Incremental Learning of Statistical Motion Patterns with Growing Hidden Markov Models

Dizan Alejandro Vasquez Govea, Thierry Fraichard, Christian Laugier

Incremental Learning of Statistical Motion Patterns with Growing Hidden Markov Models

Dizan Vasquez, Thierry Fraichard and Christian Laugier

***Abstract*—Modeling and predicting human and vehicle motion is an active research domain. Due to the difficulty of modeling the various factors that determine motion (*eg* internal state, perception, etc.) this is often tackled by applying machine learning techniques to build a statistical model, using as input a collection of trajectories gathered through a sensor (*eg* camera, laser scanner), and then using that model to predict further motion. Unfortunately, most current techniques use off-line learning algorithms, meaning that they are not able to learn new motion patterns once the learning stage has finished. In this paper, we present an approach where motion patterns can be learned incrementally, and in parallel with prediction. Our work is based on a novel extension to Hidden Markov Models –called Growing Hidden Markov models – which gives us the ability to learn incrementally both the parameters and the structure of the model.**

***Index Terms*—Hidden Markov Models, motion prediction, pattern learning.**

I. Introduction

PREDICTING the trajectories that vehicles and pedestrians are going to follow in a given environment is fundamental for effective autonomous navigation in cities, parking lots and highways. The main challenge lies in the fact that these objects move according to a diversity of complex factors – such as their intentions and internal state – which are very difficult to model and parametrize. Thus, instead of explicitly modeling these factors, the preferred approach in the literature assumes that objects tend to follow typical motion patterns; hence, if those patterns are known, it is possible to use them not only to predict further motion but also, for example, for detecting anomalous behavior, or improving visual tracking.

In practice, former knowledge about motion patterns is seldom available *a priori* and it should be obtained by applying machine learning techniques to motion data obtained through some kind of sensor system. For example: Bennewitz et al. [1] use the Expectation-Maximization algorithm to cluster trajectory data gathered with a laser scanner; and Hue et al. [2] apply a two-pass hierarchical clustering algorithm to find patterns on the output of a visual tracker.

Despite being quite diverse, most motion pattern learning approaches share the significant drawback that they operate off-line, which implies the assumption that at least one example of every possible motion pattern is contained in the learning data set. Given the enormous variety of possible human behaviors, this assumption does not hold in practice and the learned motion models have, in the best case, only limited utility.

It would be better to learn motion patterns incrementally, so that, when a new motion pattern is observed, the system is able to integrate it into its knowledge base. This paper describes such an approach: it incrementally learns motion patterns and, at the same time, uses its current knowledge to predict motion. Our approach extends further our previous work [3] by proposing a unified extension to Hidden Markov Models (HMM)[4], a probabilistic framework which is very popular in the motion pattern learning literature [*eg* 5, 6, 1]. This extension, named Growing HMM (GHMM) enables incremental and on-line learning of the parameters and the structure of the model.

The rest of this paper is structured as follows: Section II provides an overview of motion pattern learning, focusing in techniques based on HMMs. Section III presents Growing Hidden Markov Models. In Section IV the application of GHMMs to our particular problem is discussed. Our experimental results are outlined in Section V. Finally, we present our conclusions in Section VI.

II. Related work

In order to learn motion patterns it is necessary to define their meaning and to decide how they are going to be represented. In the first part of this section, we present an overview of approaches in the literature, classifying them according to the answers they provide to these questions. Then, on the second part we provide a more detailed explanation on HMM based approaches, which constitute the basis of our proposed approach.

A. Literature overview

1) Behavioral models: Approaches in this category consider motion patterns in terms of behaviors having high level semantics: a person may be following a friend, or fleeing from an attacker, a car may be passing another car or waiting for the green light, etc.

In general, these approaches deal with the evolution of the *intentional* state of the objects, often disregarding their metric or physical states (*eg* position, speed, etc.). This makes them better suited for applications like video surveillance or scene understanding than for tracking or motion prediction.

This work has been partially supported by the EC-contract BACS FP6-IST-027140 and by a Mexican CONACYT scholarship. We would like to thank Professors Roland Siegwart, Michel Devy and Wolfram Burgard for their insightful comments on this work. Finally, we want to thank Dr. Hannah Dee and the University of Leeds for kindly sharing their experimental data with us.

This paper reflects only the authors' view and funding agencies are not liable for any use that may be made of the information contained herein.

A good example of this type of approaches is described in the work of Oliver et al. [7]: they use Coupled Hidden Markov Models [8] to model interactions (*eg* approaching, meeting and fleeing) between pairs of objects. These states are defined prior to learning and the model is trained on labeled data. Similar ideas have been explored in [9, 10] allowing for interactions between more than two objects.

2) Descriptive models: This family of approaches models motion in terms of the physical state of the object without taking semantics into account. Often, motion patterns are represented as sequences of points, describing the object's state at consecutive discrete time steps. Under this representation, the learning problem is frequently addressed using some sort of clustering algorithm to extract a number of 'typical' motion patterns (*ie* trajectory prototypes) from an input data set consisting of raw trajectory data.

A representative example is the approach proposed by Bennewitz et al. [11], that uses Expectation Maximization to perform the clustering. Other algorithms include hierarchical clustering [12, 13, 14, 2], graph cutting [15], and custom pairwise clustering algorithms [16].

In order to apply the obtained trajectory prototypes to perform tracking or motion prediction, a probabilistic framework is often used. This allows one to represent explicitly the uncertainties associated with sensor noise, and to take into account the model incompleteness. Since most of these approaches are based on HMMs, we will review them in further detail in a separate section.

Some alternatives to approaches based on probabilistic frameworks exist in the literature: Neural networks are probably the most popular one, starting with the seminal work by Johnson and Hogg [17] which first proposed the use of multilayer self-organizing networks where one layer represents the states, and another corresponds to the followed path. Similar approaches have been proposed by [18] and [19], which by modeling time explicitly obtained performances comparable to that of probabilistic models. A different idea has been explored by Stauffer and Grimson [20], which no longer represented motion patterns as typical trajectories, but as a co-occurrence matrix for every different motion pattern, where every element $c_{i,j}$ roughly corresponds to the probability that an object passes through states i and j given that it is engaged in the corresponding motion pattern.

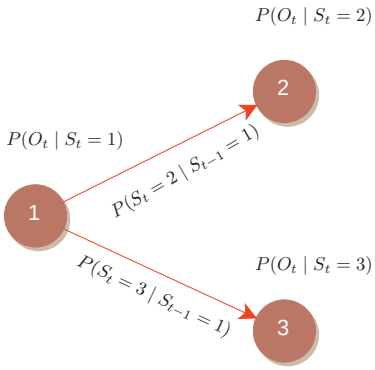
3) Hybrid models: Of course, the intentional and physical states of an object are not independent: the intentions of an object condition its physical state; conversely, information about the position and speed of an object may be used to infer the object's intentions or the behavior it is involved in. A number of approaches model to a certain extent the relationship between these two states.

The basic idea in this kind of approaches is to condition motion models on the behavior being executed. Often, the behavior is represented as the object's intention of reaching a particular place in the environment (*ie* its goal). For example, Liao et al. [21] have used a hierarchical extension to HMMs – Abstract Hidden Markov Models (AHMM)[22] – to learn and predict the motion of pedestrians in cities, where the three layers in the AHMM represented – top to down – goals, transportation modes and physical state. The approach is able to learn the goal and transportation mode structures using custom tailored algorithms, but the low-level physical structure is given *a priori* in the form of a graph. Another AHMM based approach has been proposed in[23] for indoor environments but the structure is given *a priori*. Regarding AHMMs, it is worth mentioning that they are – with respect to inference – equivalent to a Markov Decision Process [24] a probabilistic planning technique, which illustrates the connection between planning and motion prediction.

Other goal based approaches include [25, 26] and [27]. The latest is of particular interest because it represents the world from the object's perspective, which clearly contrasts with most other approaches, which are based in some sort of global view. However, these three approaches share a problem: the object's evolution towards the goal is modeled using overly simplistic mechanisms (*eg* linear interpolation for [25]), leading to unreliable physical state estimations.

B. Hidden Markov Model Based Approaches

In this section we will focus on techniques based on Hidden Markov Models, and thus closely related to the proposed approach. For the sake of clarity, our discussion of HMMs will be just a brief overview, heavily biased towards our application. The interested reader may refer to the papers by Juang et al. [28] and Rabiner [4] for a deeper introduction to the subject.



(a)

(b)

Fig. 1. a) A basic three-state HMM; b) HMM Structure embedded in a parking (only a few motion patterns are displayed)

In the context of our problem, an HMM (see fig. 1(a)) may be seen as a graph whose nodes represent states attainable by

the object (*eg* places in the environment) and whose edges represent transitions between states. The system is supposed to be at a given state and to evolve stochastically at discrete time steps by following the graph edges according to a transition probability $P(S_t|S_{t-1})$. Moreover, the object's state is not directly observable, instead, it should be measured through some kind of sensor reading (*ie* observation) which is related to the actual state through an observation probability $P(O_t|S_t)$. Often, the initial state of the system is represented stochastically with a state prior $P(S_1)$.

