Computer Vision
Project Report-1
Globally-Optimal Greedy Algorithms for Tracking a Variable Number of Objects

Shyamgopal Karthik (**20161221**)
Mohsin Mustafa (**20161131**)
Ramkishore Saravanan (**20161092**)

Github Link: `https://github.com/FloatingFowl/TrackerMan`

Introduction

The problem of multi-object tracking focuses on tracking a variable number of objects in a video sequence. Unlike the problem of single object tracking, multi object tracking(as well as most datasets prepared for it) focuses on tracking objects of a particular category(such as pedestrians or cars). The general idea to solve this problem is to generate detections throughout the video and then label each detection whether they belong to a track and then assign a label the track id it belongs it. We study and implement the formulation of this problem as a min cost flow problem [3] and see an efficient solution to this problem using the successive shortest paths approach which can solve this problem in $O(KN^2)$. Further, we also look at an approximate dynamic programming approach which solves this problem in $O(KN)$ [2]

Modeling the problem as a Directed Acyclic Graph

The first assumption that significantly simplifies this problem is the Markovian assumption. We assume the cost of a choosing a particular detection in a track is the sum of the cost of that detection and the pairwise cost of that detection and the detection in the previous frame. Mathematically, this can be represented as:

$$E(x) = \sum_t U(x_t) + P(x_t, x_t - 1)$$

where U is the unary cost of the detection x_t and P is the pairwise cost for the detections x_t and x_{t-1} To extend this formulation for K tracks, we formulate the problem as follows:

$$argmin_{x_1 \ldots x_k} \sum_k E(x_k)$$

The solve this problem, we need to infer the value of K, find the track births and deaths for each of the K tracks and solve the data association problem. This Markov Model can be formulated as a MAP(Maximum A Posteriori) inference problem which can further be cast as an ILP(Integer Linear Program). On performing the LP Relaxation, it can be shown that the solution obtained would still be optimal as the constraints are unimodular. However, the focus here is to formulate the problem and the constraints into a graph.

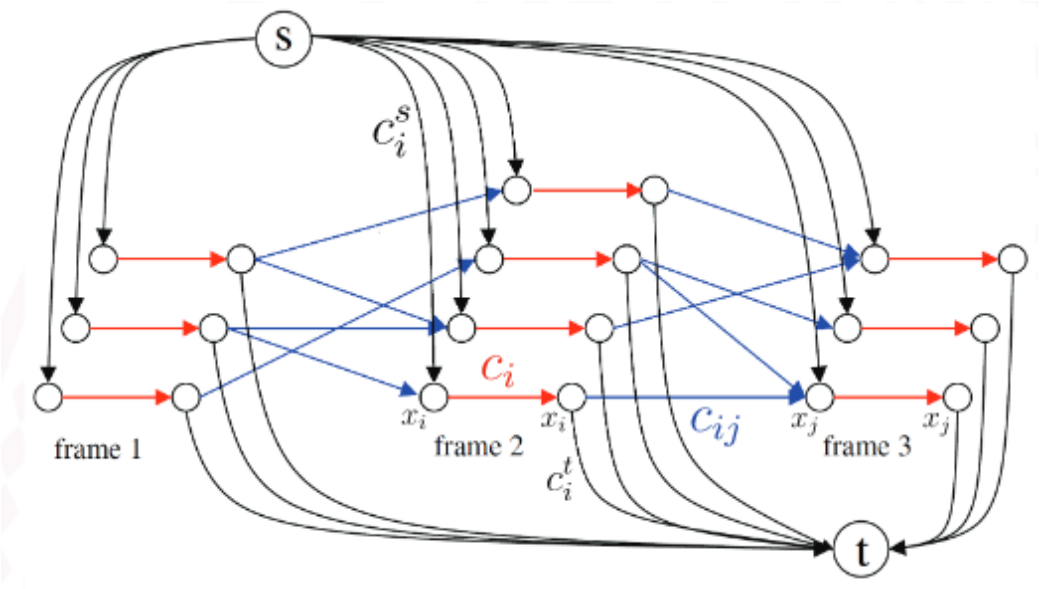

Figure 1: Modeling the tracking problem as a min cost flow graph problem

The formulation of this problem is shown in Fig.1 The structure of the graph can be explained as follows:

- Each detection in a frame corresponds to 2 nodes which are connected by an edge of weight is the negative log likelihood of the detection. Effectively it is the negative of the strength of the detection. This models the unary potentials in the energy function.

- There is an edge from the source node to the first of the 2 nodes corresponding to each detection with a fixed cost c_{en}. This is to model the track birth.
- There is an edge of fixed cost c_{ex} from the second of the 2 nodes corresponding to each detection towards the sink node. This serves to model the track death.
- There is an edge from the second node of a detection in frame i to the first node of a detection in $i+1$ if they have a spatial intersection. The cost of this edge is zero. These edges are to model the pairwise transition potentials.
- While all edges have different costs, they have unit capacity. That is, they can carry one unit of flow through them while having different costs to carry the same flow.

