HAL
archives-ouvertes.fr

Growing Hidden Markov Models: An Incremental Tool for Learning and Predicting Human and Vehicle Motion

Dizan Alejandro Vasquez Govea, Thierry Fraichard, Christian Laugier

▶ **To cite this version:**

Dizan Alejandro Vasquez Govea, Thierry Fraichard, Christian Laugier. Growing Hidden Markov Models: An Incremental Tool for Learning and Predicting Human and Vehicle Motion. The International Journal of Robotics Research, SAGE Publications, 2009, 28 (11-12), pp.1486-1506. inria-00430582

HAL Id: inria-00430582
https://hal.inria.fr/inria-00430582

Submitted on 25 Nov 2009

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Int. Journal of Robotics Research, 28(11-12), 2009

Growing Hidden Markov Models: an Incremental Tool for Learning and Predicting Human and Vehicle Motion

Dizan Vasquez* Thierry Fraichard† 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) 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. The proposed approach has been evaluated using synthetic and real trajectory data. In our experiments our approach consistently learned motion models that were more compact and accurate than those produced by two other state of the art techniques.

1 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

*Swiss Federal Institute of Technology Zurich.
†INRIA, CNRS/LIG & Grenoble University.

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 future 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. (2005) use the expectation-maximization algorithm to cluster trajectory data gathered with a laser scanner, and Hu et al. (2006) 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 techniques 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 incrementally learn motion patterns, 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. It builds on our previous work (Vasquez and Fraichard 2005; Vasquez et al. 2007) on a unified extension to Hidden Markov Models (HMM) (Rabiner 1990), a probabilistic framework which is very popular in the motion pattern learning literature (*eg* Walter et al. 1999; Makris and Ellis 2002; Bennewitz et al. 2005). This extension, named Growing HMM (GHMM) enables incremental and on-line learning of the parameters and the structure of the model.

In order to evaluate our approach, we have compared it against two other state of the art techniques (Bennewitz et al. 2005; Hu et al. 2006) using synthetic data from a simulator as well as real data obtained with a visual tracking system. In our experiments, our technique has consistently outperformed the other two approaches regarding model size and accuracy.

The rest of this paper is structured as follows: Section 2 provides an overview of motion pattern learning, focusing on techniques based on HMMs. Section 3 presents Growing Hidden Markov Models. In Section 4 the application of GHMMs to our particular problem is discussed. Our experimental test scenarios are described in Section 5, which also presents a qualitative analysis of the obtained results and discusses the real-time applicability of our approach. Section 6 presents the two approaches that have been used for comparison and evaluation purposes, it also introduces the performance measures and discusses

our quantitative results. Finally, we present our conclusions in Section 7.

2 Related work

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 of HMM-based approaches, which constitute the basis of our proposed approach.

2.1 Literature overview

2.1.1 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 undertaking 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 and speed). This makes them better suited for applications like video-surveillance or scene understanding than for tracking or motion prediction.

A good example of this type of approaches is described in the work of Oliver et al. (2000): they use Coupled HMMs (Brand et al. 1997) 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 (Gong and Xiang 2003) and (Xiang and Gong 2006), allowing for interactions between more than two objects. A more recent approach proposed by Hoogs and Perera (2008), models behavior with a Dynamic Bayesian Network containing semantic states obtained from a predefined ontology.

2.1.2 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. (2002), which uses expectation-maximization to perform the clustering. Other algorithms include hierarchical clustering (Makris and Ellis 2001; Buzan et al. 2004; Vasquez and Fraichard 2004; Hu et al. 2006), graph cutting (Junejo et al. 2004), and custom pairwise clustering algorithms (Wang et al. 2006).

To apply the obtained trajectory prototypes to perform tracking or motion prediction, a probabilistic framework is often used. This permits to represent the uncertainties associated to 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 (1995), 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 in (Sumpter and Bulpitt 2000) and (Hu et al. 2004), which by explicitly modeling time obtained performance comparable to that of probabilistic models. A different idea has been explored by Stauffer and Grimson (2000), 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.

2.1.3 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 in which it is involved. 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. (2004) have used a hierarchical extension to HMMs – Abstract Hidden Markov Models (AHMM)(Bui et al. 2002) – 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(Osentoski et al. 2004) 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 (Howard 1960) a probabilistic planning technique, which illustrates the connection between planning and motion prediction.

Other goal based approaches include (Foka and Trahanias 2002; Bruce and Gordon 2004) and (Dee and Hogg 2004). 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 the problem that the object's evolution towards the goal is modeled using overly simplistic mechanisms (*eg* linear interpolation for (Foka and Trahanias 2002)), leading to unreliable physical-state estimations.

