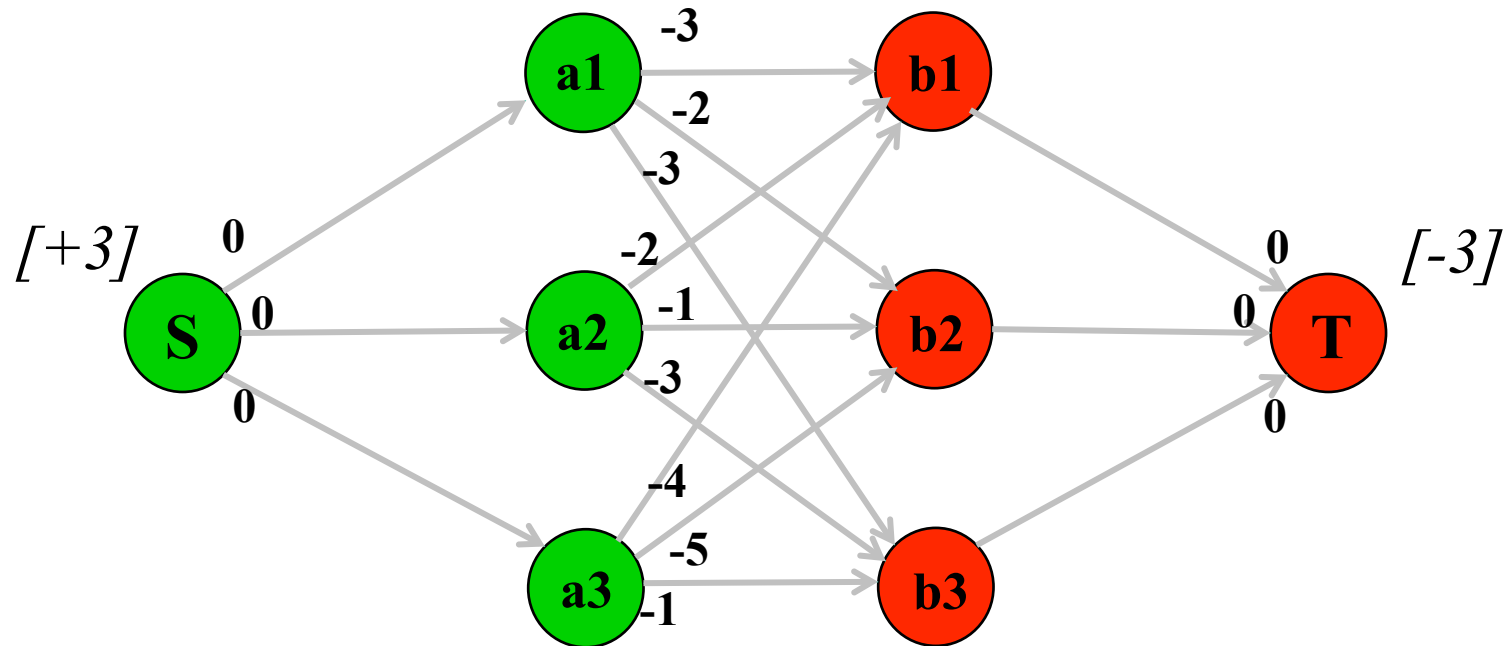
add source/sink nodes with 0 cost

directed edges with flow capacity of 1

pump N units of flow from source to sink

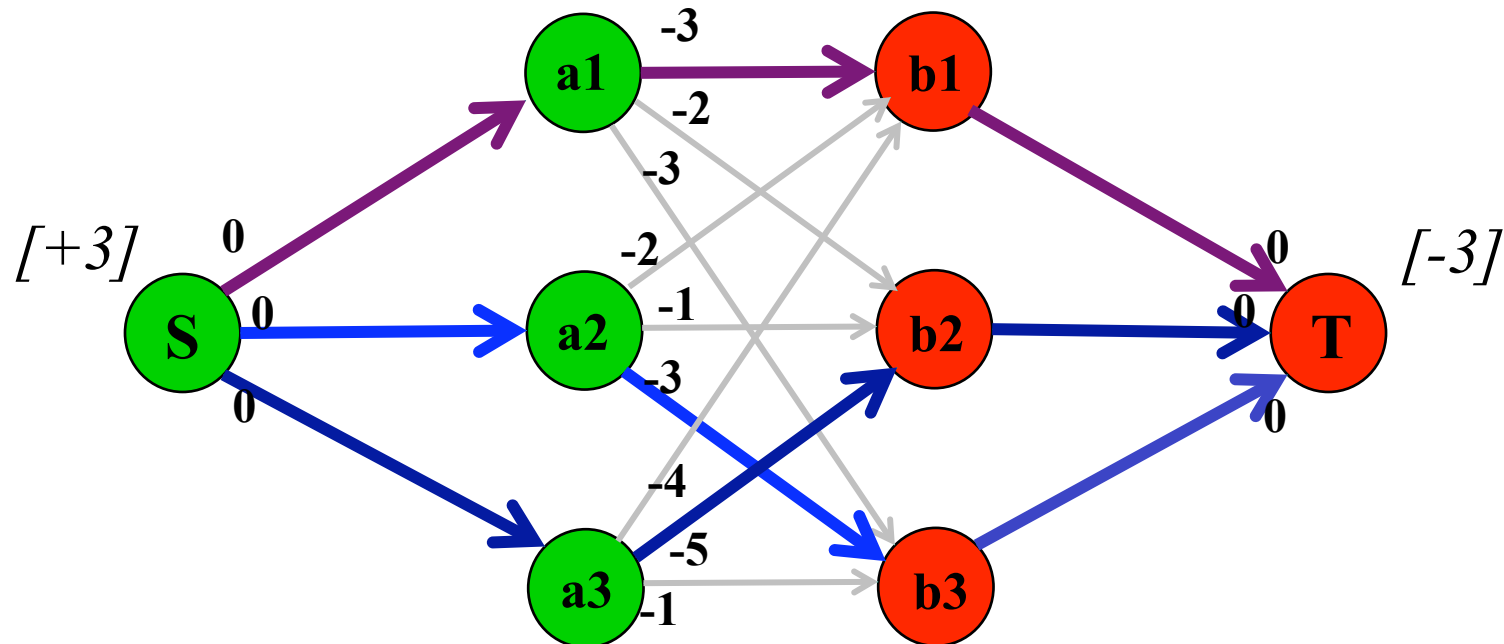


# Min-Cost Flow



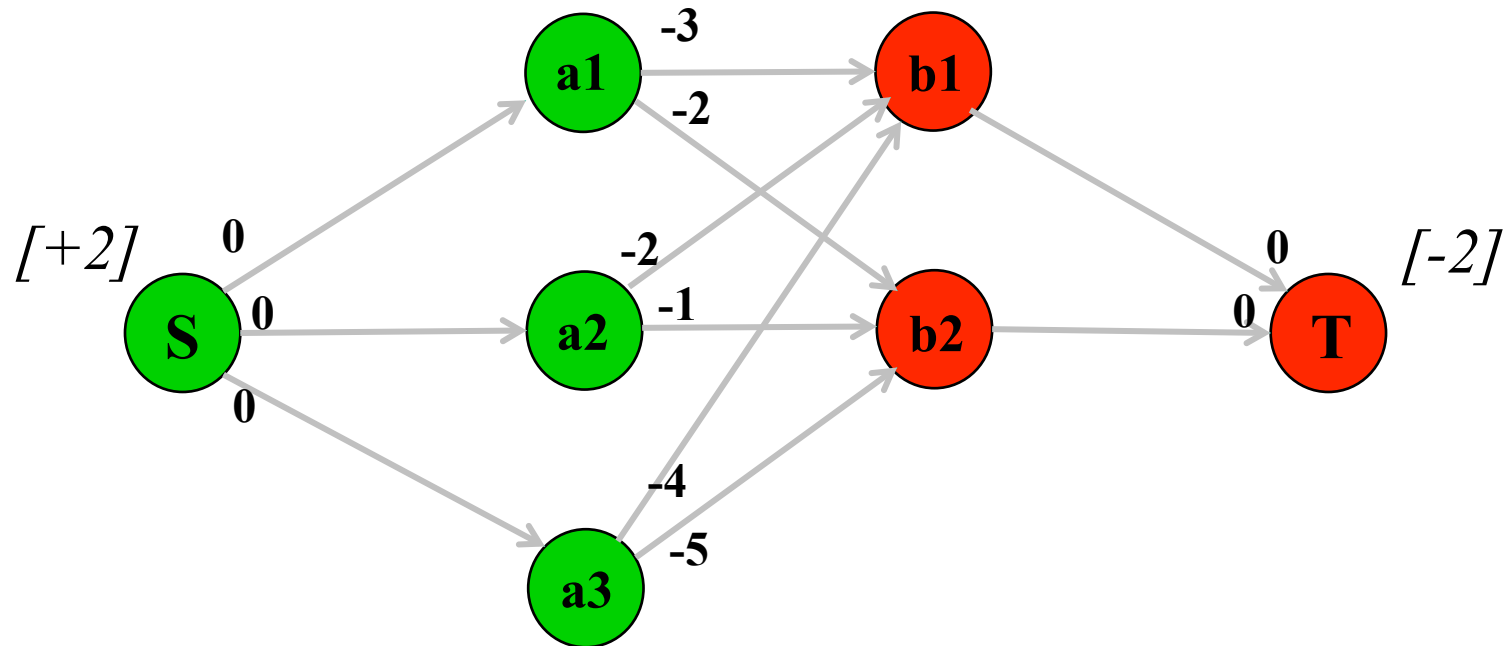
note: interior nodes become transshipment nodes  
(sum flow out = sum flow in)

# Min-Cost Flow



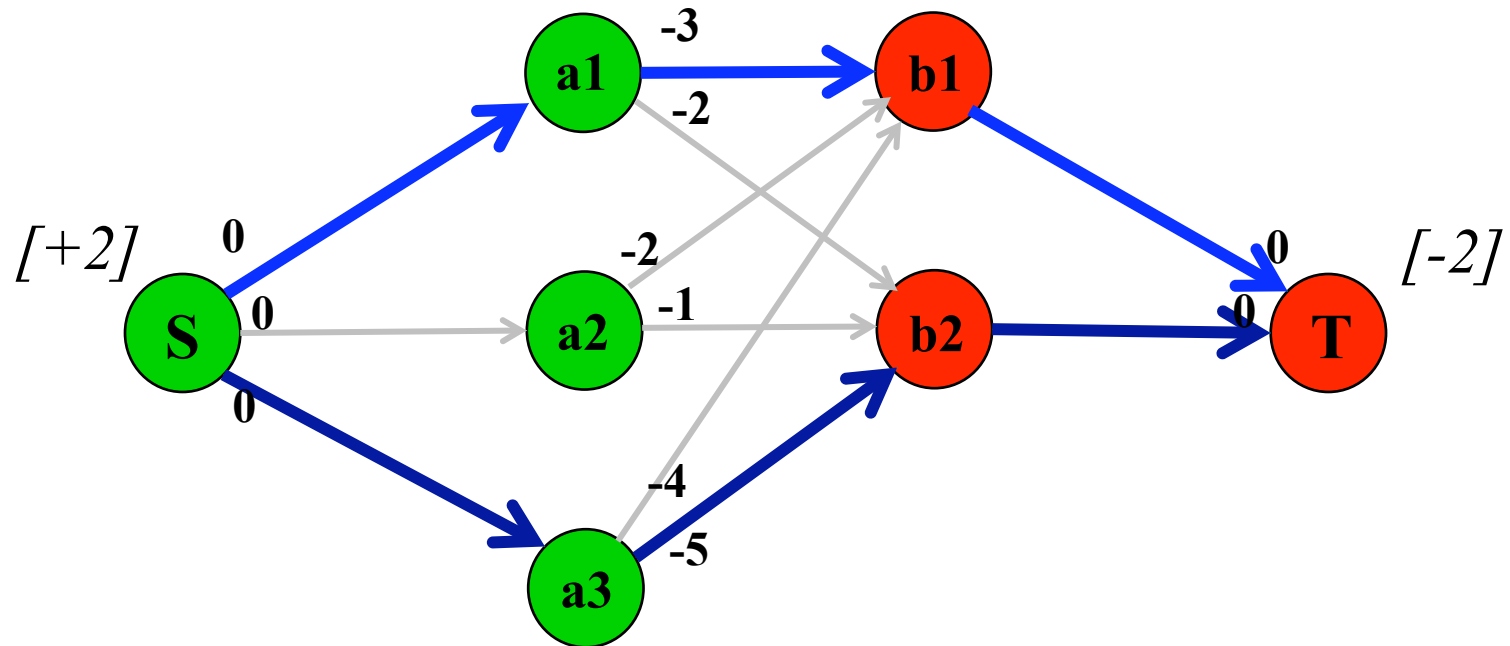
There are standard algorithms for efficiently solving min-cost network flow, e.g. push-relabel algorithm; or successive shortest paths.

# Min-Cost Flow



Nice property: The min-cost flow formalism readily generalizes to matching sets with unequal sizes.

# Min-Cost Flow



Min-cost flow formalism readily generalizes to matching between sets with unequal sizes.

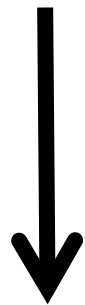
# Last Words on LAP

- As I mentioned earlier, the solution used in practice is Kuhn-Munkres (aka Hungarian) algorithm.
- Main point is that exact solution to LAP can be found efficiently (in polynomial time).

# Remainder of this Talk

- Data Association
  - associate two sets of objects
  - e.g. matching detections across frames
- Two frames: linear assignment problem
- Generalize to three or more frames

- Greedy Method
  - Mincost Network Flow
  - Multidimensional Assignment ← **NP-hard**
- Polynomial time**



**increasing  
solution  
quality**

↓ approximate solution

R.Collins, "Multitarget Data Association with Higher-Order Motion Models," *CVPR 2012*.

# Multi-Frame Generalization

Generalizing to three more more frames

- Sequential (Recursive) Greedy Method
- Mincost Network Flow
- Multidimensional Assignment

Increasing generality from top-to-bottom

# Sequential Greedy Method

- Extend a partial solution, frame-by-frame, sequentially forwards in time
- Results in a series of linear assignment problems between trajectories found up to frame  $t-1$  and new observations in frame  $t$ .

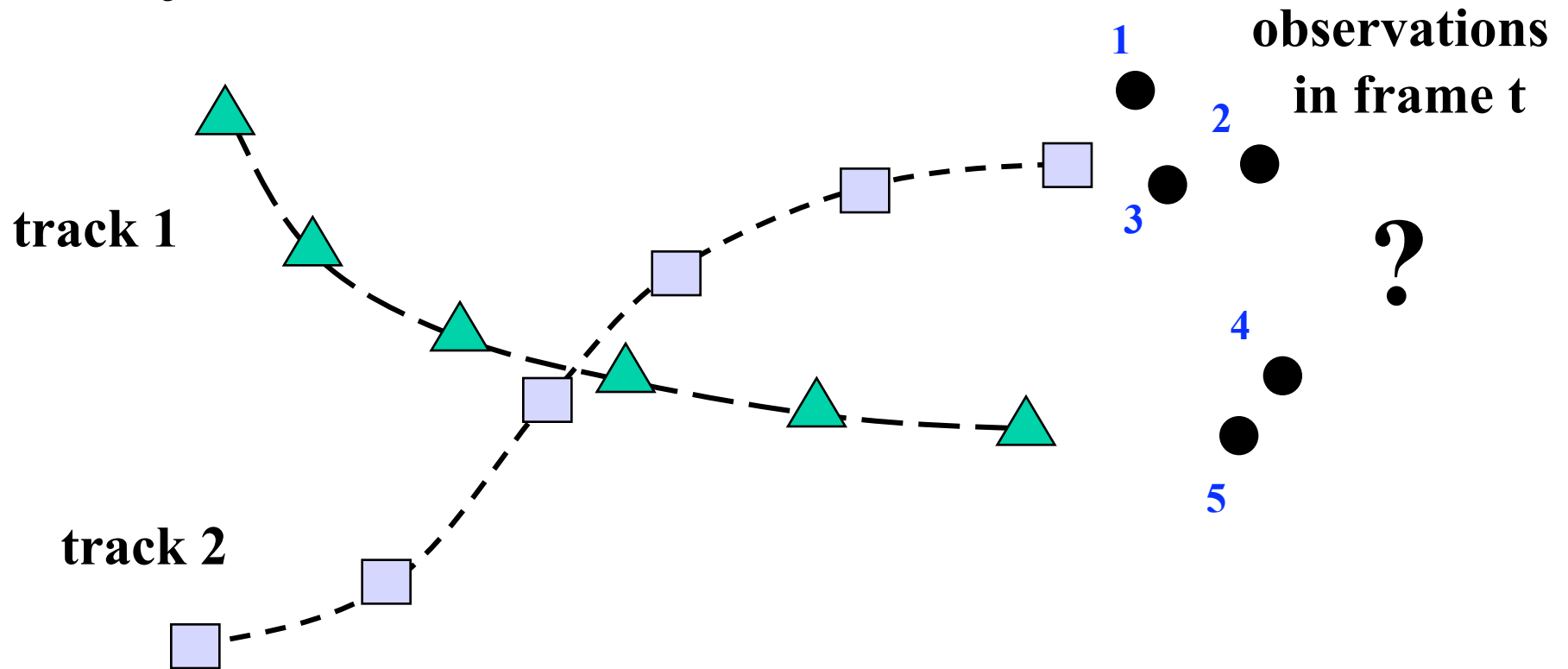
Traditional multi-target tracking approach. Often combined with recursive filtering to smooth the trajectories.

For more info: Blackman and Popoli, *Design and Analysis of Modern Tracking Systems*.