The intuition behind constructing the graph is that pushing K units of flow along this graph corresponds to having K tracks or tracking K pedestrians in the video sequence. However, we are not just looking to push K units of flow into the graph. We are looking to push K units of flow while having minimum cost. Therefore, this problem reduces to the min cost flow problem which is similar to max flow problem in terms of the techniques used to solve the problem.

However, we are not given K apriori. Instead, we need to estimate K. A good approximation that works here is to maximize K such that the cost of the flow is less than a fixed threshold. Here, the weights of the graph have been engineered to have zero as the threshold. Mathematically, we can represent this as:

$$\begin{aligned} \underset{K}{\text{maximize}} \quad & K \\ \text{subject to} \quad & \sum_{i=1}^{K} E(x_i) < 0 \end{aligned} \tag{1}$$

Modeling and Tracking in the presence of Occlusions

Solving the min cost flow problem

Successive Shortest Paths

While we could use a solver to solve the min cost flow problem, there exists a far more simpler solution to this problem namely the idea of successive

shortest paths. A key idea in flow problems right from Ford Fulkerson's, Edmond Karp or Dinic's algorithm is to find a "good" path, push the maximum possible flow along that path, update the graph(which is generally called a residual graph) until no more flow can be pushed through the network. We can find the shortest paths either using Dijkstra's algorithm or using the Bellman-Ford algorithm and then push unit flow along it and then update the residual graph. This can be done in $O(K * N^2)$ if we use Bellman Ford or in $O(K * NlogN)$ if we use Dijkstra's algorithm. This algorithm is illustrated in Fig. 2

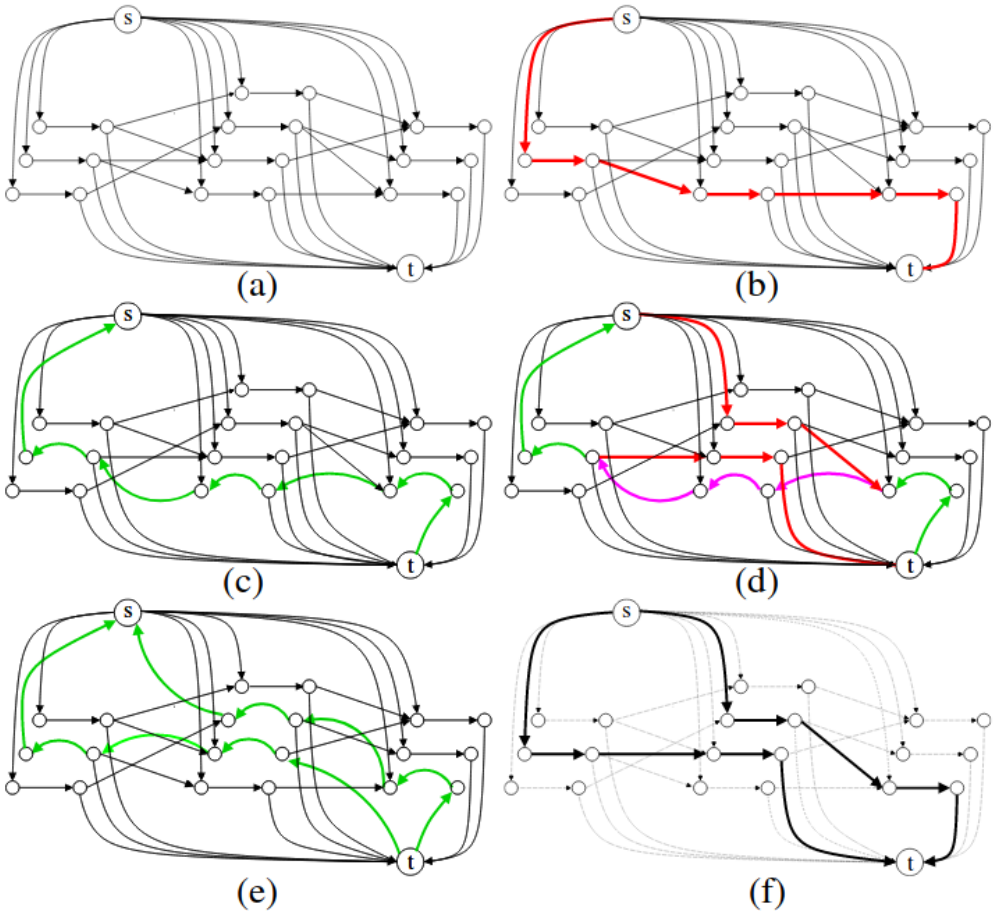

Figure 2: Solving the Problem with Successive Shortest Paths

Solving the min cost flow using Dynamic Programming

Exact Solution for K=1

For the case of 1 path, this problem reduces to the problem of finding a shortest path from the source to sink in a DAG(Directed Acyclic Graph). This can be done using Dynamic Programming. For each node i, we can initialize $cost(i) = c_i + c_i^s$ where c_i is the negative of the strength of the detection while c_i^s is the cost of the edge from the source to i. Now we formulate the DP as follows:

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

where $N(i)$ is the set of detections which are connected to the detection i with a single edge and c_{ij} is the transition cost.

