K-Shortest Path algorithm for the Multi-Modal Probabilistic Occupancy Maps

We present here the algorithm we use to reconstruct tracks from the Multi-Modal Probabilistic Occupancy Maps (MMPOMs) of Section 7.2.

KSP In the original algorithm of [1], the POMs computed at successive instants were used to produce consistent trajectories using the a K-Shortest Path (KSP) algorithm [3]. This involves building a graph in which each ground location at each time step corresponds to a node and neighboring locations at consecutive time steps are connected. KSP then finds a set of node-disjoint shortest paths in this graph where the cost of going through a location is proportional to the negative log-probability of the location in the POM [1]. The KSP problem can be solved in linear time and an efficient implementation is available online.

KSP for Multi-Modal POM Since MMMF produces multiple POMs, one for each mode, at each time-step, we duplicate the KSP graph nodes, once for each mode as well. Each node is then connected to each copy of neighboring locations from previous and following time steps. We then solve a multiple shortest-path problem in this new graph, with the additional constraint that at each time step all the paths have to go through copies of the nodes corresponding to the same mode. This larger problem is NP-Hard and cannot be solved by a polynomial algorithm such as KSP. We therefore use the Gurobi Mixed-Integer Linear Program solver [2].

More precisely, let us assume that we have a sequence of Multi-Modal POMs Q_k^t and mode probabilities m_k^t for $t \in \{1, \dots, T\}$ representing time-steps and $k \in \{1, \dots, K\}$ representing different modes. Each Q_k^t is materialized through a vector of probabilities of presence $q_{k,i}^t$, where each $i \leq N$ is indexes a location on the tracking grid.

Using the grid topology, we define a neighborhood around each variable, which corresponds to the maximal distance a walking person can make on a grid in one time step. Let us denote by $\mathcal{N}_i$ the set of indices corresponding to locations in the neighbourhood of i. The topology is fixed and hence $\mathcal{N}_i$ does not depend on the time steps. We define the following log-likelihood costs.

Using a Log-Likelihood penalty, we define the following costs:

- $C_{k,i}^t = \log\left(\frac{1 - q_{k,i}^t}{q_{k,i}^t}\right)$, representing the cost of going through variable i at time t if mode k is chosen.
- $C_k^t = \log\left(\frac{1 - m_k^t}{m_k^t}\right)$, representing the cost of choosing mode k at time t.

We solve for an optimization problem involving the following variables:

- $x_{k,i,l,j}^t$ is a binary flow variable that should be 1 if a person was located in i at t and moved to j at $t+1$, while modes k and l were respectively chosen at time t and $t+1$.

- y_k^t is a binary variable that indicates whether mode k is selected at time t.

We can then rewrite the Multi-Modal K-Shortest Path problem as the following program, were we always assume that $t \leq T$ stands for a time step, $k \leq K$ and $l \leq K$ stand for mode indices, and $i \leq N$ and $j \leq N$ stand for grid locations:

$$
\begin{aligned}
\min \quad & \sum_{t,k} C_k^t y_k^t + \sum_{t,k,l \leq K} \sum_{i,j \in \mathcal{N}_i} C_{k,i}^t x_{k,i,l,j}^t & \\
\text{s.t.} \quad & \forall (t,k,i)\ , \quad \sum_{l,j \in \mathcal{N}_i} x_{l,j,k,i}^{t-1} = \sum_{l,j \in \mathcal{N}_i} x_{k,i,l,j}^t & \texttt{flow conservation} \\
& \forall (t,k,i)\ , \quad \sum_{l,j \in \mathcal{N}_i} x_{k,i,l,j}^t \leq y_k^t & \texttt{disjoint paths + selected mode} \\
& \forall t\ , \quad \sum_k y_k^t = 1 & \texttt{selecting one mode} \\
& \forall t,k,i,l,j\ , \quad 0 \leq x_{k,i,l,j}^t \leq 1 & \\
& \forall t,k\ , \quad y_k^t \in \{0,1\} &
\end{aligned}
\tag{1}
$$