HMM learning is composed of two sub-tasks:

- *Structure learning:* Determines the number of nodes in the model – which will be called *discrete states* henceforth – as well as the edge structure for the graph.
- *Parameter learning:* Estimates the parameters for the three probability distributions (state prior, transition and observation probabilities) from data.

Different algorithms for structure and parameter learning exist in the literature, it is the choice of these algorithms what distinguishes different HMM based motion pattern learning approaches. For example, Walter et al. [5] assume that the number of motion patterns is known *a priori* and define the structure using a different chain-link graph for every motion pattern, then, parameters are learned using the Expectation-Maximization algorithm; Bennewitz et al. [1] learn the HMM structure by clustering trajectory data with the Expectation-Maximization algorithm, and then manually set the model's parameters according to assumptions about object's motion; Makris and Ellis [6] learn the HMM structure in a similar way, but also incorporate parameter learning into the algorithm.

Despite their differences, all these approaches have some points in common: a) typical motion patterns are represented with some sort of trajectory prototype; b) structure learning is independent of parameter learning; and c) learning is first performed off-line and then the system switches to a utilization stage where no further learning is performed. As we will see in the following sections, our approach behaves differently with respect to these points.

III. Growing Hidden Markov Models

In this section we present our proposed extension to HMMs: Growing Hidden Markov Models[1] (henceforth denoted GHMM); which may be described as time-evolving HMMs with continuous observation variables, where the number of discrete states, structure and probability parameters are updated every time that a new observation sequence is available.

Our approach is designed for its utilization as a discrete approximate inference tool for continuous state spaces. It is applicable to problems where the continuous state space may be discretized into a finite number of regions, so that every such region is represented by a discrete state in the GHMM.

Our approach relies on three main assumptions:

- We assume that input observation sequences correspond to complete examples (*ie* from beginning to end) of the whole process or system being modeled (*eg* in our application this corresponds to complete pedestrian trajectories).
- The evolution of the state of the modeled system or process is a continuous function.
- The observation space is a subspace of the continuous state space. This implies that, by finding a decomposition of the observation space, a decomposition is also performed on the continuous state space[2]

The key intuition behind GHMMs is that the structure of the model should reflect the spatial structure of the state space discretization, where transitions between discrete states are only allowed if the corresponding regions are neighbors. Therefore, structure learning basically consists of estimating the best space discretization from data and identifying neighboring regions. We have addressed this problem by building a *topological map* using the Instantaneous Topological Map (ITM) algorithm [30]. For parameter learning, we basically have adapted the incremental Expectation-Maximization approach proposed by Neal and Hinton [31] in order to deal with a changing number of discrete states and with continuous observations.

To avoid confusion, in the rest of this document, we will make a strict difference between *nodes* of the ITM algorithm, the *discrete states* of a GHMM; the *continuous state* of an object; and the *observations* provided by sensors.

A. Probabilistic model

Structurally GHMMs are identical to regular HMMs except for the fact that the number of states and the transition structure are not constant, but can change as more input observation sequences are processed. The other difference lies in the learning algorithm, which is able to incrementally update the model. A GHMM is defined in terms of three variables:

- S_t, S_{t-1}, the current and previous states, which are discrete variables with value $S_t, S_{t-1} \in \{1, \cdots, N_k\}$, where N_k is the number of states in the model after k observation sequences have been processed[3].
- O_t, the observation variable, which is a multidimensional vector.

The joint probability decomposition (JPD) for GHMMs is:

$$P(S_{t-1}\ S_t\ O_t) = \underbrace{P(S_{t-1})}_{\textit{state prior}}\ \underbrace{P(S_t|S_{t-1})}_{\textit{transition probability}}\ \underbrace{P(O_t|S_t)}_{\textit{observation probability}} \quad (1)$$

[1]Since space is limited, we have opted for providing a general overview on GHMMs, which omits some specific information on optimizations and data structures. The interested reader is referred to [29] for more details.

[2]It is worth noting that this assumption may be relaxed when the model is not used for prediction but – for instance – just for recognition. In that case the only requirement is the existence of a weak topological equivalence between the observation and state spaces; when the system goes through states which are near each other, the corresponding observations will also be close to each other.

[3]For the sake of notational simplicity, we will often omit the k hereafter, nevertheless, it should be noted that parameters and structure change with every new observation sequence. Also, notation $O_{1:t}$ will be used as a shortcut for the variable conjunction $O_1\ O_2\ \cdots\ O_{t-1}\ O_t$.

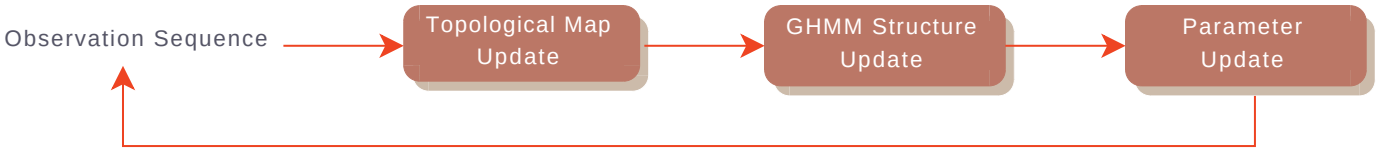


Fig. 2. Overview of the GHMM learning algorithm.

Where the state prior is simply the posterior of the previous time step:

$$P(S_{t-1}) = P(S_{t-1}|O_{1:t-1}) \tag{2}$$

Both the observation and transition probabilities are assumed to be *stationary* that is, independent of time, thus the parametric forms of the three probabilities in the JPD are the same, irrespectively of the value of the time variable:

- $P(S_0 = i) = \pi_i$. The state prior will be represented as a vector $\pi = \{\pi_1, \cdots, \pi_N\}$ where each element contains the prior probability for the corresponding discrete state.
- $P([S_t = j]|[S_{t-1} = i]) = a_{i,j}$. Transition probabilities are represented with a set of variables A, where each element $a_{i,j}$ represents the probability of reaching state j in the next time step given that the system is currently in state i.
- $P(O_t|[S_t = i]) = \mathbf{G}(O_t;\ \mu_i,\ \Sigma)$. The observation probability density function will be represented by a Gaussian distribution for every discrete state, having the same covariance matrix Σ for all discrete states. The set of all the Gaussians' parameters will be denoted by $b = \{\Sigma, \mu_1, \cdots, \mu_N\}$.

The full set of parameters for a GHMM is denoted by $\lambda = \{\pi, A, b\}$.

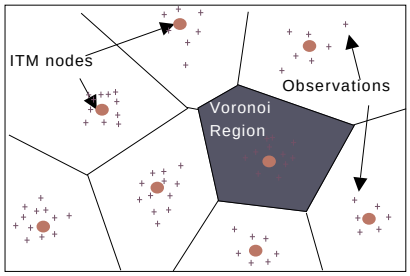
Besides its time-evolving nature, a GHMM is defined by its learning algorithm, which processes complete observations sequences as they arrive. The general steps of the algorithm are depicted in Fig. 2 and are detailed in the following subsections.

B. Updating the Topological Map

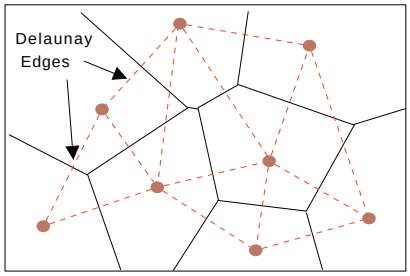
Our structure learning approach is based on the construction of a topological map: a discrete representation of continuous observation space in the form of a graph where nodes represent regions of the space, and edges connect contiguous nodes. Every node i has an associated vector w_i, corresponding to the region's centroid. The nodes are added an adapted in order to minimize the distortion of the model, *ie* the sum of the squared distances between the input (*ie* observation) vectors and the centroid of their closest node.

The topological map is updated for every available observation O_t using the ITM algorithm which has the following properties:

- It minimizes the number of nodes while trying to keep the same average distance between neighbors.
- Has linear time and memory complexity with respect to the number of nodes.
- Edges are a subset of the Delaunay triangulation, meaning that they can exist only between nodes representing adjacent Voronoi[4] regions (Fig. 3).