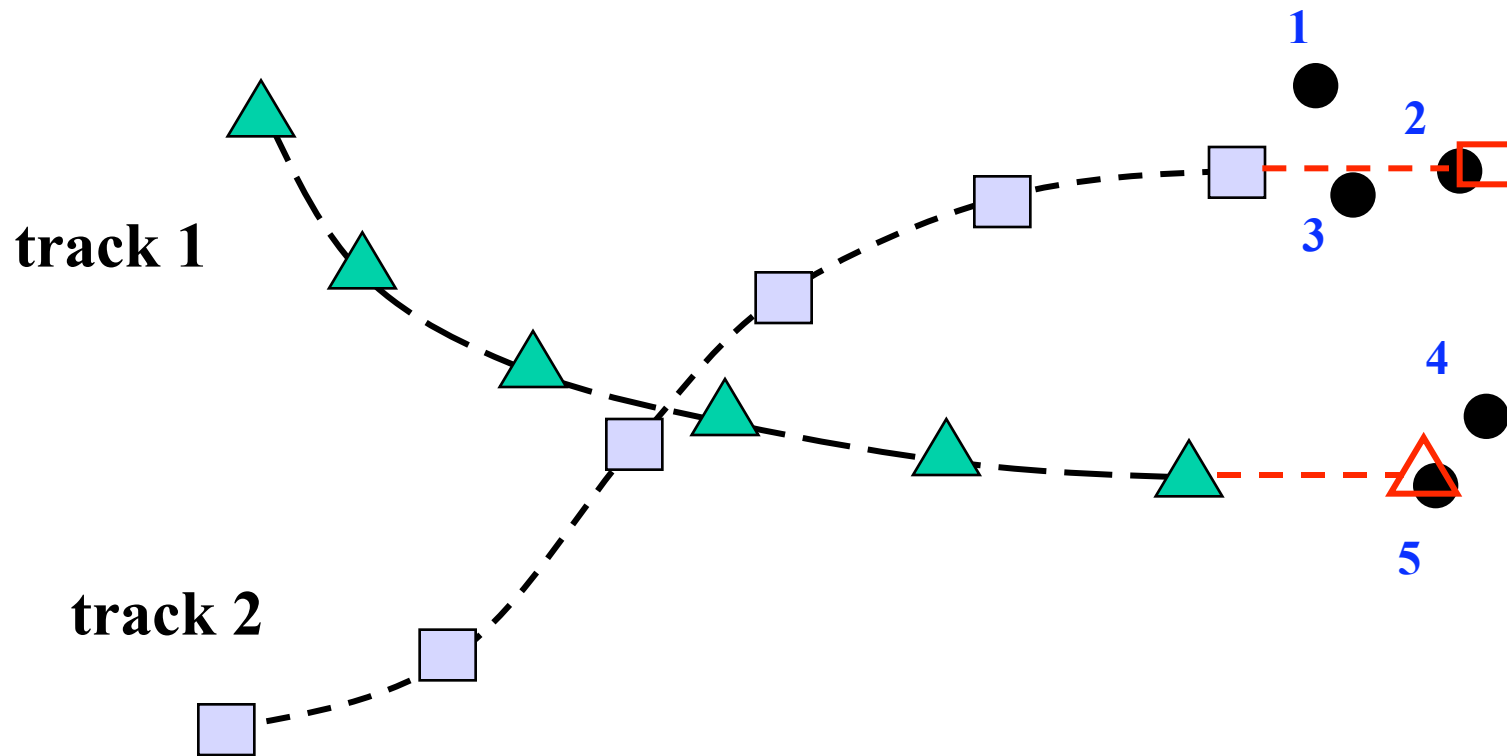
# Sequential Greedy Matching

Matching observations in a new frame to a set of trajectories from frames 1 to  $t-1$



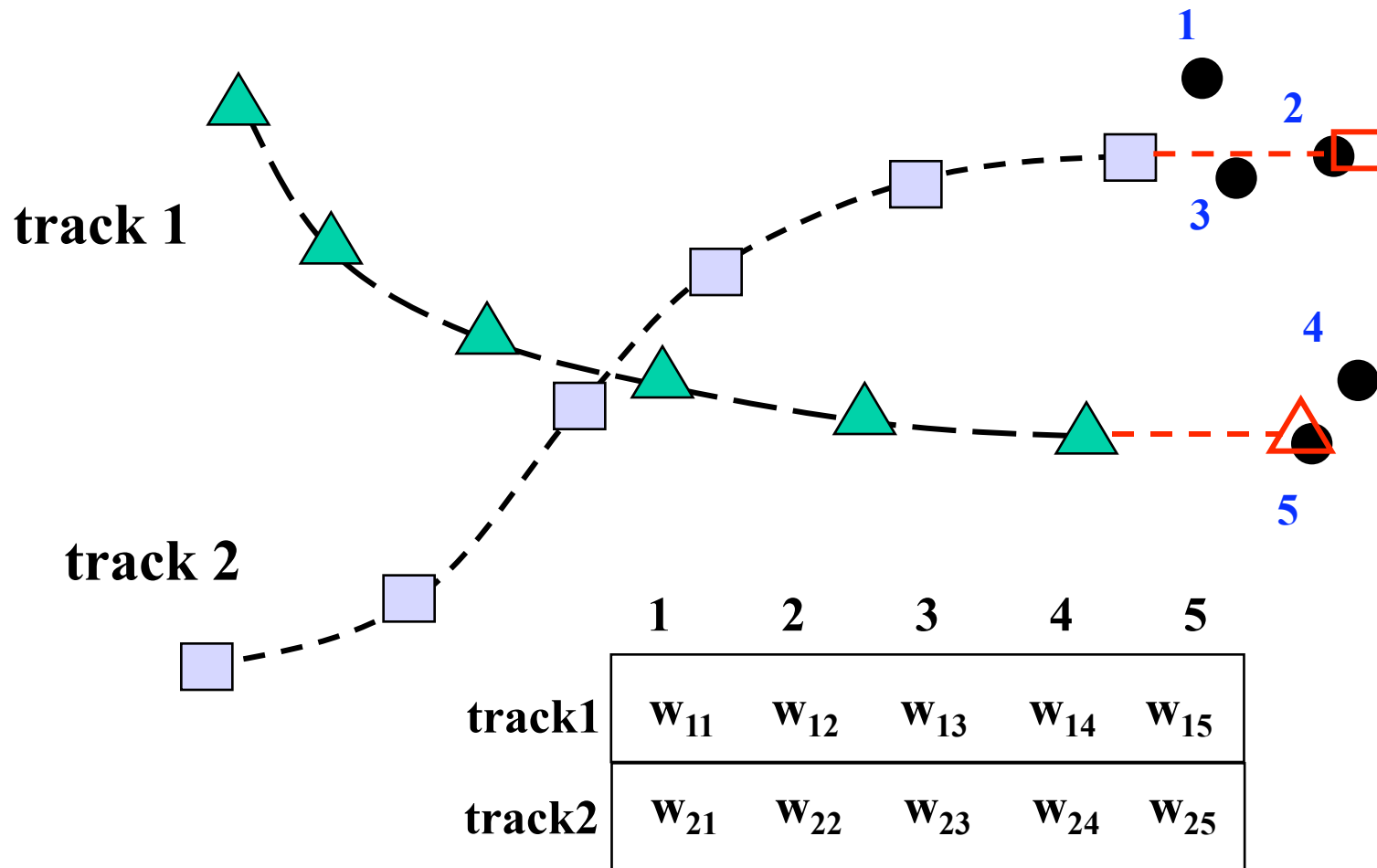
# Prediction

Predict next target position along each track via some motion model (e.g. constant velocity).



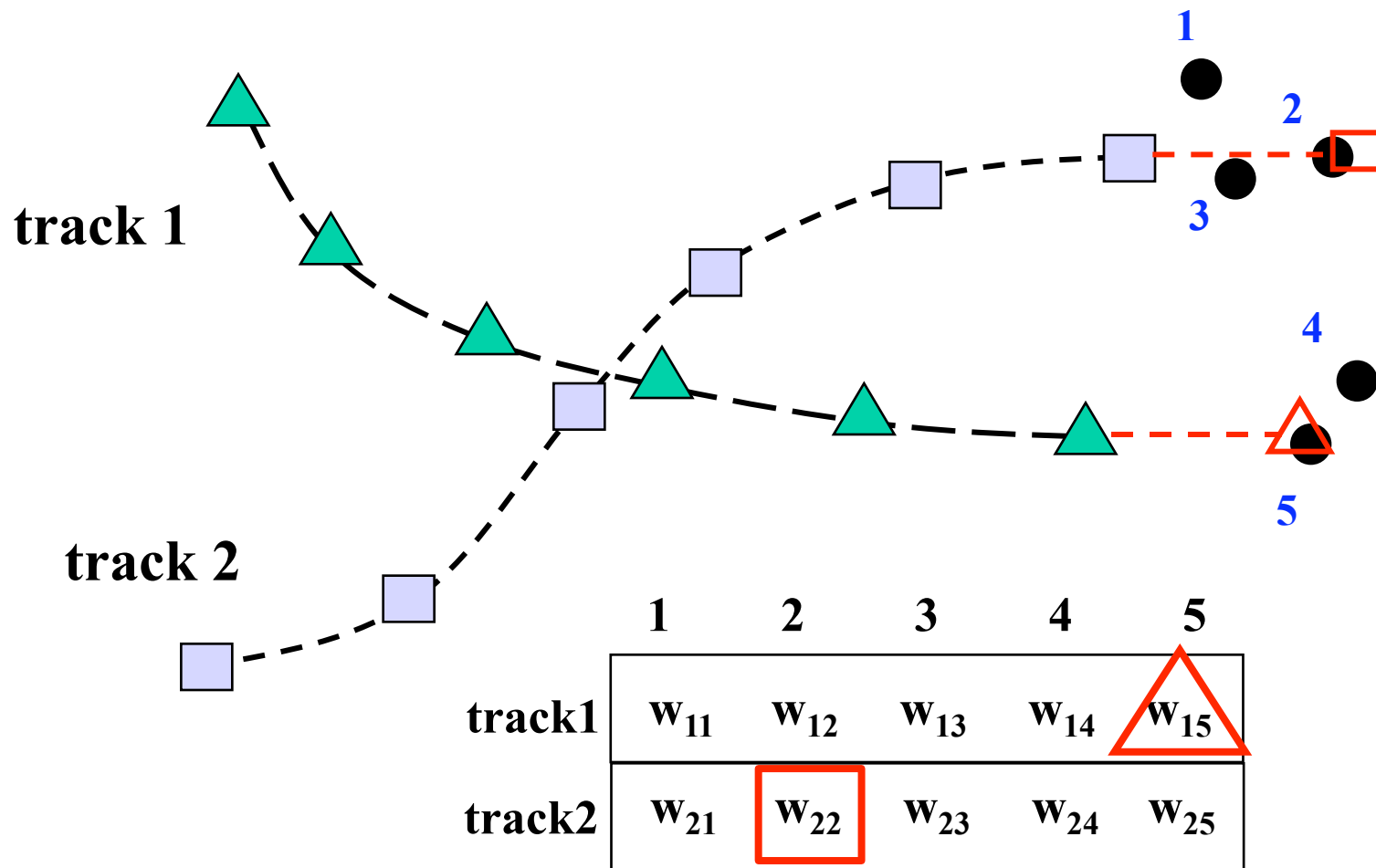
# Prediction / Scoring

Form track  $i$  to observation  $j$  scores  $w_{ij}$  based on distance and (or) appearance such that higher scores mean better matches.



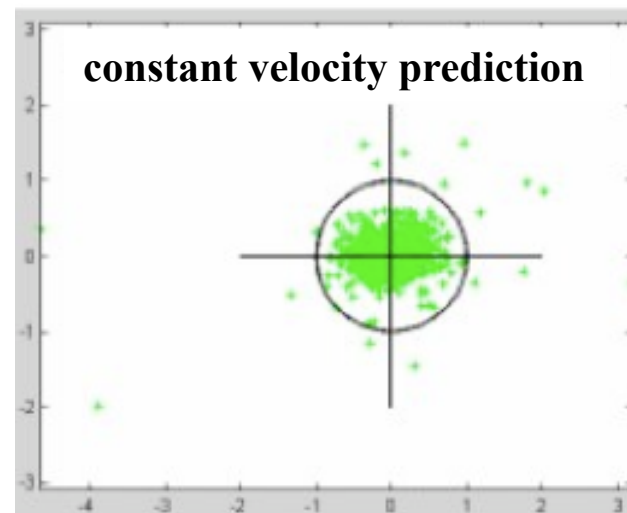
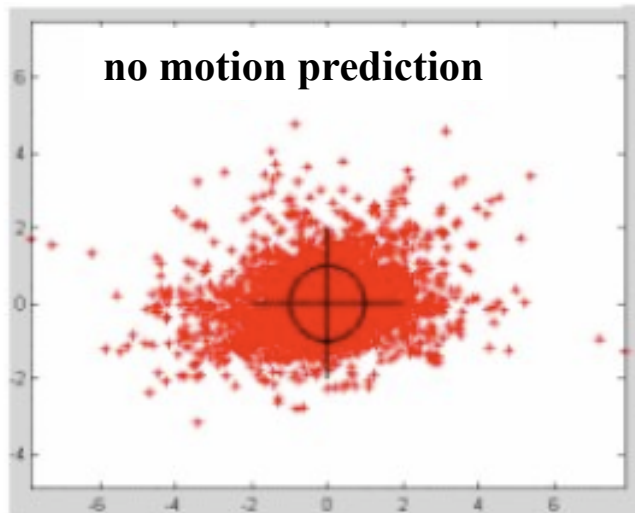
# Prediction / Scoring / Solve

Find optimal solution using Hungarian algorithm.



# Important Point

Constant velocity motion prediction greatly improves quality of matching in situations where objects are closely spaced and appearance cues are not strong.



Offset of correct match from last known location of object.

# Sequential Greedy Method

## Pros:

- Very efficient (real-time tracking)

- Very common / well-understood

## Cons:

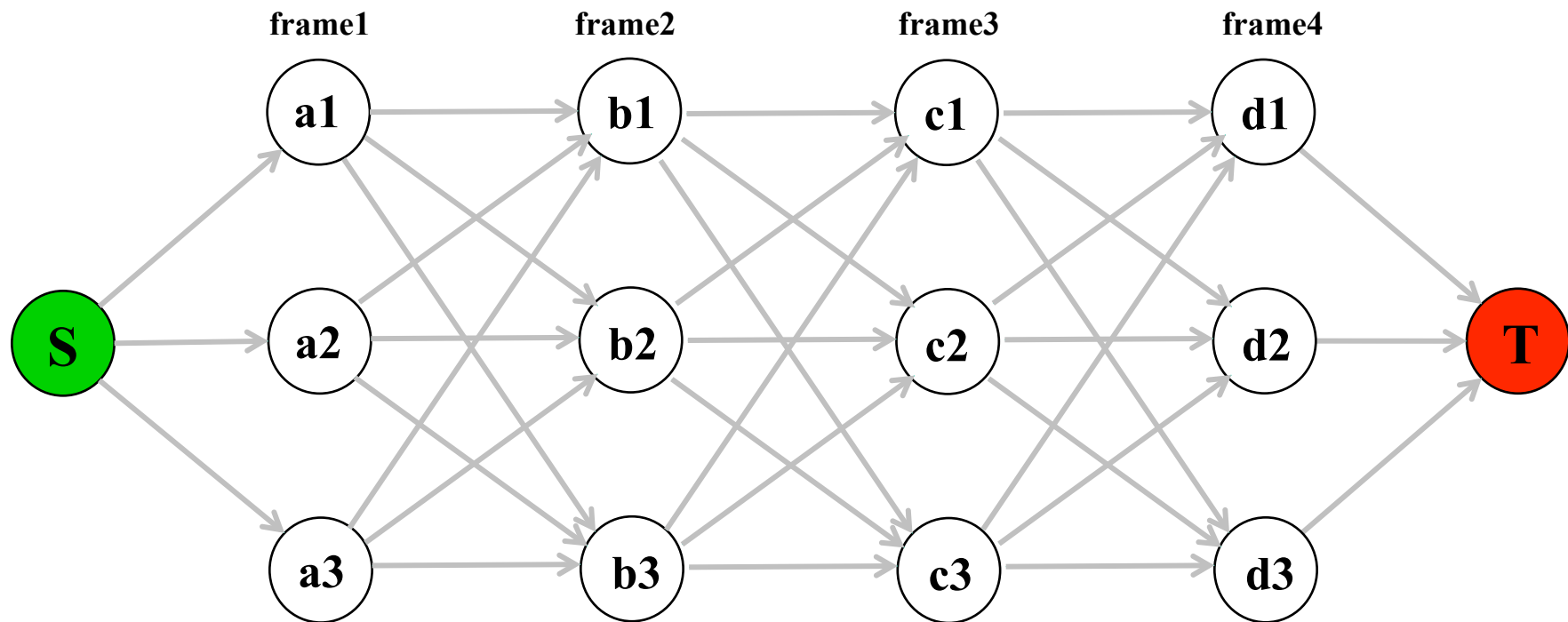
- Purely causal (no “look-ahead”)

- Matches, once made, cannot be undone if later information shows them to be suboptimal

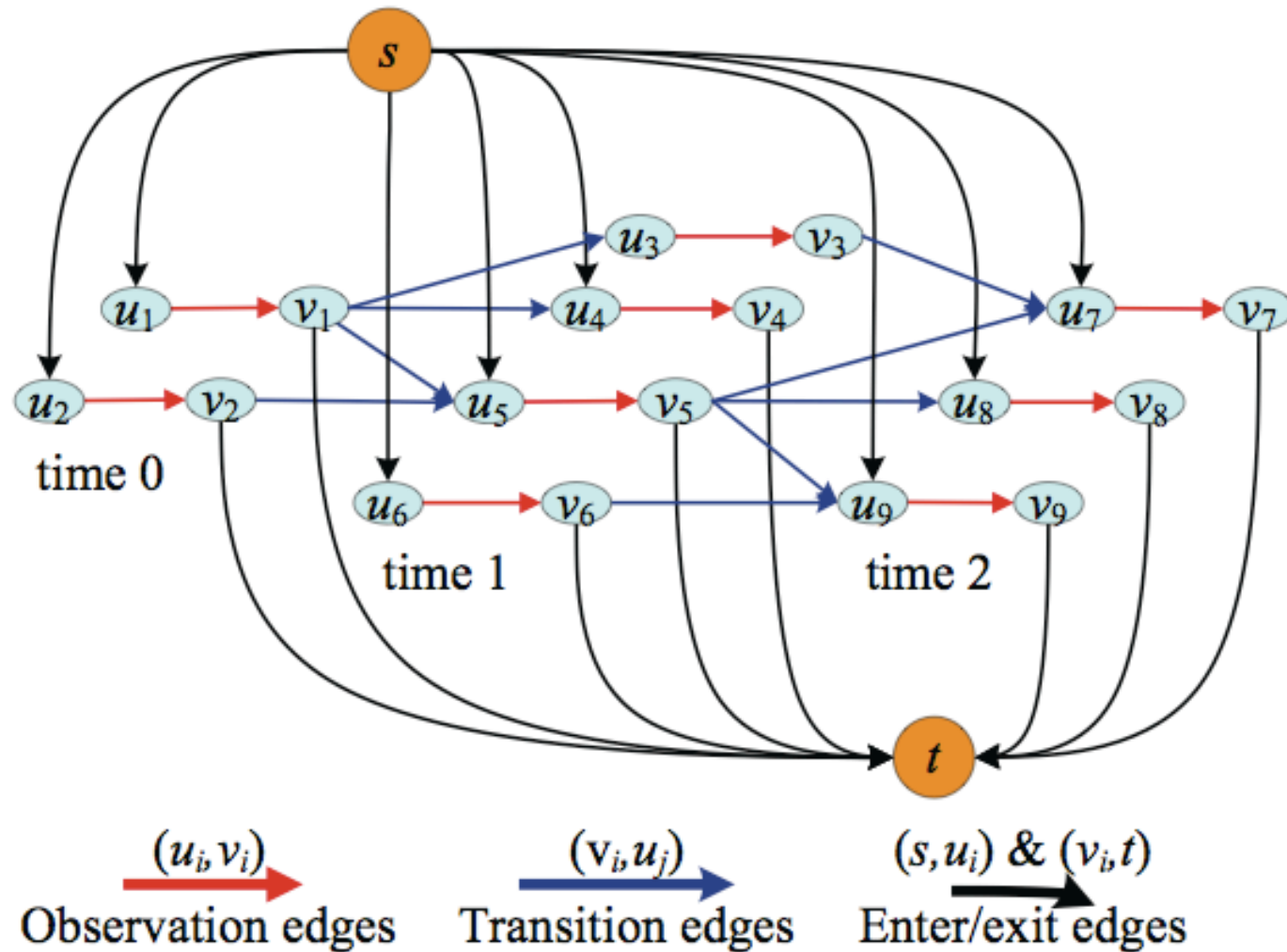
# Network Flow Approach

**Motivation:** Seeking a globally optimal solution by considering observations over all frames in “batch” mode.

**Approach:** Extend two-frame min-cost formulation by adding observations from all frames into the network



# Network Flow Approach



picture from Zhang, Li and Nevatia, “**Global Data Association for Multi-Object Tracking Using Network Flows**,” CVPR 2008.



