Globally-Optimal Greedy Algorithms for Tracking a Variable Number of Objects

Hamed Pirsiavash, Deva Ramanan, and Charless C. Fowlkes
University of California, Irvine

Presented By

Albert Haque and Fahim Dalvi

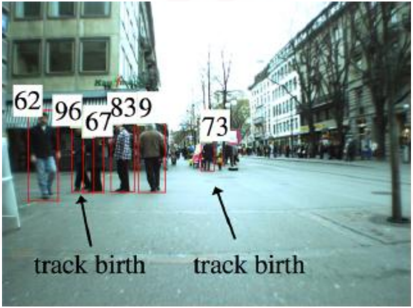
April 29, 2015

Outline

- Motivation & Related Work
- Mathematical Representation
  - Probabilistic Framework
  - ILP Formulation
- Multiple Object Tracking
  - Globally Optimal Greedy Algorithm
  - Approximate Dynamic Programming Algorithm
- Experiments and Results

Motivation

- Single object tracking isn't enough
- In reality, multiple objects appear and occlusion is present



Problem Statement

- Input: a video sequence with bounding boxes
- Output: assignment of IDs to all tracks
- Representation: a point x in *spacetime*
  - x includes pixel location, scale, time frame

Open in Adobe Reader to view in-slide video

Source: Jodoin, J. *et al.* Urban Tracker: Multiple Object Tracking in Urban Mixed Traffic. Applications of Computer Vision. 2014.

How can we solve multi-object tracking?

How can we solve multi-object tracking?

Answer: stitch together individual tracklets

- Examples: trajectory prediction, flow-networks, matching

How can we solve multi-object tracking?

Answer: stitch together individual tracklets

- Examples: trajectory prediction, flow-networks, matching

Track estimation with temporal trajectories [1]

[1] W. Choi and S. Savarese. Multiple target tracking in world coordinate with single, minimally calibrated camera. ECCV, 2010.

How can we solve multi-object tracking?

Answer: stitch together individual tracklets

- Examples: trajectory prediction, flow-networks, matching

Track estimation with temporal trajectories [1]

Joint modeling of trajectory groupings [2]

[1] W. Choi and S. Savarese. Multiple target tracking in world coordinate with single, minimally calibrated camera. ECCV, 2010.
[2] S. Pellegrini, A. Ess, and L. Gool. Improving data association by joint modeling of pedestrian trajectories and groupings. ECCV, 2010.

How can we solve multi-object tracking?

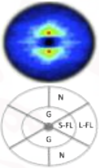
Answer: stitch together individual tracklets

- Examples: trajectory prediction, flow-networks, matching

Track estimation with temporal trajectories [1]

Joint modeling of trajectory groupings [2]

Forecasting with Social Affinity Map (SAM) descriptors [3]



[1] W. Choi and S. Savarese. Multiple target tracking in world coordinate with single, minimally calibrated camera. ECCV, 2010.
[2] S. Pellegrini, A. Ess, and L. Gool. Improving data association by joint modeling of pedestrian trajectories and groupings. ECCV, 2010.
[3] A. Alahi, V. Ramanathan, and L. Fei-Fei. Socially-aware large-scale crowd forecasting. CVPR, 2014.

How can we solve multi-object tracking?

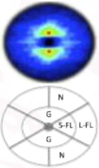
Answer: stitch together individual tracklets

- Examples: trajectory prediction, flow-networks, matching

Track estimation with temporal trajectories [1]

Joint modeling of trajectory groupings [2]

Forecasting with Social Affinity Map (SAM) descriptors [3]



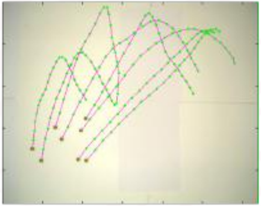
Limitations:

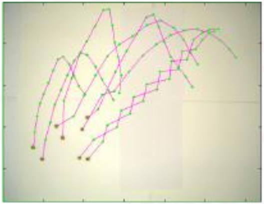
- Fails when objects move unpredictably (e.g. sports)

[1] W. Choi and S. Savarese. Multiple target tracking in world coordinate with single, minimally calibrated camera. ECCV, 2010.
[2] S. Pellegrini, A. Ess, and L. Gool. Improving data association by joint modeling of pedestrian trajectories and groupings. ECCV, 2010.
[3] A. Alahi, V. Ramanathan, and L. Fei-Fei. Socially-aware large-scale crowd forecasting. CVPR, 2014.

How can we solve multi-object tracking?

Hungarian bipartite graph matching [4, 5]



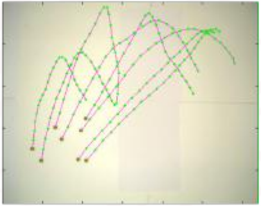


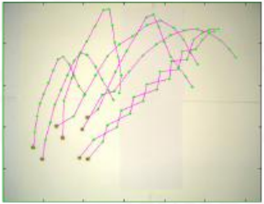
[4] H. Kuhn, *et al.* The Hungarian method for the assignment problem. 1993.

[5] M. Rowan and F. Maire. An efficient multiple object vision tracking system using bipartite graph matching. 2004.

How can we solve multi-object tracking?

Hungarian bipartite graph matching [4, 5]





Limitations:

- Is locally optimal but not globally optimal across time

[4] H. Kuhn, *et al.* The Hungarian method for the assignment problem. 1993.

[5] M. Rowan and F. Maire. An efficient multiple object vision tracking system using bipartite graph matching. 2004.

How can we find the globally optimal solution?

How can we find the globally optimal solution?

Answer: Integer linear programs (ILP)

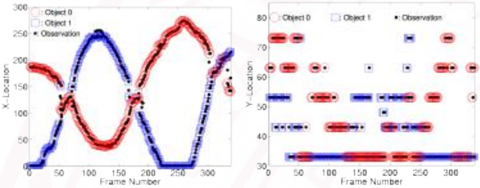
- Restrict possible locations to a finite set of candidate windows

How can we find the globally optimal solution?