(a) Nodes and regions



(b) Edges

Fig. 3. Example ITM space decomposition

The ITM algorithm consists of the following steps (*cf* [30]):

1) **Matching**: find the nearest b and second nearest s nodes to O_t. We use Mahalanobis distance with the same Σ than observation probabilities.
2) **Weight adaptation**: move w_b towards O_t by a small fraction $\Delta_b = \epsilon(O_t - w_b)$.
3) **Edge adaptation**: a) create an edge connecting b and s unless that edge exists; b) for every neighbor m of b check if O_t lies in the Thales sphere going through w_m and w_b and delete the corresponding edge if that is the case. Delete any node which has no neighbors.
4) **Node adaptation**: a) if O_t lies outside the Thales sphere going through w_b and w_s and its distance from w_b is greater than a given threshold τ, create a new node n with $w_n = O_t$. Connect nodes b and n. Remove node s if it's distance from b is smaller than $\frac{\tau}{2}$.

A crucial part of the algorithm is that, besides the matching step, all the operations needed to maintain the Delaunay triangulation depend only on nodes and edges in a local

[4]The Voronoi region associated with a node is defined by the set of all the points which are closer to that node's centroid than to any other centroid in the graph. Delaunay edges link the centroids of Voronoi regions that have a common border.

neighborhood. There is a minor problem though, since node adaptation takes place after edge adaptation, it is possible that some of the edges connected to b become non Delaunay. However, these edges are latter deleted by the edge adaptation step when new observations fall in the same region.

It is important to note that, due to the assumption that the observation space is actually a subspace of the continuous state space, the obtained ITM is also a representation of the latter. This makes it possible to use it directly to update the GHMM structure, as described in the following section.

C. Updating the Model's Structure

During the topological map update, nodes and edges may be added or deleted, these changes in the topological map are reflected in the GHMM structure as follows:

1) For every new node i in the topological map, add a corresponding discrete state in the GHMM, initializing its prior to a preset value: $\pi_i = \pi_0$. Do the same for the self-transition probability: $a_{i,i} = a_0$. Note that in this and the two following steps, the values are not strictly a probability because the corresponding sums do not add to one. This is corrected by a normalization step that takes place at the beginning of parameter update (*cf* § III-D).
2) For every new edge (i, j) in the topological map, initialize the corresponding transition weights to: $a_{i,j} = a_0$ and $a_{j,i} = a_0$. As in the previous step, this values will be normalized later to obtain true probabilities.
3) For every deleted node and edge in the topological map, assign a value of zero (*ie* delete) to the corresponding state prior and transition weights.
4) For every added or modified centroid w_i, set the corresponding Gaussian mean value: $\mu_i = w_i$.

D. Updating the Parameters

Parameter learning is performed immediately after structure learning. The GHMM learning algorithm reestimates the parameters using an incremental version of the Baum-Welch technique based on the work from Neal and Hinton [31] extending it for continuous observation variables and an evolving number of states. The basic idea of these algorithms is to use inference to compute, for every state an transition, the likelihood that it belongs to the state (or transition) sequence that best explains the observation sequence. Then, these likelihoods are used as weights to update the model.

A particularity of our approach is that all of the observation probabilities' mean values have been assigned during structure update (see § III-C) and that their covariance Σ is fixed. Hence, only the state prior and transition probabilities need to be reestimated. This is done in four steps:

1) Normalize the state prior and transition values. This is necessary because structure update does not guarantee that the corresponding probabilities add up to one, as explained in § III-C.
2) Precompute α_i (forward probabilities), β_i (backward probabilities) and p_O (joint observation probability) for the observation sequence $O_{1:T}$ (see the appendix).
3) For every discrete state i in the GHMM, reestimate the state prior:

$$\hat{\pi}_i \leftarrow \frac{\alpha_1(i)\ \beta_1(i)}{P_O} \tag{3}$$

$$\pi_i \leftarrow \frac{(k-1)\pi_i + \hat{\pi}_i}{k} \tag{4}$$

where k is the number of observation sequences that have been observed so far.
4) Reestimate every non-zero transition probability in the GHMM using equations (5) and (6).

$$\hat{a}_{i,j} \leftarrow \frac{\sum_{t=2}^{T} \alpha_{t-1}(i) a_{i,j} P(O_t | [S_t = j]) \beta_t(j)}{\sum_{t=2}^{T} \alpha_{t-1}(i) \beta_{t-1}(i)} \tag{5}$$

$$a_{i,j} \leftarrow \frac{(k-1)a_{i,j} + \hat{a}_{i,j}}{k} \tag{6}$$

The reason to use Equations (4) and (6) is that they are equivalent to dividing the sum of the weight by the number of trajectories in order to obtain an average weight value.

For the sake of comparison, we also performed early tests with straightforward Baum-Welch, *ie* using only (3) and (5) to make the update, but the learned parameters were too heavily biased towards recent observation sequences.

IV. Learning and Predicting Motion with GHMMs

Having presented GHMMs, this section focuses in their concrete application to learning and predicting the motion of vehicles and pedestrians. This application is based on the key observation that, often, objects move as a function of their intention to reach a particular state (*ie* their goal). Accordingly, we model the object's motion in terms of an augmented continuous state vector, composed of two sets of variables describing its *current* and *intended* state (*ie* goal), respectively.

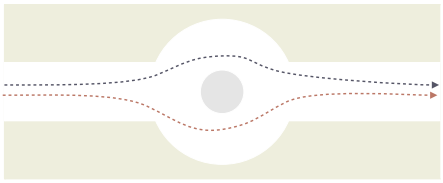
Due to the fact that our model is goal-oriented, in our approach a motion pattern is no longer a trajectory prototype, but a directed graph indicating all the possible ways in which a goal may be reached (Fig. 4).

A. Notation and Basic Assumptions

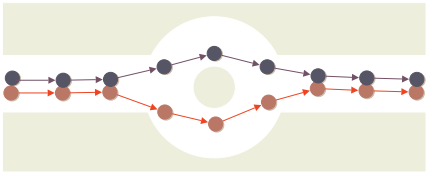
We assume that tracking data are available as a collection of observation sequences (*ie* trajectories). Every individual sequence $O^k_{1:T^k} = \{O_1, \cdots, O_{T^k}\}$ corresponds to the tracker's output for a single object and its observations are evenly spaced in time. Different observation sequences may have different lengths T^k.

In the rest of this section, we assume that the state of an object is defined by its position (x, y) and thus, that the augmented state of the object consists of its current and goal position (x, y, x', y'). It should be noted, however, that our approach is applicable to spaces of arbitrary dimensions, for example, in our experiments, we have included the instantaneous velocities of the vehicle in the continuous state vector.

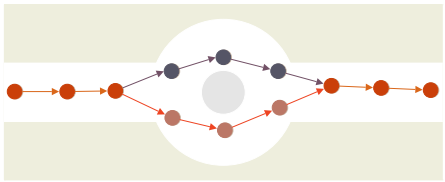
We assume that observations are available in the form of estimates of the object's coordinates $O_t = (x_t, y_t)$ – although, as in the case of the continuous state, they can also include other variables, such as velocities estimated by a tracking system. Since learning is performed on the basis of complete



(a) Input trajectories



(b) Trajectory based



(c) Goal based

Fig. 4. Pattern representations generated from input data: (b) trajectory prototypes; (c) directed graph (the goal is the rightmost node).

observation sequences, we assume that the last observation $O_T = (x_T, y_T)$ of each sequence corresponds to the object's goal. Hence, it is possible to build an augmented observation sequence, which constitutes the actual input to our algorithm:

$$\bar{O}_{1:T} = \{(x_1, y_1, x_T, y_T), (x_2, y_2, x_T, y_T), \cdots, (x_T, y_T, x_T, y_T)$$

B. Intended state

To gain a better understanding of the effect of including the intended state in our model, we will discuss the example of a simple one dimensional environment in which objects can only move from a point A to a point B or vice versa as depicted in Figure 5(a).

We will first analyze what happens with trajectories that go from A to B. From our definition the extended state is two dimensional and consists of the current and intended coordinates (x, x'). Let us suppose that $A = -5$ and $B = 5$, then the extended state trajectories will go from $(-5, 5)$ to $(5, 5)$. After training several $A \rightarrow B$ trajectories we could expect to have a topological map like the one depicted in Figure 5(b). Notice that, although the ITM does not explicitly represent the direction of motion, this notion is somehow implicit in the x' coordinate. The corresponding GHMM is shown in Figure 5(c). In it, left-right transition probabilities are high, self transition probabilities are much lower, and right-left transitions probabilities are negligible (but never zero).

If we now train on $B \rightarrow A$ trajectories, we will obtain something similar to the lower part of Figure 5(d). The first thing to notice is that extended state trajectories go now from $(5, -5)$ to $(-5, -5)$ and belong to a different one-dimensional manifold than $A \rightarrow B$ trajectories. Second, the dominant transition probabilities on that manifold correspond to right-left motion.

Thus, it can be seen that the effect of the inclusion of the intended state is the appearance of distinctive manifolds describing how objects move to reach the intended state. Of course, this is useful in the measure that there is only a limited number of typical final states or goals in the environment which, in practice, is often the case.

