Globally-Optimal Greedy Algorithms for Tracking Variable Number of Objects

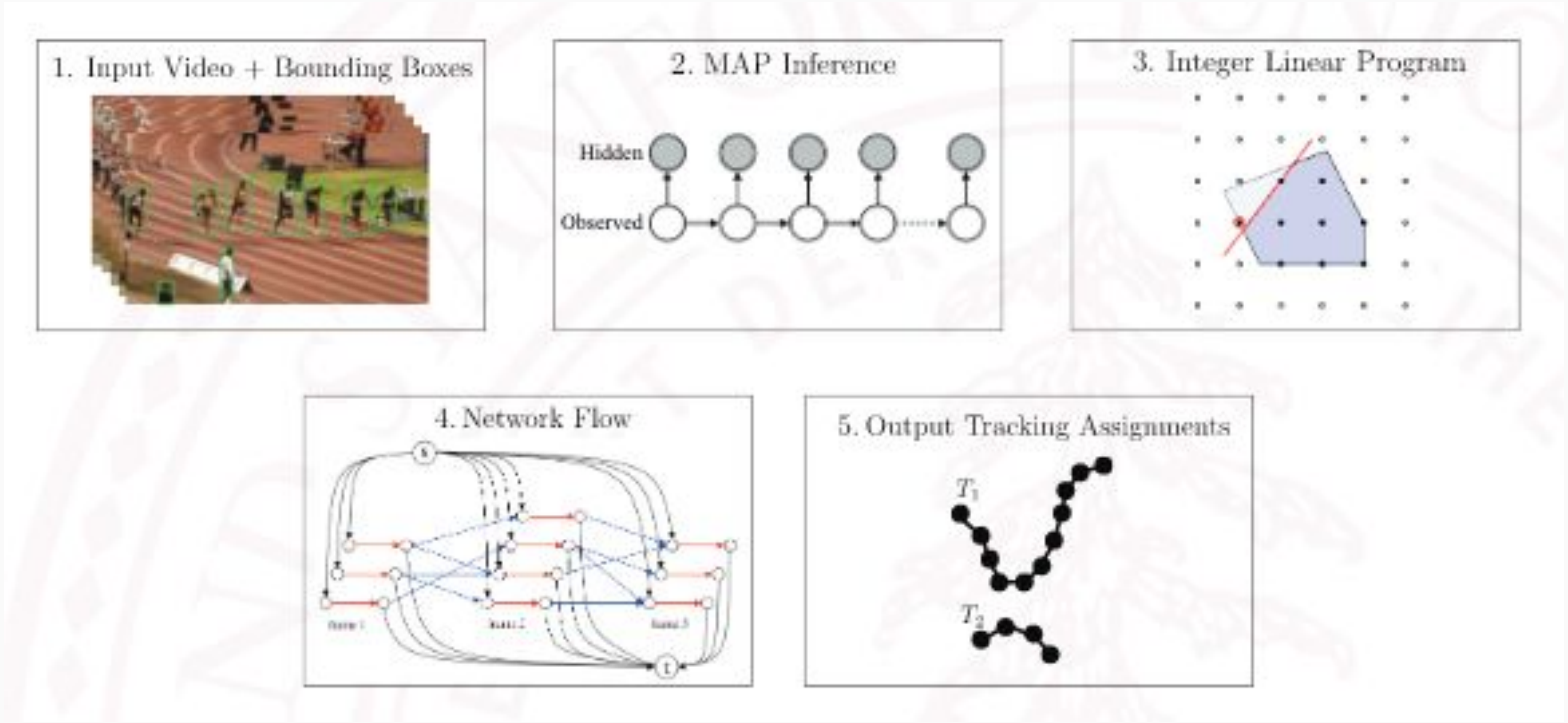
C.V. Project: Fourier and Duals

Mohsin Mustafa, Ramkishore Saravanan and Shyamgopal Karthik

Brief description of the paper

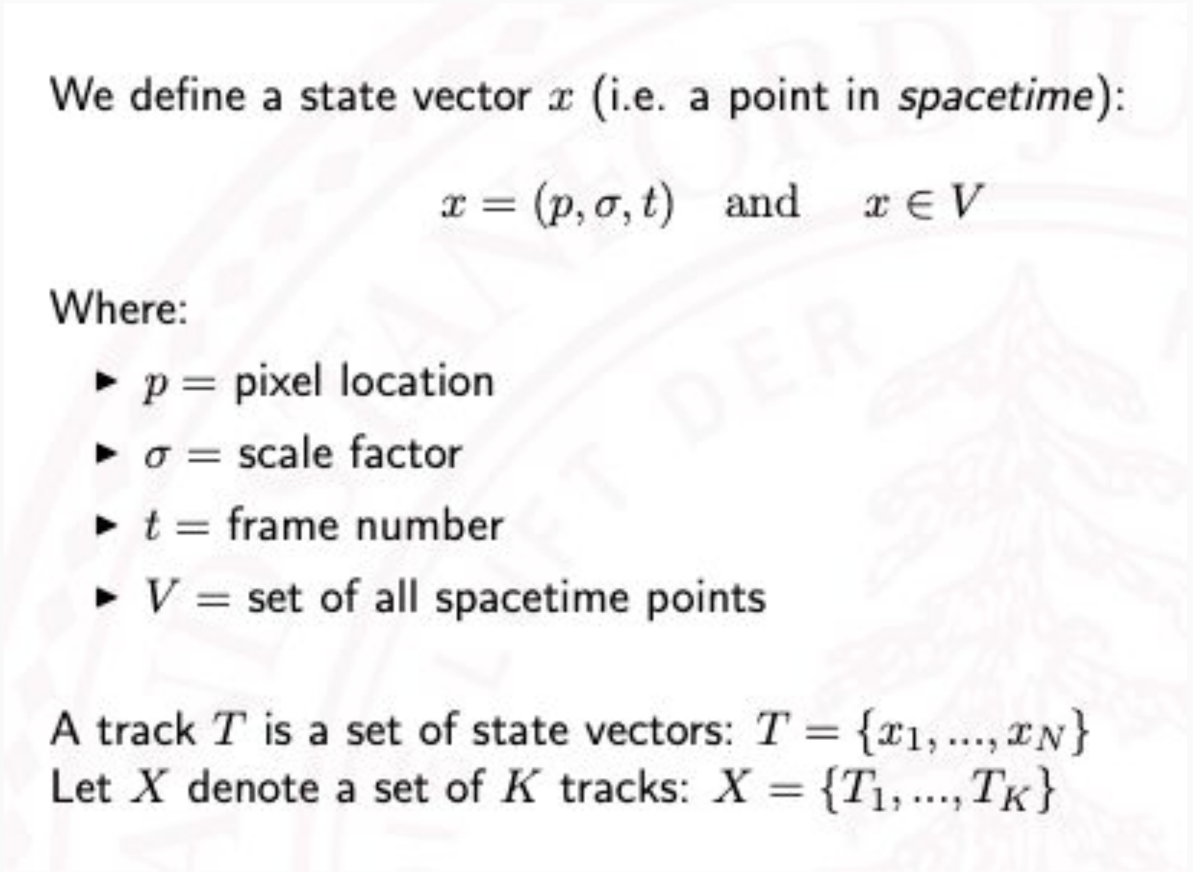
- Find the objects in each frame.
- Formulate observations of transitions, start and terminations of tracks as a Markov Model. Use MAP to maximize observed probabilities.
- Formulate ILP to select optimal tracks from MM.
- Convert ILP to DAG, with edge weights as negative of transition probabilities.
- Solve ILP by finding shortest paths, paths with highest probabilities.

Algorithm Pipeline


1. Input Video + Bounding Boxes
2. MAP Inference
Hidden
Observed
3. Integer Linear Program
4. Network Flow
s
t
frame 1
frame 2
frame 3
5. Output Tracking Assignments
T_1
T_2

