Problem input:
- Detector scores for all space time windows in a video

Problem output:
- Number of objects (tracks)
- Starting and ending frames for each track
- Tracks themselves

Our contributions:
- Globally Optimal (for a common class of objective functions)
- Locally Greedy (and hence straightforward to implement)
- Scale linearly in the number of objects and (quasi)linearly with video-length

Prior

Observation:

Likelihood:

Model:

A detection window (position, scale, time): \( x = (p, s, t) \) \( x \in V \)

A track: \( T = \{T_1, \ldots, T_N\} \) and a collection of tracks: \( X = \{X_1, \ldots, X_M\} \)

Prior:

\( \text{Likelihood: } P(Y|X) = \prod_{T \in X} P_T(Y_T) \prod_{i \in \mathcal{V}} P_{\theta_i}(y_{\theta_i}) = \sum_{x \in X} \prod_{T \in X} P_T(x_T) \prod_{i \in \mathcal{V}} l_{\theta_i}(y_{\theta_i}) \)

MAP inference:

\( X = \arg \max \prod_{T \in X} P_T(x_T) \prod_{i \in \mathcal{V}} l_{\theta_i}(y_{\theta_i}) \)

\( X = \arg \max \sum_{x \in X} \prod_{T \in X} P_T(x_T) \prod_{i \in \mathcal{V}} l_{\theta_i}(y_{\theta_i}) \)

Equivalent graph problem: Min-cost flow

\( \text{Solutions: } \)

- Globally optimal
  - Push-relabel algorithm
    - Used in Zhang et al CVPR’08
    - Solves for known \( d \); Does binary search to find \( K=\text{optimum } d \)
  - Successive shortest path
    - Special structure in our graph (DAG, unit-capacity)
    - Is greedy
  - Approximate
    - Dynamic programming
      - \( O(N^2 K) \)

Successive shortest path:

- At each iteration, find the shortest path and augment the graph by reversing edges along it.
- Iterate as long as the cost of shortest path is positive.

Dynamic programming:

- Sweep the graph and update paths to find the shortest one
- Variable length DP because of “s” and “f” nodes
- Is optimum for \( d = 1 \) since the graph is DAG.
- Is not optimum for next iterations \( d > 1 \)
- Ignore backward edges: works since the optimum solution rarely uses them.
- 2-pass DP: Run one more DP along backward edges
  - Is not optimum because of very rare cases.
  - Can do multi-pass DP

Caching:

- 1000X faster:
  - Use shortest paths from previous iterations. Keep track of the origin for each path
  - After instancing a track, update only paths which share the same origin.

Non-max-suppression in the loop:

- At each iteration, suppress all windows overlapping with the instanced track.

Handling occlusion:

- Adding more than one-frame jumps to transition edges.

Datasets:

- Caltech Pedestrian Dataset and ETHMS Pedestrian Dataset

Comparing objective function for all three algorithms: (on Caltech dataset)

Comparison with the related work on ETHMS dataset

Conclusion:

- Novel scalable, greedy algorithm for multi-object tracking
- Scalable: Process large inputs (millions of detections) and model long occlusions
- Greedy: Embed pre-processing steps (NMS) within algorithm.
- State-of-the-art results on benchmark data