C. Probabilistic Model

Since our approach is based on GHMMs, it uses the same probabilistic model that has been described in §III-A. Nevertheless, we also need to distinguish between current and intended components of the state. Thus, we will decompose the augmented observation variable into its current O'_t and its intended O''_t component: $O_t = [O'_t, O''_t]$.

In order to define the JPD, we will assume that the current and intended components of observations are conditionally independent given the current state, enabling us to rewrite the observation probability as:

$$P(O_t|S_t) = P(O'_t\ O''_t|S_t) = P(O'_t|S_t)P(O''_t|S_t) \quad (7)$$

and the whole JPD as:

$$P(S_{t-1}\ S_t\ O'_t\ O''_t) = P(S_{t-1})P(S_t|S_{t-1})P(O'_t|S_t)P(O''_t|S_t) \quad (8)$$

Since the observation probability is now written as a product of probabilities, $P(O'_t\ O''_t|S_t) = P(O'_t|S_t)P(O''_t|S_t)$ we need to define their parametric forms:

$$P(O'_t \mid [S_t = i]) = \mathbf{G}(O'_t;\ \mu'_i,\ \Sigma') \quad (9)$$

and:

$$P(O''_t \mid [S_t = i]) = \begin{cases} \mathbf{U}_{O''_t} & \text{if } O''_t \text{ is not available} \\ \mathbf{G}(O''_t;\ \mu''_i,\ \Sigma'') & \text{otherwise} \end{cases} \quad (10)$$

where $\mathbf{U}_{O''_t}$ is a uniform distribution over the goal domain, μ'_i and μ''_i are the mean values of the current and goal components for discrete state i; and Σ' and Σ'' are the respective values of the covariance matrix for all the states.

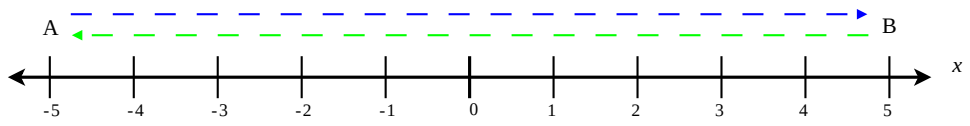
By noting that $P(O_t \mid S_t)$ is either a product of Gaussians, or a product of a constant and a Gaussian, we may rewrite this probability as a single Gaussian:

$$P(O_t \mid [S_t = i]) = \frac{1}{Z}\mathbf{G}(O_t;\ \mu_i,\ \Sigma) \quad (11)$$

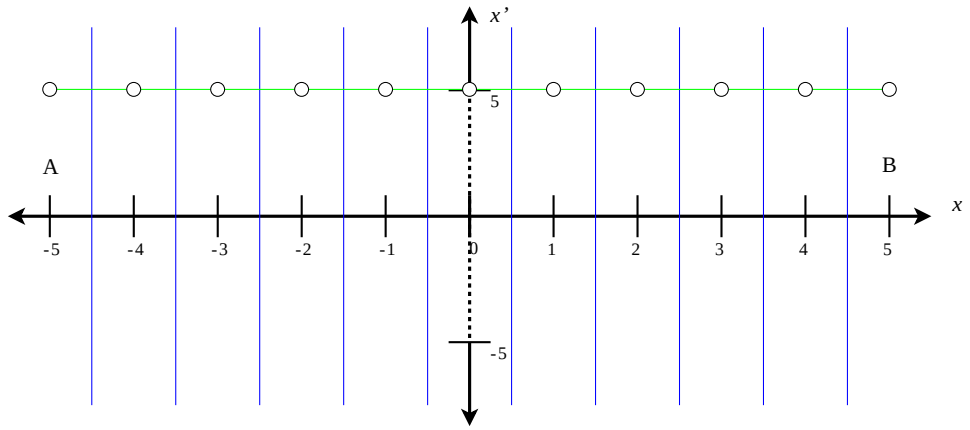
where $\mu_i = [\mu'_i, \mu''_i]$, and Σ is a block diagonal matrix having the form:

$$\Sigma = \begin{bmatrix} \Sigma' & 0 \\ 0 & \Sigma'' \end{bmatrix} \quad (12)$$

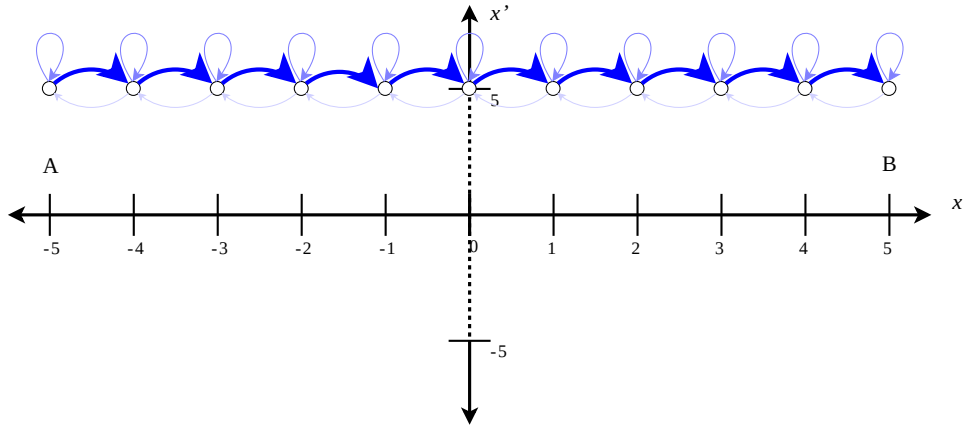
and Z is a normalization variable, that enables computation of the uniform on the goal component using the same Gaussian



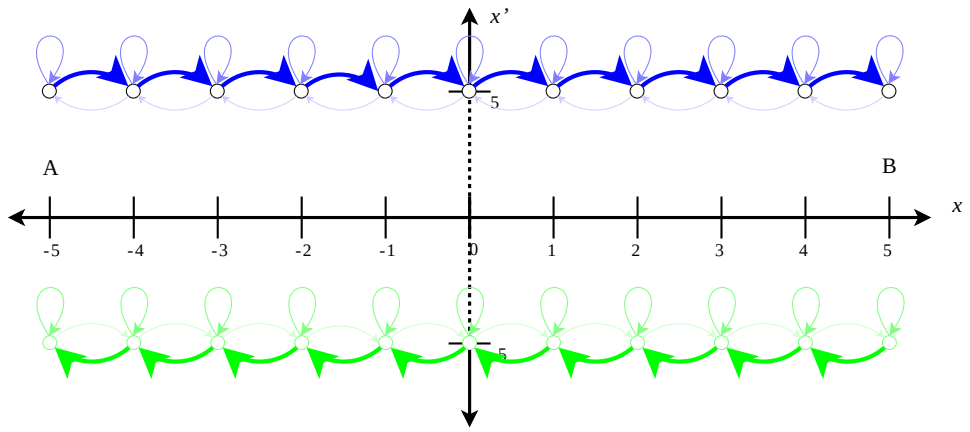
(a) Example unidimensional environment. Objects move from A to B (top arrow) or from B to A (bottom arrow).



(b) Extended observation sequence plot and Voronoi Diagram. Voronoi region's borders are depicted in blue and Delaunay links in green.

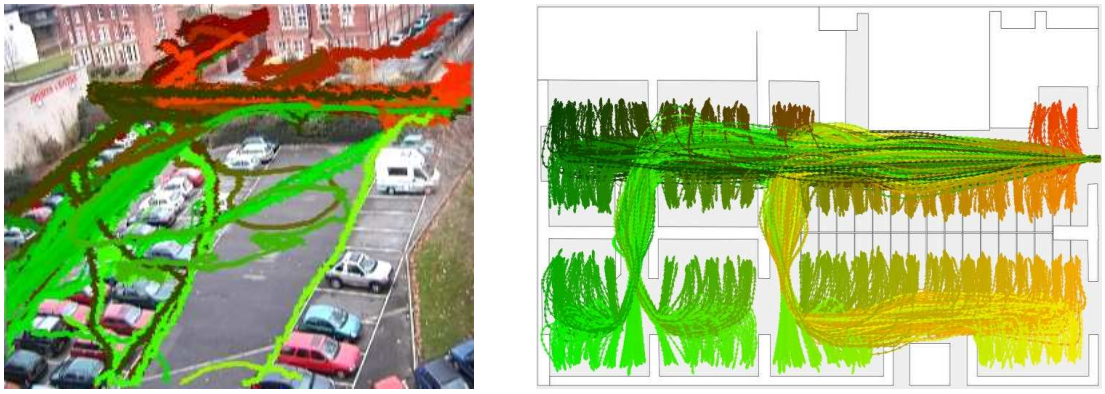


(c) Learned GHMM after processing one kind of observation sequence. The size and color of the arrows represent the probability.



(d) Learned GHMM after processing the two kinds of observation sequences. The size and color of the arrows represent the probability.