Mathematical Representation

Notation



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

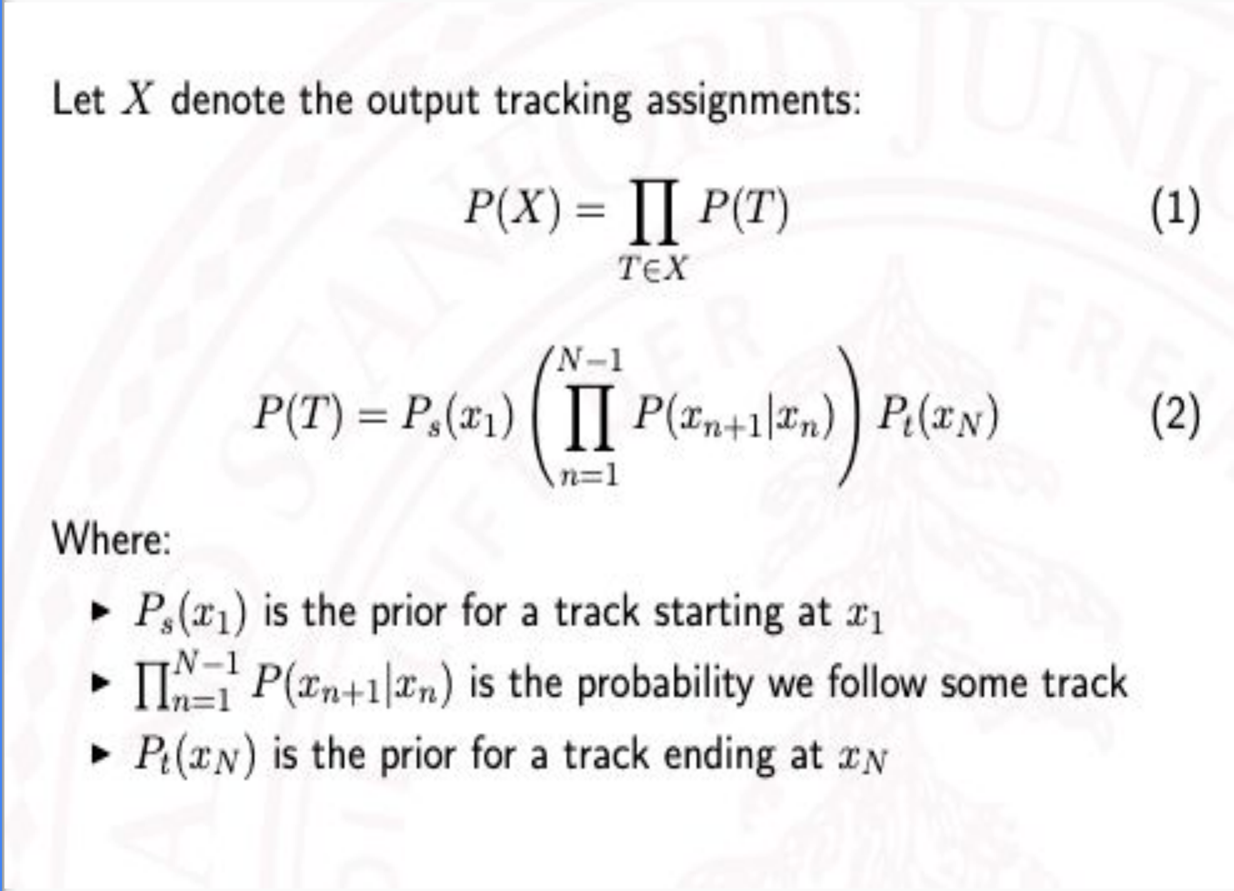
$$x = (p, \sigma, t) \quad \text{and} \quad x \in V$$

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: $X = \{T_1, ..., T_K\}$

Markov Model



Let X denote the output tracking assignments:

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

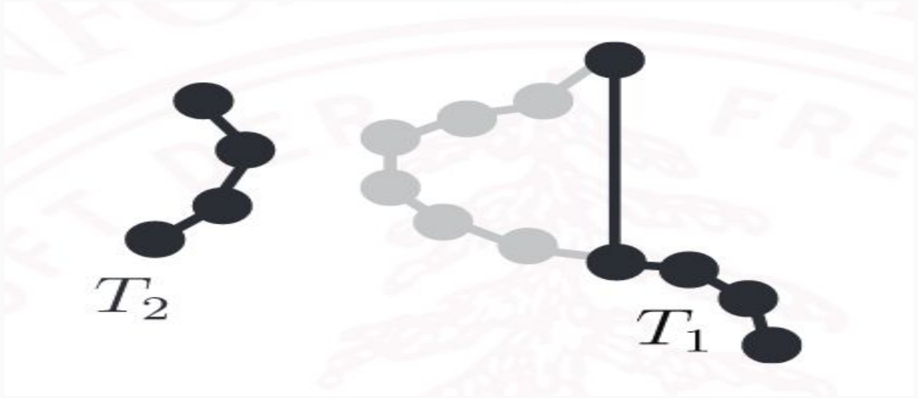
$$P(T) = P_s(x_1) \left(\prod_{n=1}^{N-1} P(x_{n+1}|x_n) \right) P_t(x_N) \qquad (2)$$

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

Modelling Occlusion

- Tracks can be made of non consecutive frames.
- An upper limit is set on the number of frames that can be skipped to deal with time complexity.