# Network Flow Solutions

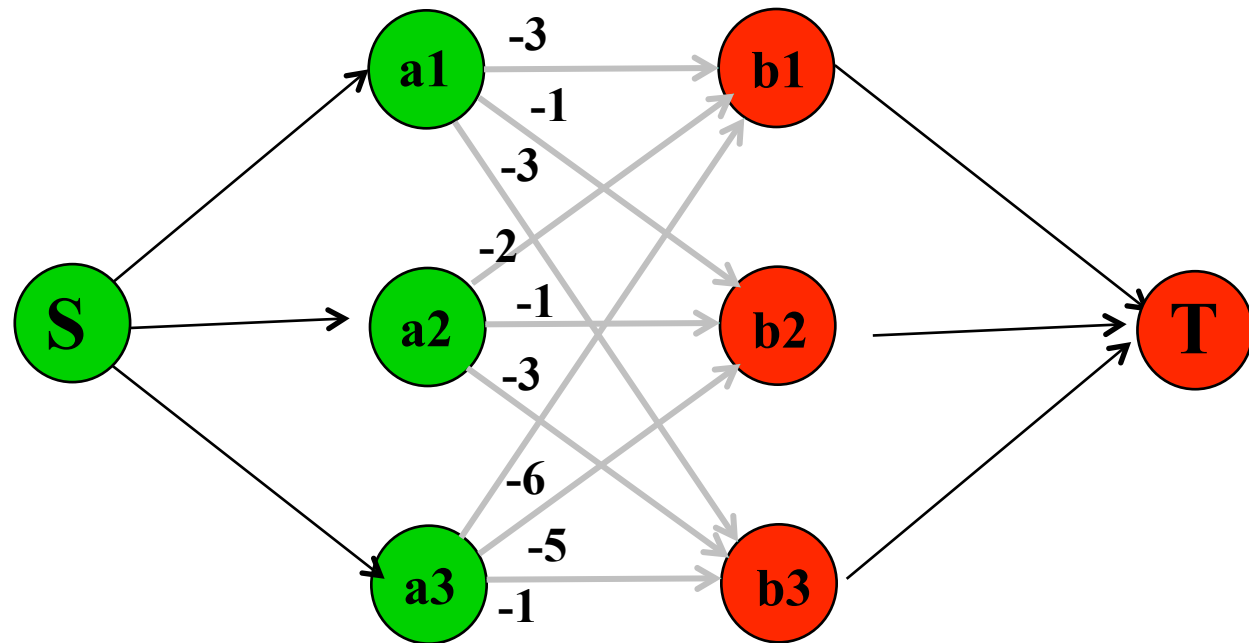
- push-relabel method
  - Zhang, Li and Nevatia, “**Global Data Association for Multi-Object Tracking Using Network Flows,**” CVPR 2008.
- successive shortest path algorithm
  - Berclaz, Fleuret, Turetken and Fua, “**Multiple Object Tracking using K-shortest Paths Optimization,**” IEEE PAMI, Sep 2011.
  - Pirsiavash, Ramanan, Fowlkes, “**Globally Optimal Greedy Algorithms for Tracking a Variable Number of Objects,**” CVPR 2011.
  - these both include approximate dynamic programming solutions

# Successive Shortest Path Example

Small (3x3) example

	b1	b2	b3
a1	3	1	3
a2	2	1	3
a3	6	5	1

min-cost flow network

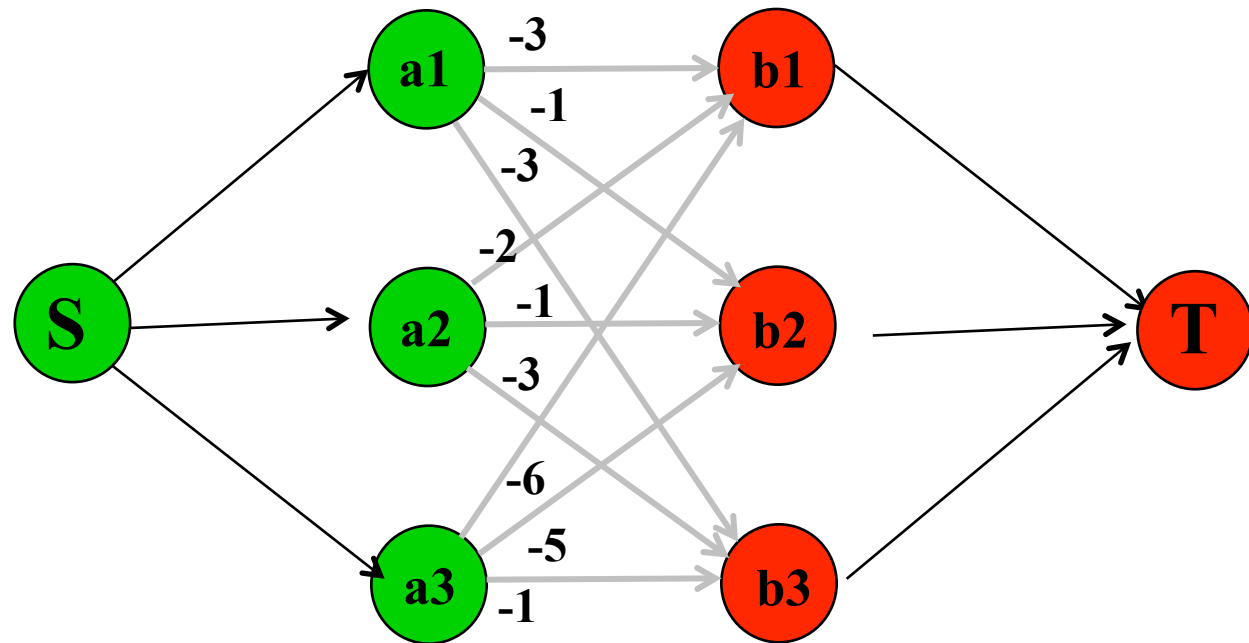


# Successive Shortest Path Example

Small (3x3) example

	b1	b2	b3
a1	3	1	3
a2	2	1	3
a3	6	5	1

min-cost flow network



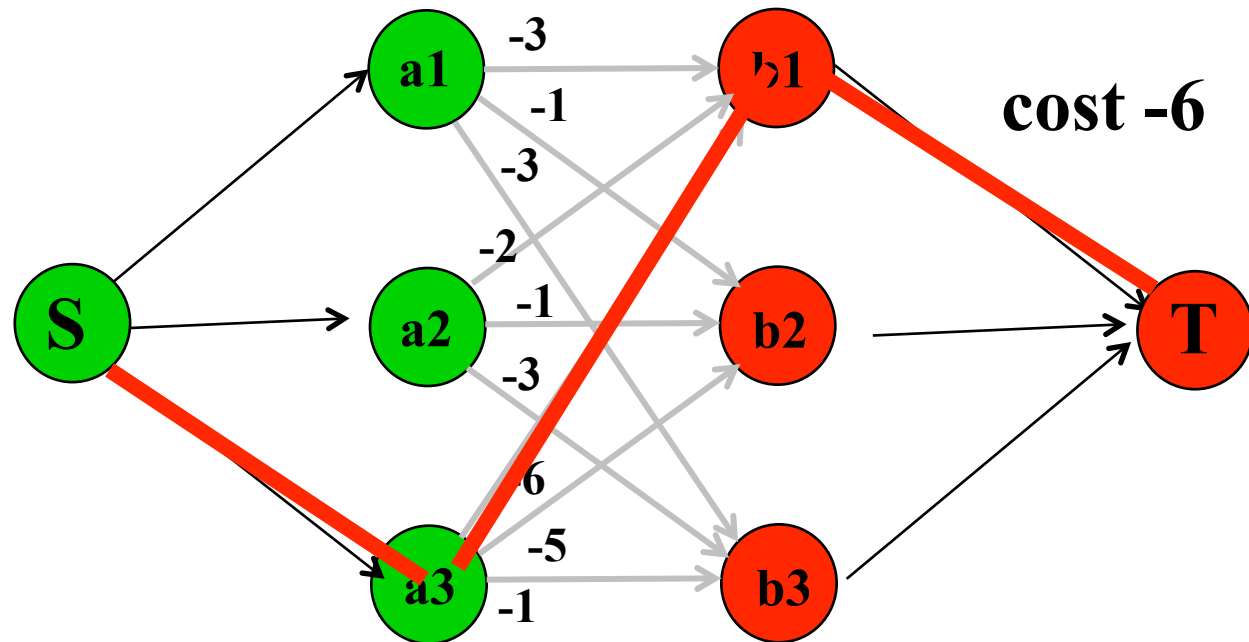
Find the minimum cost path.

# Successive Shortest Path Example

Small (3x3) example

	b1	b2	b3
a1	3	1	3
a2	2	1	3
a3	6	5	1

min-cost flow network



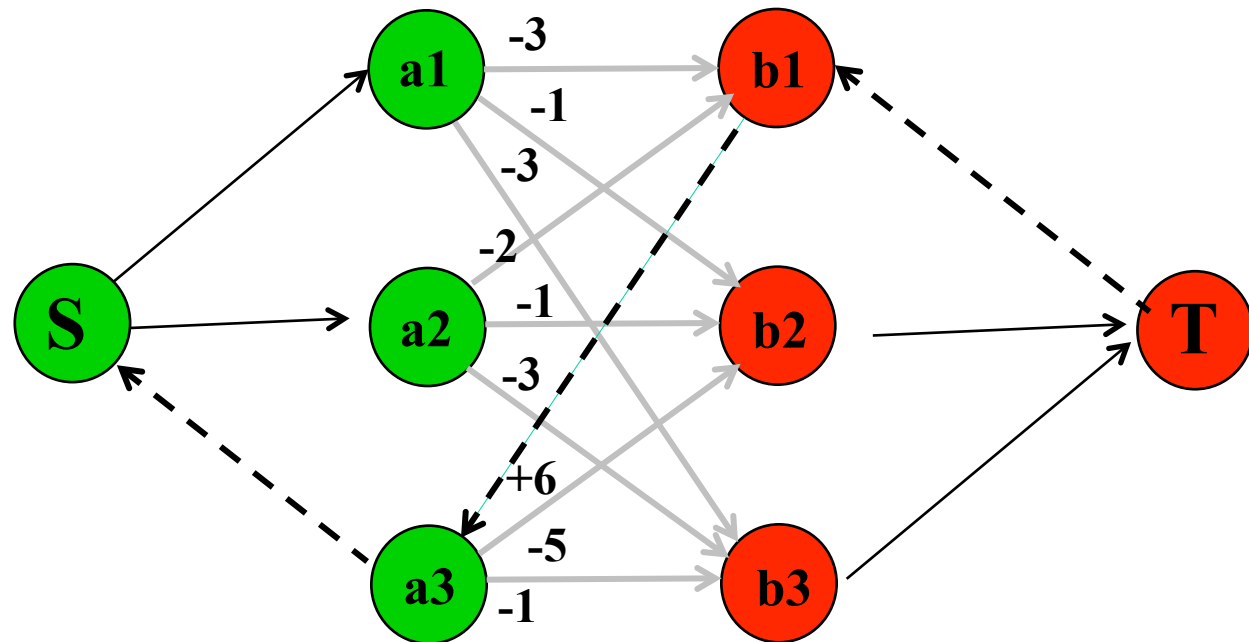
Find the minimum cost path.

# Successive Shortest Path Example

Small (3x3) example

	b1	b2	b3
a1	3	1	3
a2	2	1	3
a3	6	5	1

min-cost flow network



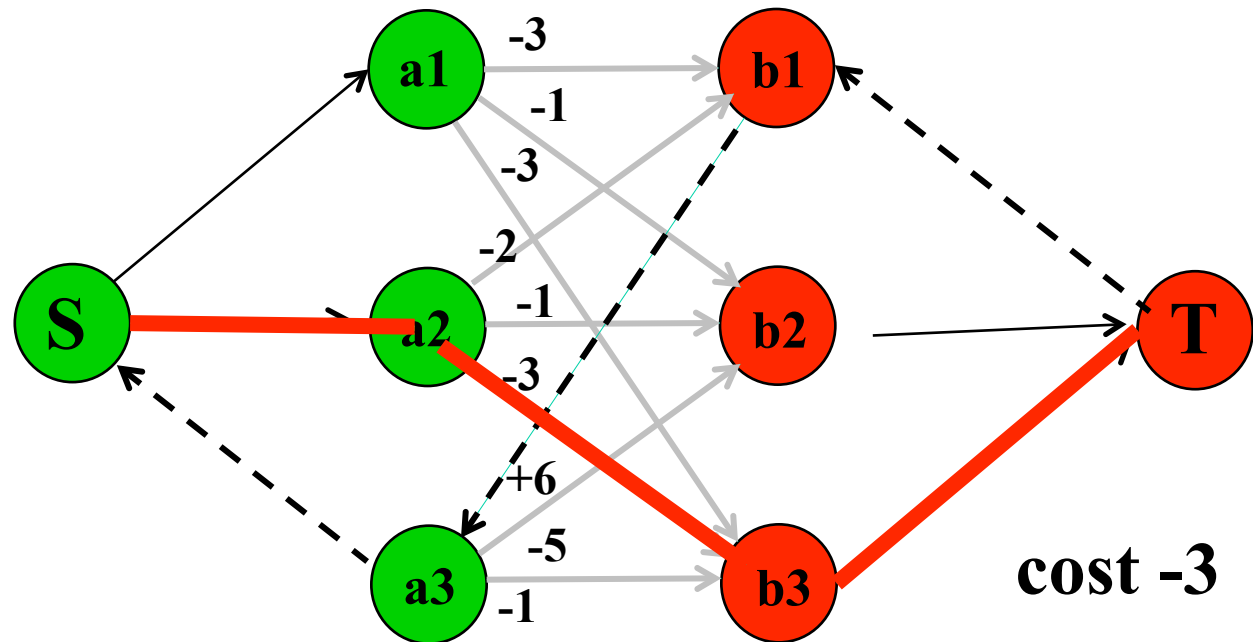
Reverse each edge along that path, and negate the score on each reverse edge.

# Successive Shortest Path Example

Small (3x3) example

	b1	b2	b3
a1	3	1	3
a2	2	1	3
a3	6	5	1

min-cost flow network



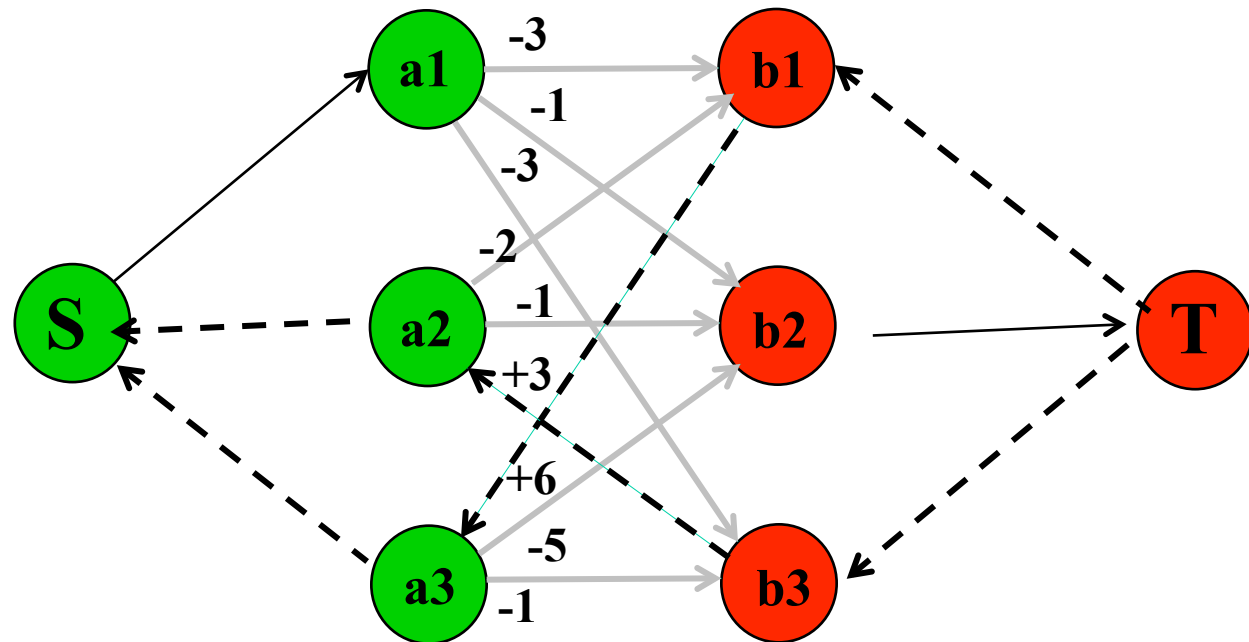
Find a remaining minimum cost path.

# Successive Shortest Path Example

Small (3x3) example

	b1	b2	b3
a1	3	1	3
a2	2	1	3
a3	6	5	1

min-cost flow network



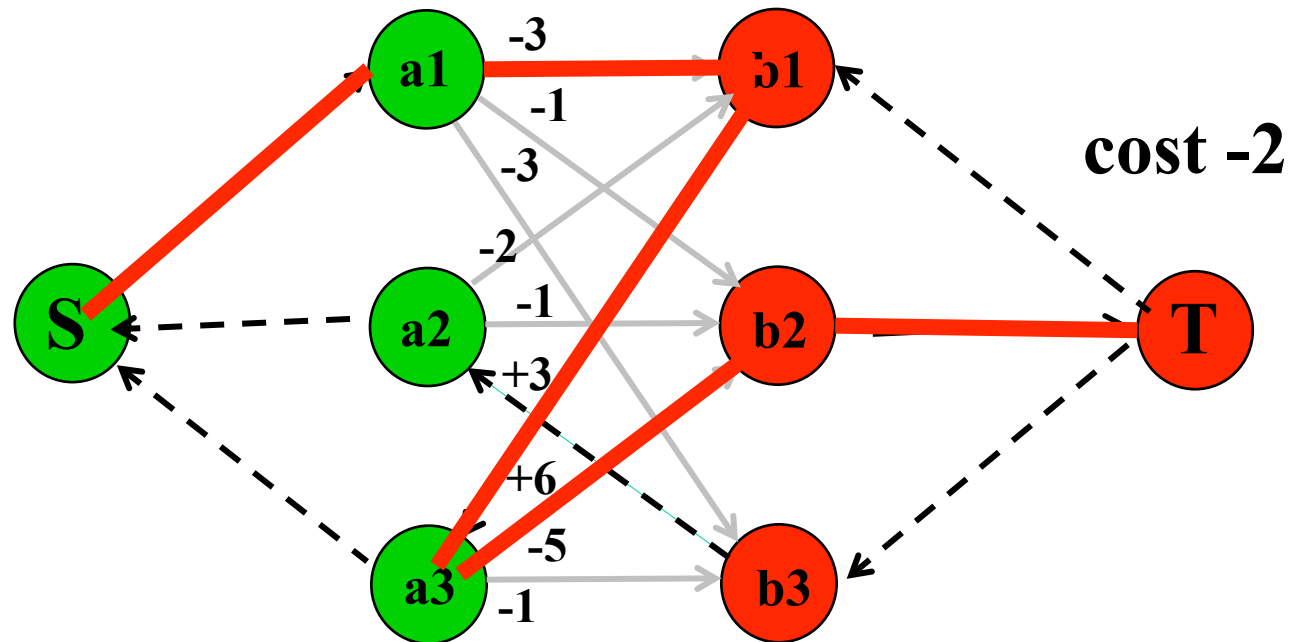
Reverse each edge along that path, and negate the score on each reverse edge.