Fig. 5. Example of our motion learning algorithm in a unidimensional environment. x and x' axes have not the same scale.



(a) (b)

Fig. 6. Data sets: a) real data; b) synthetic data.

representation. Since, during prediction, the intended part of the augmented observation is not available, this is done by setting[5] $O''_t = 0$.

D. Prediction

We have not yet discussed prediction, which can be performed using the same algorithms that are used for standard HMMs, without interfering with learning. This is possible because learning takes place immediately after an observed trajectory has finished and, thus, it does not affect prediction in any way. For our particular case, we have chosen to apply exact inference:

For every new observation, the current belief state for the object is re-estimated using:

$$P(S_t \mid O_{1:t}) = \frac{1}{Z} P(O_t|S_t) \sum_{S_{t-1}} [P(S_t|S_{t-1}) P(S_{t-1}|O_{1:t-1})] \tag{13}$$

where $P(S_{t-1} \mid O_{1:t-1})$ comes from the state estimation for the previous time step (or from the state prior, in the case of the first observation in a sequence). Then, prediction is performed by propagating this estimate H time steps ahead into the future using:

$$P(S_{t+H} \mid O_{1:t}) = \sum_{S_{t+H-1}} P(S_{t+H}|S_{t+H-1}) P(S_{t+H-1}|O1:t) \tag{14}$$

Sometimes, we are interested in knowing the probability of the continuous state probability distribution – as opposed to the discrete space, shown above. Since in our case observations are expressed in terms of the state variable, the continuous state probability can be approximated by the probability that a given state is *observed* in the future, which may be computed from the predicted discrete state as follows:

[5]It is easy to show that this is equivalent to a multiplication by a constant, and – when normalized – becomes effectively equivalent to the uniform on (10).

$$P(O_{t+H} \mid O_{1:t}) = \frac{1}{Z} \sum_{S_{t+H}} P(S_{t+H} \mid O_{1:t}) P(O_{t+H} \mid S_{t+H}) \tag{15}$$

V. Experimental Results

We have implemented our approach and conducted extensive experiments using several real and synthetic data sets. In this section we will discuss some of the results we have obtained on two of these data sets (Fig. 6): a) real data obtained through a visual tracker in a parking environment and then tracking errors have been corrected by humans when necessary by inspecting the original video sequence, as described in [32]; and b) synthetic data, generated by a simulator. As mentioned in §IV we have included velocity information in the simulation, and extended the state and observation vectors accordingly. This allows, for example, to infer that the probability that a car is going to park is higher if it starts to slow down near a parking place.

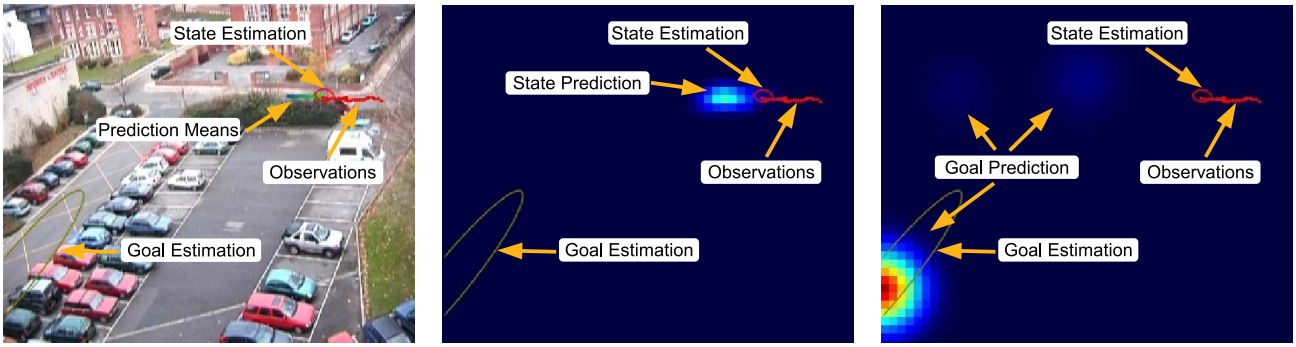
Both data sets are sampled at 10 Hz, and the tests have been executed in a 512 MB Athlon XP 2800 computer running Linux.

A. Qualitative Results

As a result of the learning step the GHMM's structure and parameters are updated, Fig. 8 shows the resulting structure after applying the learning step for 100 trajectories of the Leeds data set.

Figure 9 shows a typical example of the prediction process on the real data set. It consists of a set of images arranged in columns and rows. Rows correspond to different values of t.

In each row, the left image shows an actual picture of the parking lot featuring different overlays, as shown in Fig. 7(a): a) the current and previous observations, depicted as red dots, b) the current state estimation approximated by a Gaussian indicated with a red ellipse, c) the current goal estimation also approximated by a Gaussian, represented by a golden ellipse, and d) the mean value of the predicted states for different time horizons going from $H = 1$ to $H = 15$, where H represents the number of time steps to look ahead in the future. These



(a) Left Column (b) Center Column (c) Right Column

Fig. 7. Explanation of Fig. 9, see § V-A as well.

mean values are displayed as points colored from blue (for $H = 1$) to green (for $H = 15$).

The center and right images are 2D projections in the image space of the state and goal estimations. As depicted in Figs. 7(b) and 7(c) they display – in addition to the previously mentioned overlays – the probability distribution for the predicted position for $H = 15$ and final goal in the environment, respectively. Higher probabilities are indicated with 'warmer' tones (closer to red).

The state prediction probability displayed in the center row has been computed by applying Eq. (15) to the cells of a regular grid. Since the augmented state is 4-dimensional, we have chosen to project the probability over the current position plane, thus not showing the predicted goal.

For the right column, we have applied Eq. (15) to the cells of a regular grid, much like for the center column, but this time we have projected the probability over the intended position (goal) plane.

An interesting feature of our environment is that it includes two types of moving objects (*ie* pedestrians and vehicles). Since these objects follow different motion patterns, this has considerable influence in the prediction process. For example, for $t = 10$, we may see that there are two highly probable goals. This is interesting because they correspond to a pedestrian's destination (the building entrance) and a vehicle's destination (a lane's end). As the vehicle moves further, it becomes quickly associated with a vehicles' goal and, by $t = 82$ the only two goals with a significant probability correspond to vehicles' destinations.

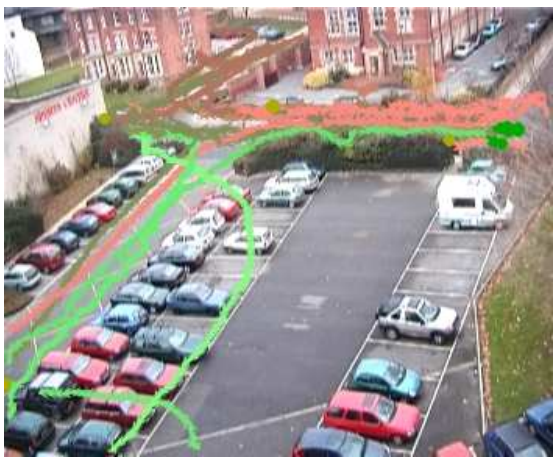


Fig. 8. Projection on the image of the learned structure after 100 trajectories have been processed (Leeds data set).

Also noteworthy, is that predicted states at $H = 15$ seem to be considerably close from the current object position. The reason is that, in this data set, objects move very slowly with respect to the size of the image, especially when they are far from the camera. This effect is much less noticeable on simulated data (not shown here), because all its coordinates are referred to the ground plane.