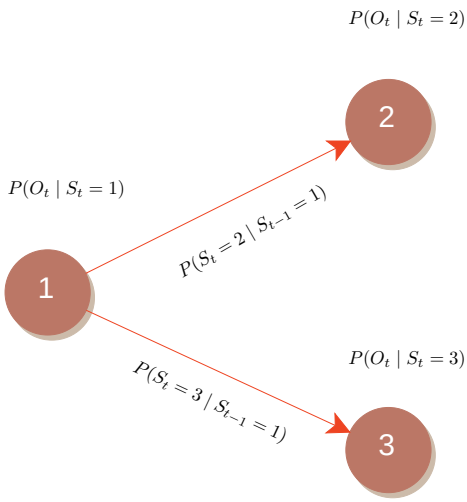
2.1.4 Other approaches

The problem of building models of temporal processes has also been studied in other disciplines. For example, Dixon et al. (2004) have proposed a programming-by-demonstration framework that represents motion using HMMs that are learned with a model merging algorithm very similar in spirit to the one proposed by Stolcke and Omohundro (1993). Another programming-by-demonstration framework has been proposed by Calinon and Billard (2007) it represent trajectories as Gaussian Mixture Models and uses Gaussian Mixture Regression for inference. The paper proposes two incremental parameter learning strategies, while the model structure (*ie* number of Gaussians) is fixed *a priori*.

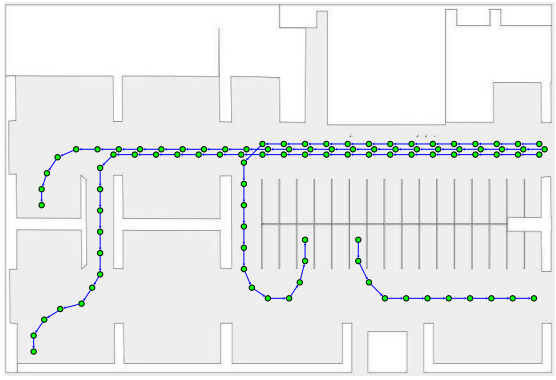
Finally, we would like to mention the work of Kulic et al. (2008) in the context of whole body motion. They propose an approach based on a generalization of HMMs called Factorial Hidden Markov Models. The approach assumes a chain-like structure and fixed number of discrete states and uses an incremental algorithm to estimate the model parameters and to instantiate new motion models. The approach builds an HMM out of every input observation sequence and uses a symmetric version of the Kullback-Leibler divergence as the basis for a Hierarchical Agglomerative clustering algorithm where new motion models are inserted according to a threshold.

2.2 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. (1986) and Rabiner (1990) for a deeper introduction to the subject.



(a)



(b)

Figure 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 may be done at two different levels:

- *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. (1999) assume that the number of motion patterns is known *a priori* and define the structure using a different chain-like graph for every motion pattern, then, parameters are learned using the Expectation-Maximization algorithm; Bennewitz et al. (2005) 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 (2002) 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.

3 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

[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 (Vasquez 2007) for more details.

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. The learning algorithm can be considered incremental according to the three-point definition proposed by Langley (1995) since: a) it inputs one observation sequence at a time, b) it does not reprocess any previous data, and c) it retains only one knowledge structure in memory.

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, which make it less general than conventional HMMs:

- 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 (Jockusch and Ritter 1999). For parameter learning, we basically have adapted the incremental Expectation-Maximization approach proposed by Neal and Hinton (1998) 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.

[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.

3.1 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:

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

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 $b = \{\Sigma, \mu_1, \cdots, \mu_N\}$.

[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 $O_1\ O_2\ \cdots\ O_{t-1}\ O_t$.

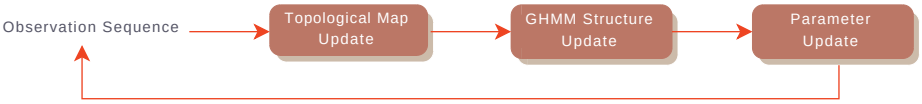


Figure 2: Overview of the GHMM learning algorithm.

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.

3.2 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.

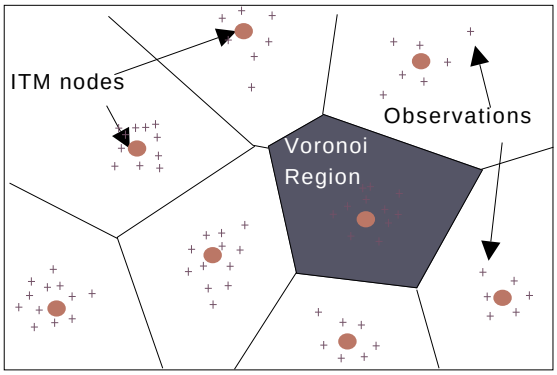
The topological map is updated for every available observation O_t using the ITM algorithm which has the following properties:

- 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).