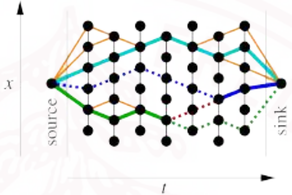
Answer: Integer linear programs (ILP)

- Restrict possible locations to a finite set of candidate windows

LP-approach to multiple tracking [6]



Integer programming approach [7]



[6] H. Jiang. *et al.* A linear programming approach for multiple object tracking. CVPR, 2007.

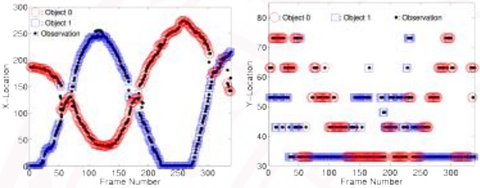
[7] A. Andriyenko and S. Konrad. Globally optimal multi-target tracking on a hexagonal lattice. ECCV, 2010.

How can we find the globally optimal solution?

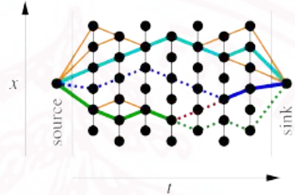
Answer: Integer linear programs (ILP)

- Restrict possible locations to a finite set of candidate windows

LP-approach to multiple tracking [6]



Integer programming approach [7]



Limitations:

- Doesn't scale well
- Limited or no occlusion modeling

[6] H. Jiang. *et al.* A linear programming approach for multiple object tracking. CVPR, 2007.

[7] A. Andriyenko and S. Konrad. Globally optimal multi-target tracking on a hexagonal lattice. ECCV, 2010.

Contributions

Past attempts require **prior information** while some methods are **not globally optimal**. Linear programs are optimal but **not efficient**.

Contributions

This paper proposes an ILP tracking formulation that:

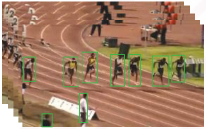
- is globally optimal
- is locally greedy
- scales linearly in the number of objects
- scales quasi-linearly in the number of frames

Research Questions

- How can we represent tracking as a probabilistic framework?
- How can we formulate this as an ILP?
- How can we efficiently solve it?
- How can we guarantee optimality?

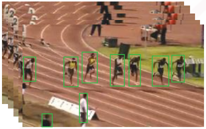
Algorithm Pipeline

1. Input Video + Bounding Boxes

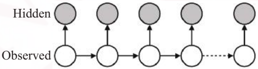


Algorithm Pipeline

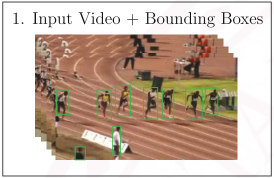
1. Input Video + Bounding Boxes

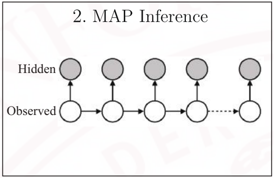


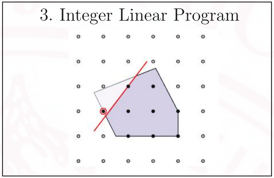
2. MAP Inference



Algorithm Pipeline

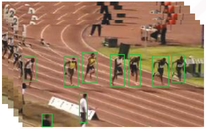




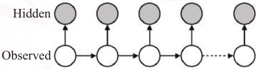


Algorithm Pipeline

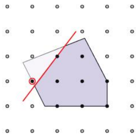
1. Input Video + Bounding Boxes



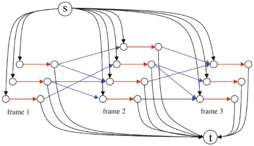
2. MAP Inference



3. Integer Linear Program

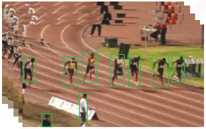


4. Network Flow

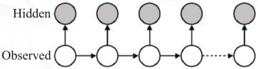


Algorithm Pipeline

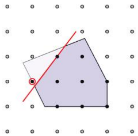
1. Input Video + Bounding Boxes



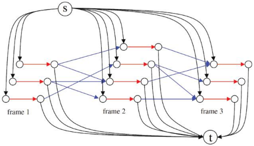
2. MAP Inference



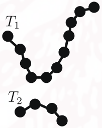
3. Integer Linear Program



4. Network Flow



5. Output Tracking Assignments



Outline

- Motivation & Related Work
- Mathematical Representation
  - Probabilistic Framework
  - ILP Formulation
- Multiple Object Tracking
  - Globally Optimal Greedy Algorithm
  - Approximate Dynamic Programming Algorithm
- Experiments and Results

Notation

We define a state vector x (i.e. a point in *spacetime*):

$$x = (p, \sigma, t) \quad \text{and} \quad x \in V$$

Where:

- $p =$ pixel location
- $\sigma =$ scale factor
- $t =$ frame number
- $V =$ set of all spacetime points

A track T is a set of state vectors: $T = \{x_1, ..., x_N\}$
Let X denote a set of K tracks: $X = \{T_1, ..., T_K\}$

Hidden Markov Model

Let X denote the output tracking assignments:

$$P(X) = \prod_{T \in X} P(T) \tag{1}$$

Hidden Markov Model

Let X denote the output tracking assignments:

$$P(X) = \prod_{T \in X} P(T) \tag{1}$$

$$P(T) = P_s(x_1) \left(\prod_{n=1}^{N-1} P(x_{n+1}|x_n) \right) P_t(x_N) \tag{2}$$

Where:

- $P_s(x_1)$ is the prior for a track starting at x_1
- $\prod_{n=1}^{N-1} P(x_{n+1}|x_n)$ is the probability we follow some track
- $P_t(x_N)$ is the prior for a track ending at x_N

Modeling Occlusion

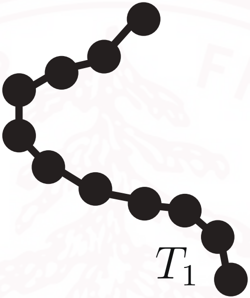
To model occlusion:

- Allow tracks to be composed of non-consecutive frames

Modeling Occlusion

To model occlusion:

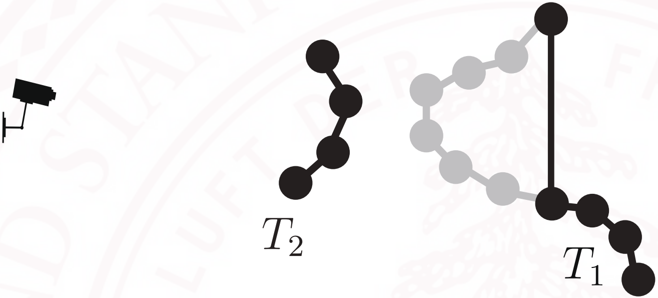
- Allow tracks to be composed of non-consecutive frames



Modeling Occlusion

To model occlusion:

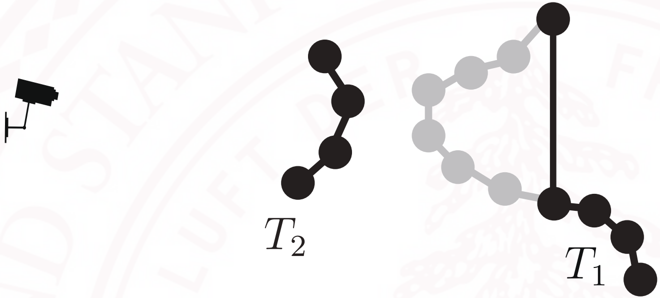
- Allow tracks to be composed of non-consecutive frames



Modeling Occlusion

To model occlusion:

- Allow tracks to be composed of non-consecutive frames



Note: $P(x_{n+1}|x_n)$ does not refer to the next frame but rather the next spacetime location in the track

MAP Inference

- Y = all features y_i observed at all spacetime points $i \in V$ in a video
- Goal: Select X^* such that it maximizes the likelihood of Y

MAP Inference

- Y = all features y_i observed at all spacetime points $i \in V$ in a video
- Goal: Select X^* such that it maximizes the likelihood of Y

$$X^* = \underset{X}{\operatorname{argmax}}\, P(X)P(Y|X) \tag{3}$$

MAP Inference

- Y = all features y_i observed at all spacetime points $i \in V$ in a video
- Goal: Select X^* such that it maximizes the likelihood of Y

$$X^* = \underset{X}{\operatorname{argmax}}\, P(X)P(Y|X) \tag{3}$$

$$= \underset{X}{\operatorname{argmax}} \prod_{T \in X} P(T) \prod_{x \in T} l(y_x) \tag{4}$$

- where $l(y_x) = \frac{P_{\mathrm{FG}}(y_x)}{P_{\mathrm{BG}}(y_x)}$ and $P_{\mathrm{FG}}, P_{\mathrm{BG}} \sim \mathcal{N}$

MAP Inference

- Y = all features y_i observed at all spacetime points $i \in V$ in a video
- Goal: Select X^* such that it maximizes the likelihood of Y

$$X^* = \underset{X}{\operatorname{argmax}}\, P(X)P(Y|X) \tag{3}$$

$$= \underset{X}{\operatorname{argmax}} \prod_{T \in X} P(T) \prod_{x \in T} l(y_x) \tag{4}$$

$$= \underset{X}{\operatorname{argmax}} \sum_{T \in X} \log P(T) + \sum_{x \in T} \log l(y_x) \tag{5}$$

- where $l(y_x) = \frac{P_{\mathrm{FG}}(y_x)}{P_{\mathrm{BG}}(y_x)}$ and $P_{\mathrm{FG}}, P_{\mathrm{BG}} \sim \mathcal{N}$

How can we find the MAP?

How can we find the MAP?

Answer: Represent MAP as an integer linear program (ILP)

How can we find the MAP?

Answer: Represent MAP as an integer linear program (ILP)

- Let i, j be different frames

How can we find the MAP?

Answer: Represent MAP as an integer linear program (ILP)

- Let i, j be different frames
- $f_{ij}, f_i, f_i^s, f_i^t$ are indicator variables

How can we find the MAP?

Answer: Represent MAP as an integer linear program (ILP)

- Let i, j be different frames
- $f_{ij}, f_i, f_i^s, f_i^t$ are indicator variables
- $c_i^s = -\log P_s(x_i) \quad c_i^t = -\log P_t(x_i)$

How can we find the MAP?

Answer: Represent MAP as an integer linear program (ILP)

- Let i, j be different frames
- $f_{ij}, f_i, f_i^s, f_i^t$ are indicator variables
- $c_i^s = -\log P_s(x_i) \quad c_i^t = -\log P_t(x_i)$
- $c_{ij} = -\log P(x_j | x_i)$

How can we find the MAP?

Answer: Represent MAP as an integer linear program (ILP)

- Let i, j be different frames
- $f_{ij}, f_i, f_i^s, f_i^t$ are indicator variables
- $c_i^s = -\log P_s(x_i) \quad c_i^t = -\log P_t(x_i)$
- $c_{ij} = -\log P(x_j|x_i)$
- $c_i = -\log l(y_i)$

How can we find the MAP?

Formal ILP definition:

Where E is the set of permissible state transitions given by dynamic model

How can we find the MAP?