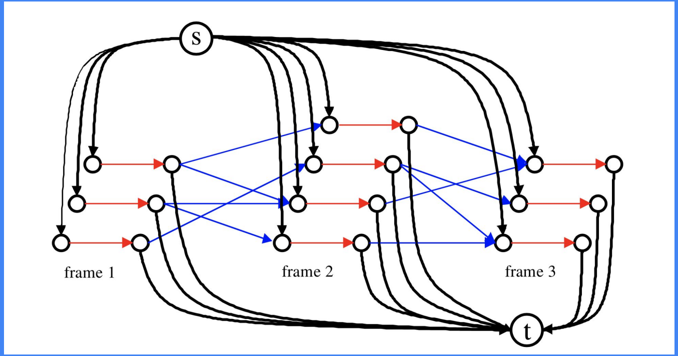
MAP inference

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

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

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

MAP to ILP



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

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

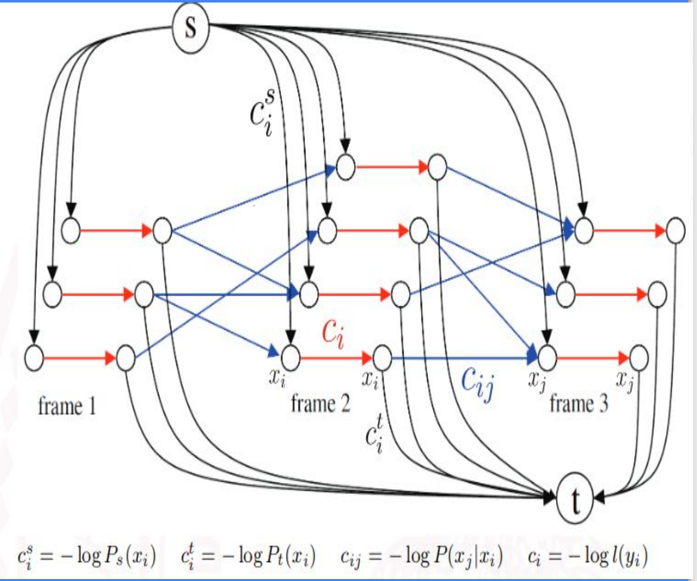
$$\text{s.t.} \quad f_{ij}, f_i, f_i^s, f_i^t \in \{0, 1\}$$

$$\text{and} \quad f_i^s + \sum_j f_{ji} = f_i = f_i^t + \sum_j f_{ij} \tag{8}$$

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

$$F_K = \left\{ \begin{array}{c} f_{ij}, f_i, f_i^s, f_i^t \in [0, 1], \quad \sum_i f_i^s = K, \\ f_i^s + \sum_j f_{ji} = f_i = f_i^t + \sum_j f_{ij}, \quad \sum_i f_i^t = K \end{array} \right.$$

ILP to network flow



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

- Create a pair of vertices for every detection in a frame.
- Add an edge from the source to the first vertex
- Add an edge from the second vertex to the sink.
- Add an edge between the 2 vertices which represents the cost of using that detection in the track
- Add edges to vertices of future frames to model the transitions between frames
- All edges have unit weight, but a certain cost to pick the edge

Sending a flow of "K" from the source to sink on this graph corresponds to having "K" tracks. And picking the flow with minimum cost corresponds to a globally optimal solution.

Effectively, the problem has now been reduced to a min cost flow problem.

