On the Graph Building Problem in Camera Networks [⋆]

Angelo Cenedese [*] **Riccardo Ghirardello** [**] **Roberto Guiotto** [**] **Fabio Paggiaro** [**] **Luca Schenato** [**]

[*] *Dept. of Engineering and Management, University of Padova, Italy.*
[**] *Dept. of Information Engineering, University of Padova, Italy.*

Abstract: In this paper, the problem of building a model of a sensor (camera) network from observations is considered. By model, we mean a graph where the nodes represent states that are observable and distinguishable by the sensor network and edges are the feasible transitions among these states: the edges are also weighted by the probability of transition from one state to another. Remarkably, since merely static observations are not sufficient to discern all states in the networked system, the dynamics of transition is also considered. In this respect, the proposed graph model appears falling into the class of hidden Markov models, where the discover of hidden states is made possible by exploiting the temporal evolution of the transitions and the implementation of a splitting procedure of previously identified graph nodes.

Keywords: Graph models for networks; Sensor networks; Camera networks; Hidden Markov Models; Model Identification; Model Optimization.

1. INTRODUCTION

In a distributed scenario, the knowledge of the topology of the network by the acting agents is of paramount importance: in fact, the agents have to share information locally and need to coordinate with neighbors to attain the global performance. While in general this represents an advantage of the distributed architecture in terms of parsimony of computational and communication resources, it also becomes a need in situations where hundreds of nodes are spread across the environment. In this situation, in fact, it is practically impossible to have a manual setup of the network, and at the same time it is not convenient to rely on a topology that is a priori defined during the design phase. There is therefore the necessity of learning the topology of the network after it has been deployed in the environment. A practical example of such situation is that of the control room of a surveillance network of hundreds of sensors (cameras). The real world scenario refers to monitoring vast indoor or outdoor area (e.g. malls, natural or amusement parks, city neighborhoods, docks and large industrial sites, airport terminals, university campuses) with Pan-Tilt-Zoom (PTZ) or fixed cameras. There is a feasible possibility of visualizing only a subset of these signals, and the human operator attention can focus only on an even smaller set of information. In particular, once an event of interest has been localized in a specific area, it would be interesting to know which are the locations where the same event is more likely to appear in the near future, or better which are the sensors (cameras) that can provide information on this movement. Therefore, the problem consists in understanding how the sensor scene relate to the environment topology. In the case of camera networks, the sensor scene is meant as the whole of visibility areas obtained through the union of the camera fields of view. We refer to this problem as the *graph building problem*, since the solution is given in terms of a graph where each node represents a "sensor area" and each edge stands for an admissible physical transition from one area to another. In particular, the aim is to associate to each graph edge a probability of transition that describes the possibility of the event to move from the current position. The estimation of the graph structure in terms of both nodes (i.e. areas of interest) and edges (i.e. the transition probability map) has to be made from the information obtained from camera observations, during the calibration phase of the system. In this phase, with reference to the camera network application, we suppose to have a known target that moves across the scene. The aim is to devise a procedure that takes the observations in input in order to identify an initial model able to describe the visible areas and the adjacency relationship among them, i.e. the probability of transition from one to the others. Then, while more observations are received by the network, the system should learn how to adaptively correct the initial model making it in some sense increasingly homeomorphic to the monitored environment (as far as this is possible, w.r.t. unobservable states).

2. STATE OF ART

The general subject of multicamera networks is widely treated in the scientific literature (e.g. [Javed et al. (2003)], [Collins et al. (2001)], and references within). The focus, though, is often on the target matching problem (e.g. [Chilgunde et al. (2004)]), after the event of interest have been detected in more than one field of view, and is based on several approaches to feature matching among different views. Differently, the problem addressed in this work regards the topological association of the fields of view themselves, and is related more to the structure of the camera network than to the dynamic instances that occur in the scene. More in detail, this issue can be seen from a theoretical point of view as a model selection problem. In this context, graphical models, "a marriage between probability theory and graph theory" [Jordan (1999)], are a popular research subject [Bishop (2006)][Wainwright and Jordan (2008)] where the problem of fitting and selecting models on the basis of data is tackled for example by

[⋆] The research leading to these results has received funding from the European Community's Seventh Framework Programme under agreement n. FP7-ICT-223866-FeedNetBack.

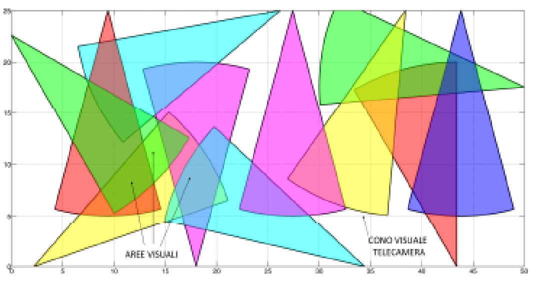

Fig. 1. A simplified camera network scenario. Camera fov's provide the observations to identify system states.

resorting to graph based structures (Markov random field) and to the formulation of a maximum likelihood optimization problem with a regularization term. An in principle similar approach from the application perspective is given in [Vasquez et al. (2009)] where a growing Hidden Markov Model is capable of learning incrementally the model and is proposed to study and predict the motion of vehicles and pedestrians: this work makes use of tools of the same kind than those developed for our case study. A graphical model is also used in [Farrell et al. (2007)] for multicamera surveillance systems, where a dynamic Bayesian network is used to learn the model parameter incrementally by accumulating observations on the tracked object trajectories. In [Ellis et al. (2003)] the problem of a multicamera framework is posed, where the aim is to establish correspondences among neighboring fields of view, for dynamic object tracking and to estimate the periods of blind-view due to the presence of occlusions. The approach is based on the identification of the entrance/exit points in each field of view and in studying the temporal correlation of objects as seen by different cameras.