Formal ILP definition:

$$f^* = \operatorname*{argmin}_{x} C(f) \tag{6}$$

Where E is the set of permissible state transitions given by dynamic model

How can we find the MAP?

Formal ILP definition:

$$f^* = \operatorname*{argmin}_x C(f) \tag{6}$$

$$\text{with } C(f) = \sum_i c_i^s f_i^s + \sum_{ij \in E} c_{ij} f_{ij} + \sum_i c_i f_i + \sum_i c_i^t f_i^t \tag{7}$$

Where E is the set of permissible state transitions given by dynamic model

How can we find the MAP?

Formal ILP definition:

$$f^* = \operatorname*{argmin}_x C(f) \tag{6}$$

$$\text{with } C(f) = \sum_i c_i^s f_i^s + \sum_{ij \in E} c_{ij} f_{ij} + \sum_i c_i f_i + \sum_i c_i^t f_i^t \tag{7}$$

$$\text{s.t.} \quad f_{ij}, f_i, f_i^s, f_i^t \in [0, 1] \tag{8}$$

Where E is the set of permissible state transitions given by dynamic model

How can we find the MAP?

Formal ILP definition:

$$f^* = \operatorname*{argmin}_x C(f) \tag{6}$$

$$\text{with } C(f) = \sum_i c_i^s f_i^s + \sum_{ij \in E} c_{ij} f_{ij} + \sum_i c_i f_i + \sum_i c_i^t f_i^t \tag{7}$$

$$\text{s.t.} \quad f_{ij}, f_i, f_i^s, f_i^t \in [0, 1] \tag{8}$$

$$\text{and} \quad f_i^s + \sum_i f_{ji} = f_i = f_i^t + \sum_j f_{ij} \tag{9}$$

Where E is the set of permissible state transitions given by dynamic model

How can we find the MAP?

Formal ILP definition:

$$f^* = \underset{x}{\operatorname{argmin}}\, C(f) \tag{6}$$

$$\text{with } C(f) = \sum_i c_i^s f_i^s + \sum_{ij \in E} c_{ij} f_{ij} + \sum_i c_i f_i + \sum_i c_i^t f_i^t \tag{7}$$

$$\text{s.t.} \quad f_{ij}, f_i, f_i^s, f_i^t \in [0, 1] \tag{8}$$

$$\text{and} \quad f_i^s + \sum_i f_{ji} = f_i = f_i^t + \sum_j f_{ij} \tag{9}$$

Where E is the set of permissible state transitions given by dynamic model

How can we find the MAP?

Formal ILP definition:

$$f^* = \operatorname*{argmin}_x C(f) \tag{6}$$

$$\text{with } C(f) = \sum_i c_i^s f_i^s + \sum_{ij \in E} c_{ij} f_{ij} + \sum_i c_i f_i + \sum_i c_i^t f_i^t \tag{7}$$

$$\text{s.t.} \quad f_{ij}, f_i, f_i^s, f_i^t \in [0, 1] \tag{8}$$

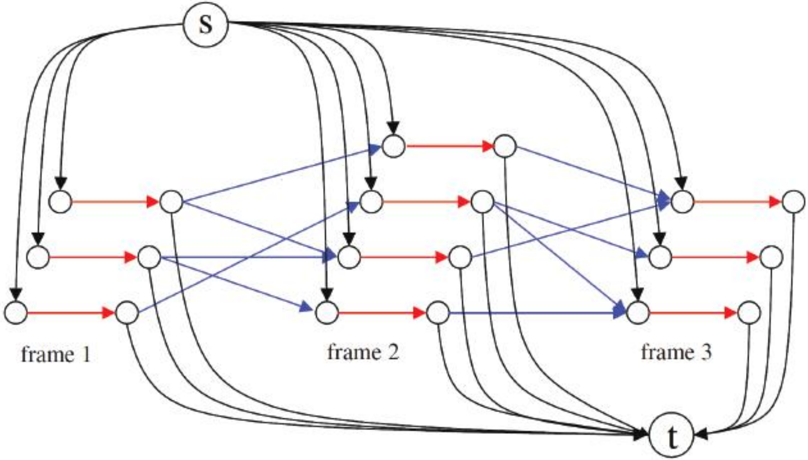
$$\text{and} \quad f_i^s + \sum_i f_{ji} = f_i = f_i^t + \sum_j f_{ij} \tag{9}$$

Where E is the set of permissible state transitions given by dynamic model

How can we solve the ILP?

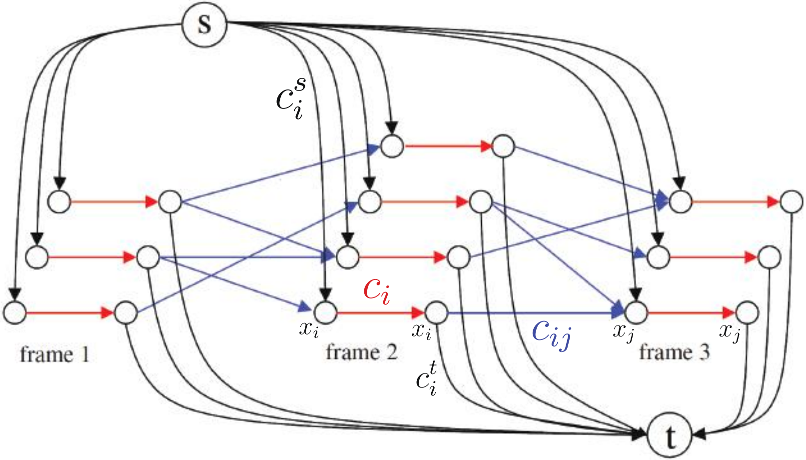
How can we solve the ILP?

- Answer: Represent ILP as a network-flow problem



How can we solve the ILP?