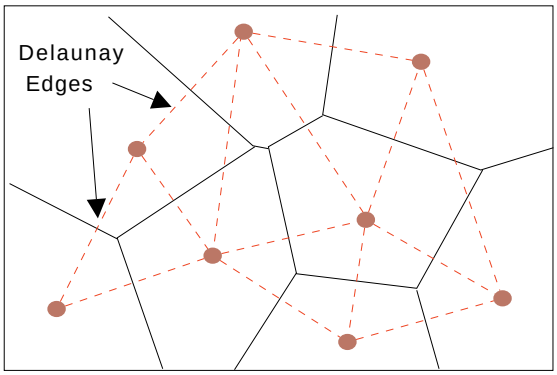
The ITM algorithm consists of the following steps (*cf* (Jockusch and Ritter 1999)):

1. **Matching**: find the nearest b and second nearest s nodes to O_t. We use the Mahalanobis distance – Eq. (3) – with the same Σ than for observation probabilities.

[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.



(a) Nodes and regions



(b) Edges

Figure 3: Example ITM space decomposition

$$d^2_\Sigma(u, v) = (u - v)^T \Sigma^{-1} (u - v) \tag{3}$$

where u and v are two reference vectors.

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 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.

3.3 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* § 3.4).

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: $\mu_i = w_i$.

3.4 Updating the Parameters

Parameter learning takes place once per input sequence, after all the observations have been processed by the structure learning step. The GHMM learning algorithm reestimates the parameters using an incremental version of the Baum-Welch technique based on the work from Neal and Hinton (1998) 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 § 3.3) 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 § 3.3.

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:

$$\hat{\pi}_i \leftarrow \frac{\alpha_1(i)\ \beta_1(i)}{P_O} \tag{4}$$

$$\pi_i \leftarrow \frac{(k-1)\pi_i + \hat{\pi}_i}{k} \tag{5}$$

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 (6) and (7).

$$\hat{a}_{i,j} \leftarrow \frac{\sum_{t=2}^{T} \alpha_{t-1}(i) a_{i,j} P(O_t|[S_t = j]) \beta_t(j)}{\sum_{t=2}^{T} \alpha_{t-1}(i) \beta_{t-1}(i)} \tag{6}$$

$$a_{i,j} \leftarrow \frac{(k-1)a_{i,j} + \hat{a}_{i,j}}{k} \tag{7}$$

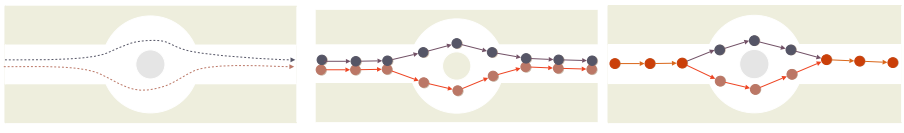
These steps constitute a single iteration of incremental Baum-Welch. The reason to use Equations (5) and (7) is that they are equivalent to dividing the sum of the weight by the number of trajectories in order to obtain an average weight value; these equations, together with the preset values π_0 and a_0, are the elements that enable incremental learning with an evolving number of states.

For the sake of comparison, we also performed early tests with straightforward Baum-Welch, *ie* using only (4) and (6) to make the update, but the learned parameters were too heavily biased towards recent observation sequences.

4 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 in function of their intention to reach a particular state (*ie* their goal). Accordingly, we model the object's motion as a sequence of *augmented state vectors*, composed of two sets of variables describing its *current* and *goal* state, 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) Input trajectories (b) Trajectory based (c) Goal based

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

4.1 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 and velocities (x, y, x', y') and thus, that the augmented state of the object consists of its current position and velocities, as well as its goal position $(x, y, x', y', \hat{x}, \hat{y})$. It should be noted, however, that our approach is applicable to spaces of arbitrary dimensions.

We assume that observations are available in the form of estimates of the object's coordinates and velocities $O_t = (x_t, y_t, x'_t, y'_t)$ – although, as in the case of the continuous state, is it also possible to include other variables, such as the size or orientation of the object. Since learning is performed on the basis of complete observation sequences, we assume that the position of the last observation $O_T = (x_T, y_T, x'_T, y'_T)$ 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:

$$\bar{O}_{1:T} = \{(x_1, y_1, x'_1, y'_1, x_T, y_T), (x_2, y_2, x'_2, y'_2, x_T, y_T), \cdots, (x_T, y_T, x'_T, y'_T, x_T, y_T)\}$$

4.2 Probabilistic Model

Since our approach is based on GHMMs, it uses the same probabilistic model that has been described in §3.1. Nevertheless, we also need to distinguish between the 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: $O_t = [O'_t, O''_t]$.