3. FORMALIZATION OF THE PROBLEM

3.1 Work Hypotheses and Camera Model

In this work, without loss of generality, a simplified scenario is considered, where the monitored area is a two-dimensional domain Ω of $25 \times 50m^2$ and $K = 11$ fixed cameras $\{\mathcal{A}_i, i = 1, \dots, K\}$ are positioned around the perimeter, facing inwards with circular sector field of view (fov) Δ_i, as shown in Fig. 1. Other camera parameters of interest are known and set as follows:

- focal length $F = 20m$;
- fov angle $\theta = 30°$;
- frame rate $f = 25fps$.

Conversely, the camera position and orientation are unknown, and also the shape of the fov overlapping regions is a priori not known. The states of the system are the visible areas obtained by considering all overlapping and non overlapping fov regions, in addition to the (null) state corresponding to no observation and region $S_0 = (\Omega \setminus \bigcup \Delta_i)$.

3.2 Rationale

In this context, we are not interested in solving the computational vision problem of *how* the camera sees the object of interest, but we are interested in *if* the camera sees it. The information datum (observation) at time t is given as a binary string $O_t \in \{0,1\}^K$ whose entry in the i-th position is 1 if the i-th camera sees the target object, while it is 0 otherwise [1]. Since this binary observation represents a signature of the states, different observations state that the corresponding states of the system are distinguishable.

Therefore, the problem of graph selection can be formulated as follows: given an observation sequence $\mathcal{O}$ in the finite interval $[1, T]$, infer the set of states $\mathcal{S}$ and the underlying graph $\mathcal{G}$ that constraints their transitions (and whose node set is $\mathcal{S}$).

In the adopted formulation, the state graph construction follows a two step approach: firstly, a strong correspondence between states and (static) observations is enforced and for every different observation at time t a state is generated and a transition probability is evaluated. Then, some of these states undergo a splitting procedure, giving rise to the revelation of initially hidden states, and in this ways being replaced by two or more novel states: this procedure is attained exploiting the dynamics of the observation by considering not only present but also past and future values.

3.3 Hidden Markov Models

Hidden Markov Model (HMM) [Rabiner (1989)][Rabiner and Juang (2003)] is the model that better describes the problem we are dealing with, being actually characterized by the fact that the state is not known a priori but need to be inferred from observations.

An HMM is characterized by the following:

- the number N of the states of the model (although they are hidden, usually their number is known); we denote the state set as $\mathcal{S} = \{S_1, S_2, \dots, S_N\}$ and the state at time t as $q_t \in \mathcal{S}$.
- the number M of distinct observation symbols, $M \leq 2^K$. This dictionary is denoted by $\mathcal{V} = \{v_1, v_2, \dots, v_M\}$, being $v_i \in \{0,1\}^K$; an (ordered) sequence of observations in the interval $\mathcal{T} = [1, T] \subset \mathbb{Z}_+$ is given by $\mathcal{O} = [O_1, \dots, O_T]$, $O_i \in \mathcal{V}$;
- the state transition probability distribution $A \in \mathbb{R}^{N \times N}$ whose elements are
$$a_{ij} = \mathbf{P}[q_{t+1} = S_j | q_t = S_i] \qquad 1 \leq i, j \leq N\,, \qquad (1)$$
with $0 \leq a_{ij} \leq 1$ and $\sum_{j=1}^{N} a_{ij} = 1$;
- the observation symbol probability distribution $B \in \mathbb{R}^{M \times N}$, whose elements $\{b_{ij}\}$ are
$$b_j(v_i) = \mathbf{P}[O_t = v_i | q_t = S_j] \qquad 1 \leq i \leq M\,, \quad 1 \leq j \leq N\,, \qquad (2)$$
with $0 \leq b_j(v_i) \leq 1$ and $\sum_{i=1}^{M} b_j(v_i) = 1$ [2] ;
- the initial state distribution $\pi \in \mathbb{R}^N$, whose elements are
$$\pi_i = \mathbf{P}[q_1 = S_i] \qquad 1 \leq i \leq N\,, \qquad (3)$$
$\sum_{i=1}^{N} \pi_i = 1\,.$

Given the state space $\mathcal{S}$, the Markov model is the triple $\lambda = (A, B, \pi)$, and a directed graph $\mathcal{G}$ is associated to the related Markov chain.The event $[q_t = S_i]$ represents the fact that the chain is in the state S_i at time t, while the event $[q_t = S_j | q_{t-1} = S_i]$ represents the transition from the state S_i to the state S_j. In other words, $\mathcal{G}$ is obtained as follows:

[1] This quite simplifying assumption appears reasonable as the setup of complex multicamera system may boil down to the mere detection of a known target as the input of some indicator function.
[2] This condition means that being in a state S_j generates *an* observation with probability 1.

(1) for every state S_i in the model a graph node is set;
(2) for each pair of nodes (S_i, S_j) in the graph, the directed arc between the two is labeled with the transition probability a_{ij}.

3.4 Objective and Design Strategy

At this stage, it is useful to point out how the work rationale of Subsec. 3.2 is embodied by the Markov modeling just presented. In this respect, the following procedure is implemented:

(1) a camera configuration is setup and a trajectory in the Ω domain is considered, to provide observation data;
(2) a first topological graph is built directly as a first guess Markov model, in terms of nodes (states) and edges (state transitions);
(3) the refinement of the model is obtained through an iterative procedure according to some norm, updating the model parameters to better fit the observation sequence;
(4) the identification of hidden states is then carried out by observing the evolution in time of the trajectory.