- Answer: Represent ILP as a network-flow problem



$c_i^s = -\log P_s(x_i) \quad c_i^t = -\log P_t(x_i) \quad c_{ij} = -\log P(x_j|x_i) \quad c_i = -\log l(y_i)$

Outline

- Motivation & Related Work
- Mathematical Representation
  - Probabilistic Framework
  - ILP Formulation
- Multiple Object Tracking
  - Globally Optimal Greedy Algorithm
  - Approximate Dynamic Programming Algorithm
- Experiments and Results

How can we find the min-cost flows?

- Answer: Push a flow of K through the graph

[8] L. Zhang *et al*. Global data association for multi-object tracking using network flows. CVPR, 2008.
[9] A. Goldberg. An efficient implementation of scaling minimum-cost flow algorithm. Journal of Algorithms, 1997.

How can we find the min-cost flows?

- Answer: Push a flow of K through the graph
- Using properties of our network:
  - All edges are unit capacity
  - Network is a DAG

[8] L. Zhang *et al*. Global data association for multi-object tracking using network flows. CVPR, 2008.
[9] A. Goldberg. An efficient implementation of scaling minimum-cost flow algorithm. Journal of Algorithms, 1997.

How can we find the min-cost flows?

- Answer: Push a flow of K through the graph
- Using properties of our network:
  - All edges are unit capacity
  - Network is a DAG
- Best previous method $[8, 9]$: $O(N^3 \log N)$

[8] L. Zhang *et al*. Global data association for multi-object tracking using network flows. CVPR, 2008.
[9] A. Goldberg. An efficient implementation of scaling minimum-cost flow algorithm. Journal of Algorithms, 1997.

How can we find the min-cost flows?

- Answer: Push a flow of K through the graph
- Using properties of our network:
  - All edges are unit capacity
  - Network is a DAG
- Best previous method $[8, 9]$: $O(N^3 \log N)$
- We achieve a $O(KN \log N)$ algorithm

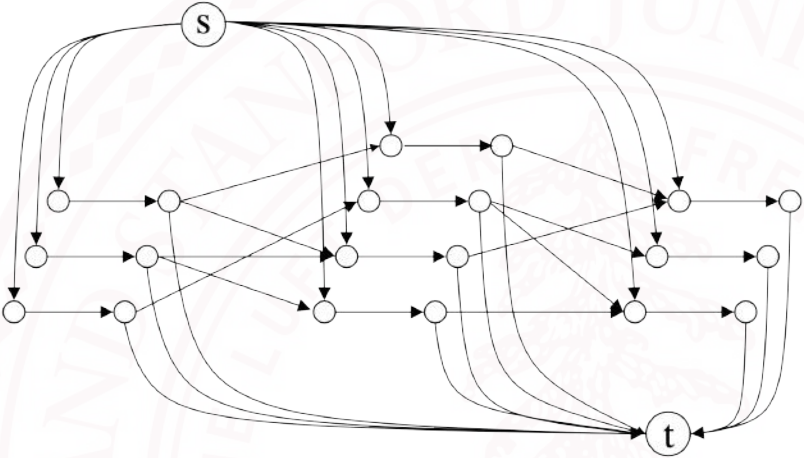
[8] L. Zhang *et al.* Global data association for multi-object tracking using network flows. CVPR, 2008.
[9] A. Goldberg. An efficient implementation of scaling minimum-cost flow algorithm. Journal of Algorithms, 1997.

How can we find the min-cost flows?

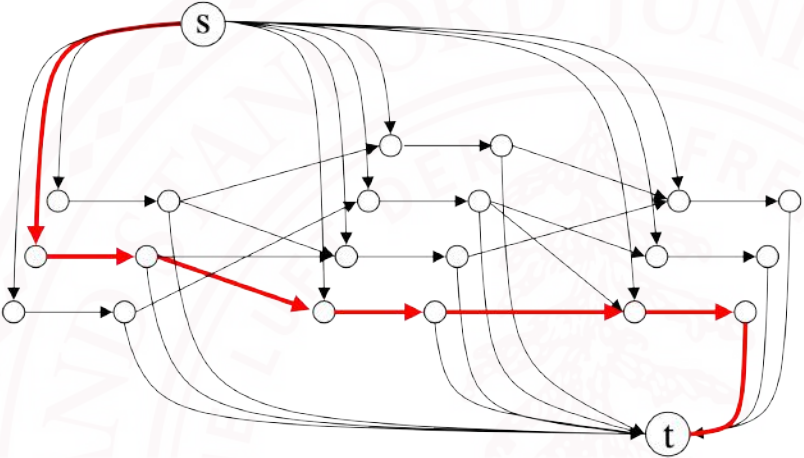
This paper proposes three algorithms:

- Successive shortest-paths
- Approximate One-Pass DP for $K > 1$
- Approximate Two-Pass DP for $K > 1$