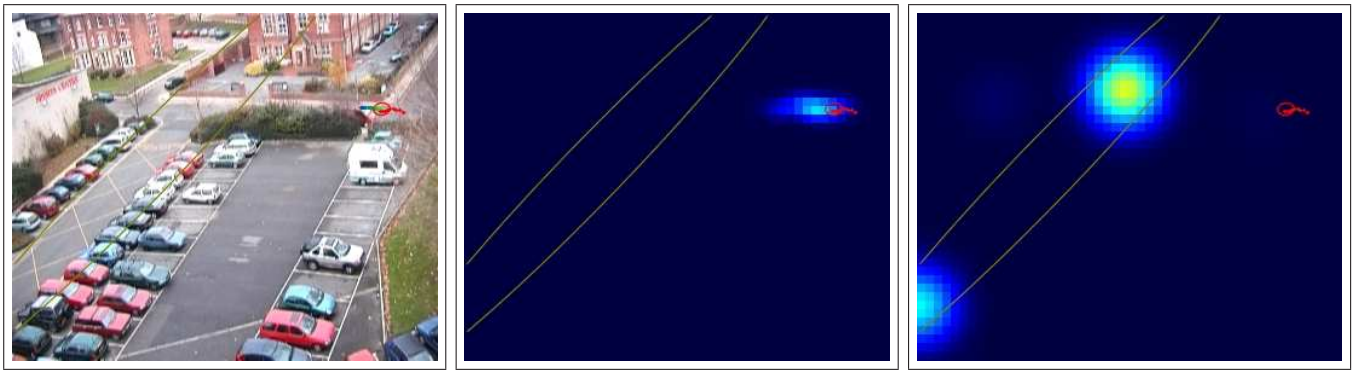
B. Measuring Prediction Accuracy

We have evaluated our prediction results using the average error for a complete data set containing K observation sequences. This has been computed as the expected distance between the predicted position for a time horizon H and the effective observation O_{t+H}:

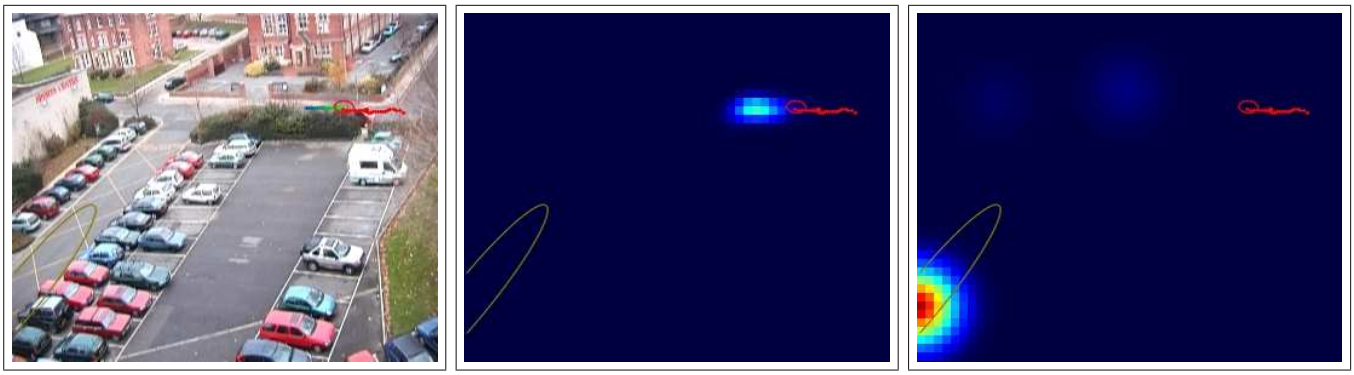
$$\langle E \rangle = \frac{1}{K} \sum_{k=1}^{K} \frac{1}{T^k - H} \sum_{t=1}^{T^k - H} \sum_{i \in \mathcal{S}} P([S_{t+H} = i] \mid O^k_{1:t}) \| O^k_{t+H} - \mu_i \|^{1/2} \tag{16}$$

C. Model size vs. Prediction Accuracy

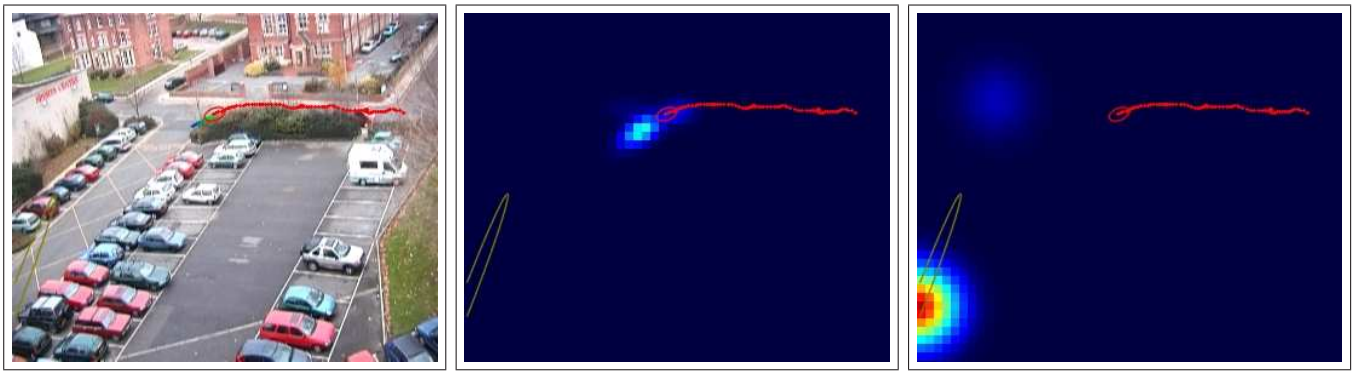
Figures 10(a) and 10(c) show the model size (number of edges) and the average prediction error as a function of the total number of processed trajectories, for the real and simulated environments, respectively. As one could expect, the average error decreases as more trajectories have been processed and, at the same time, the model's growth quickly decelerates as existing knowledge covers more of the observed motion patterns. In the case of real data, it is also possible to see some sudden jumps in the size of the model that correspond to the addition of motion patterns that occur in regions where no motion has been observed previously or that differ significantly from all the rest.



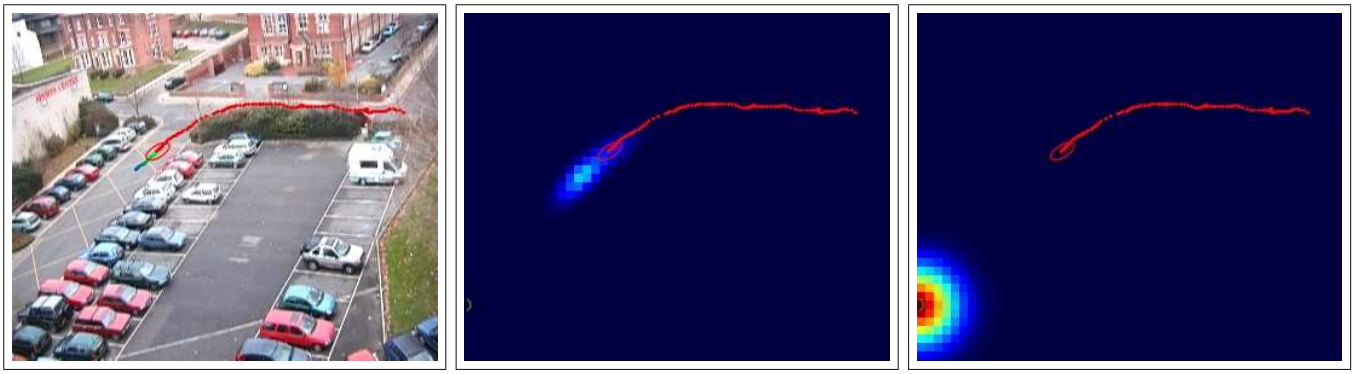
(a) t = 10



(b) t = 30



(c) t = 82



(d) t = 110

Fig. 9. Example of a sequence of predictions for an obstacle moving in the Leeds environment. See § V-A for details.

D. Model size vs. Processing time

Figures 10(b) and 10(d) plot the time taken by prediction (middle line) and learning (lower line) with respect to the number of processed trajectories. The model size (upper line) is also given as a reference.