# Successive Shortest Path Example

Small (3x3) example

	b1	b2	b3
a1	3	1	3
a2	2	1	3
a3	6	5	1

min-cost flow network



Find a remaining minimum cost path.

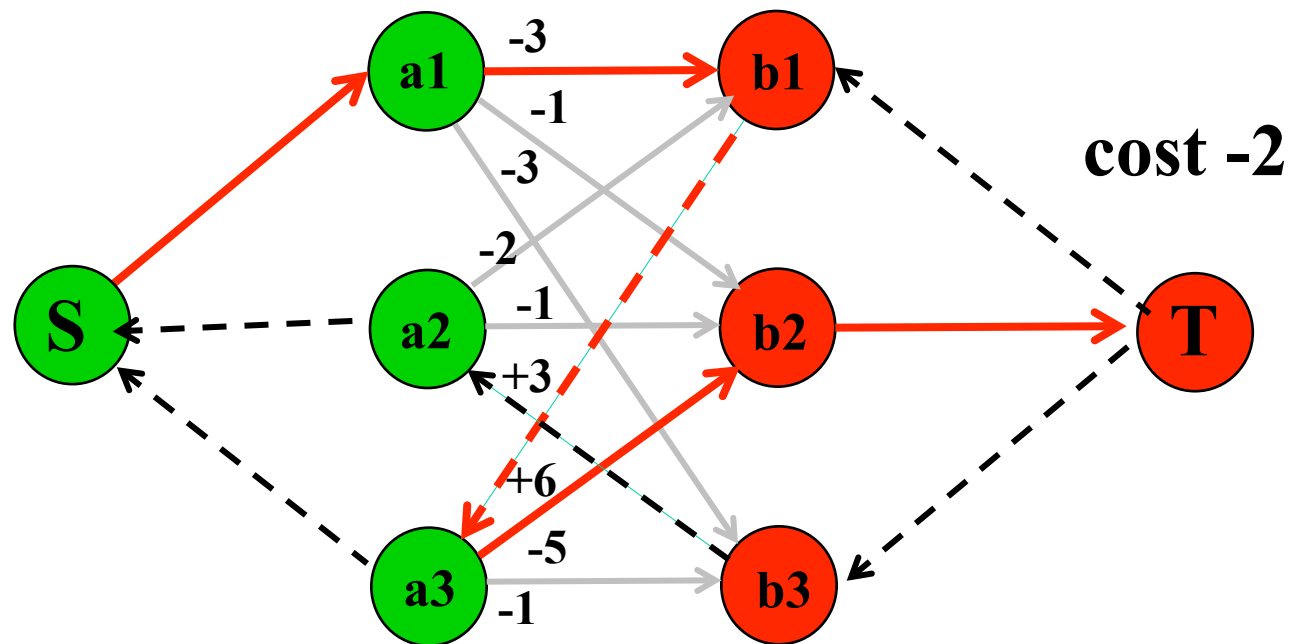


# Successive Shortest Path Example

Small (3x3) example

	b1	b2	b3
a1	3	1	3
a2	2	1	3
a3	6	5	1

min-cost flow network



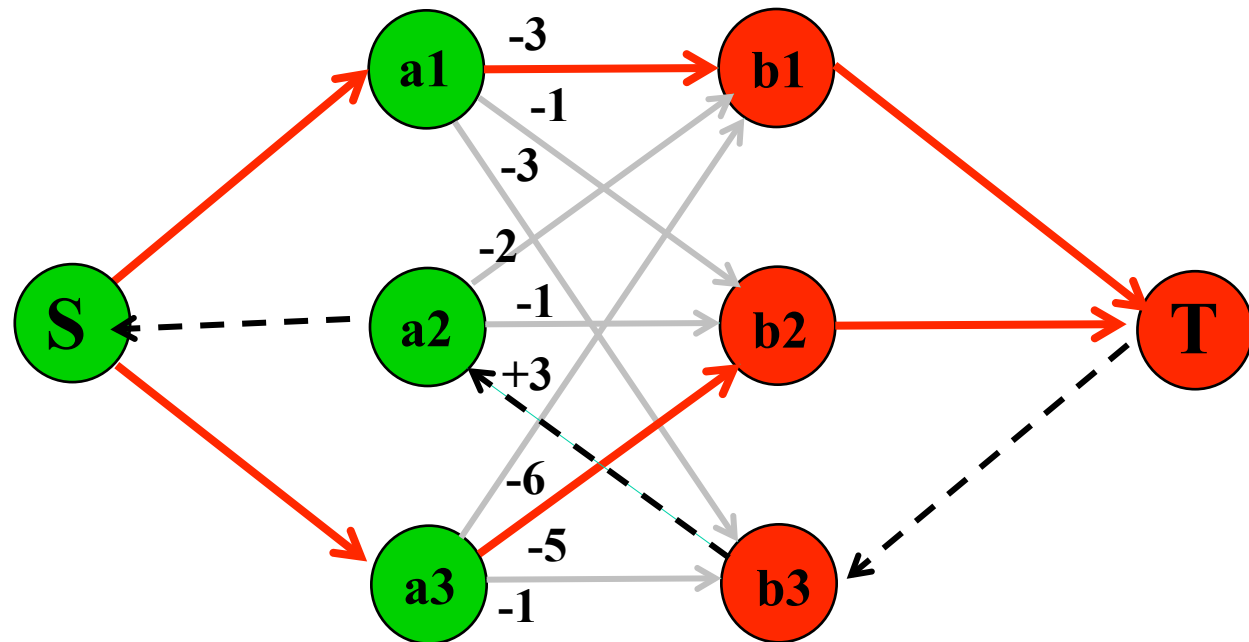
**Notice that this path includes a reversed edge. When this happens, edit rules are applied to splice and correct the previous and new paths.**

# Successive Shortest Path Example

Small (3x3) example

	b1	b2	b3
a1	3	1	3
a2	2	1	3
a3	6	5	1

min-cost flow network



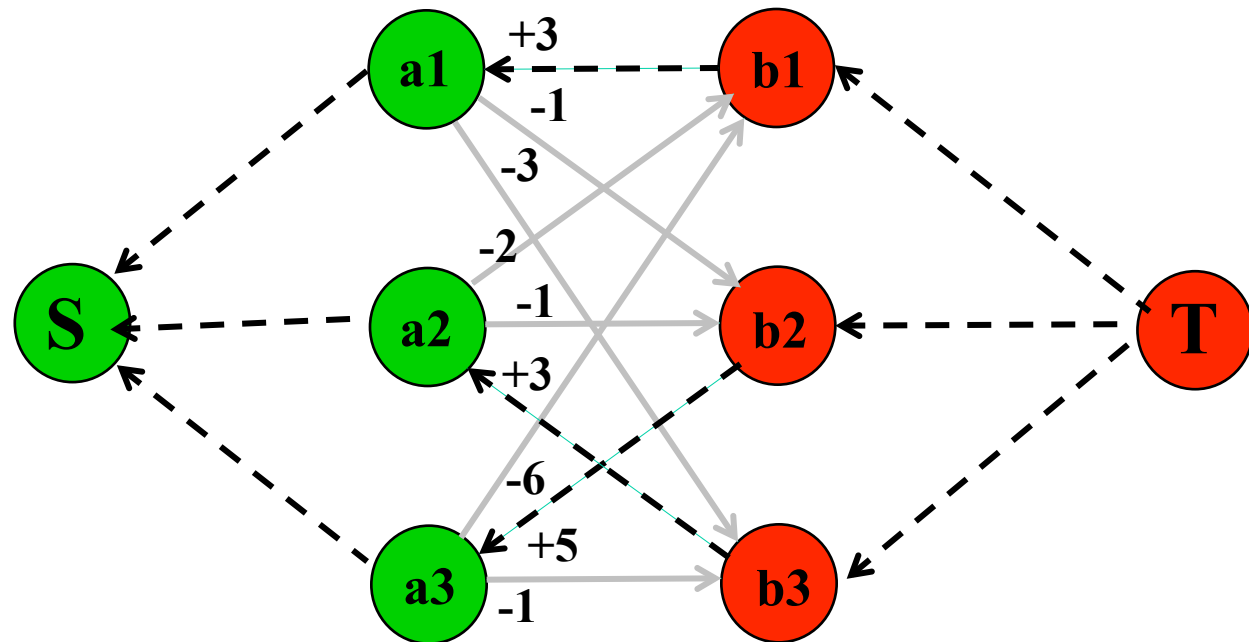
**Notice that this path includes a reversed edge.  
When this happens, edit rules are applied to  
splice and correct the previous and new paths.**

# Successive Shortest Path Example

Small (3x3) example

	b1	b2	b3
a1	3	1	3
a2	2	1	3
a3	6	5	1

min-cost flow network



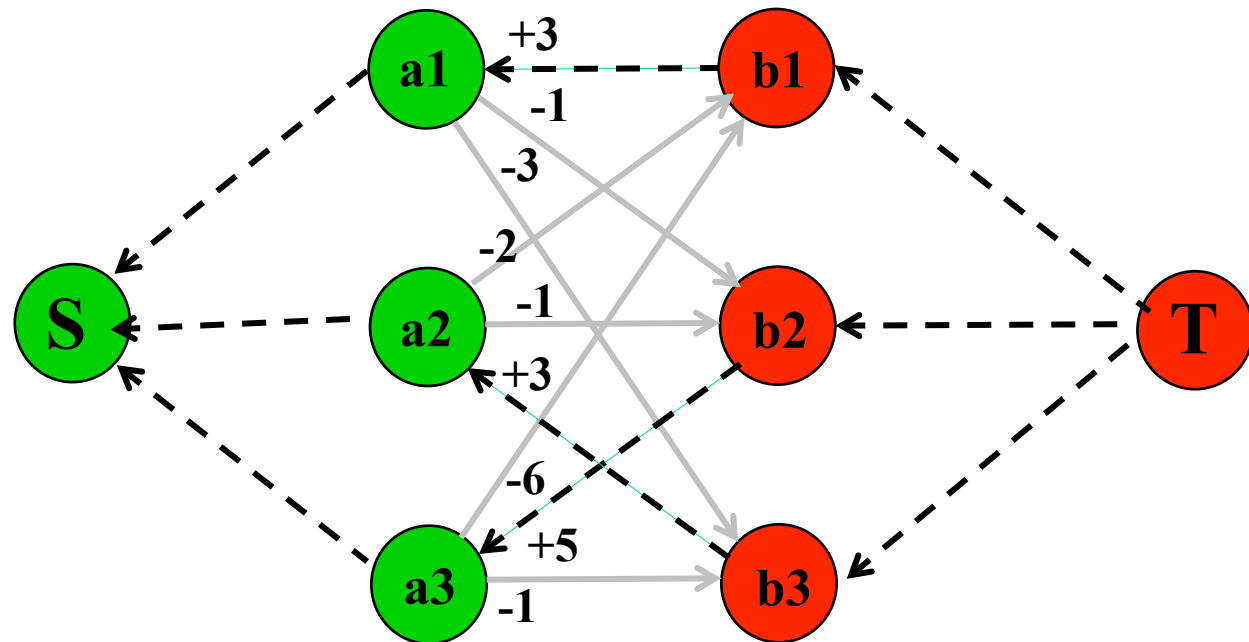
Reverse each edge along that path, and negate the score on each reverse edge.

# Successive Shortest Path Example

Small (3x3) example

	b1	b2	b3
a1	3	1	3
a2	2	1	3
a3	6	5	1

min-cost flow network



Now we are done, since there are no more directed paths from S to T.

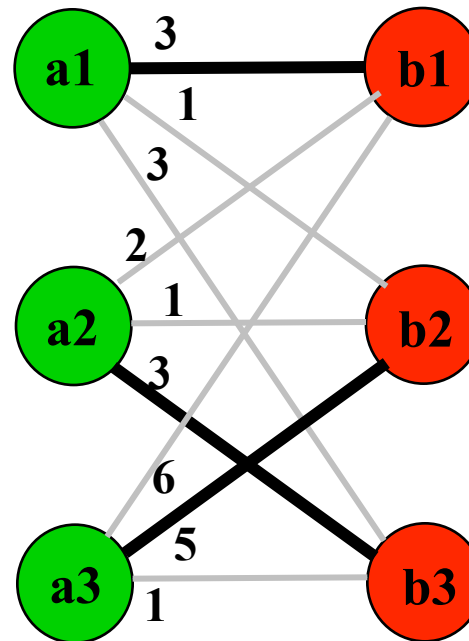
# Successive Shortest Path Example

Small (3x3) example

	b1	b2	b3
a1	3	1	3
a2	2	1	3
a3	6	5	1

corresponding  
solution

min-cost flow network



maximum weight matching (thick edges).  
Total weight = 11

# Network Flow

## Pros:

- Efficient (polynomial time)

- Uses all frames to achieve a global batch solution

## Cons:

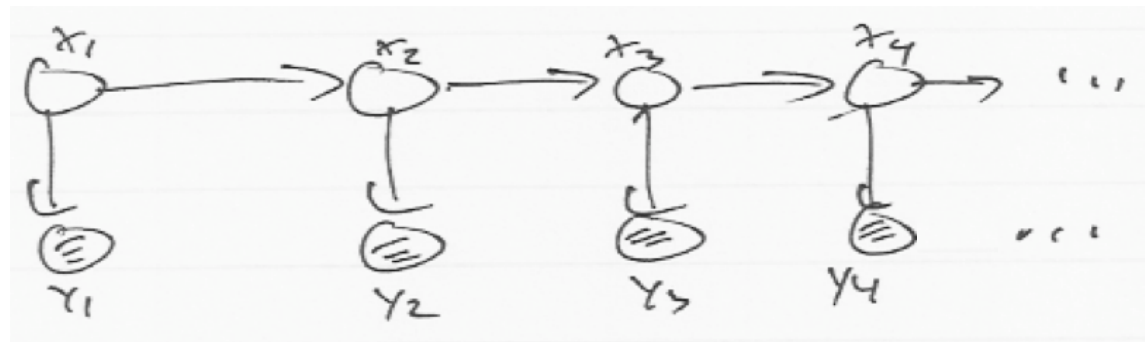
- Cost function is limited to pairwise terms

- Cannot represent constant velocity or other higher-order motion models

- Will therefore have trouble when appearance information is not discriminative and/or frame rate is low

# Confusion Alert

- Wait... we know the graphical model for Kalman filter has only pairwise links, yet KF is able to represent constant velocity motion models. Why do you say a network flow graph cannot?



- KF is a recursive estimator that can propagate past location information forward in time. Network flow graphs are static structures; costs on edges are fixed in advance, but constant velocity is a function of 3 nodes (if only x,y data observed).

# Multi-Dimensional Assignment

Recall the elements of 2D assignment problem:

- observations are nodes in bipartite graph
- edges contain one node from each partite set
- each edge has an associated cost / weight
- each edge has a binary decision variable (on/off)
- each node can only be part of one “on” edge  
(i.e. edges in the solution set are disjoint)

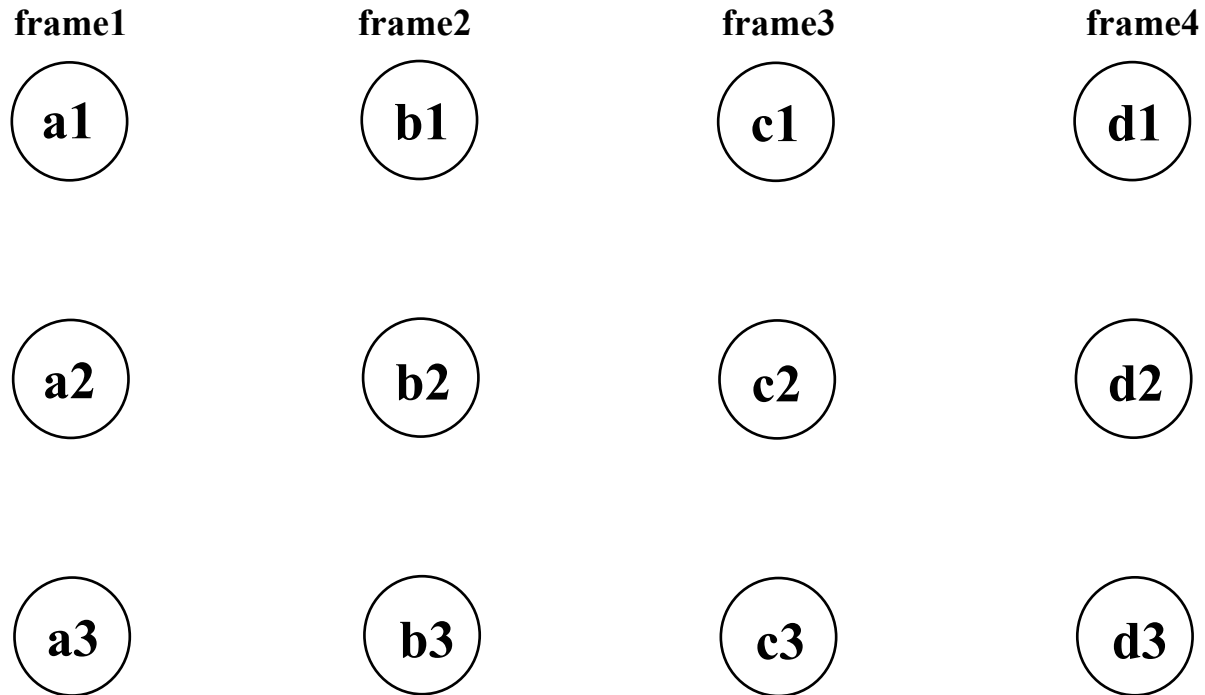


# Multi-Dimensional Assignment

Generalization to k-frame assignment problem:

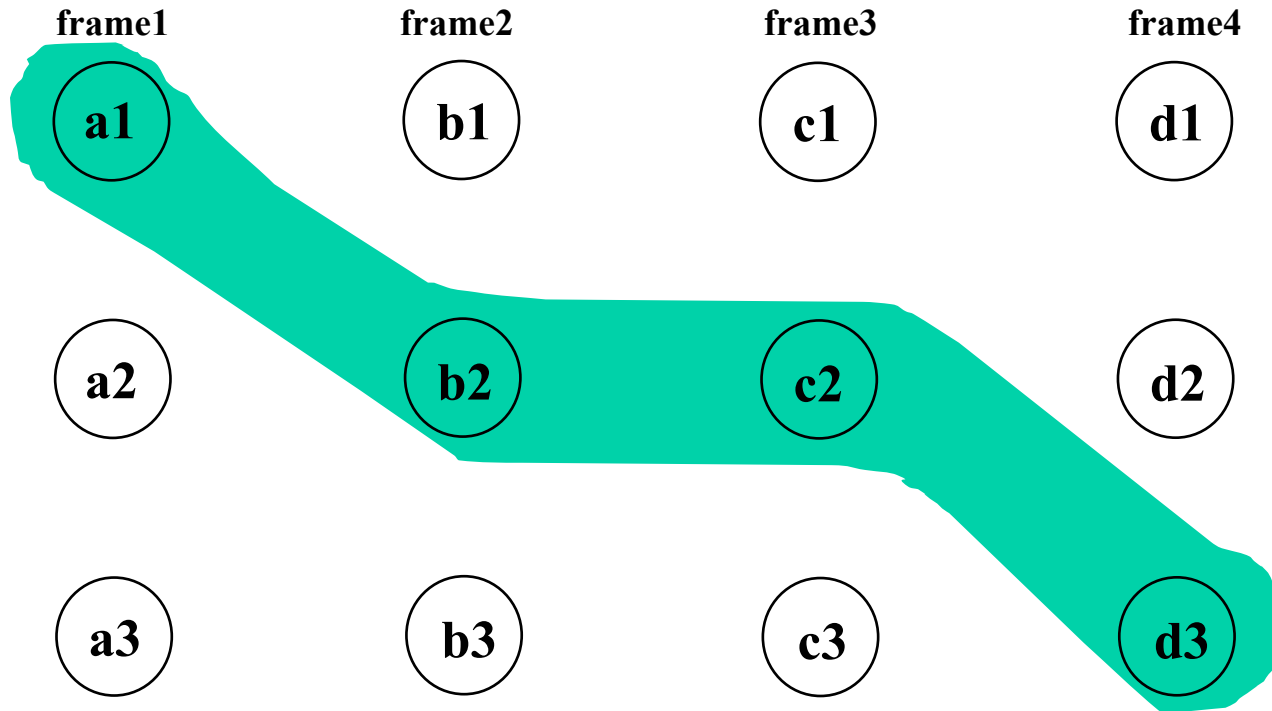
- observations are nodes in k-partite graph
- hyperedges contain one node from each partite set
- each hyperedge has an associated cost / weight
- each hyperedge has a binary decision variable
- hyperedges in solution set must be disjoint

# Four-Frame Example



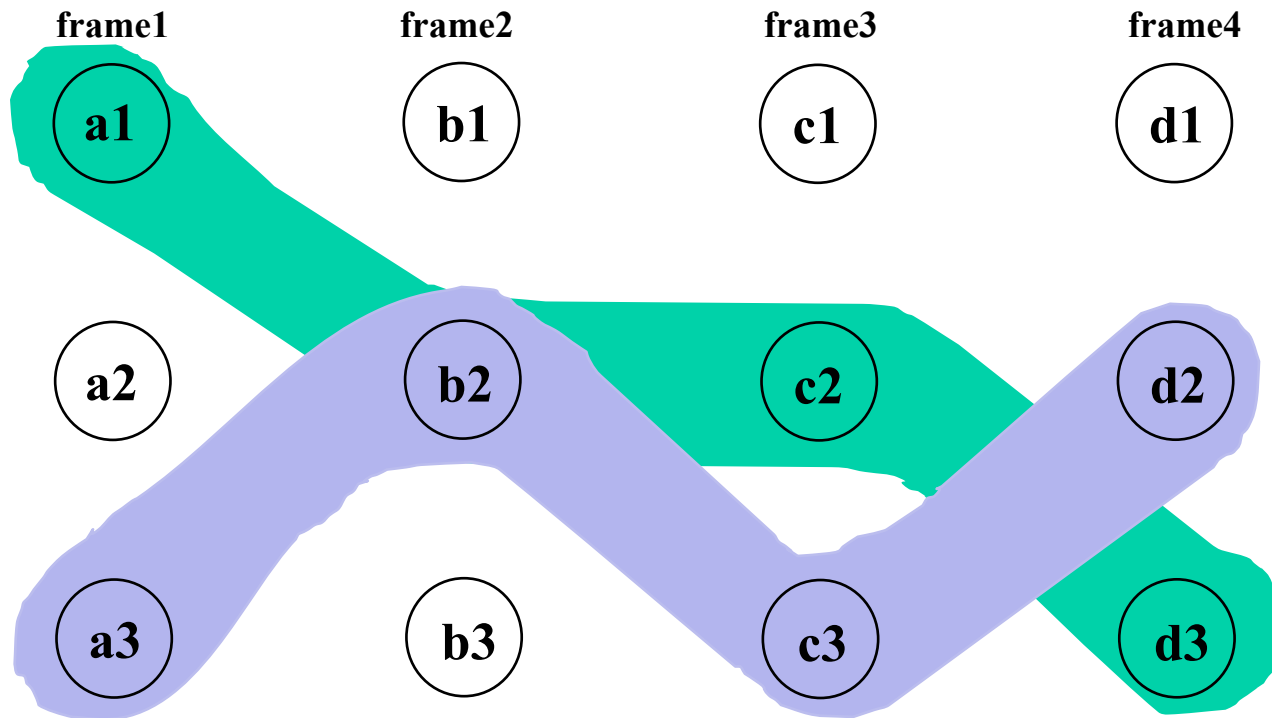
observations arranged in four sets (4-partite graph)

# Four-Frame Example



hyperedge 1-2-2-3 { decision variable  $x_{1223}$   
associated cost  $c_{1223}$

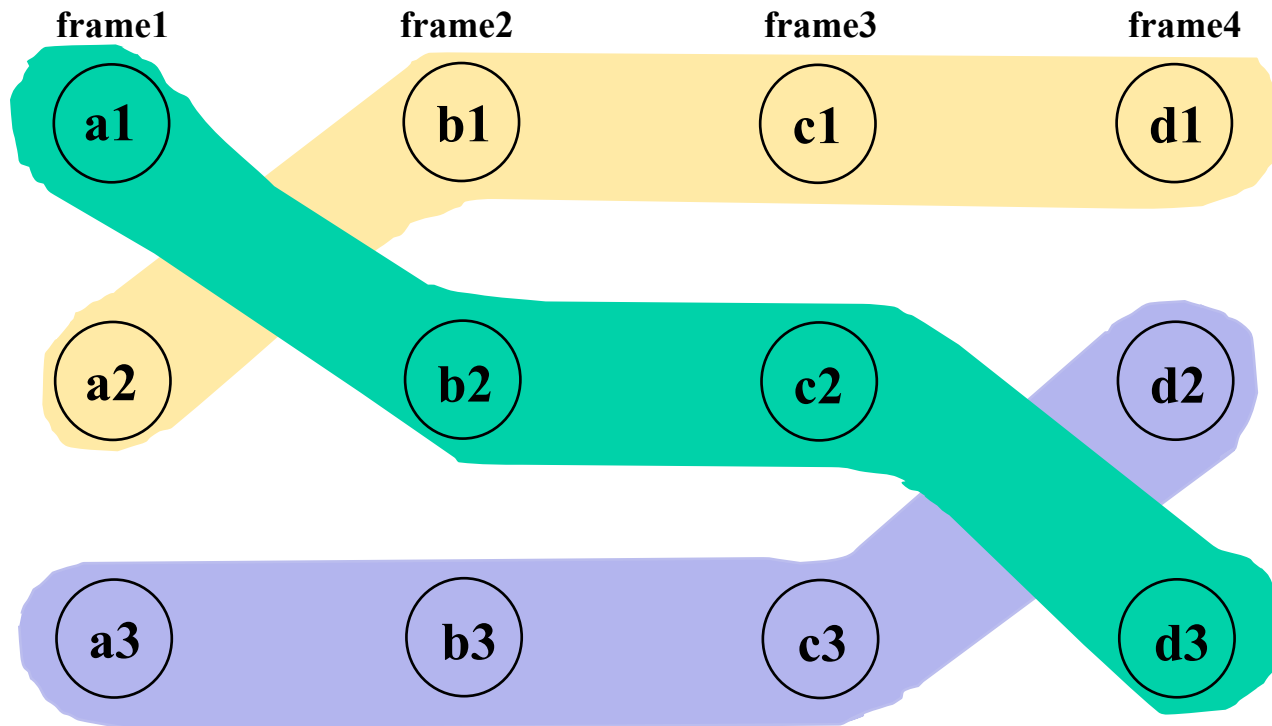
# Four-Frame Example



hyperedges 1-2-2-3 and 3-2-3-2 are incompatible

- they share observation 2 in frame 2
- $x_{1223} + x_{3232} > 1$  is disallowed

# Four-Frame Example



1-2-2-3 , 3-3-3-2 , 2-1-1-1 is a feasible solution

- all observations used once and only once [partitioning]
- $x_{1223} = x_{3332} = x_{2111} = 1$  ; all others 0
- solution cost is  $c_{1223} + c_{3332} + c_{2111}$

# Four-Frame Example

- Mathematical formulation (an ILP)

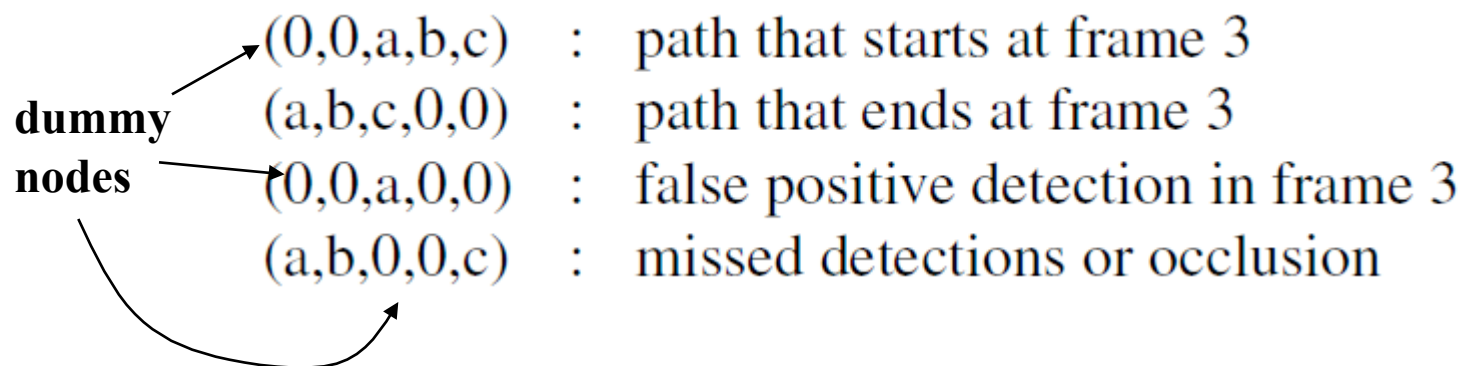
$$\begin{array}{ll} \min & \sum_{a=0}^{n_a} \sum_{b=0}^{n_b} \sum_{c=0}^{n_c} \sum_{d=0}^{n_d} c_{abcd} x_{abcd} \\ \text{s.t} & Ax = 1 \\ & x \in \{0, 1\} \end{array}$$

## Intuition:

We want to find a minimum cost partitioning of observations into disjoint hyperedges.

## Some Implementation Details

I am glossing over some details. For example, each partite set includes a dummy node (index 0) that can be assigned multiple times to allow objects to appear, disappear, and to explain missing observations.



This is why we are able to require a partitioning of the observations (each observation used once and only once)

# General MDA as an ILP

$$\min \sum_{i_1=0}^{n_1} \sum_{i_2=0}^{n_2} \cdots \sum_{i_k=0}^{n_k} c_{i_1 i_2 \dots i_k} x_{i_1 i_2 \dots i_k}$$

subject to

$$\sum_{i_2=0}^{n_2} \sum_{i_3=0}^{n_3} \cdots \sum_{i_k=0}^{n_k} x_{i_1 i_2 \dots i_k} = 1; \quad i_1 = 1, 2, \dots, n_1$$

$$\sum_{i_1=0}^{n_1} \sum_{i_3=0}^{n_3} \cdots \sum_{i_k=0}^{n_k} x_{i_1 i_2 \dots i_k} = 1; \quad i_2 = 1, 2, \dots, n_2$$

⋮

⋮

$$\sum_{i_1=0}^{n_1} \sum_{i_2=0}^{n_2} \cdots \sum_{i_{k-1}=0}^{n_{k-1}} x_{i_1 i_2 \dots i_k} = 1; \quad i_k = 1, 2, \dots, n_k$$



# MDA Solutions

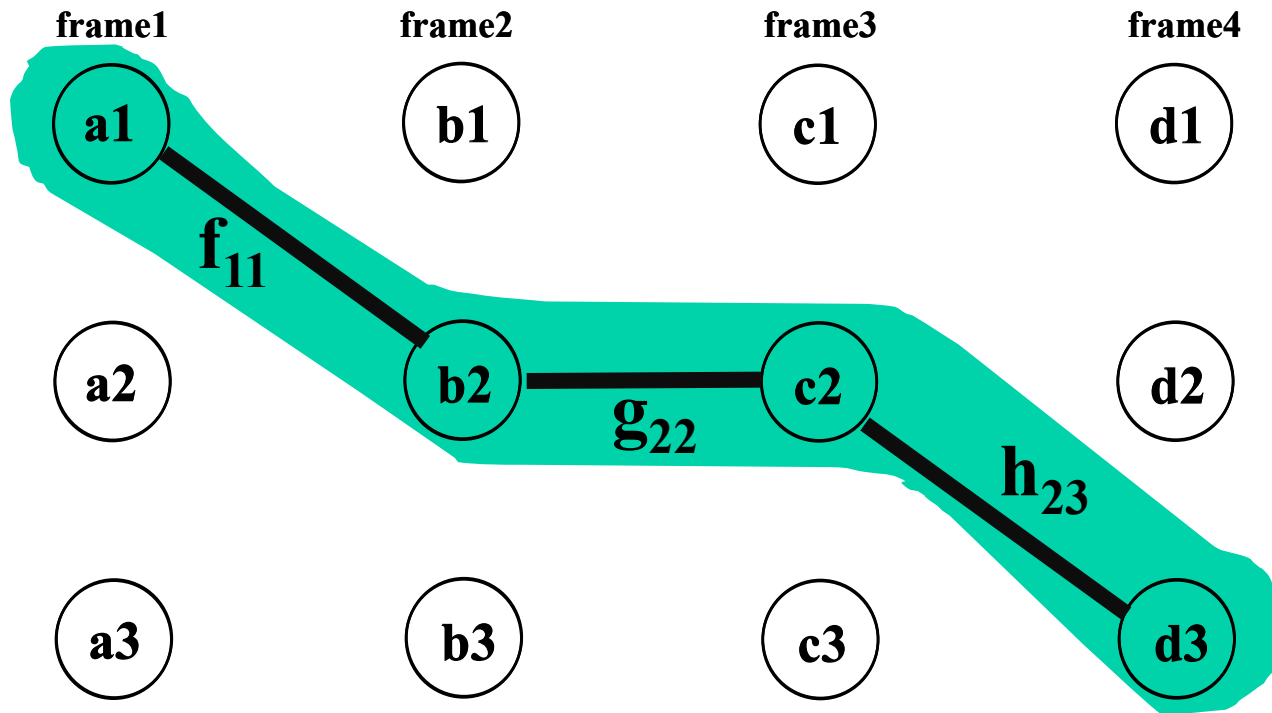
- MDA is NP-hard
- three-frame decision version is one of Karp's 21 NP-complete problems
- Previous approaches
  - branch-and-bound: multi-hypothesis tracking (MHT) is a version of this
  - Lagrangean relaxation (Aubrey Poore)
  - complicated, time-consuming algorithms

intuition: there is an exponential number of decision variables that you have to reason about.  $k$  objs in  $f$  frames  $\rightarrow k^f$  hyperedges

# Our Approach

- We propose an approximation algorithm
- Starting with an initial feasible solution, it does a series of improvement steps
- Always maintains a feasible solution
- Guaranteed to converge to an optimum
- Caveat: it is a local optimum
- In practice it converges quickly to very reasonable solutions