To define the JPD, we will assume that the current and intended components of observations are conditionally independent given the current state[5], enabling us to rewrite the observation probability as:

$$P(O_t|S_t) = P(O'_t\ O''_t|S_t) = P(O'_t|S_t)P(O''_t|S_t) \tag{8}$$

and the whole JPD as:

[5]When using a visual tracker this is reasonable, since it can be safely assumed that the current estimate of the pose depends only on the current pose of the object and its projection on the camera image, not on its goal. The same can be said about the goal, which depends only on the final pose of the object and not on the intermediate readings of the camera

$$P(S_{t-1}\ S_t\ O'_t\ O''_t) = P(S_{t-1})P(S_t|S_{t-1})P(O'_t|S_t)P(O''_t|S_t) \tag{9}$$

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:

$$P(O'_t \mid [S_t = i]) = \mathbf{G}(O'_t;\ \mu'_i,\ \Sigma') \tag{10}$$

$$P(O''_t \mid [S_t = i]) = \begin{cases} \mathbf{U}_{O''_t} & \text{if } O''_t \text{ is not available} \\ \mathbf{G}(O''_t;\ \mu''_i,\ \Sigma'') & \text{otherwise} \end{cases} \tag{11}$$

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:

$$P(O_t \mid [S_t = i]) = \frac{1}{Z}\mathbf{G}(O_t;\ \mu_i,\ \Sigma) \tag{12}$$

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

$$\Sigma = \begin{bmatrix} \Sigma' & 0 \\ 0 & \Sigma'' \end{bmatrix} \tag{13}$$

and Z is a normalization variable, which enables computation of the uniform on the goal component using the same Gaussian representation. Since during prediction the intended part of the augmented observation is not available, this is done by setting[6] $O''_t = 0$.

4.3 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 immediately takes place 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:

$$P(S_t \mid O_{1:t}) = \frac{1}{Z}P(O_t|S_t)\sum_{S_{t-1}}[P(S_t|S_{t-1})P(S_{t-1}|O_{1:t-1})] \tag{14}$$

[6]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 (11).

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:

$$P(S_{t+H} \mid O_{1:t}) = \sum_{S_{t+H-1}} P(S_{t+H}|S_{t+H-1})P(S_{t+H-1}|O1:t) \tag{15}$$

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:

$$P(O_{t+H} \mid O_{1:t}) = \frac{1}{Z} \sum_{S_{t+H}} P(S_{t+H} \mid O_{1:t})P(O_{t+H} \mid S_{t+H}) \tag{16}$$

5 Experimental Results

We have implemented our approach and conducted extensive experiments with both synthetic and real data sets. This section starts by describing the scenarios and the data sets used (§5.1), then it discusses qualitative results in order to provide an intuition of how the approach works (§5.2). Finally, the real-time performance of our approach is studied in §5.3. Performance evaluation regarding prediction accuracy and model size are deferred to §6 wherein our approach will also be compared to two other motion prediction techniques.

5.1 Test Scenarios

All the experiments discussed on this paper are based on data obtained in two different parking lot environments. A real one and a simulated one.

5.1.1 Real Environment: Leeds

The first experimental environment is a parking lot located at the University of Leeds[7] and a small street section where both pedestrians and vehicles move.

The video input has been captured by a camera located high above the parking and covering a wide area, which has allowed us to conduct our experiments in the image plane, without any projection on a world reference system.

[7]We would like to thank Hannah Dee and the University of Leeds for letting us use this data set.

The tracking system proposed by Magee (2004) has been used to obtain the observation sequences, however, it is important to note that then trajectories have been hand-edited to correct tracking problems. As an indicator, approximately 20% of the 269 trajectories in the data set have been altered in some way, and some have been entirely tracked by hand (Dee 2005).

Solving these tracking problems is an open issue in the multi object tracking community, and is out of the scope of this paper. This is a drawback of most approaches in the literature – including ours and the two other approaches presented in section 6 – which depend on an accurate segmentation of the observation sequences. On the other hand, this problem does not have any effect in our comparison, which focuses on the evaluation of the learning algorithm and compare the different approaches using the same input data sets.

The observations on this dataset have been sampled at approximately 10 Hz and contain only position data, therefore, we have used a Kalman Filter to estimate the corresponding velocities. The complete data set is depicted in fig. 5.

Figure 5: Leeds data set.

5.1.2 Synthetic Environment: INRIA

Due to the difficulty of obtaining real data, we have decided – in addition to the real data set that we have described above – to develop a parking lot and trajectory simulator in order to have a more thorough experimental testbed. This allows us to test particular situations without the logistic difficulties posed

by executing scripted actions in a real environment as the Leeds parking lot. It is also a practical way of generating big amounts of data which are free of the problems that are common to tracking systems.

A graph-like structure has to be defined first (Fig. 6). Nodes of the graph represent control points with an associated speed and position variance, *ie* way points in the environment. They also have flags indicating whether they are start, intermediate or end points of a trajectory. Nodes are connected by oriented edges, which indicate the possibility to go from one node to another.