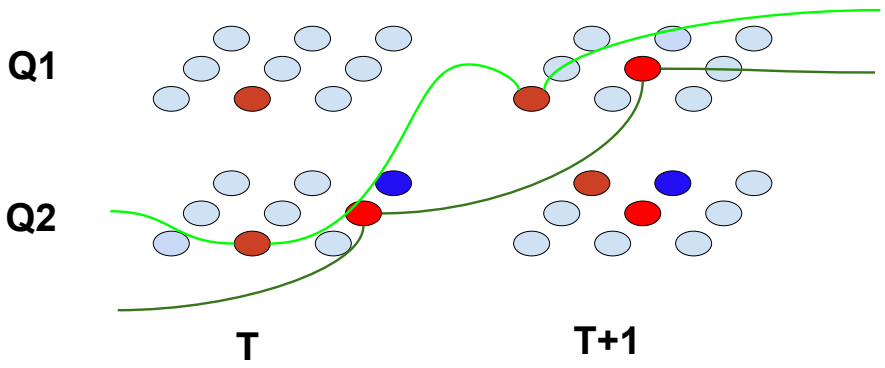

Figure 1: Illustration of the output of our K-Shortest Path algorithm in the case of multiple modes.

KSP prunning However, the problem as written above, may involve several tens millions of flow variables and therefore becomes intractable, even for the best MILP solvers. We therefore first prune the graph to drastically reduce its size.

The obvious strategy would be by thresholding the POMs and removing all the outgoing and incoming edges from locations which have probabilities below q_{thresh}. However, this would be self-defeating as one of the main strengths of the KSP formulation is to be very robust to missing-detections and be able to reconstruct a track even if a detection is completely lost for several frames.

We therefore resort to a different strategy. More precisely, we initially relax the constraint `disjoint paths + selected mode`, to a simple disjoint path constraint, and remove the constraint `selecting one mode`. We therefore obtain a relaxed problem

$$
\begin{aligned}
\min \quad & \sum_{t,k} \sum_{t,k,l \leq K} \sum_{i,j \in \mathcal{N}_i} C^t_{k,i} x^t_{k,i,l,j} \\
\text{s.t.} \quad & \forall (t,k,i) \ , \quad \sum_{l,j \in \mathcal{N}_i} x^{t-1}_{l,j,k,i} = \sum_{l,j \in \mathcal{N}_i} x^t_{k,i,l,j} \qquad \texttt{flow conservation} \\
& \forall (t,k,i) \ , \quad \sum_{l,j \in \mathcal{N}_i} x^t_{k,i,l,j} \leq 1 \qquad \texttt{disjoint paths} \\
& \forall t,k,i,l,j \ , \quad 0 \leq x^t_{k,i,l,j} \leq 1
\end{aligned}
\tag{2}
$$

which is nothing but a vanilla K-Shortest Path Problem. It can be solved using our linear-time KSP algorithm. This KSP problem will output a very large number of paths, going through all the different modes simultaneously. From, this output, we extract the set of grid locations which are used, in any mode, at each time step, and select them as our potential locations in the final program. In our current implementation, we add to these locations, the ones for which $q^t_{k,i} \geq q_{thresh}$ for any mode at time-step t.

We can finally solve Program 1, where non-selected locations are pruned from the flow graph. We don't know if our strategy, based on a relaxation and pruning, provides a guaranteed optimal solution to 1, but this is an interesting question.

References

[1] J. Berclaz, F. Fleuret, E. Türetken, and P. Fua. Multiple Object Tracking Using K-Shortest Paths Optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(11):1806–1819, 2011. 1

[2] I. Gurobi Optimization. Gurobi optimizer reference manual, 2016. 1

[3] J. W. Suurballe. Disjoint Paths in a Network. *Networks*, 4:125–145, 1974. 1