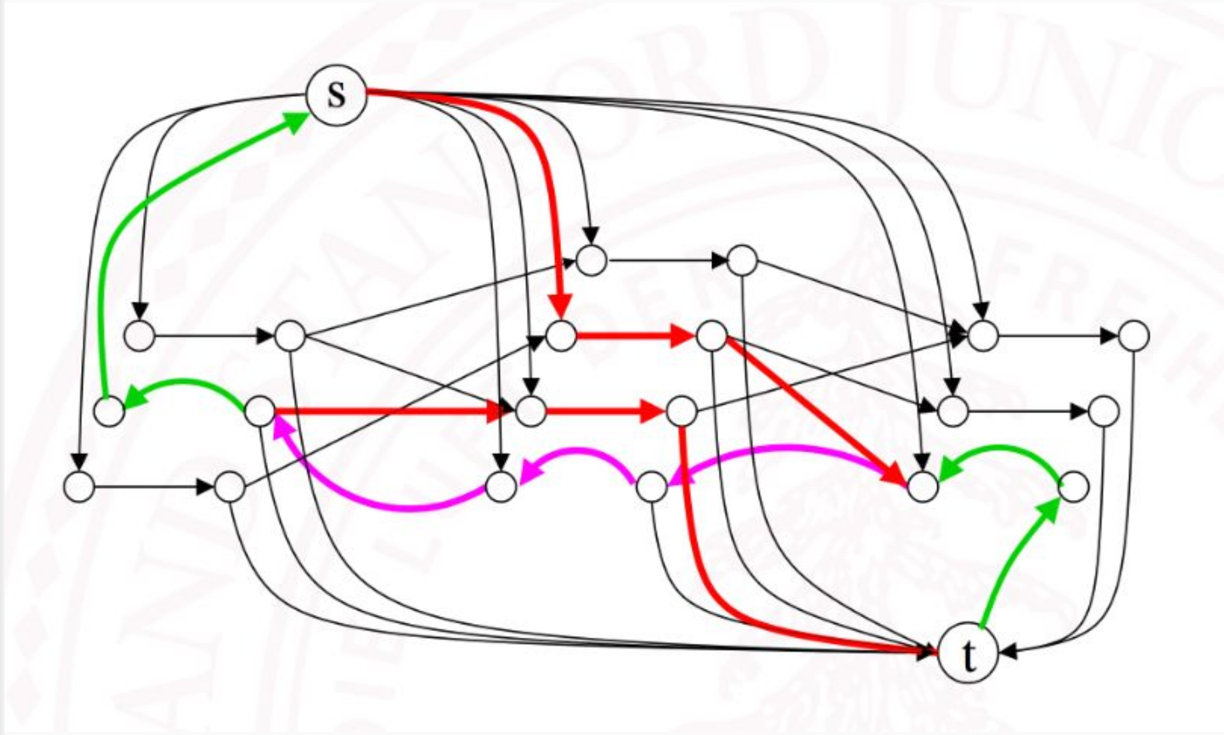
Since the graph has edges of unit capacity and is a DAG, we can solve this problem efficiently.

Solving using Successive Shortest Paths

1. Find minimum cost from s to t in the residual graph
2. If the cost of this path is negative, update the residual graph by pushing a unit flow along this path.

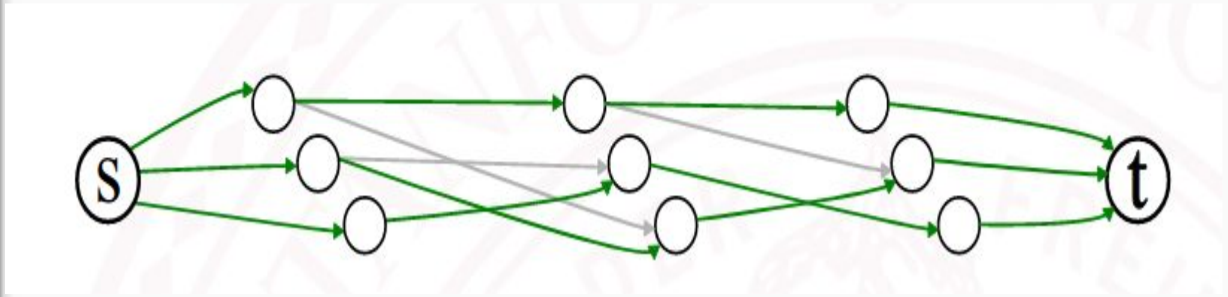
This algorithm can be done in O(KNlogN) using Dijkstra's algorithm for finding the shortest paths.

However, we can come up with a simpler solution for this Problem!



Solving for K=1 using Dynamic Programming

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



Approximate Solution for K>1

1. Use the previous dynamic programming algorithm to compute the shortest path
2. Remove the edges of the path and repeat
3. Stop when the overall cost increases

Experimental Details

- Caltech Pedestrian Dataset: 71 videos with 1800 frames at 30 fps.
- ETHMS Dataset: 4 videos 1000 frames at 14fps
- Generate the bounding boxes(vertices of the graph) using HOG based pedestrian detector from OpenCV
- Evaluation Metrics: Detection Rate (Number of Correct ID Labelings/Total no. of ID Labelings) and False Positives Per Frame(Total No. of False Positives/No. Of Frames)

Further Scope

Implement "Joint Tracking and Segmentation (CVPR 2015)" if time permits.

The key addition in this new work is to use superpixel level information in addition to the bounding boxes. Instead of formulating an ILP, a Conditional Random Field (CRF) is used and is solved using the alpha-expansion algorithm.