This final model should provide a better description of data, in the sense of prediction of the trajectory used for the graph construction. In this respect, given the observation O and its estimate $\hat{O}$ the probability that they coincide is considered as the utility function:

$$\max \mathbf{P}[\hat{O}_{t+1} = O_{t+1} | O_{1:t}] \,. \tag{4}$$

4. GRAPH BUILDING ALGORITHM

Given an initial model,$\lambda_0 = (A, B, \pi)_0$, one canonical problem for HMMs is to determine a method to adjust the model parameters $\{A, B, \pi\}$ to maximize the probability of the observation sequence $\mathcal{O}$ given the model:

$$\max_{\lambda} \mathbf{P}[\mathcal{O}|\lambda]. \tag{5}$$

In this context, we make use of the Baum-Welch algorithm [Baum et al. (1970)][Welch (2003)] to find and adjust the unknown parameters of the HMM of interest. The algorithm refers to expectation-maximization methods to compute posterior estimates for the model parameters, based on a maximum likelihood approach. In order to do so, we first introduce some preliminary concepts in the following.

To compute the probability of the observation sequence, given the model, $\mathbf{P}[\mathcal{O}|\lambda]$, the forward-backward procedure is employed [Baum et al. (1970)].

Be the observation interval $\mathcal{T} = [1, T]$: the forward variable $\alpha_t(i)$ is defined for $t \in \mathcal{T}$ as

$$\alpha_t(i) = \mathbf{P}[O_1, O_2, \dots, O_t, q_t = S_i|\lambda] \,, \tag{6}$$

i.e. it is the probability of the partial observation sequence $[O_1, O_2, \dots, O_t]$ and state S_i at time t, given the model λ.

The solution can be found inductively for $t \in [1, T-1]$

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij} \right] b_j(O_{t+1}) \quad 1 \le j \le N \,, \tag{7}$$

with initialization as

$$\alpha_1(i) = \pi_i b_i(O_1) \,, \qquad 1 \le i \le N \,. \tag{8}$$

Similarly, the backward variable $\beta_t(i)$, $t \in \mathcal{T}$, is defined as

$$\beta_t(i) = \mathbf{P}[O_{t+1}, O_{t+2}, \dots, O_T | q_t = S_i, \lambda] \,, \tag{9}$$

i.e. it is the probability of the partial observation sequence in the interval $[t+1, T]$, given the state S_i at time t and the model λ. Again, we can solve it inductively for $t \in \{T-1, T-2, \dots, 1\}$ as

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(O_{t+1}) \beta_{t+1}(j) \quad 1 \le i \le N \,, \tag{10}$$

with initialization as

$$\beta_T(i) = 1 \,, \qquad 1 \le i \le N \,. \tag{11}$$

Finally, $\gamma_t(i)$, $t \in \mathcal{T}$, is defined as the probability of being in state S_i at time t, given the observation sequence and the model:

$$\gamma_t(i) = \mathbf{P}[q_t = S_i|\mathcal{O}, \lambda]. \tag{12}$$

Remarkably, the probability of the whole observation sequence $\mathcal{O}$ is given in terms of forward and backward variables either by summing over the state space the terminal variables $\alpha_T(i)$, or, alternatively, by summing over the state space the partial observation variables $\alpha_t(i)$ multiplied by the correspondent $\beta_t(i)$ accounting for the remainder of the observation interval:

$$\mathbf{P}[\mathcal{O}|\lambda] = \sum_{i=1}^{N} \alpha_T(i) = \sum_{i=1}^{N} \alpha_t(i)\beta_t(i) \,. \tag{13}$$

Also, $\gamma_t(i)$ relates to $\alpha_t(i)$ and $\beta_t(i)$, being:

$$\gamma_t(i) = \mathbf{P}[q_t = S_i|\mathcal{O}, \lambda] = \frac{\alpha_t(i)\beta_t(i)}{\mathbf{P}[\mathcal{O}|\lambda]}. \tag{14}$$

To determine the model that maximizes the probability of the observation sequence there is no known analytical way, nonetheless the Baum-Welch algorithm locally maximizes $\mathbf{P}[\mathcal{O}|\lambda]$, bases on two steps: first it computes the forward (6) and backward (9) probabilities for each state of the model, then the expected count of the transition to a state and emission of an observation pair is evaluated, to provide a new estimation of the model parameters $\{\bar{A}, \bar{B}, \bar{\pi}\}$.

The variable $\xi_t(i,j)$ is introduced as the probability of been in state S_i at time t and in state S_j at time $t+1$, given the model λ and the observation sequence $\mathcal{O}$:

$$\xi_t(i,j) = \mathbf{P}[q_t = S_i, q_{t+1} = S_j|\mathcal{O}, \lambda] \,. \tag{15}$$

From the chain rule, the definitions (6)-(9), and (13), this can be rewritten as

$$\xi_t(i,j) = \frac{\mathbf{P}[q_t = S_i, q_{t+1} = S_j, \mathcal{O}|\lambda]}{\mathbf{P}[\mathcal{O}|\lambda]} \tag{16}$$

$$= \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)} \,, \tag{17}$$

while from (12) and (14) it follows:

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j) \,. \tag{18}$$

Integrating over time, the summation of γ gives either the expected number of visits of state S_i if summing over $[1, T]$ or the expected number of transition made from state S_i if summing over $[1, T-1]$, while the summation of ξ over $[1, T-1]$ provides the expected number of transitions from state S_i to state S_j, namely:

$$\sum_{t=1}^{T} \gamma_t(i) = \mathbb{E}[\# \text{ of } S_i] \tag{19}$$

$$\sum_{t=1}^{T-1} \gamma_t(i) = \mathbb{E}[\# \text{ of } S_i \to \bullet] \tag{20}$$

$$\sum_{t=1}^{T-1} \xi_t(i,j) = \mathbb{E}[\# \text{ of } S_i \to S_j]\,, \tag{21}$$

with straightforward interpretation of the adopted notation.