Approximate Solution for K>1

To extend this for $K > 1$ we can do the following:

- Find the shortest path from s to t using the algorithm presented previously.
- Remove the nodes and repeat until the cost of the path remains negative

This runs in $O(NK)$ unlike the Successive Shortest Paths algorithm which either runs in $O(NK^2)$ or $O(NlogN)$ However while being much faster, this is misses the crucial insight exploited by the Successive Shortest Paths algorithm. The idea of having a residual graph and introducing back edges is that pushing any flow along the back edge effectively amounts "editing" the paths previously chosen. This is the reason why Successive Shortest Paths based flow algorithms return the optimal solution. However, introducing back edges violates the DAG property and therefore the Dynamic Programming algorithm would not work in that case.

Improved Approximate Solution for K>1

To improve the performance of the approximate Dynamic Programming solution, we another pass. The idea is to have a forward pass to compute $cost(i)$ as shown before. However, we also add a backward pass to compute the cost in that direction for each node as well and the $cost'(i)$ is the sum of the costs. To also not have to recompute $cost(i)$ for every iteration, since many of them would remain unchanged over time, we can also make use of some message passing techniques.

Improving Performance using Non-Max Suppression

After computing a track in an iteration of the DP, we can suppress all the detections which have high overlap with the detections chosen to form the track. This can be done in the DP, however cannot be done in the flow algorithms as this would be updating the graph on the fly. This trick can both improve performance and speed. This is illustrated in Fig.3

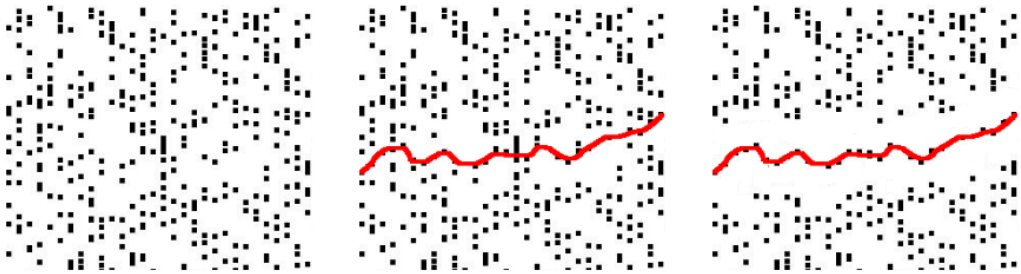

Figure 3: DP+NMS

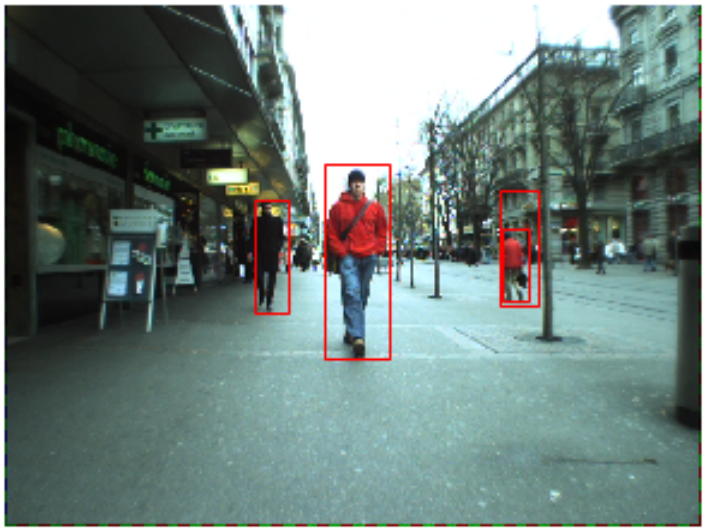

Figure 4: Detections on a sample frame

Figure 5: Tracking using DP for K=1

Experimental details and tasks implemented so far

We use HOG based pedestrian detector to obtain the detections. The graph is constructed as explained below. This is shown in Fig. 4

We finished implementing the Approximate $K > 1$ DP (Fig. 5) and our sample results can be seen for K=1 `https://github.com/FloatingFowl/TrackerMan/blob/master/README.md` and K>1 `https://youtu.be/K1Uouo-hXp0` We tested our implementation on the ETHMS [1] dataset which has a moving camera mounted on a stroller. However, benchmarks, detailed results and metrics shall be given for the final evaluation.

Figure 6: Tracking using DP for K>1

References

[1] Andreas Ess, Bastian Leibe, and Luc Van Gool. "Depth and appearance for mobile scene analysis". In: *2007 IEEE 11th International Conference on Computer Vision*. IEEE. 2007, pp. 1–8.

[2] Hamed Pirsiavash, Deva Ramanan, and Charless C Fowlkes. "Globally-optimal greedy algorithms for tracking a variable number of objects". In: *CVPR 2011*. IEEE. 2011, pp. 1201–1208.

[3] Li Zhang, Yuan Li, and Ramakant Nevatia. "Global data association for multi-object tracking using network flows". In: *2008 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2008, pp. 1–8.