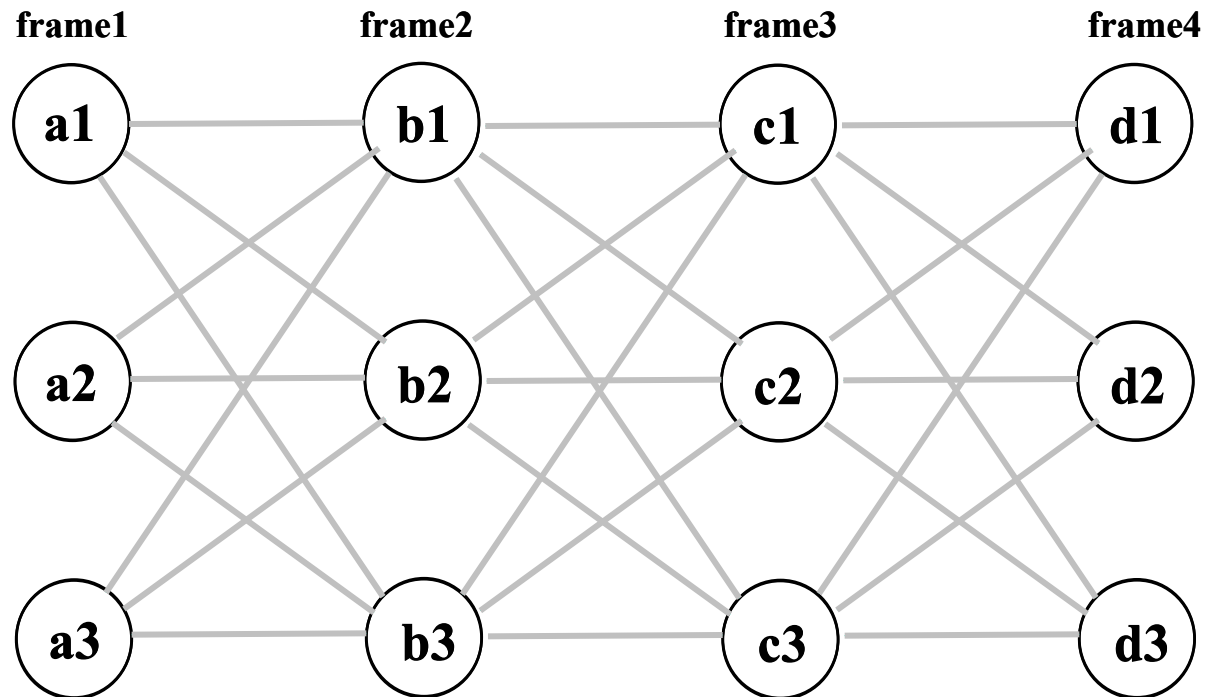
# Factoring Hyperedges



hyperedge 1-2-2-3  $\rightarrow$  (a1-b2) (b2-c2) (c2-d3)

decision variable  $x_{1223} = f_{12} * g_{22} * h_{23}$

# Factoring Hyperedges

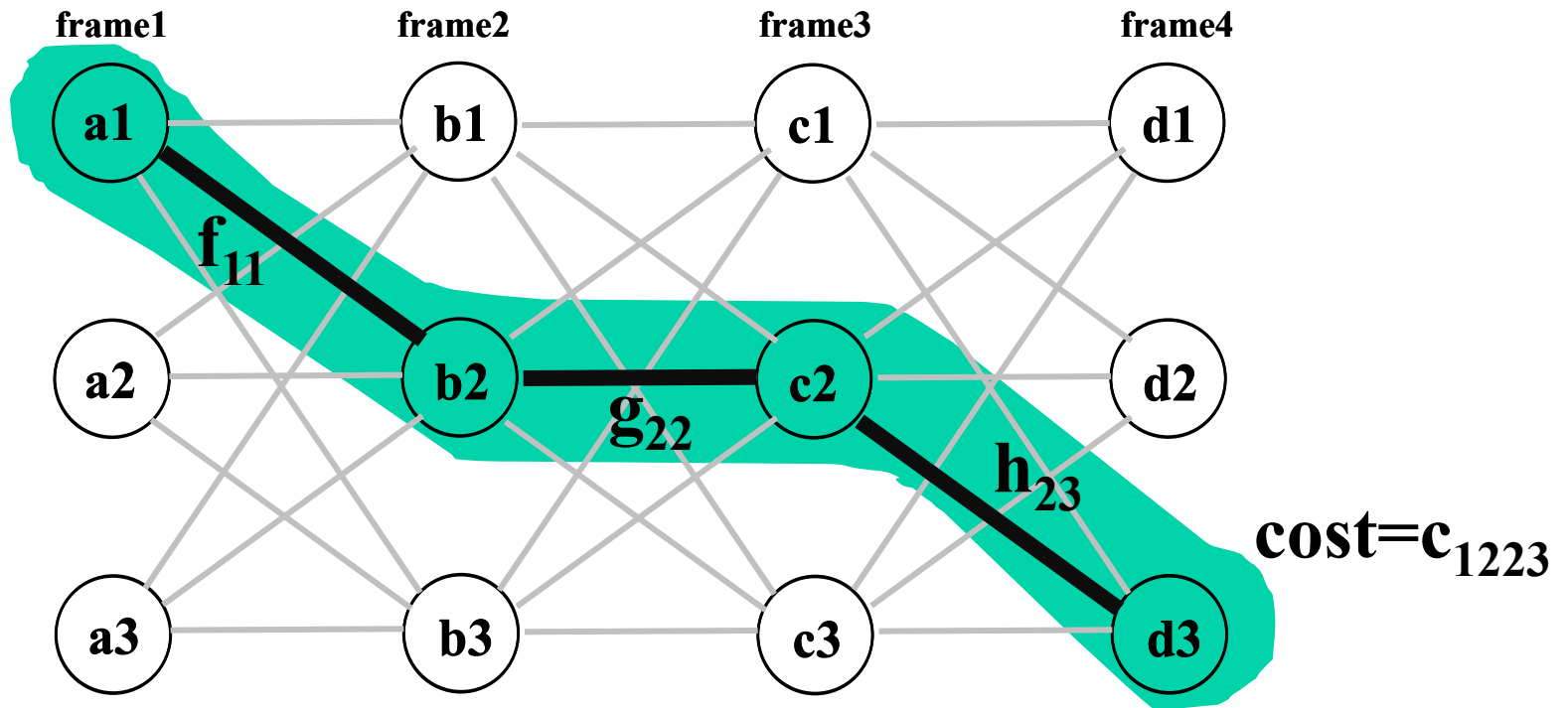


Hyperedges become k-paths; decision variables factor.

Reintroduces graph structure of network flow approach

**Key point: factoring allows us to consider local updates!**

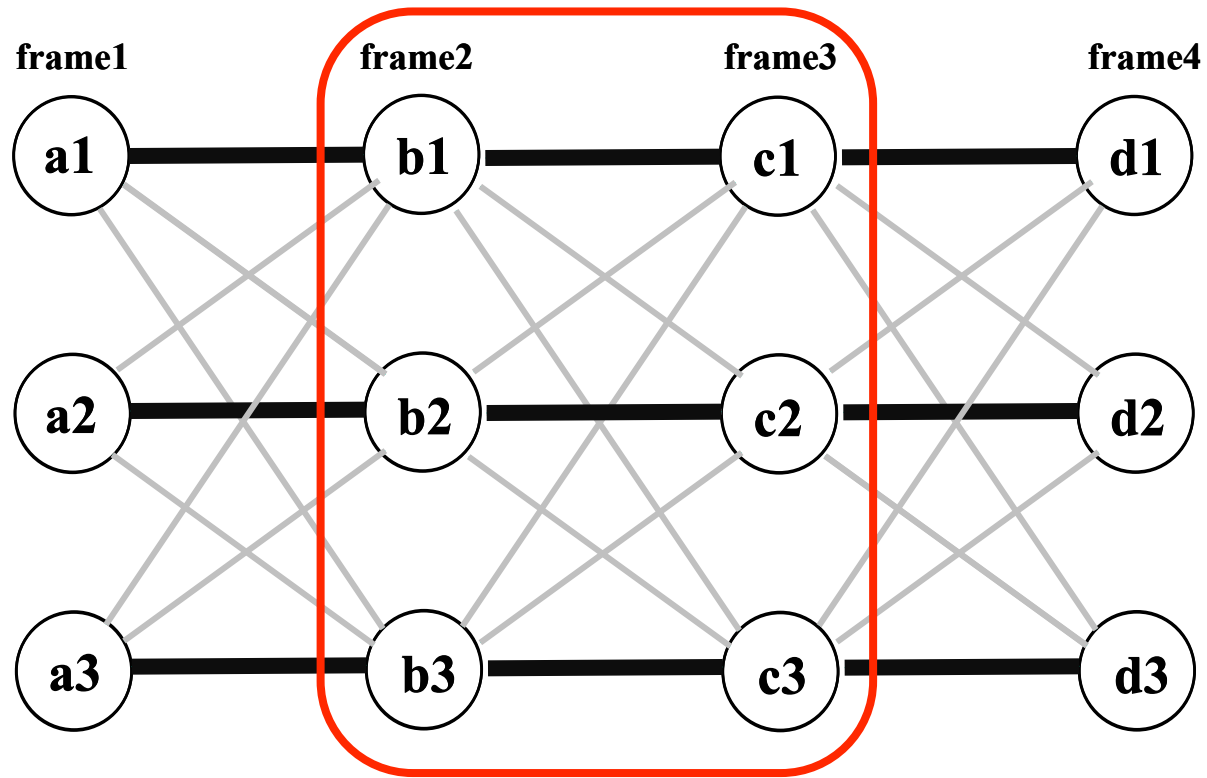
# Factoring Hyperedges



Another important point: costs remain unfactored.

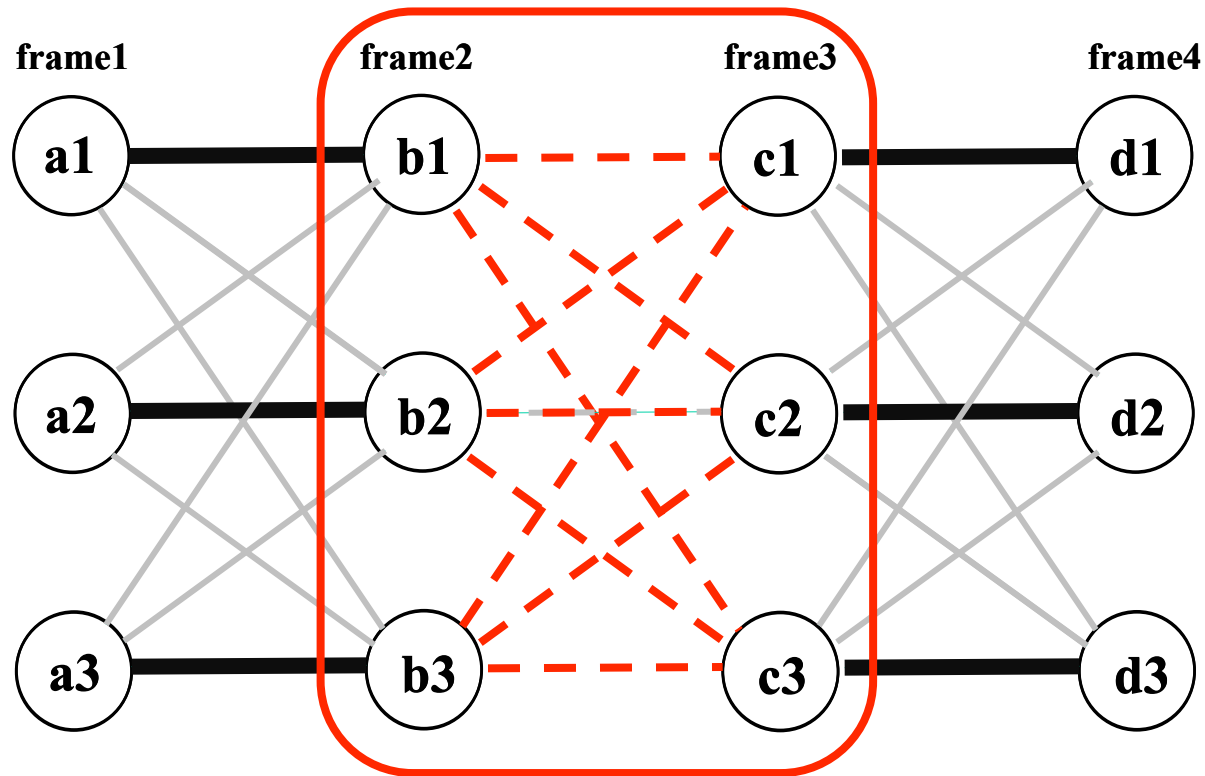
This allows us to use arbitrary cost functions, defined over entire trajectories.

# Local Updates



- Pick a pair of adjacent frames (e.g. 2 and 3)

# Local Updates



- Pick a pair of adjacent frames (e.g. 2 and 3)
- Revise the decision variables (edges) between those frames while holding the rest of the solution fixed

# Local Updates

- Rewriting objective function in terms of edges between frames 2 (indexed by  $b$ ) and 3 (indexed by  $c$ ):

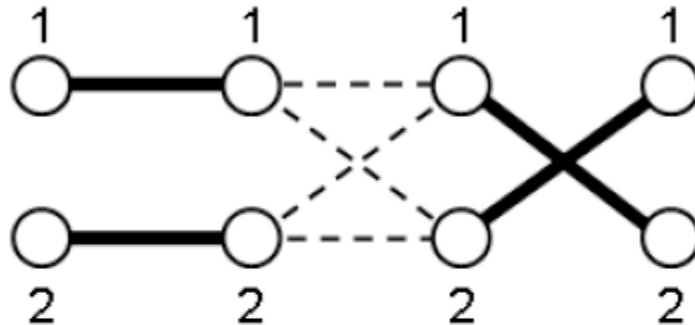
$$\begin{aligned}
 & \sum_a \sum_b \sum_c \sum_d c_{abcd} x_{abcd} \\
 &= \sum_a \sum_b \sum_c \sum_d c_{abcd} \underbrace{f_{ab}}_{\text{fixed}} \underbrace{g_{bc}}_{\text{variables being updated}} \underbrace{h_{cd}}_{\text{fixed}} \\
 &= \sum_b \sum_c g_{bc} \sum_a \sum_d f_{ab} h_{cd} c_{abcd} \\
 &= \sum_b \sum_c g_{bc} C_{(a \rightarrow b)bc(c \rightarrow d)} \\
 &= \sum_b \sum_c g_{bc} \omega(b, c)
 \end{aligned}$$

where  $(a \rightarrow b)$  means current solution path that ends at observation  $b$  in frame 2, and  $(c \rightarrow d)$  means current path that starts at observation  $c$  in frame 3.



# Example

Consider two targets viewed through four frames as in the sketch below:



We are solving for edges between frames 2 and 3 (dashed), holding all other edges (thick) fixed. The reduced cost matrix  $w(b,c)$  is:

$$\omega(b, c) = \begin{array}{c} \mathbf{b=1} \\ \mathbf{b=2} \end{array} \begin{array}{cc} \mathbf{c=1} & \mathbf{c=2} \\ \left[ \begin{array}{cc} C_{1112} & C_{1121} \\ C_{2212} & C_{2221} \end{array} \right] \end{array}$$

# Local Updates

- In the paper, we show that updates to edge decision variables between two adjacent frames:

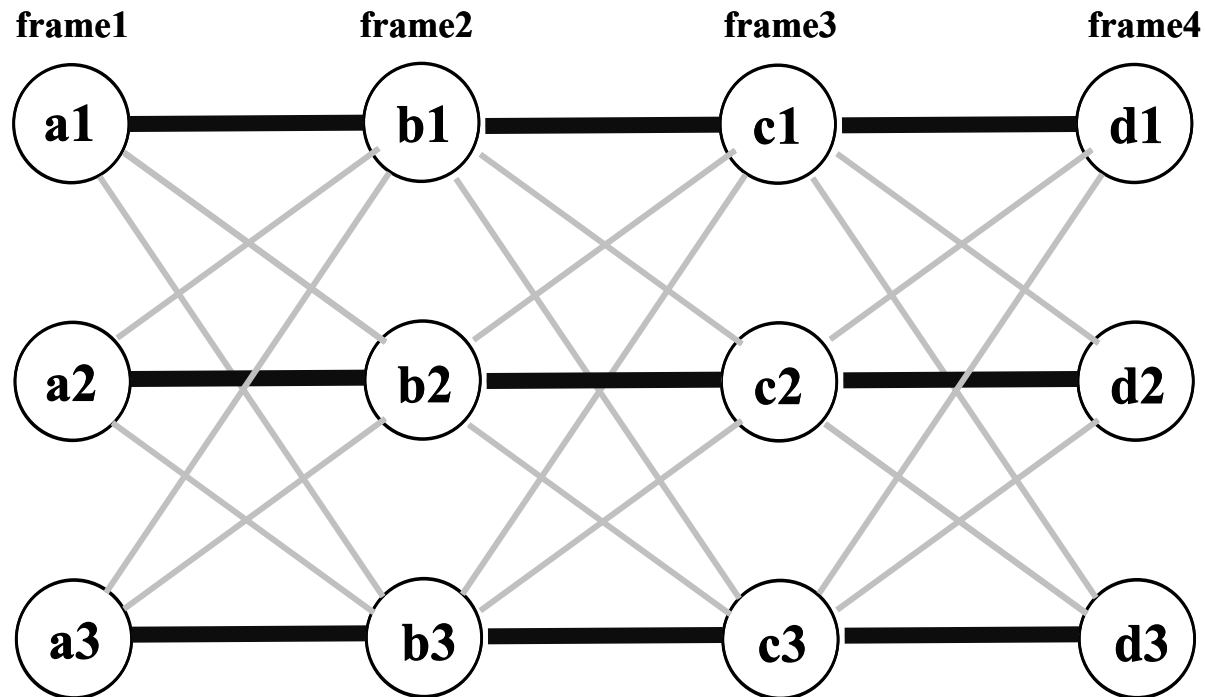
$$\text{minimize } \sum_a \sum_b \sum_c \sum_d c_{abcd} \underbrace{f_{ab}}_{\text{fixed}} \underbrace{g_{bc}}_{\text{variables being updated}} \underbrace{h_{cd}}_{\text{fixed}}$$

reduces to:

$$\text{minimize } \sum_b \sum_c g_{bc} \omega(b, c)$$

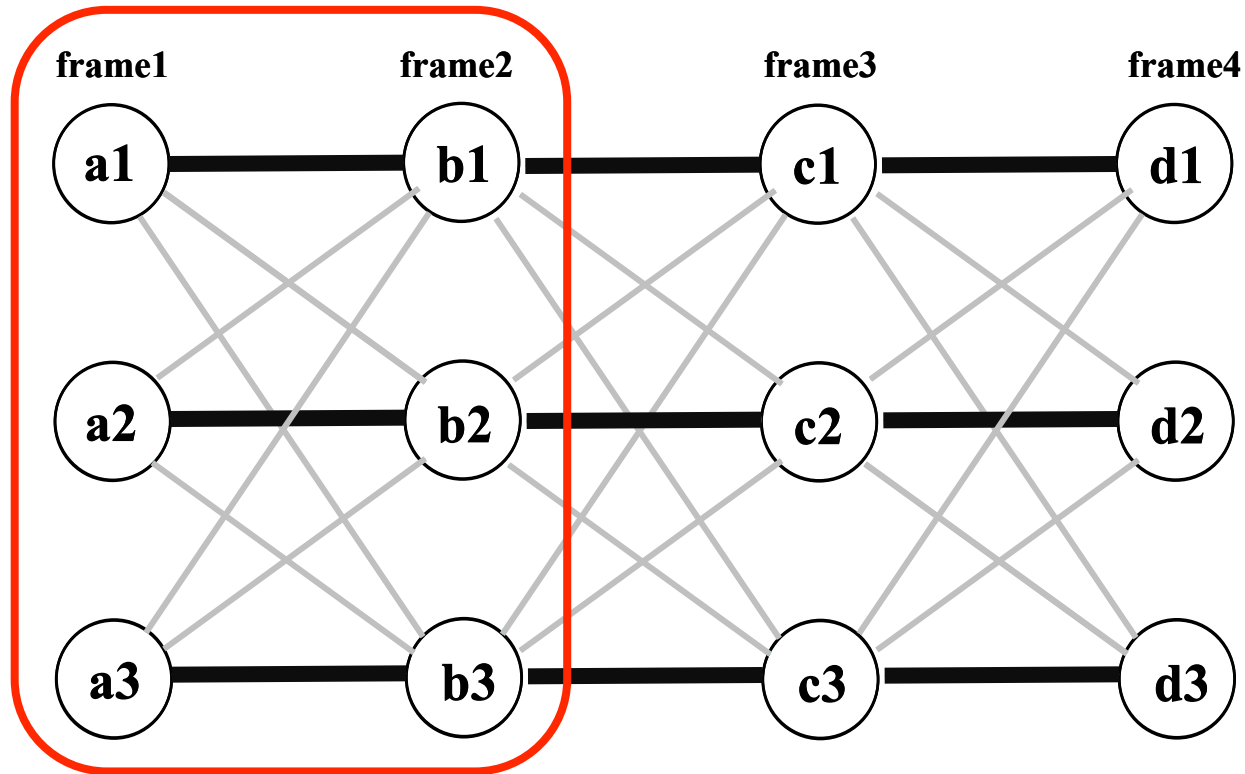
The bottom line is that local updates to edges between two frames reduces to a two-frame linear assignment problem! We solve this using Hungarian algorithm.