Finally, simply by counting the event occurrences, it is possible to formulate a procedure for the re-estimation of the model parameters, $\{\bar{A}, \bar{B}, \bar{\pi}\}$ as:

$$\bar{\pi}_i = \mathbb{E}[\# \text{ of } S_i \text{ at } t = 1] = \gamma_1(i) \tag{22}$$

$$\bar{a}_{ij} = \frac{\mathbb{E}[\# \text{ of } S_i \to S_j]}{\mathbb{E}[\# \text{ of } S_i \to \bullet]} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \tag{23}$$

$$\bar{b}_j(v_i) = \frac{\mathbb{E}[\# \text{ of } S_j \text{ and } v_i]}{\mathbb{E}[\# \text{ of } S_j]} = \frac{\sum_{\substack{t=1 \\ \text{s.t. } O_t = v_i}}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)} \tag{24}$$

In [Baum and Sell (1968)][Baker (2003)] it is shown that the new model $\bar{\lambda} = (\bar{A}, \bar{B}, \bar{\pi})$ is more likely than the initial model λ in the sense that it increases (at worst it equalizes) the probability of the observation given the model:

$$\mathbf{P}[\mathcal{O}|\bar{\lambda}] \geq \mathbf{P}[\mathcal{O}|\lambda]\,, \tag{25}$$

i.e., the new model $\bar{\lambda}$ is more likely than the original λ to produce the observation sequence (or at least equally likely).

A couple of notes are in order: firstly, since this iterative procedure converges to local minima and the optimization manifold may be of some complexity, the initial condition (state) should be a fair guess of the true model, in order to provide a good model adjustment. Also, working with long observation sequence, scaling is required for the re-estimation computation, because the parameter values become numerically close to machine precision.

Some implementations can be found in [Rabiner (1989)], [Stamp (2004)], [Mann (2006)].

5. NODE-SPLITTING FOR HMM IDENTIFICATION

As previously mentioned in Subsec. 3.2 once a first instance of the graph $\mathcal{G}$ has been retrieved, some of the nodes undergo a splitting procedure, which is the subject of this section. Here and in the remainder of the paper we neglect self loops in the graph structure.

5.1 Topological and Logical Node-Splitting

Starting from the observation data, it is possible to infer a probabilistic graph structure that represents the sensor-observation relations, which can be not exactly consistent with the topology of the camera network and does not

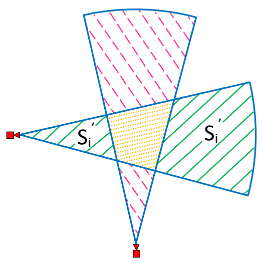

Fig. 2. Schematic drawing for topological node-splitting. The fov's of two cameras identify an overlapping area, while the other areas of interest are seen by one camera only. Areas colored with the same pattern style generate the same observation symbol.

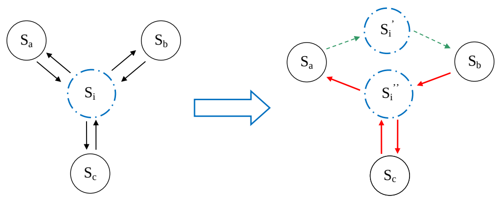

Fig. 4. Node splitting example. The target trajectory at past and future times determines the node splitting.

provide accurate information about space locations. To exemplify, given the assumptions of having binary observations, it is possible to get the same observation symbol while being the target in distinct positions (e.g. in Fig. 2). In this case, it is preferable to have two distinct states S_i' and S_i'' that relate to the same observation, because the future trajectory could evolve very differently. We refer to this situation as *topological node-splitting*. A second situation is related to the target trajectory, since the past transitions may suggest a preferential direction and allow the estimation of the future state visits (see Fig. 3). This is a motivation to split a state, i.e. the probability of future state, given the present position (observation) and the past trajectory. We indicate this case as *logical node-splitting*. Note that logical splitting include also topological splitting. In both case, it follows an increase in the number of states and arcs of the graph, as well as an appropriate arrangement of the transition probabilities. The purpose is now to design a procedure that autonomously finds the states to split and performs the splitting. To this aim, we first need to enrich the graph with more information so we decide to consider for each state S_i all the possible paths of length 2 (i.e. combinations of past/future transitions that can occur).

5.2 Orthogonality Measure

With reference to the example in Fig. 4, for a node $q_t = S_i$ a table is built with the possible past and future nodes (q_{t-1} and q_{t+1}), and the related transition probabilities, as the following:

		q_{i+1}		
		S_a	S_b	S_c
q_{i-1}	S_a	0	$\mathbf{P}_{a*b}$	0
	S_b	$\mathbf{P}_{b*a}$	0	$\mathbf{P}_{b*c}$
	S_c	$\mathbf{P}_{c*a}$	0	$\mathbf{P}_{c*c}$

Table 1.