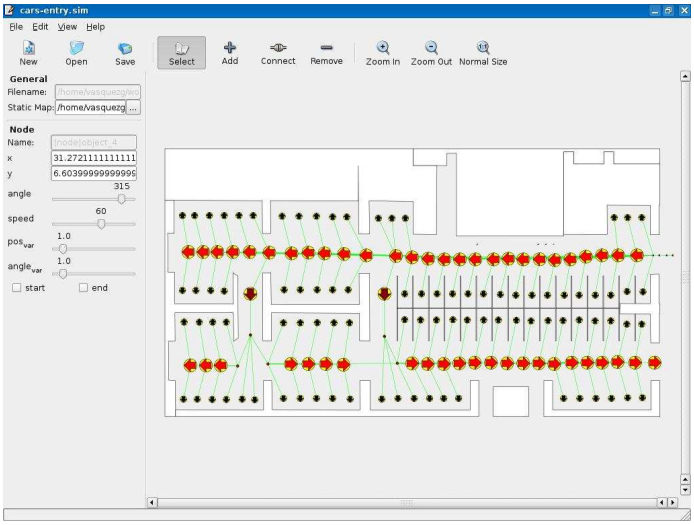


Figure 6: The parking lot simulator showing a model of the INRIA lab parking lot. Point size and color correspond to position variance and speed, respectively.

Once this graph has been defined, the simulator works by choosing at random a start and an end point (according to the corresponding flags), and then obtaining a sequence of nodes by finding the shortest path between them. For every node in the sequence, a point is generated by adding Gaussian noise to the node's position. Finally, the trajectory is generated by applying spline interpolation to the point sequence and sub-sampling it according to the simulated sampling rate and object's speed. As in the case of the Leeds data set, we have assumed a frame rate of 10 Hz. Fig. 7 depicts an example trajectory data set generated by the simulator.

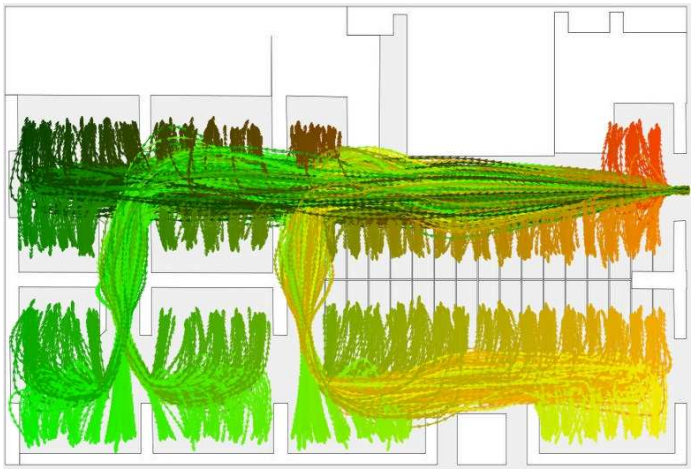


Figure 7: Synthetic parking data set, from the model displayed in fig. 6

5.2 Qualitative Results

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

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

Figures 9 and 10 illustrate a typical example of the prediction process on the real data set. Fig. 10 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 fea-

turing different overlays, as shown in Fig. 9(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 mean values are displayed as points colored from blue (for $H = 1$) to green (for $H = 15$).

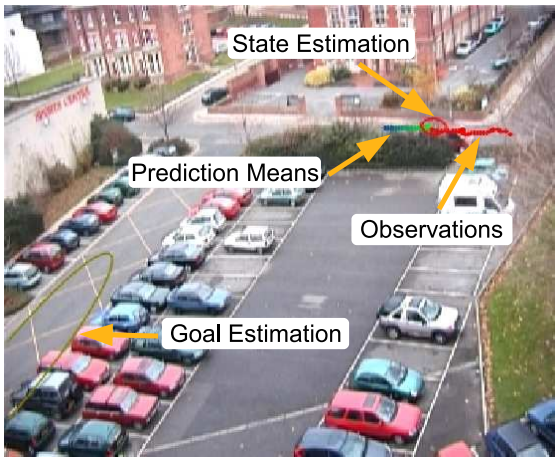
The center and right images are 2D projections in the image space of the state and goal estimations. As depicted in Figs. 9(b) and 9(c) they display – in addition to the previously mentioned overlays – the probability distribution for the predicted position for $H = 15$ and final goal in the environment, 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. (16) to the cells of a regular grid. Since the augmented state is 6-dimensional, we have chosen to project the probability over the current position plane, thus not showing the predicted goal.