# To Summarize Our Approach



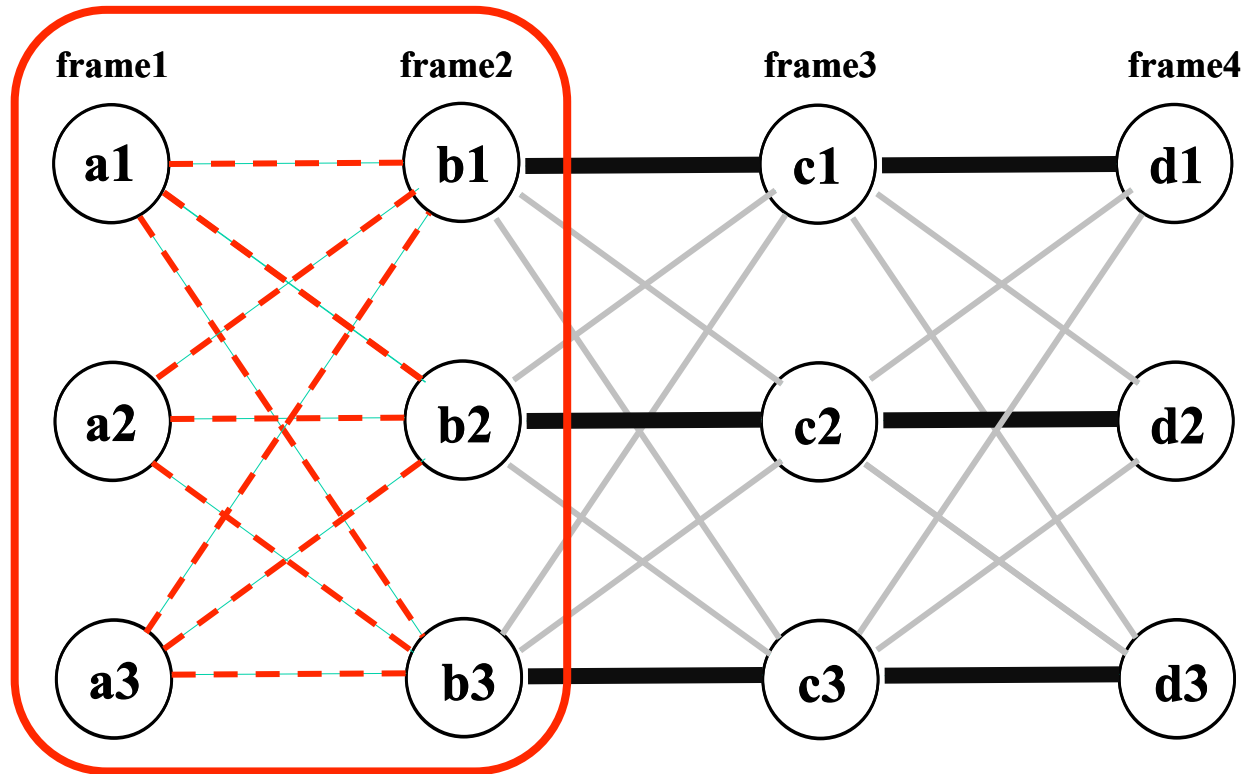
- Start with an initial feasible solution.

# To Summarize Our Approach



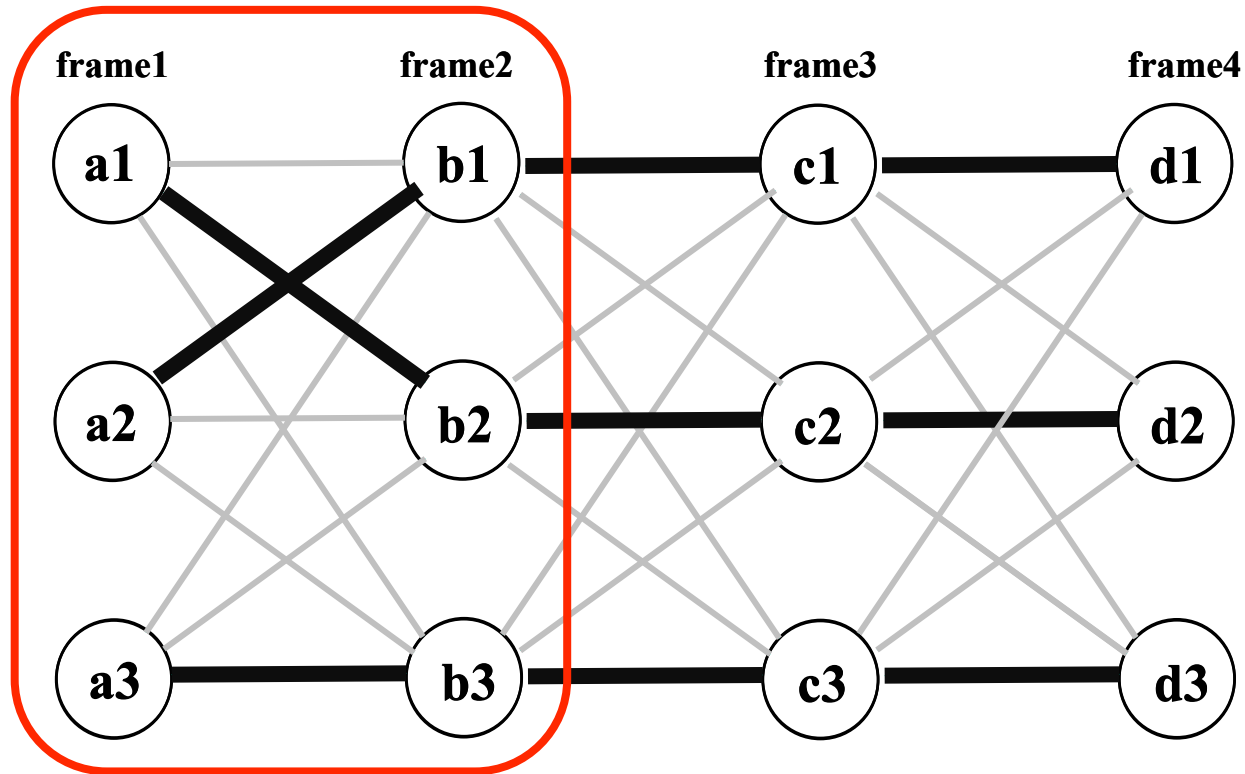
- Begin stepping through adjacent pairs of frames, solving updating edge decisions to improve the objective function value

# To Summarize Our Approach



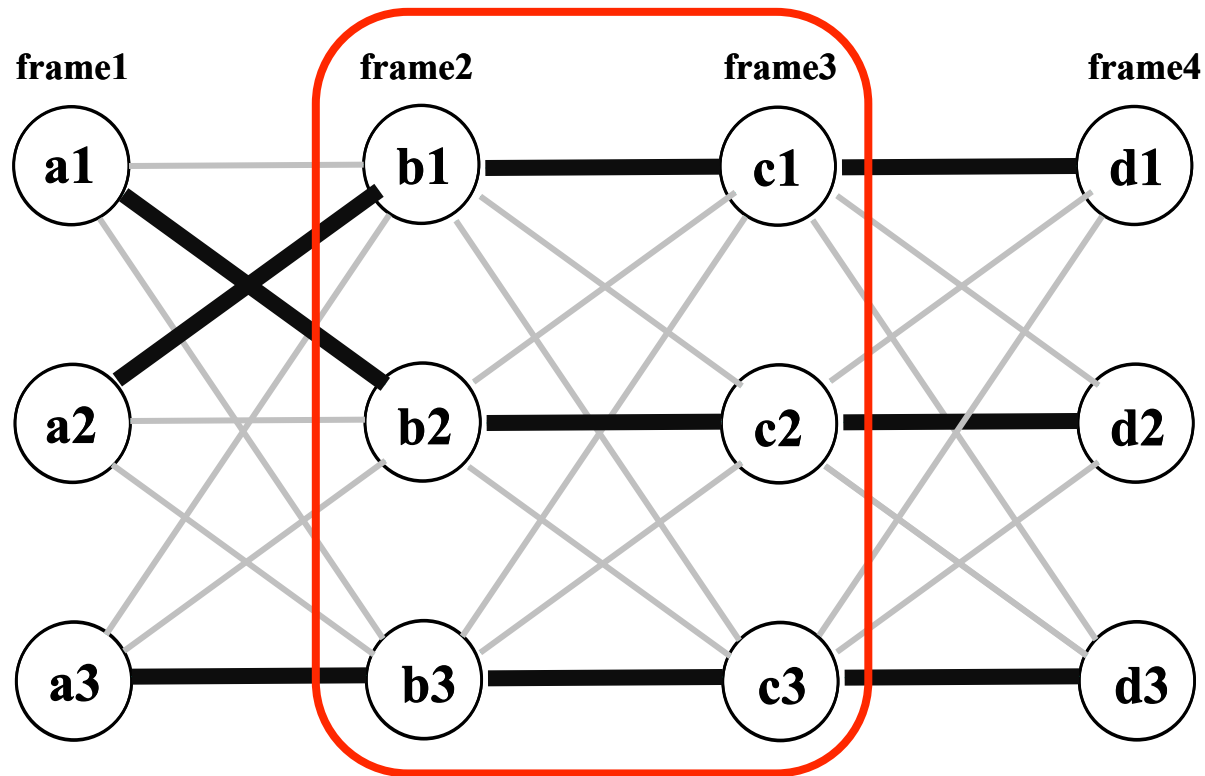
- Begin stepping through adjacent pairs of frames, solving updating edge decisions to improve the objective function value

# To Summarize Our Approach



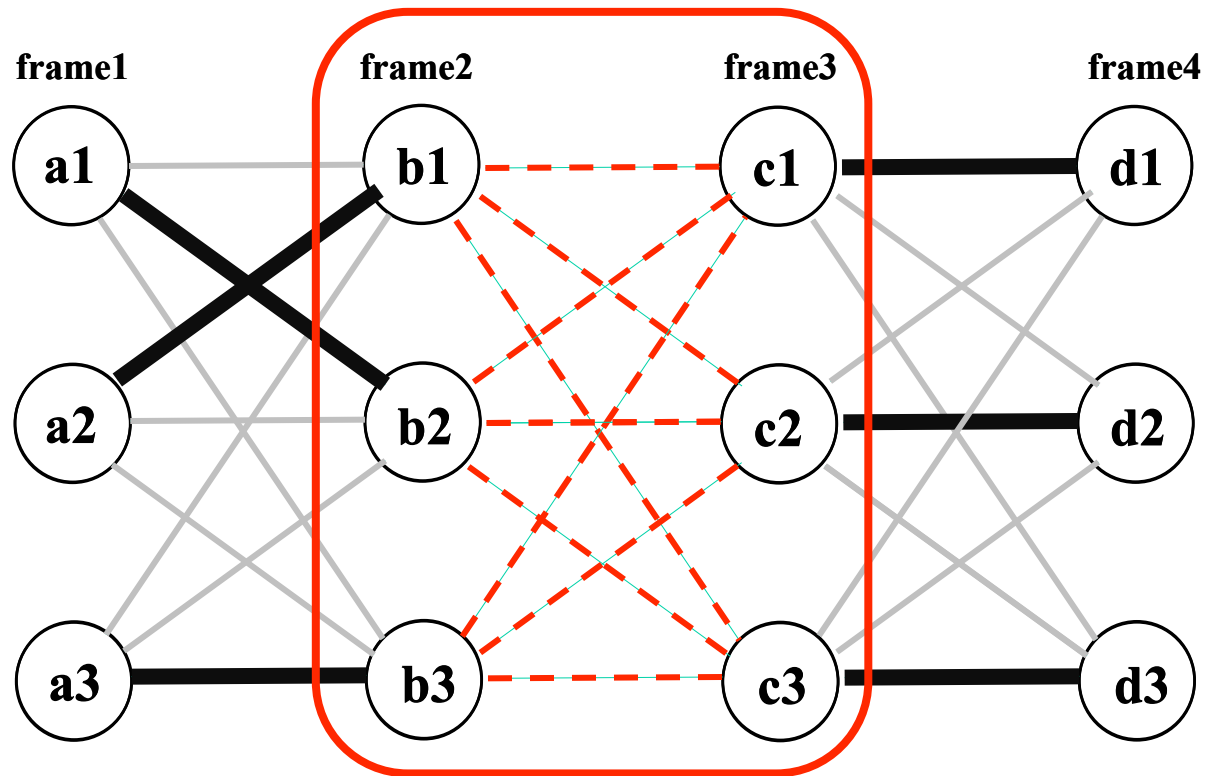
- Begin stepping through adjacent pairs of frames, solving updating edge decisions to improve the objective function value

# To Summarize Our Approach



- Begin stepping through adjacent pairs of frames, solving updating edge decisions to improve the objective function value

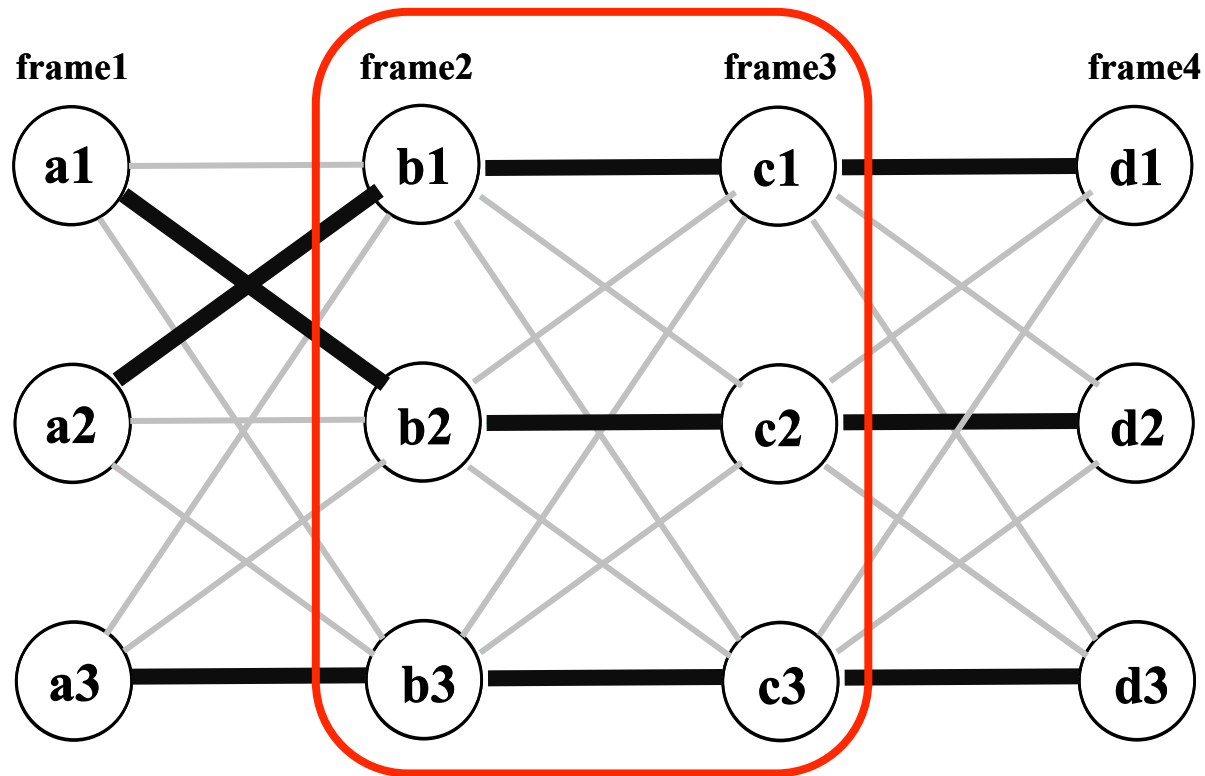
# To Summarize Our Approach



- Begin stepping through adjacent pairs of frames, solving updating edge decisions to improve the objective function value

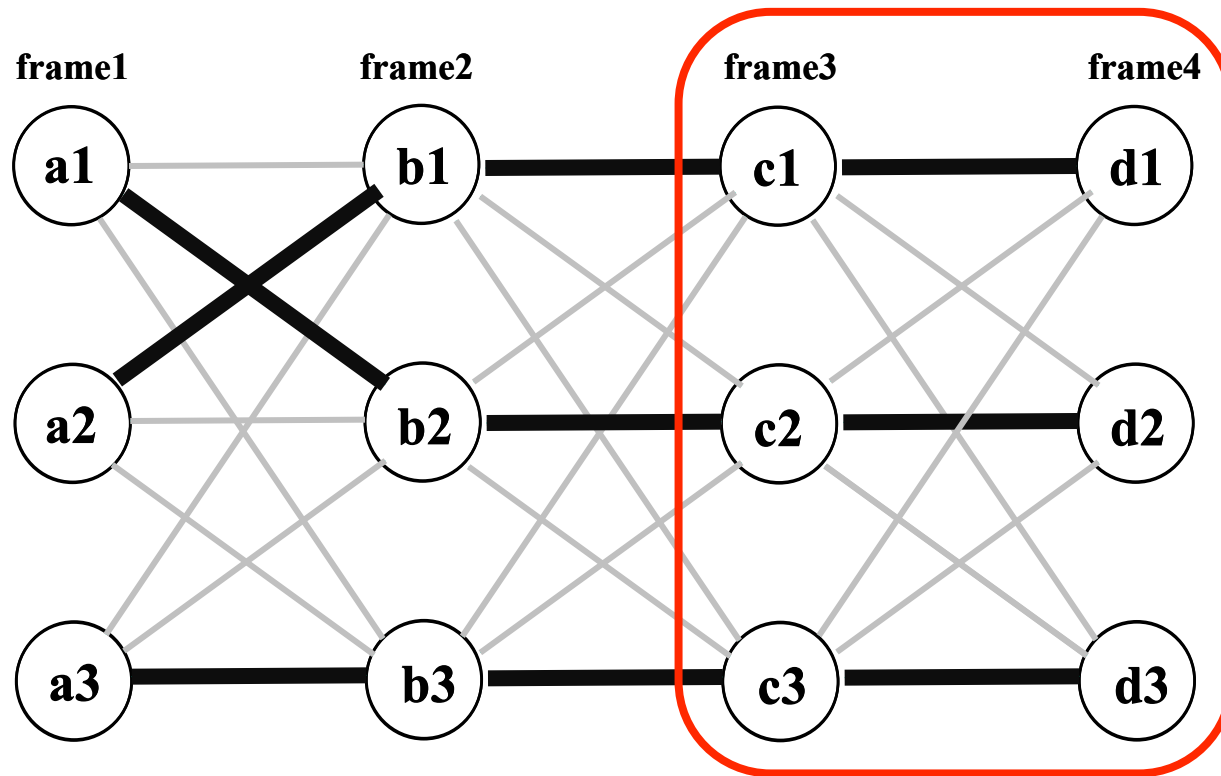


# To Summarize Our Approach



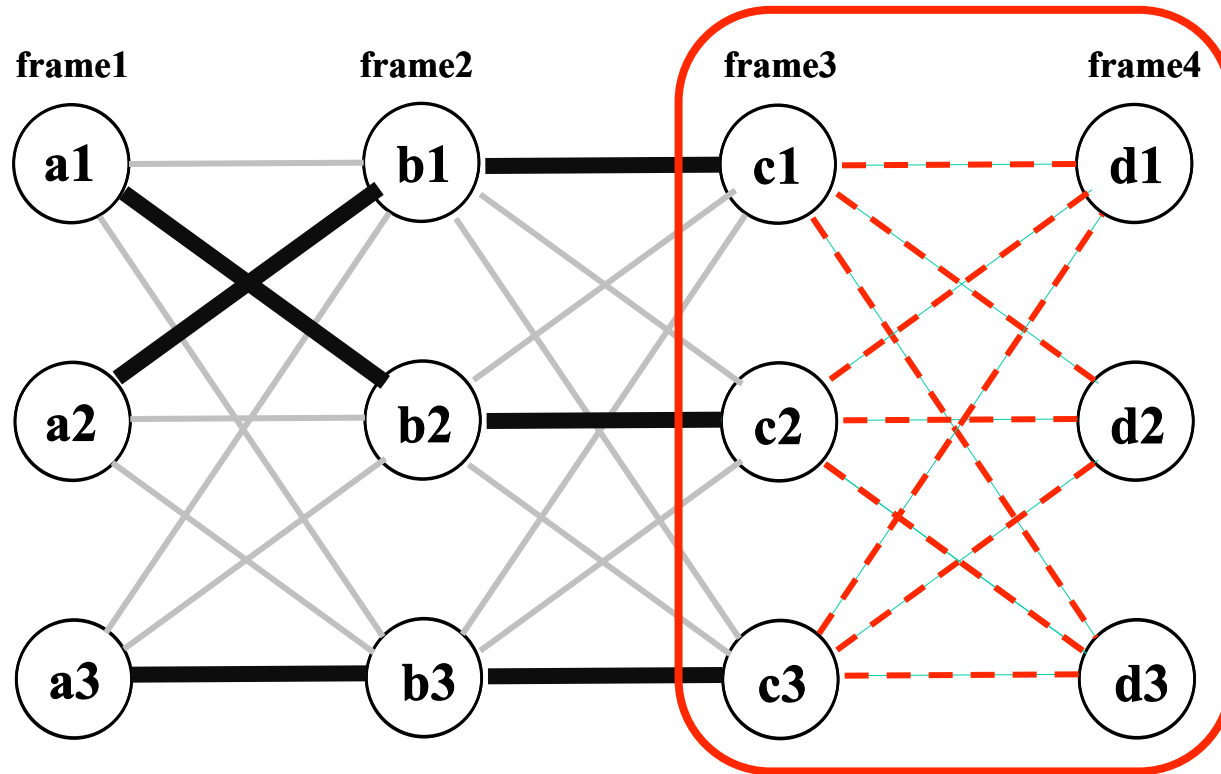
- Begin stepping through adjacent pairs of frames, solving updating edge decisions to improve the objective function value