Here, it appears that the future state is S_b only if the past state is S_a (dashed arrows in figure on the right);

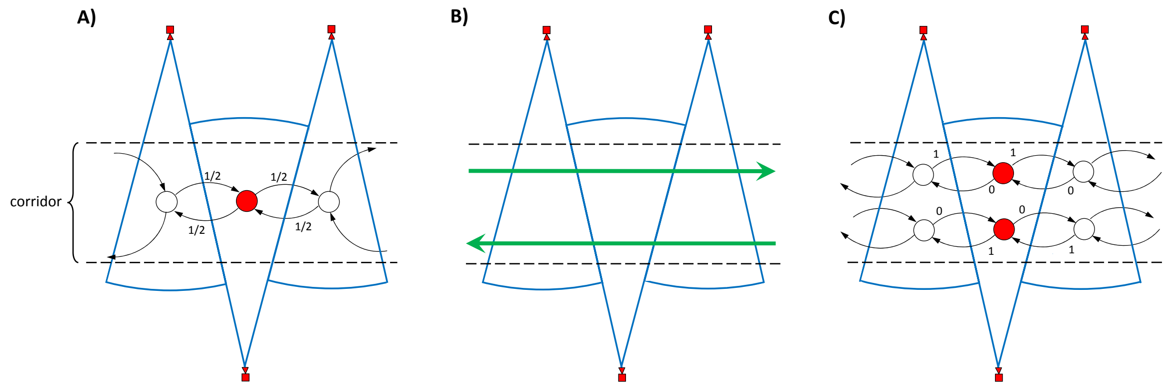

Fig. 3. Schematic drawing for logical node-splitting. A corridor is seen by a set of cameras in the orthogonal direction w.r.t. the target trajectory. A) Without trajectory information, probabilities to go either way are equal. B) Trajectory pattern: left to right or viceversa. C) The state is split into two according to what is the origin of the trajectory and the probabilities of transition are updated accordingly.

instead future states are $\{S_a, S_c\}$ only if the past states are $\{S_b, S_c\}$ (continuous arrows in figure on the right).

From a projection point of view, the past-future transition table of state S_i (Tab. 1) is prone to an interesting interpretation: the first row of the table is in fact orthogonal to the others, which conversely lie on the same 2D space, and are in this sense "similar". The state S_i can be therefore split into two separate states S_i' and S_i'' to take into account this feature: S_i' will refer to the transition $S_a \rightarrow S_i \rightarrow S_b$ while S_i'' will be related to $\{S_b, S_c\} \rightarrow S_i \rightarrow \{S_a, S_c\}$. In general, a "measure of orthogonality" between row vectors v_h and v_l is given by the cosine of the angle between the two:

$$\sigma_{hl} = \frac{v_h \cdot v_l^T}{\|v_h\|\|v_l\|}, \tag{26}$$

that is $\sigma_{hl} = 0$ if v_h and v_l are orthogonal, $0 < \sigma_{hl} \leq 1$ if they are coplanar, $\sigma_{hl} = 1$ being the parallel condition.

A node-splitting occurs when $\sigma_{hl} = 0$, which is a strict condition, but at the same time it assures the benefit of the operation (although in Sec. 6 this condition will be relaxed to $\sigma_{hl} \leq 0.1$ and $\sigma_{hl} \leq 0.2$). In fact, this means that the states in $t-1$ (S_a and $\{S_b, S_c\}$ in the example) give rise to very different evolutions in $t+1$ that need to be treated separately at t ($q_t = S_i$). Then, new transition probabilities are assigned redistributing the transition probability through S_i according to the occurrence of transitions from one state to the other. This procedure is described in the left drawing of Fig. 5, with S_i split into S_i' and S_i''. Nonetheless, a non-null probability ϵ (i.e. low values 10^{-6}) among apparently non related states (i.e. unobserved transitions), is admitted with the twofold purpose of accounting for non perfect orthogonality among vectors and of allowing some "genetic variability" among transitions since the whole procedure is based on a finite sampling (see Fig. 5).

In the end modifications to the elements of A are in the rows and columns corresponding to the two new states. In particular rows and columns related to S_i are doubled. In B only the number of columns changes due to the fact that the observation cardinality remains the same. The column corresponding to S_i splits into one for S_i' and a new equal column for S_i''.

5.3 Algorithm

The splitting procedure described in the previous section is not complete yet. It only proposes a strategy which tells when and where to add an hidden state (a new row and column to matrix A, which have some elements forced to zero). However, the numerical values of the non-zero entries are not known, therefore it is necessary to re-estimated the enlarged A based on the observations. Baum-Welch's algorithm gives the solution to this problem. More precisely, the Baum-Welch's algorithm is applied repeatedly after each split until the parameters converge within a certain tolerance ν. In particular, we cycle until no adjustment of values between A in input and $\bar{A}$ in output exceeds a chosen ν (initially set equal to 10^{-4} but then increased to 10^{-3} in the experimental part to reduce the computational burden). We repeat the steps σ-splitting+Baum-Welch until there is some σ_{hl} (approximately) equal to 0.

More precisely, the algorithm proceeds as follows. At the beginning the states are identified by the distinct observation, i.e. $\mathcal{S} = \mathcal{V}$, i.e. $N = M$ and the observation matrix is set to identity, $B = I$, since each state is observed exactly. The transition matrix A and the initial state distribution π are obtained by applying the Baum-Welch's algorithm, which in this particular setup has a unique solution independent of the initial condition of A and π since the states are observable. Indeed the entries a_{ij} are just the relative frequency of the observed transitions from state i to state j, and π_i are all zero except for the the state $S_{\bar{i}} = y(0) = x(0)$. After this initialization step, when a split is performed with respect to a state S_i, then a new state S_{N+1} is added, i.e. a new column and a new row are appended to the matrix A, and a new row is appended to the matrix B. In particular this new $N+1$ for B is a copy of the row corresponding to the splitting node S_i, i.e. all elements are zeros except for a one, corresponding to the unique observation. Similarly, the new $N+1$ row and column inherit the same zeros of the splitting node S_i, however the values of these entry are set quite arbitrarily, with the only constraint that the matrix A is still stochastic. A sensible choice for these entries is illustrated in Fig. 5. The Baum-Welch's algorithm, will then adjust these values to better predict the observed measurements. This procedure implemented by the main program is described in Algorithm 1.