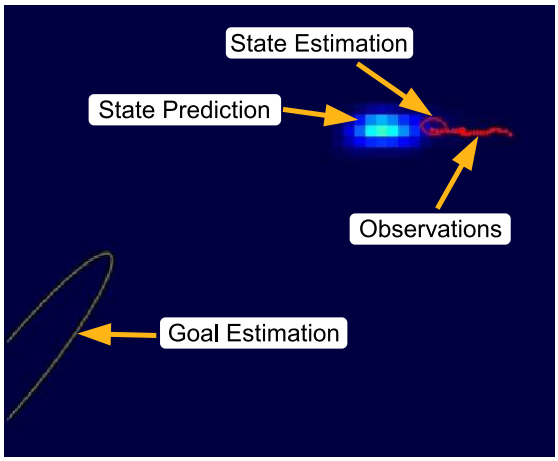
For the right column, we have applied Eq. (16) to the cells of a regular grid, much like for the center column, but this time we have projected the probability over the intended position (goal) plane.

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.

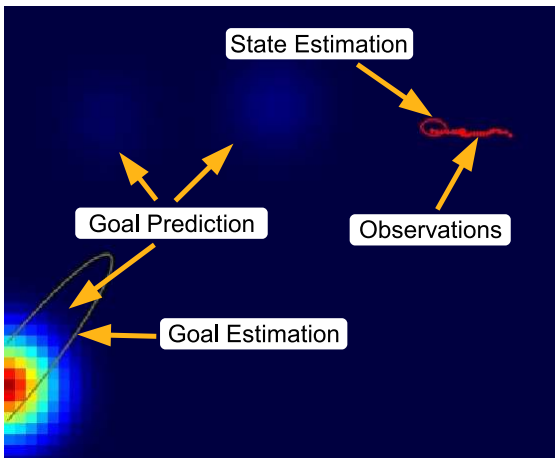
Also noteworthy, is that predicted states at $H = 15$ seem to be considerably close from the current object position. The reason is that, in this data set, objects move very slowly with respect to the size of the image, especially when they are far from the camera. This effect is much less noticeable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



(a) Left Column

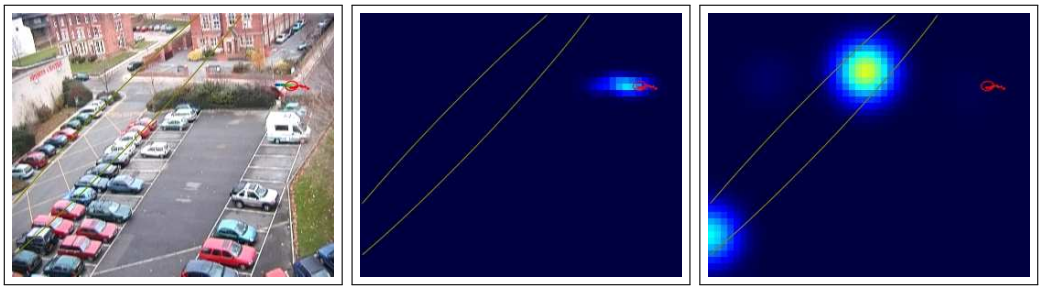


(b) Center Column

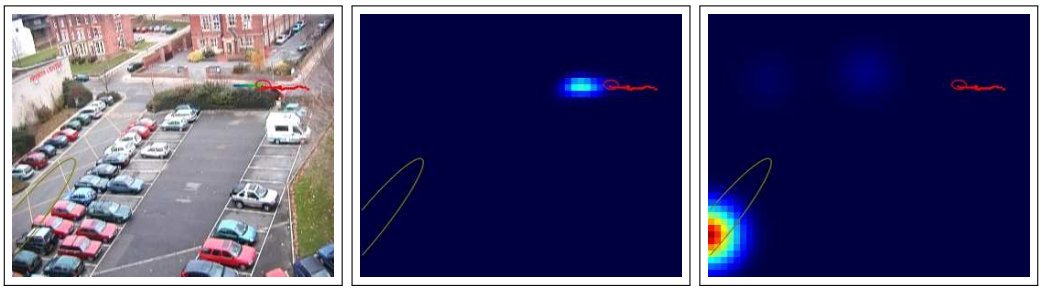


(c) Right Column

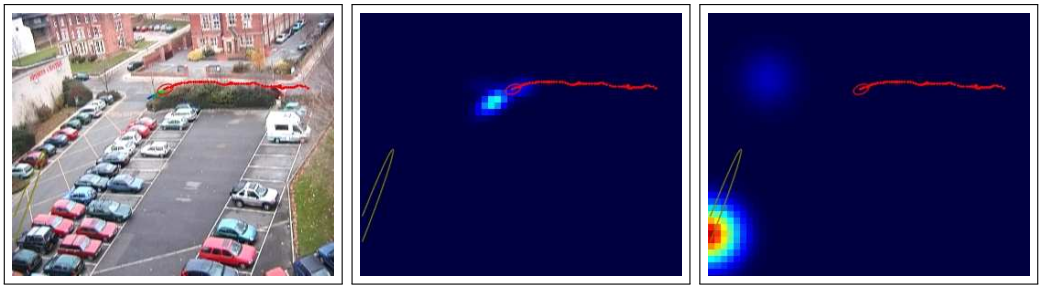
Figure 9: Explanation of Fig. 10, see § 5.2 as well.