As it may be expected, time seems to depend linearly on the

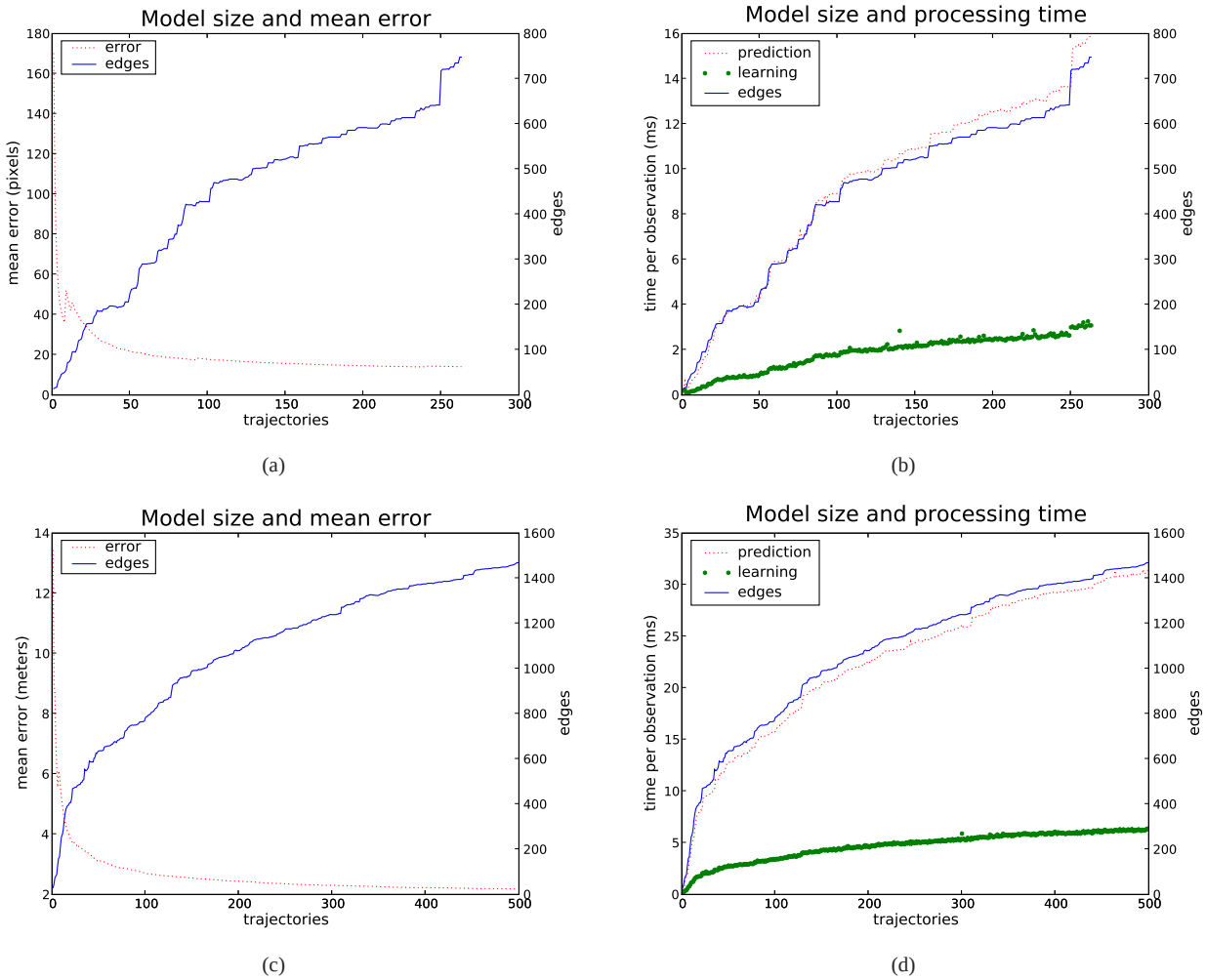


Fig. 10. Error and computation times. First row: real data. Bottom row: synthetic data.

model size. Moreover, prediction times are below 16 ms per observation for real data and 35 ms for simulated data. This is explained in part by the fact that the simulated environment is much bigger than the real one. Even in this case, prediction times are obtained at full camera frame rate.

It is also interesting to note that in the case of learning, times per observation are below 5 ms which means that a ten second trajectory requires about slightly more than one second to be learned.

E. Modeling motion with cycles

An interesting question is to see if an how our approach could deal with motion patterns containing cycles. As an example, let us imagine an 800 m track running event, where runners have to go twice around the race-track in order to finish the race. It is evident that the situations at 395 and 795 meters are very different, in the first one, the competitor will continue to run for another turn, while in the second one it will stop after five meters. Since our motion prediction approach as it is defined, only distinguishes discrete states in terms of their position and speed, it is not able to deal with

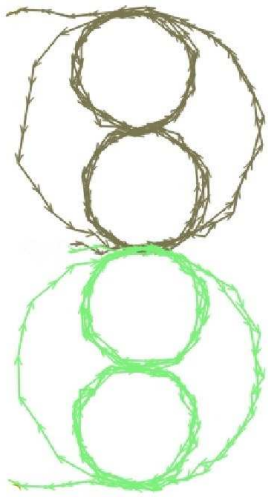
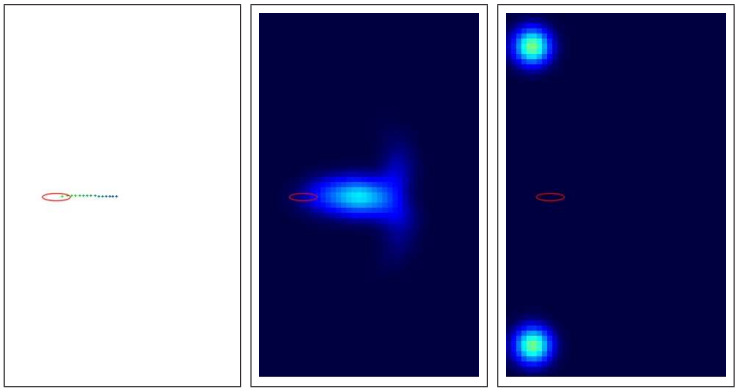
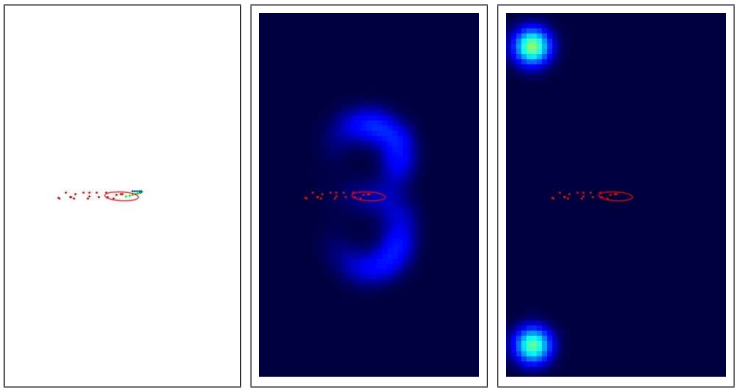


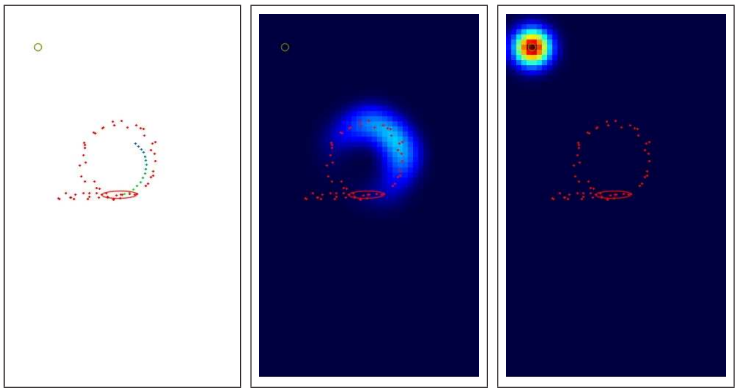
Fig. 11. Examples of cyclic trajectories. One trajectory per motion pattern is shown, the upper (gray) and lower (green) trajectories corresponds to left and right turns, respectively. See § V-E for further details.



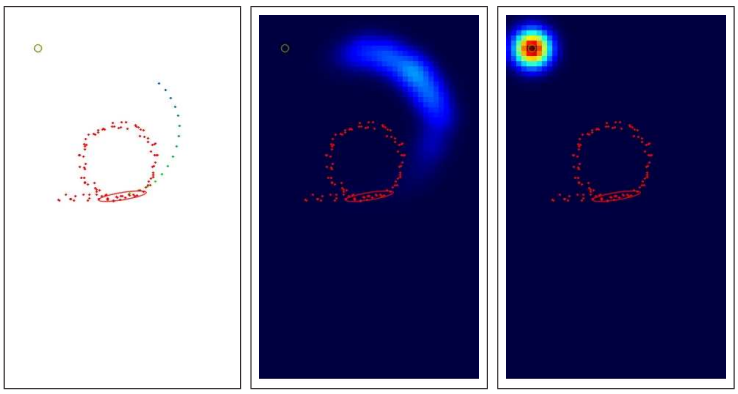
(a) t = 1



(b) t = 20



(c) t = 60



(d) t = 100

Fig. 12. Prediction example for the cycle data set.

this situation, resulting in equiprobable predictions of stopping and continuing at both 395 and 795 meters.

Although this example may seem contrived, this situation arises frequently for other kinds of motion, for example for gesture recognition. Here, we propose a simple solution to this problem: including the time variable t in observations. To illustrate this we have prepared a synthetic data set consisting of two symmetric motion patterns (see Fig. 11) where the object describes one and a half big circles, interleaved with two smaller cycles at the top and bottom. The only difference between both patterns is that one always turns to the left and the other one to the right.

Of course, including the $time$ variable in observations implies augmenting the observation covariance matrix, to include the time variable. Without going into the parameter details, Fig. 12 illustrates our algorithm working in this data set.

Notice how object's positions for $t = 20$, $t = 60$ and $t = 100$ are very similar. Nevertheless predictions are clearly different. At $t = 20$ the object has not even turned, hence, the estimated probability for both goals practically the same. After one turn, at $t = 20$, the algorithm has identified the left turning motion pattern, and it predicts that the object will still perform one more small turn. Finally, at $t = 100$ the algorithm recognizes that the object has performed two small turns and thus predicts that the object will continue to follow the big one.

As our example shows, by simply redefining observations, our algorithm is able to deal with cyclic motion patterns without further modifications. Moreover, moderated temporal misalignments may be easily dealt with by modifying the covariance matrix.