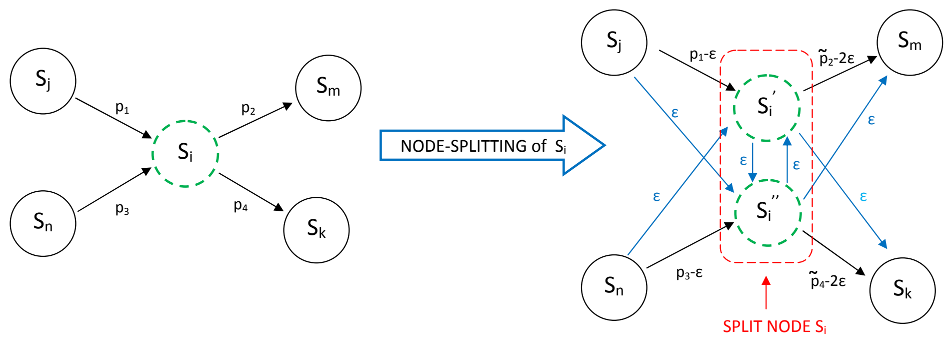

Fig. 5. Drawing for a general node-splitting of a node S_i.

Algorithm 1 σ-splitting HMM identification

```
Require: Set $\mathcal{S} = \mathcal{V}, B = I$, run Baum-Welch's algorithm
  for all states $S_i$ do
    while $\sigma_{hl} = 0$ for $a_{hi} > 0, a_{il} > 0$ do
      split node $S_i$ into $S_i$ and $S_{N+1}$
      set new $N+1$ row $B$ equal to row $i$
      set new $N+1$ row and column of $A$ (e.g. in Fig. 5)
      repeat
        Baum-Welch's algorithm
      until convergence for $A$ is not reached (w.r.t. $\nu$)
    end while
  end for
```

5.4 Performance index

As mentioned in Section 3.4, our goal is to identify a HMM which has good predictive properties. Therefore, a natural index of performance in the normalized correct prediction probability defined as

$$\eta(\mathcal{O}, \lambda) = \sqrt[T]{\prod_{t=0}^{T-1} \mathbf{P}[\hat{y}_{t+1} = y_{t+1} | y_{0:t}]} = \sqrt[T]{\mathbf{P}[\mathcal{O}|\lambda]}. \quad (27)$$

which corresponds to the (geometric) average probability of making the correct prediction for a specific sequence of observations $\mathcal{O}$ of length T based on the identified model λ. This index can be interpreted as a sort of efficiency. In fact it is easy to see that $\eta \in [0, 1]$ for any observation sequence $\mathcal{O}$ and model λ, and it is equal to one if the model is able to perfectly predict all observations y_{t+1} based on the past, and zero if there is at least one t for which. Therefore, in general, the closer is this index to one, the better are its prediction capabilities.

It is also important to notice that the σ-splitting procedure described above should generate a sequence of models λ_k, where k is the algorithm iteration step, for which the performance index η increases when evaluated over the training observation sequence $\mathcal{O}^{tr}$, i.e. $\eta(O^{tr}, \lambda_{k+1}) \geq \eta(O^{tr}, \lambda_k)$. However, we should evaluate the performance based on a different validation sequence of observation $\mathcal{O}^{val}$. In this case, we expect that $\eta(O^{val}, \lambda_k)$ will initially increase, and then it will start decreasing due to the fact that the σ-splitting procedure starts overfitting the training observation $\mathcal{O}^{tr}$. As a consequence, the splitting procedure should be stopped when the prediction performance η start decreasing when computed for the validation set $\mathcal{O}^{val}$.

6. SIMULATIONS

In this section, we provide some numerical simulations of the proposed algorithm and we evaluate its performance under different scenarios. We will start with a simple model of corridor where targets move back and forth from one side to the other. This example clearly exhibits the need of both topological splitting and logical splitting, and we expect that the splitting procedure exactly captures this structure. In the second example, we consider a rectangular area where targets move at different velocities along random directions for a random period of time. This scenario is representative of large areas where there are no preferential paths as inside an airport. In this scenario we expect no substantial improvement from the splitting procedure as targets dynamics is representative of a random walk. In the third example we consider again a rectangular area where targets moves randomly along preferential paths. This could be the typical behavior in a park where there are preferential paths for pedestrians and green areas where people do not generally walk. In this scenario we expect that the splitting procedure is able to better capture this structure of target motions. This scenario is in between the constrained motion of targets along a corridor and the totally random motion of targets on a planar area.

6.1 Scenario 1: Corridor

As explained above, in this scenario targets moves from one side to the other at approximately the same speed. The cameras are positioned as in Fig. 6, i.e. they are not overlapping and cover most of the corridor. In this case, there are 5 distinct observations y_t, which correspond to the initial guess of the states. More precisely S_1 is the state relative to the areas on the left and on the right that are not seen by any camera, while S_2, S_3, S_4, S_5 are the states corresponding to the areas seen by one camera only. After applying our splitting procedure procedure with the threshold for splitting set to $\sigma = 0$, the graph had 10 states, as expected. In particular, initial states are split as follows:

$$\begin{array}{lcl} S_1 & \rightarrow & S_1, S_6 \\ S_2 & \rightarrow & S_2, S_{10} \\ S_3 & \rightarrow & S_3, S_9 \\ S_4 & \rightarrow & S_4, S_8 \\ S_5 & \rightarrow & S_5, S_7 \end{array}$$

as shown in the bottom panel of Fig. 6. In fact, one splitting corresponds to a topological node-splitting (state S_1), while the others correspond to a logical node-splitting (states S_2 to S_5).