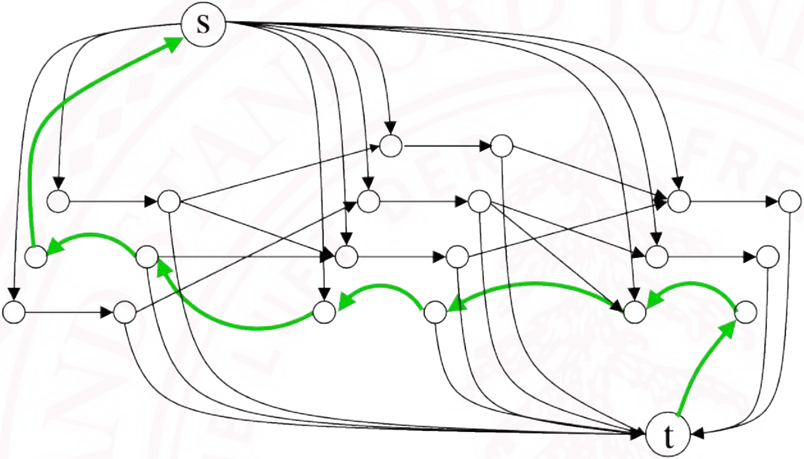
Successive Shortest Paths



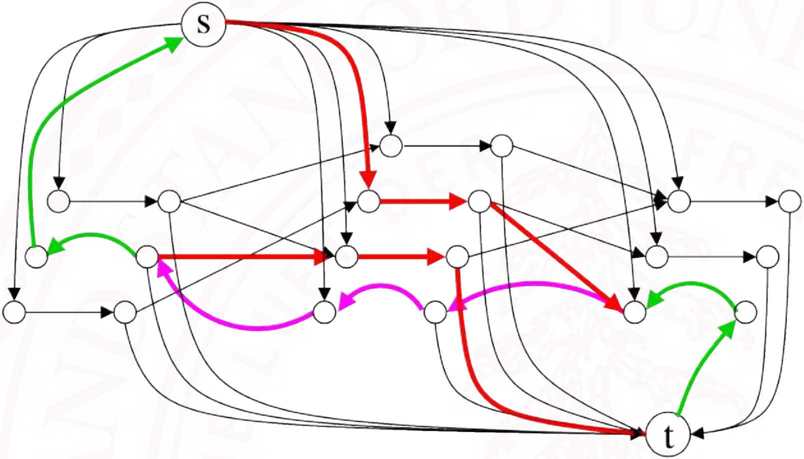
Successive Shortest Paths



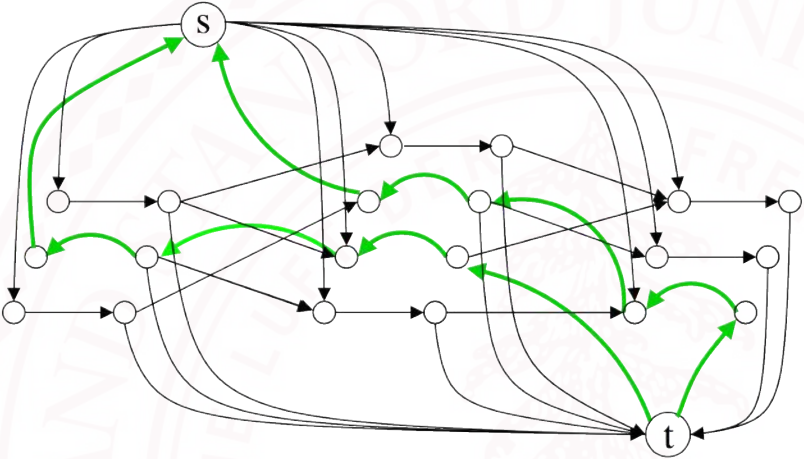
Successive Shortest Paths



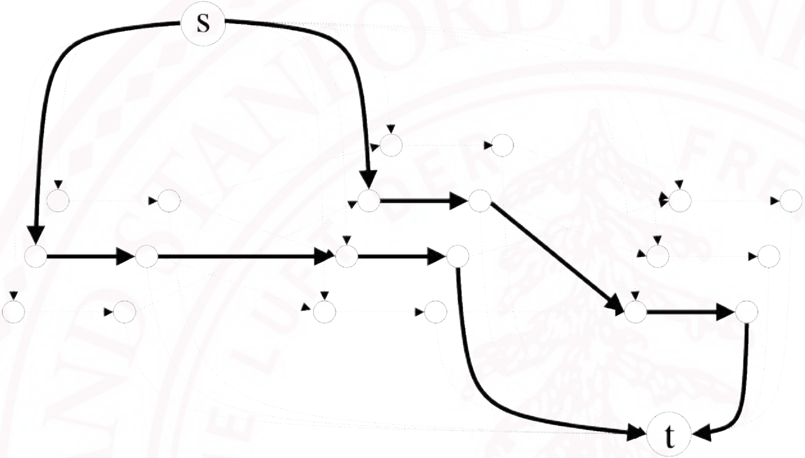
Successive Shortest Paths



Successive Shortest Paths



Successive Shortest Paths



Successive Shortest Paths

- Problem: Using residual graph introduces negative costs

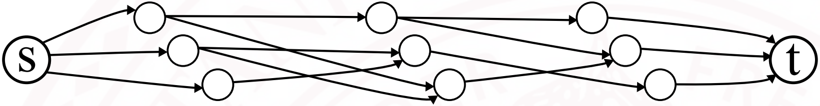
Successive Shortest Paths

- Problem: Using residual graph introduces negative costs
- Solution: Convert residual graph to positive costs only
  - Requires computing shortest path from source to all nodes
  - $O(N^2)$ with Bellman-Ford

Successive Shortest Paths

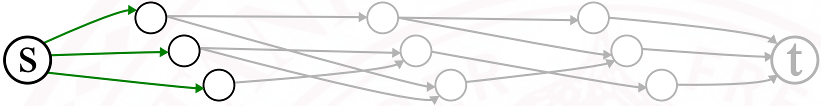
- Problem: Using residual graph introduces negative costs
- Solution: Convert residual graph to positive costs only
  - Requires computing shortest path from source to all nodes
  - $O(N^2)$ with Bellman-Ford
- Our DP approach: $O(N)$

Dynamic Programming Approach



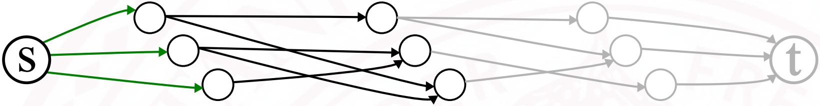
Start with a partial ordering of the nodes based on time