VI. Conclusions

We have developed a novel extension to HMMs which is able to learn both the models parameters and structure incrementally. We have applied this extension to vehicle and pedestrian motion by defining an augmented state which adds the intended goal to the classic state variables. We have validated our approach using real and synthetic data and the obtained results show both good prediction accuracy and real-time applicability. Moreover, our approach improves upon other HMM based techniques by implementing a model of the object's intentions.

In the medium term, future work includes applying our approach to gesture recognition on video sequences, with the intention of validating its applicability to higher dimensional spaces. On the practical side, we will work in performing in-depth comparison with other motion learning and prediction approaches.

Appendix
Forward, Backward and Joint Observation Probabilities

For the sake of completeness, we include here the pseudo code for the computation of the forward (Alg. 1) and backward (Alg. 2) probabilities.

The joint observation probability (*ie* the probability of an observation sequence given the model) is computed from the forward probabilities using (17).

$$\begin{aligned} P(O_{1:T}|\lambda) &= \sum_{i=1}^{N} P(O_{1:T}, S_T = i|\lambda) \\ &= \sum_{i=1}^{N} \alpha_T(i) \end{aligned} \tag{17}$$

References

[1] M. Bennewitz, W. Burgard, G. Cielniak, and S. Thrun, "Learning motion patterns of people for compliant robot motion," *Internationl Journal of Robotics Research*, vol. 24, no. 1, pp. 31–48, January 2005.

Algorithm 1: $Forward\_probabilities(O_{1:T}, \lambda)$

input :
- An observation sequence $O_{1:T}$
- HMM parameters $\lambda = \{\pi, b, A\}$

output : Forward Probabilities $\alpha_t(i)$

begin
  for $i = 1$ *to* N **do**
    $\alpha_1(i) = P([S_1 = i])P(O_1|[S_1 = i])$
  end
  for $t = 2$ *to* T **do**
    for $j = 1$ *to* N **do**
      $\alpha_t(j) = \left[\sum_{i=1}^{N} \alpha_t(i)P([S_t = j]|[S_{t-1} = i])\right] P(O_t|[S_t = j])$
    end
  end
end
return *all* $\alpha_t(i)$

Algorithm 2: $Backward\_probabilities(O_{1:T}, \lambda)$

input :
- An observation sequence $O_{1:T}$
- HMM parameters $\lambda = \{\pi, b, A\}$

output : Backward probabilities $\beta_t(i)$

begin
  for $i = 1$ *to* N **do**
    $\beta_T(i) = 1$
  end
  for $t = T - 1$ *down to* 1 **do**
    for $i = 1$ *to* N **do**
      $\beta_t(i) = \sum_{j=1}^{N} P([S_{t+1} = j]|[S_t = i])P(O_{t+1}|[S_{t+1} = j])\beta_{t+1}(j)$
    end
  end
end
return *all* $\beta_t(i)$

[2] W. Hu, X. Xiao, Z. Fu, D. Xie, T. Tan, and S. Maybank, "A system for learning statistical motion patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 9, pp. 1450–1464, September 2006.

[3] D. Vasquez and T. Fraichard, "Intentional motion on-line learning and prediction," in *Field and Service Robotics*, Port Douglas, Australia, July 2005. [Online]. Available: http://emotion.inrialpes.fr/bibemotion/2004/VF04

[4] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Readings in speech recognition*, pp. 267–296, 1990.

[5] M. Walter, A. Psarrow, and S. Gong, "Learning prior and observation augmented density models for behaviour recognition," in *Proc. of the British Machine Vision Conference*, 1999, pp. 23–32.

[6] D. Makris and T. Ellis, "Spatial and probabilistic modelling of pedestrian behavior," in *Proc. of the British Machine Vision Conference*, Cardiff, UK, 2002, pp. 557–566.

[7] N. M. Oliver, B. Rosario, and A. P. Pentland, "A bayesian computer vision system for modeling human interactions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 831–843, August 2000.

[8] M. Brand, N. Oliver, and A. Pentland, "Coupled hidden markov models for complex action recognition," in *In Proc. of the 1997 Conf. on Computer Vision and Pattern Recognition*, San Juan (PR), 1997, pp. 994–999.

[9] S. Gong and T. Xiang, "Recognition of group activities using dynamic probabilistic networks," in *In Proceedings of the Ninth IEEE International Conference on Computer Vision*, vol. 2, Washington (US), 2003, pp. 742–749.

[10] T. Xiang and S. Gong, "Beyond tracking: Modelling activity and understanding behaviour." *International Journal of Computer Vision*, 2006.

[11] M. Bennewitz, W. Burgard, and S. Thrun, "Learning motion patterns of persons for mobile service robots," in *Proc. of the IEEE Int. Conf. On Robotics and Automation*, Washington, USA, 2002, pp. 3601–3606.

[12] D. Makris and T. Ellis, "Finding paths in video sequences," in *Proc. of the British Machine Vision Conference*, 2001, pp. 263–272.

[13] D. Buzan, S. Sclaroff, and G. Kollios, "Extraction and clustering of motion trajectories on video," in *Proc. of the Int. Conf. on Pattern Recognition*, Cambridge, UK, 2004.

[14] D. Vasquez and T. Fraichard, "Motion prediction for moving objects: a statistical approach," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, New Orleans, LA (US), April 2004, pp. 3931–3936. [Online]. Available: http://emotion.inrialpes.fr/bibemotion/2004/VF04

[15] I. Junejo, O. Javed, and M. Shah, "Multi feature path modeling for video surveillance," in *Proc. of the 17th conference of the International Conference on Pattern Recognition (ICPR)*, 2004, pp. 716–719.

[16] X. Wang, K. Tieu, and E. Grimson, "Learning semantic models by trajectory analysis," Massachussets Institute of Technology, Tech. Rep., 2006.

[17] N. Johnson and D. Hogg, "Learning the distribution of object trajectories for event recognition," in *Proc. of the British Machine Vision Conference*, vol. 2, September 1995, pp. 583–592.

[18] N. Sumpter and A. Bulpitt, "Learning spatio-temporal patterns for predicting object behaviour," *Image and Vision Computing*, vol. 18, no. 9, pp. 697–704, 2000.

[19] W. Hu, D. Xie, T. Tieniu, and S. Maybank, "Learning activity patterns using fuzzy self-organizing neural network," *IEEE Trans. on Systems, Man and Cybernetics*, vol. 34, no. 3, pp. 1618–1626, June 2004.

[20] C. Stauffer and E. Grimson, "Learning patterns of activity using real-time tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 747–757, August 2000.

[21] L. Liao, D. Fox, and H. Kautz, "Learning and inferring transportation routines," in *Proc. of the National Conf. on Artificial Intelligence AAAI-04*, 2004.

[22] H. Bui, S. Venkatesh, and G. West, "Policy recognition in the abstract hidden markov models," *Journal of Artificial Intelligence Research*, vol. 17, pp. 451–499, 2002.

[23] S. Osentoski, V. Manfredi, and S. Mahadevan, "Learning hierarchical models of activity," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Sendai, Japan, 2004.

[24] R. A. Howard, *Dynamic Programming and Markov Process*. Cambridge (US): MIT Press, 1960.

[25] A. F. Foka and P. E. Trahanias, "Predictive autonomous robot navigation," in *In Proc of the IEEE/RSJ Int. Conf. on Intelligent Robots and System*, vol. 1, Lausanne (CH), October 2002, pp. 490–495.

[26] A. Bruce and G. Gordon, "Better motion prediction for people-tracking," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, New Orleans, US, April 2004.

[27] H. Dee and D. Hogg, "Detecting inexplicable behaviour," in *In Proceedings of the British Machine Vision Conference*, Kingston (UK), 2004, pp. 49–55.

[28] B.-H. Juang, S. E. Levinson, and M. M. Sondhi, "Maximum likelihood estimation for multivariate mixture observations of markov chains," *IEEE Transactions on Information Theory*, vol. 32, no. 2, pp. 307–309, March 1986.

[29] D. Vasquez, "Incremental learning for motion prediction of pedestrians and vehicles," Ph.D. dissertation, Institut National Polytechnique de Grenoble, Grenoble, FR, February 2007. [Online]. Available: http://emotion.inrialpes.fr/bibemotion/2007/Vas07

[30] J. Jockusch and H. Ritter, "An instantaneous topological map for correlated stimuli," in *In Proc. of the International Joint Conference on Neural Networks*, vol. 1, Washington (US), July 1999, pp. 529–534.

[31] R. M. Neal and G. E. Hinton, "A new view of the em algorithm that justifies incremental, sparse and other variants," in *Learning in Graphical Models*, M. I. Jordan, Ed. Kluwer Academic Publishers, 1998, pp. 355–368.

[32] H.-M. Dee, "Explaining visible behaviour," Ph.D. dissertation, University of Leeds, 2005.