Fig. 6 illustrates the performance η as a function of the iteration step of Algorithm 1. The solid line correspond to the performance for the training trajectory and as expected is monotonically increasing. Also the performance on a different validation trajectory computed only after a

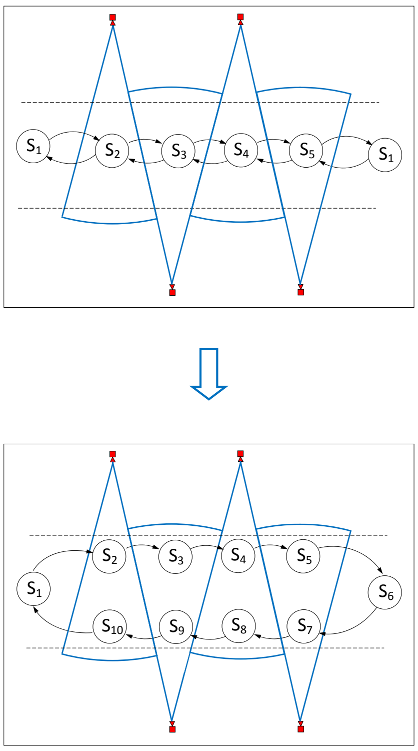

Fig. 6. States for corridor scenario: initial states *(top)* and after the splitting procedure with $\sigma = 0$ *(bottom)*

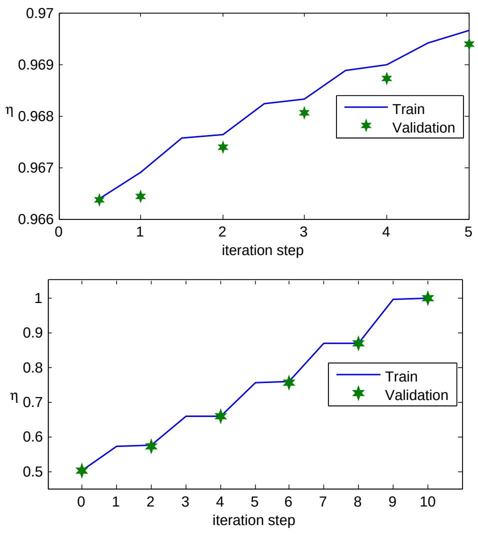

Fig. 7. Prediction performance η as a function of iteration time of Algorithm 1 for corridor scenario: including self-loops *(top)* and without self-loops *(bottom)*. Star symbols correspond to prediction performance on a validation trajectory after each state splitting step.

state splitting is very close to the training performance. However, it seems that the improvement is only marginal since the performance increases from $\eta = 0.966$ to $\eta = 0.970$ in top panel of Fig. 6. This is due to the fact that the sampling time is very small, therefore the target remains in a state for a very long time, and transitions between two distinct states are rare. A more informative approach is obtained by removing from the training trajectories all observations for which $y_{t+1} = y_t$. In this way, only the transitions are recorded. The negative side of this

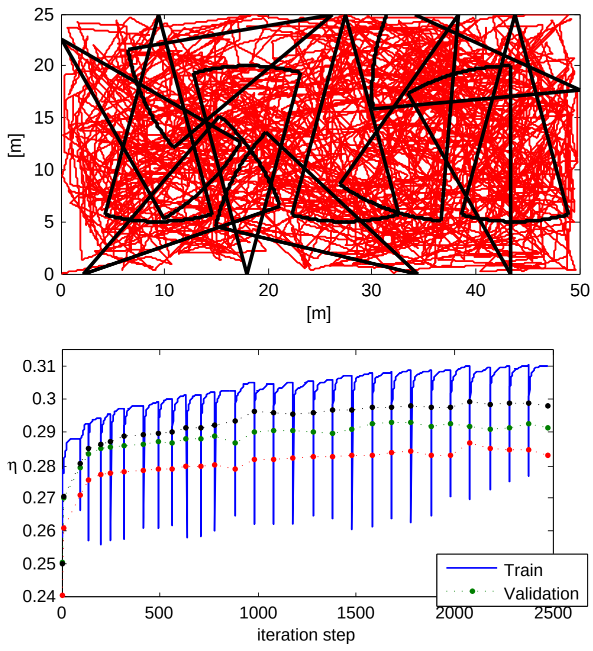

Fig. 8. Trajectories (*top*) and performance (*bottom*) for the 2-D random motion with $\sigma = 0.2$.

modeling is that permanence time in each state is lost, but it is much more useful if the objective is to *alert* the cameras that are more likely to see the target next. The performance with the self-loop removal is shown in the bottom panel of Fig. 6. As expected at the beginning, the prediction performance is $\eta = 0.5$ since a Markov modeling predicts with the same probability that the target can change direction at every step. At the end of the splitting procedure the "hidden states" are properly identified and the prediction capability is now perfect $\eta = 1$, since the identified HMM now correctly estimates the direction of the target.

6.2 Scenario 2: 2D Random motion

In this scenario targets move in a rectangular area with random direction and with random velocities as shown in top panel of Fig. 8. The black conic areas are the fields of view of 11 cameras (as in Fig. 1). Clearly, there are areas that are not seen by any cameras and others that are seen by multiple cameras. In this case, the initial number of distinct observations is 35. The performance based on trajectories without self-loops as a function of the iteration time by setting the splitting threshold to $\sigma = 0.2$ is shown in the bottom panel of Fig. 8. The solid line corresponds to the performance on the training trajectory. It is evident from the figure that after each splitting step the performance drops considerably due to the incorrect initialization of the matrix A, however, after a few steps of the Baum-Welch's algorithm, the performance improves becoming higher than before the splitting. It might happen that the splitting procedure leads to a performance which is slightly smaller than before the splitting. This is probably due the fact that the Baum-Welch is only locally optimal and it is guaranteed to converge only to a local maximum. Also from the panel it is clear that the splitting procedure in this scenario does not lead to much improvement (the performance goes from $\eta = 0.25$ to $\eta = 0.3$), thus confirming the intuition that the random walk of the targets do not allow good prediction.