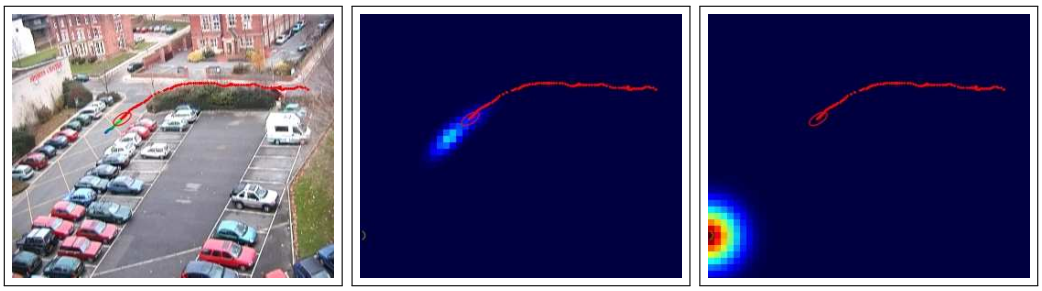
(a) t = 10



(b) t = 30

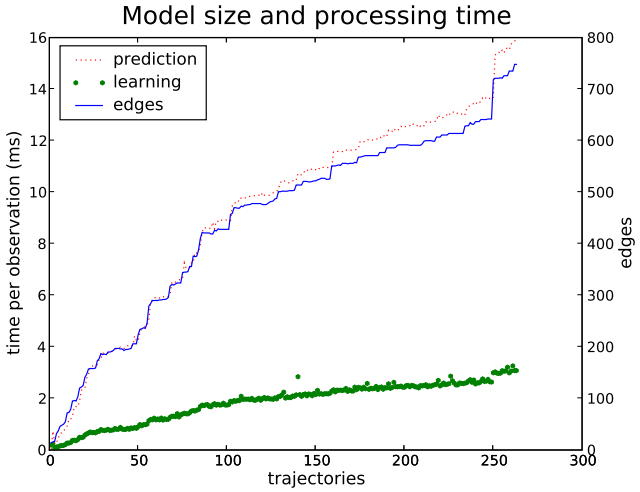


(c) t = 82

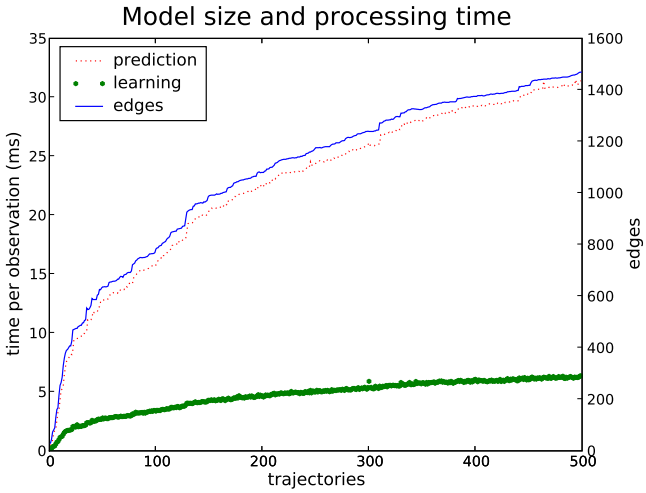


(d) t = 110

Figure 10: Example of a sequence of predictions for an obstacle moving in the Leeds environment. See § 5.2 for details.



(a)



(b)

Figure 11: Computation times. (a) Leeds data set, (b) INRIA simulator.

5.3 Processing Time

Figs. 11(a) and 11(b) plot the time taken by prediction (upper dotted line) and learning (lower dotted line) with respect to the number of processed trajectories. The model size, expressed as the number of edges in the GHMM structure (upper line) is also given as a reference. Tests have been executed in a 2 GHz Intel Duo computer running Linux.

As expected, running time seems to depend linearly on the model size. More-

over, prediction times are below 16 ms per observation for real data and 35 ms for simulated data. The longer times for the simulated environment are explained in part by the fact that this 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 slightly more than one second to be learned. Thus, the algorithm is well adapted for on-line usage in environments where it is likely to observe less than one trajectory per second.

6 Performance Evaluation

This section provides a quantitative comparison of our approach against two other recent techniques. First, we introduce those approaches, then, we describe the proposed performance measures, finally we discuss the obtained results with both real and simulated data.

6.1 Compared Approaches

We have implemented two other approaches in order to compare them to the proposed technique. These approaches have been selected among the many different approaches discussed in §2 on the basis of three criteria:

- Unsupervised learning. Since our approach learns from unlabeled data, we are interested in comparing it against similar approaches.
- Structure learning. The compared approaches should be able to estimate the size and structure of the learned model, not only the parameter values.
- Suitability for prediction. Not all the existing approaches are suited to motion prediction, which, from our point of view, requires at least two conditions: the approach should model time, at least implicitly, and it should be able to produce multi-modal predictions.