Dynamic Programming Approach



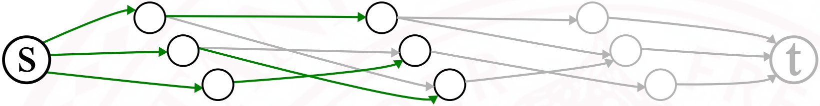
$$cost(i) = c_i + c_i^s$$

Dynamic Programming Approach



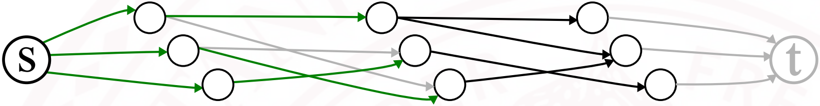
$$cost(i) = c_i + \min(\pi, c_i^s)$$
$$\pi = \min_{j \in N(i)} c_{ij} + cost(j)$$

Dynamic Programming Approach



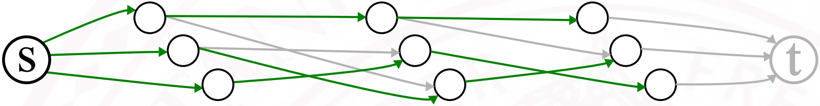
$$cost(i) = c_i + \min(\pi, c_i^s)$$
$$\pi = \min_{j \in N(i)} c_{ij} + cost(j)$$

Dynamic Programming Approach



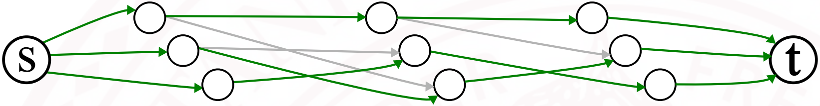
$$cost(i) = c_i + \min(\pi, c_i^s)$$
$$\pi = \min_{j \in N(i)} c_{ij} + cost(j)$$

Dynamic Programming Approach



$$cost(i) = c_i + \min(\pi, c_i^s)$$
$$\pi = \min_{j \in N(i)} c_{ij} + cost(j)$$

Dynamic Programming Approach



$$cost(i) = c_i + \min(\pi, c_i^s)$$
$$\pi = \min_{j \in N(i)} c_{ij} + cost(j)$$

How can we track one object?

How can we track one object?

- Shortest path corresponds to optimal solution for $K = 1$

How can we track one object?

- Shortest path corresponds to optimal solution for $K = 1$
- Conversion algorithm gives us shortest path from source node to terminal node

How can we track multiple objects?

Approximate One-Pass DP $O(KN)$ Algorithm:

How can we track multiple objects?

Approximate One-Pass DP $O(KN)$ Algorithm:

- Start with original flow graph, perform $K + 1$ iterations:

How can we track multiple objects?

Approximate One-Pass DP $O(KN)$ Algorithm:

- Start with original flow graph, perform $K + 1$ iterations:
  - Find shortest path from s to t

How can we track multiple objects?

Approximate One-Pass DP $O(KN)$ Algorithm:

- Start with original flow graph, perform $K + 1$ iterations:
  - Find shortest path from s to t
  - If path cost is negative, remove nodes on the path

How can we track multiple objects?

Approximate One-Pass DP $O(KN)$ Algorithm:

- Start with original flow graph, perform $K + 1$ iterations:
  - Find shortest path from s to t
  - If path cost is negative, remove nodes on the path
- At each iteration, we instance one track

Outline

- Motivation & Related Work
- Mathematical Representation
  - Probabilistic Framework
  - ILP Formulation
- Multiple Object Tracking
  - Globally Optimal Greedy Algorithm
  - Approximate Dynamic Programming Algorithm
- Experiments and Results

Datasets

- Caltech Pedestrian Dataset [7]: 71 videos, 1800 frames each, 30 fps

- ETHMS Dataset [8]: 4 videos, 1000 frames each, 14 fps

[7] Dollar, P. *et al.* Pedestrian detection: A benchmark. CVPR, 2009.

[8] Ess, A. *et al.* A Mobile Vision System for Robust Multi-Person Tracking. CVPR, 2008.

Evaluation Metrics

Evaluation Metrics

$$\text{Detection rate (recall)} = \frac{\text{Number of correct ID labelings}}{\text{Total number of ID labelings}}$$

Evaluation Metrics

$$\text{Detection rate (recall)} = \frac{\text{Number of correct ID labelings}}{\text{Total number of ID labelings}}$$

$$\text{False positives per image (FPPI)} = \frac{\text{Total number of false positives}}{\text{Number of images (frames)}}$$

Evaluation Metrics

$$\text{Detection rate (recall)} = \frac{\text{Number of correct ID labelings}}{\text{Total number of ID labelings}}$$

$$\text{False positives per image (FPPI)} = \frac{\text{Total number of false positives}}{\text{Number of images (frames)}}$$

$$\text{Identification error} = \frac{\text{Number of incorrect ID labelings}}{\text{Total number of ID labelings}}$$

Performance Comparison

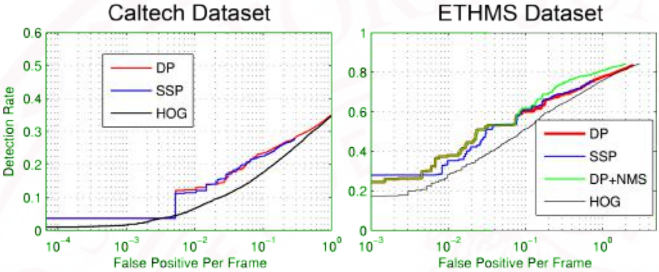
Algorithm	Detection Rate	FPPI
Stereo Algorithm [10]	47.0	1.50
MAP/Min-Cost Flow [11]	68.3	0.85
MAP/Min-Cost Flow + Occlusion Handling [11]	70.4	0.97
Two-Stage + Occlusion Handling [12]	75.2	0.93
Our DP	**76.6**	**0.85**
Our DP + NMS	**79.8**	**0.85**

[10] A. Ess *et al.* Depth and appearance for mobile scene analysis. ICCV, 2007.