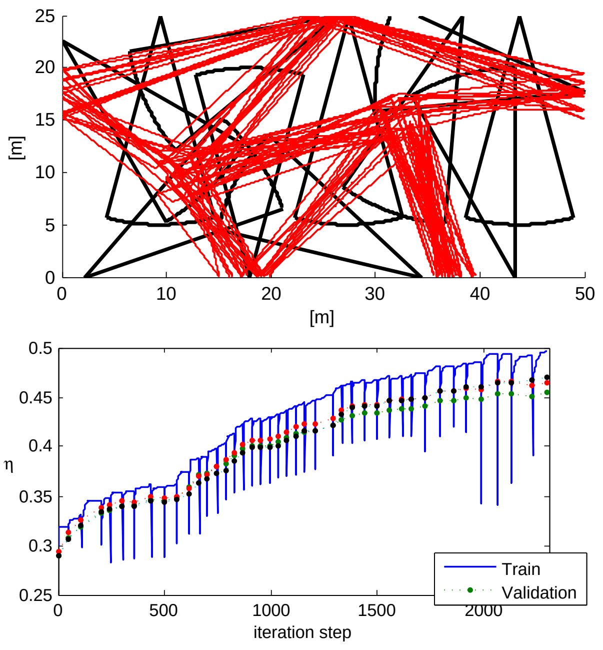

Fig. 9. Trajectories (*top*) and performance (*bottom*) for the park-like motion with $\sigma = 0.2$.

6.3 Scenario 3: Park-like

This scenario setup is similar to the previous one in the sense that targets moves with random speed around a region with the same positioning of the cameras. However, now the targets move along preferential path as illustrated in the top panel of Fig. 9. Interestingly, despite the fact that the targets explore only a fraction of the total region, the number of distinct observations are exactly the same of those relative to the previous example, namely 35. However, in this scenario the proposed HMM identification algorithm identifies about 40 additional "hidden" states, and performance improves from about $\eta = 0.3$ to $\eta = 0.45$, i.e. the prediction capability has improved of about 50% from the initial step. This confirms the intuition that this scenario represents a situation in between the corridor-like scenario for which the trajectory can be exactly predicted and the 2D random target motion, for which it is not possible to predict the future trajectory based on the two-step observation history, but depends only on the current observation.

7. CONCLUSIONS AND FUTURE WORK

The problem of determining the graph structure in a sensor network is addressed in the context of camera networks: this represents an important issue during the installation phase of medium-scale and large networks, to avoid tiring, expensive, and often unfeasible manual setups. The proposed strategy makes use of HMM modeling and the combination of the Baum-Welch algorithm to optimize the model and a node splitting procedure to discover hidden states and augment the model. A measure of performance is given in terms of the capability of the model to predict the observations. Numerical simulations are provided trying to reproduce different paradigmatic scenarios to prove the effectiveness of the methodology. The results shows that the approach is consistent with intuition and able to well recover the observable topology of the network.

Future work will include a better insight into the convergence features of the procedure and the implementation of the system in a real world scenario.

REFERENCES

Baker, J. (2003). The dragon system–an overview. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 23(1), 24–29.

Baum, L.E., Petrie, T., Soules, G., and Weiss, N. (1970). A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *The Annals of Mathematical Statistics*, 41(1), 164–171.

Baum, L.E. and Sell, G.R. (1968). Growth transformations for functions on manifolds. *Pacific Journal of Mathematics*, 27(2), 211–227.

Bishop, C.M. (2006). *Pattern Recognition and Machine Learning*. Springer, New York.

Chilgunde, A., Kumar, P., Ranganath, S., and Weimin, H. (2004). Multicamera target tracking in blind regions of cameras with nonoverlapping fields of view. In *British Machine Vision Conference (BMVC04*, 397–406.

Collins, R.T., Lipton, A.J., Fujiyoshi, H., and Kanade, T. (2001). Algorithms for cooperative multisensor surveillance. In *Surveillance, Proceedings of the IEEE*.

Ellis, T.J., Makris, D., and Black, J.K. (2003). Learning a multi-camera topology. In *in Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, 165–171.

Farrell, R., Doermann, D., and Davis, L.S. (2007). Learning higher-order transition models in medium-scale camera networks. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, 1–8.

Javed, O., Rasheed, Z., Alatas, O., and Shah, M. (2003). Knight^{m}: A real time surveillance system for multiple overlapping and non-overlapping cameras. In *in IEEE Conference on multi media and expo*, 6–9.

Jordan, M.I. (1999). *Learning in graphical models*. MIT Press, Cambridge, MA.

Mann, T.P. (2006). Numerically stable hidden markov model implementation.

Rabiner, L. and Juang, B. (2003). An introduction to hidden markov models. *ASSP Magazine, IEEE*, 3(1), 4–16.

Rabiner, L.R. (1989). A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2), 257–286.

Stamp, M. (2004). A revealing introduction to hidden markov models.

Vasquez, D., Fraichard, T., and Laugier, C. (2009). Incremental learning of statistical motion patterns with growing hidden markov models. *Trans. Intell. Transport. Sys.*, 10(3), 403–416.

Wainwright, M.J. and Jordan, M.I. (2008). Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1–2), 1–305.

Welch, L.R. (2003). Hidden markov models and the baum-welch algorithm. *IEEE Information Theory Society Newsletter*, 53(4).