# To Summarize Our Approach



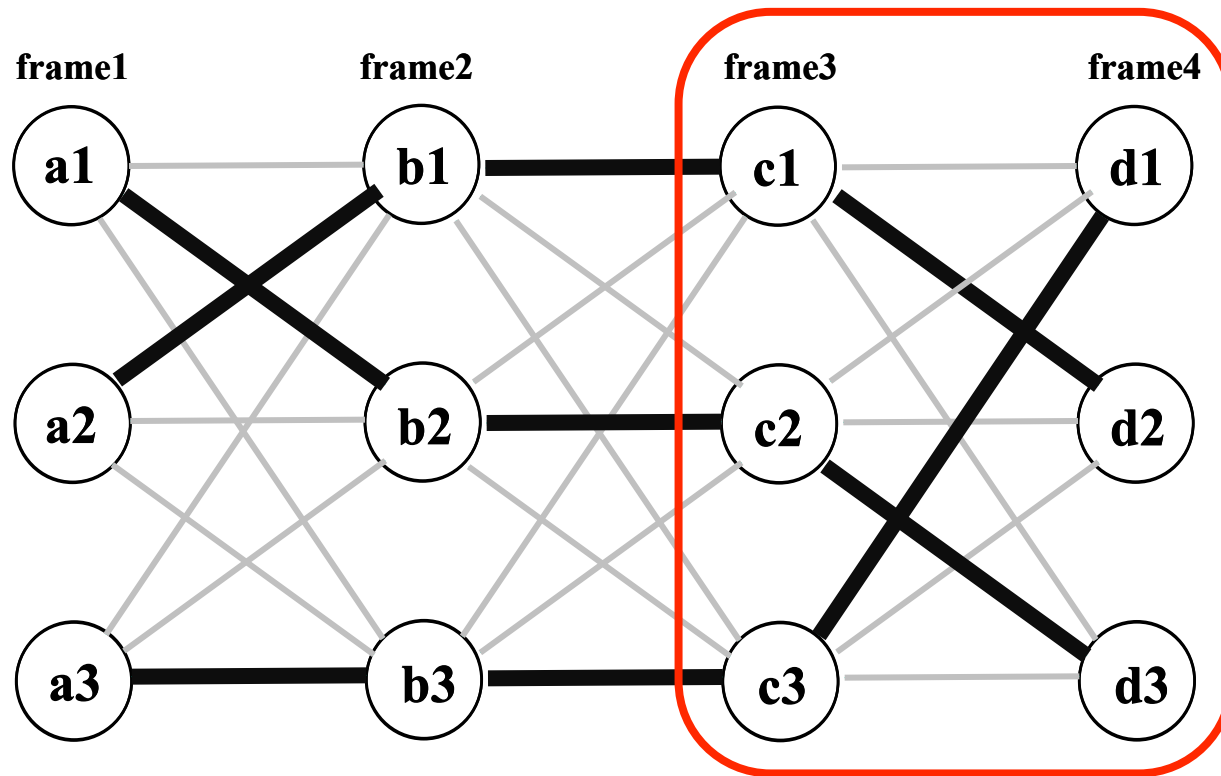
- Begin stepping through adjacent pairs of frames, solving updating edge decisions to improve the objective function value

# To Summarize Our Approach



- Begin stepping through adjacent pairs of frames, solving updating edge decisions to improve the objective function value

# To Summarize Our Approach



- Begin stepping through adjacent pairs of frames, solving updating edge decisions to improve the objective function value

# To Summarize Our Approach

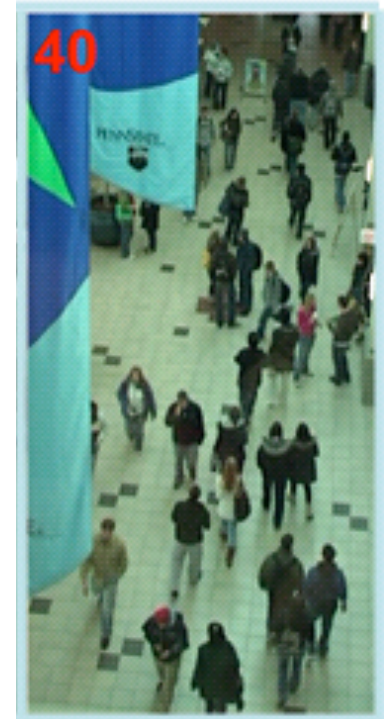
- Step through all pairs of frames, from beginning to end of sequence. This is one cycle.
- If no edge variables changed value during the cycle, we have converged and can stop.
- Otherwise, go back to first pair of frames and step through all pairs for another cycle.

## Relation to ICM

- This method is very much related to the iterated conditional modes algorithm of Besag (1986) for computing MAP estimate of an MRF
- But instead of optimizing a single variable at a time, we do a block update of all edge variables between two adjacent frames. [leads to faster convergence, and maintains a feasible solution]
- Inherits convergence properties (and proof) from ICM [guaranteed to converge to a local optimum]

# Validation

- Two datasets with ground-truth trajectories collected. One is relatively sparse (average of five people per frame); one is dense (average of 20 people per frame). Each sequence is 15 minutes long.
- We temporally subsampled the data into testsets having 3, 2 and 1 frames per sec.
- Only location data used; no appearance.
- Our approach is compared to two baseline methods
  - greedy sequential filtering (using constant velocity)
  - network flow algorithm (successive shortest paths)



sample frame  
dense sequence

# Validation

- We use a higher-order trajectory cost function based on the “snake” energy function of Kass, Witkin and Terzopoulos (1987)

$$\text{cost}(P) = \alpha E_{\text{cont}} + \beta E_{\text{curv}}$$

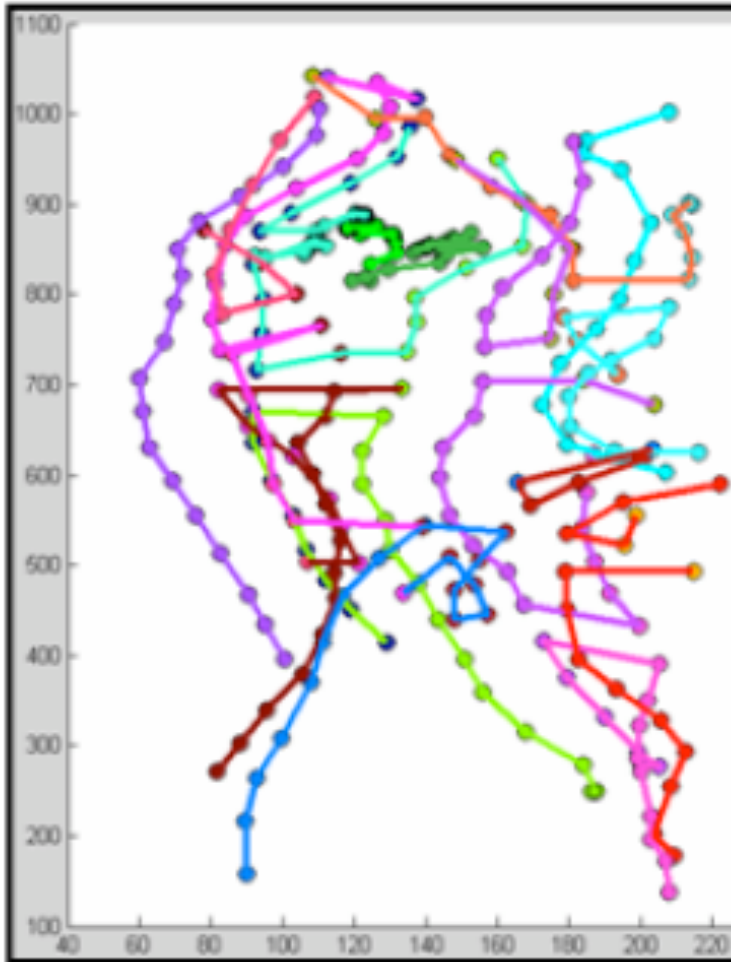
where

$$E_{\text{cont}} = \frac{1}{n-1} \sum_{i=2}^n \|p_i - p_{i-1}\| \quad \left. \vphantom{\sum} \right\} \text{distance}$$
$$E_{\text{curv}} = \sum_{i=2}^{n-1} \|p_{i+1} - 2p_i + p_{i-1}\|^2 \quad \left. \vphantom{\sum} \right\} \text{curvature}$$

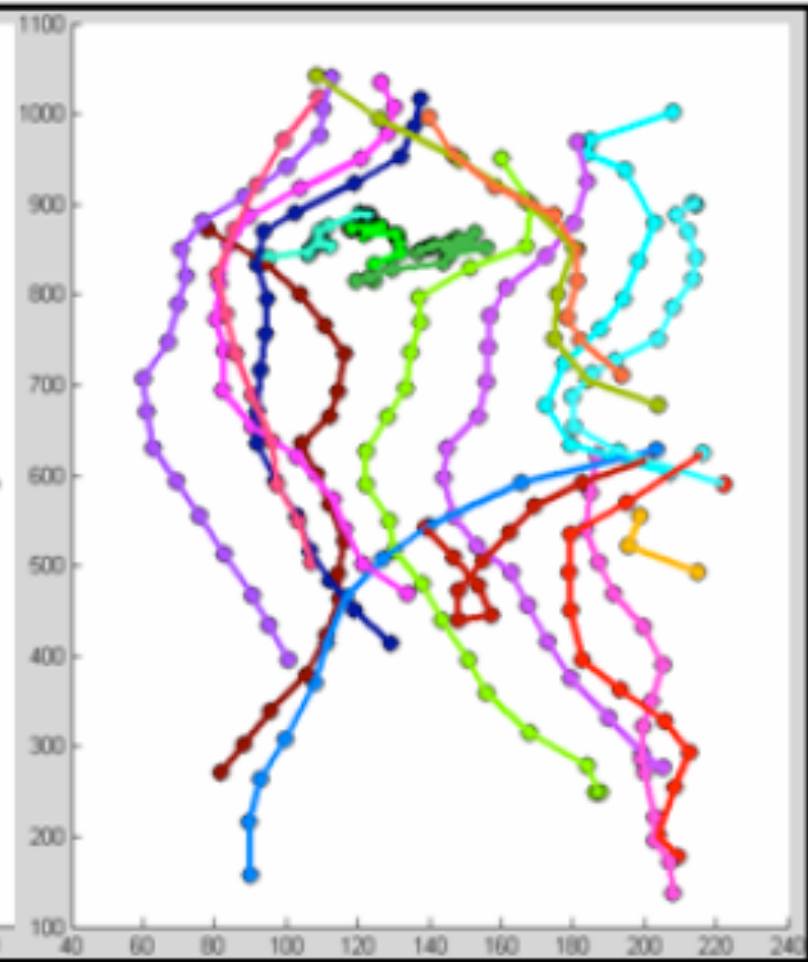


# Sample Results Sparse Dataset

**Network Flow**  
(22 ID swaps)

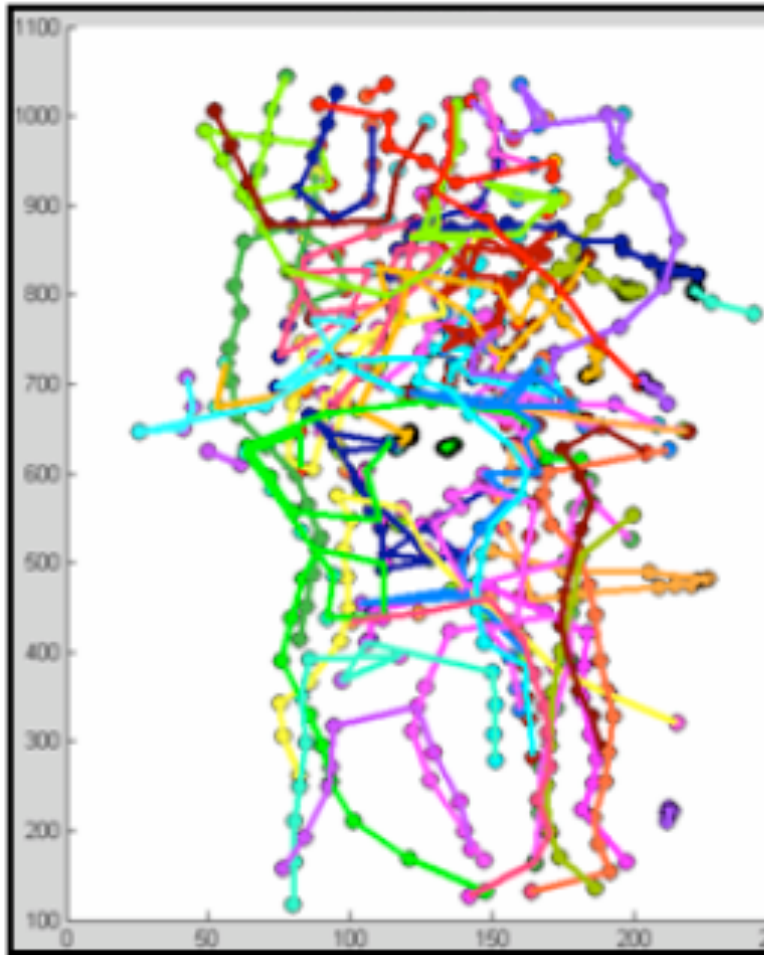


**Our Approach**  
(2 ID swaps)

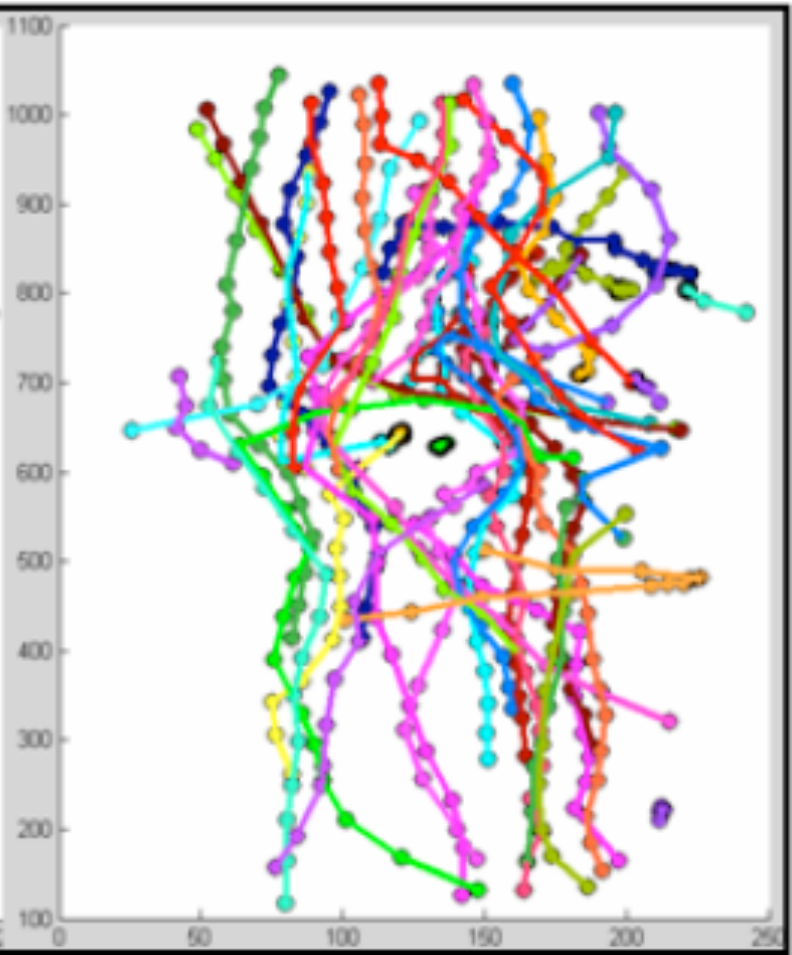


# Sample Results DenseDataset

**Network Flow**  
(116 ID swaps)



**Our Approach**  
(71 ID swaps)



# Validation

- Tabulating mismatch error percentage (one component of the MOTA tracking accuracy measure)

	Sparse Trajectories			Dense Trajectories		
	Flow	Greedy	Ours	Flow	Greedy	Ours
3fps	0.04	0.00	0.00 (1)	0.23	0.12	0.13 (2)
2fps	0.28	0.06	0.12 (1)	1.36	0.34	0.25 (2)
1fps	5.43	1.36	0.80 (2)	21.35	6.83	4.13 (5)

**smaller  
numbers  
are  
better**

$$mmep = 100 \times (\sum_t g(t) / \sum_t mme(t))$$
  
where  $g(t)$  is number of objects at time  $t$  and  $mme(t)$  is number of ID swaps at time  $t$

# Tracking and Data Association

- Tracking
  - Bayesian filtering
  - (continuous) Probability Theory
- Data Association
  - Assignment problems
  - (discrete) Combinatorics