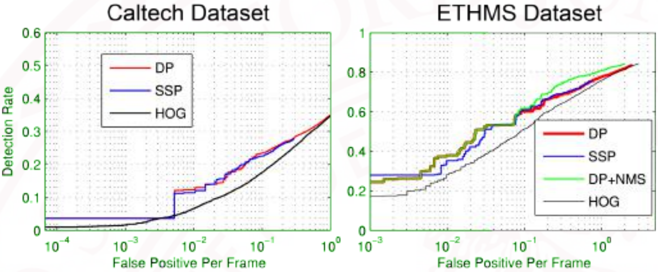
[11] L. Zhang *et al.* Global data association for multi-object tracking using network flows. CVPR, 2008.

[12] J. Xing *et al.* Multi-object tracking through occlusions by local tracklets filtering and global tracklets association. CVPR, 2009.

Detection Rate vs False Positives per Image (FPPI)



Detection Rate vs False Positives per Image (FPPI)



Key Insights:

- SSP produces short tracks due to 1st order Markov property
- DP produces longer tracks because tracks are never cut or edited

Track Label Error vs Allowed Occlusion

Results on ETHMS Dataset (Ideal Detector)

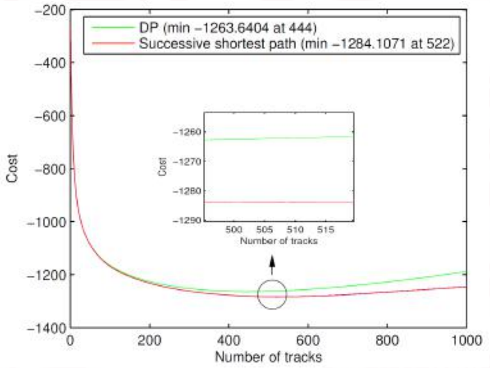
Length of Allowable Occlusion	Windows with ID Errors
1	14.69%
5	13.32%
10	9.39%

Key Insight:

- Larger occlusion windows improve performance

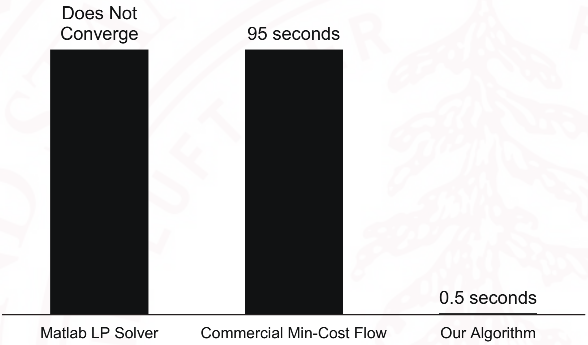
Cost versus iteration number

- DP algorithm is close to optimal (SSP) while being orders of magnitude faster



Algorithm Runtime

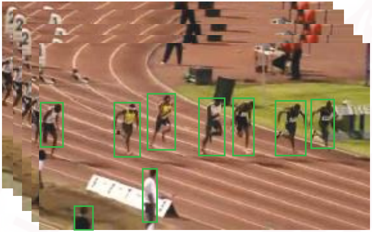
- DP algorithm is two orders of magnitude faster than commercial solvers



Conclusion

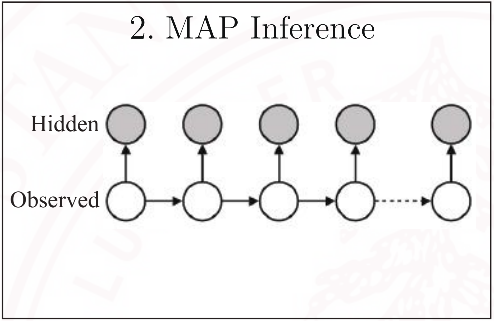
Given the input, we answered several research questions:

1. Input Video + Bounding Boxes



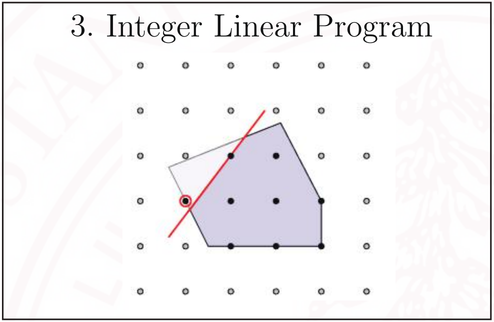
Conclusion

How can we represent tracking as a probabilistic framework?



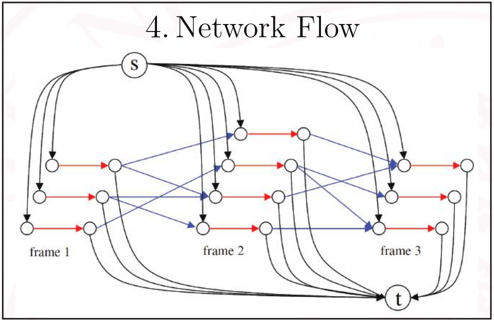
Conclusion

How can we formulate this as an ILP?



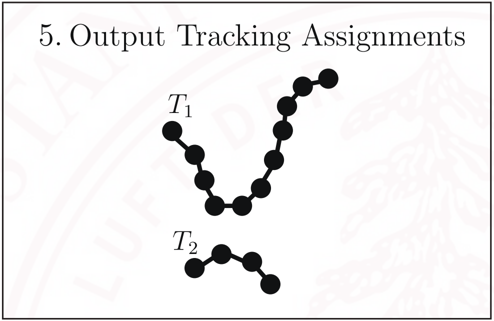
Conclusion

How can we efficiently solve it?



Conclusion

This allowed us solve the multi-object tracking problem:



Questions?