From the approaches that verify those criteria, we have selected an HMM-based approach, which applies the expectation-maximization algorithm for learning, and a second approach that is based on hierarchical fuzzy K-means clustering. We will describe them now in further detail.

6.1.1 Expectation-Maximization Clustering

This approach, proposed by Bennewitz et al. (2005) uses the expectation-maximization (EM) algorithm to cluster together trajectories into trajectory prototypes, which

are then transformed into HMMs with the same semantics than in our approach, but limited to chain-like structures where the only valid transitions out of a state go to itself and to the next state in the chain.

We will discuss here the overall structure of the clustering algorithm, the interested reader is referred to the original paper for further details:

1. Initialize K clusters from randomly chosen observation sequences.
2. Run optimization.
3. Compute model score.
4. If the score may be improved by adding the worse scored trajectory as a cluster, add it and go to step 2.
5. If the score may be improved by removing one of the two more similar clusters, delete it and go to step 2, else stop.

Step two is the most important, it estimates the cluster parameters by iterating through two steps: a) for every observation sequence and cluster, compute the expected membership, *ie* the likelihood that the observation sequence belongs to that cluster, and b) reestimate cluster representations as an average of observation sequences, weighted by the corresponding likelihoods that were computed on the previous step. The process is stopped when parameter changes between two consecutive iterations are negligible.

Since the EM algorithm requires the number of clusters to be known *a priori*, it is necessary to estimate it somehow. This is the purpose of steps 3 to 5, which score the model using the Bayesian Information Criterion (Schwarz 1978) and then try to improve that score by adding or removing clusters. It is worth noting that our implementation differs slightly from the original paper, which worked with predefined insertion and deletion thresholds for steps four and five. Since we have found it difficult to fix those thresholds, we have preferred to always try to increase and decrease the cluster number to see if the score is improved.

Once the clusters are found, they are converted into HMMs by applying prior knowledge about how objects move. HMM states are uniformly spaced along the found clusters, and transition probabilities are fixed according to average object velocities. A final note about this approach is that it does not use velocity information on the observations – although the authors mention the possibility to extend the approach to use such information.

6.1.2 Hierarchical Fuzzy K-Means Clustering

The second approach we have implemented (Hu et al. 2006) is based on the fuzzy K-means clustering algorithm (HFKM). It has the same overall structure

than expectation-maximization, but uses different expressions to compute the expected memberships and cluster representations in step 2. Also, a different measure – the Tightness and Separation Criterion (Xie and Beni 1991) – is used to find the optimal number of clusters on steps 3 through 5.

The approach works in a hierarchical manner: a first clustering is performed on coarsely subsampled observation sequences using only pose information. Then, for every obtained cluster, the belonging observation sequences are clustered again, this time without sub-sampling and taking into account the object's velocities.

In this case, our implementation differs again from the original paper: after conducting an initial round of tests with very bad clustering results, we concluded that there is a typo in cluster representation computation formula that appears in the paper (see the appendix). After correcting it, the results improved significantly. A final difference with the EM approach is that, after the final clustering step, "anomalous" trajectories are filtered out by deleting clusters that have less than a given number of trajectories. In our experiments we have fixed this threshold to three.

6.2 Performance Measures

Comparing heterogeneous techniques – even if they solve a common problem – is a difficult task. Often their theoretical bases are too different, making it difficult to evaluate them fairly. Here we propose two measures that, despite their simplicity, still provide useful indicators about the accuracy and the parsimony of the models produced by the different approaches.

6.2.1 Measuring Model Size

The structure of the transition matrix is the fundamental complexity factor for inference on Hidden Markov Models, a sparse transition matrix implies much lower computation cost than a dense one. Therefore, we measure the size of the model by the number of edges in the transition graph. For GHMMs this is the sum of the number of neighbors for every discrete state in the model. In the case of EM and HFKM this is the sum of the transitions for all clusters, which for a single cluster is equal to the number of points in the cluster minus one plus the number of points, to account for self-transitions.

6.2.2 Measuring Prediction Accuracy

To evaluate prediction accuracy, the average expected prediction error has been computed from a test data set containing K observation sequences. The pre-

diction error is the expected distance between the predicted position for a time horizon H and the corresponding observation O_{t+H}:

$$\langle E \rangle = \frac{1}{K} \sum_{k=1}^{K} \frac{1}{T^k - H} \sum_{t=1}^{T^k - H} \sum_{i \in \mathcal{S}} P([S_{t+H} = i] \mid O^k_{1:t}) \| O^k_{t+H} - \mu_i \|^{1/2} \qquad (17)$$

The case of the HFKM algorithm is particular, since the algorithm only outputs the probability of following a given motion pattern and is not able to predict predictions at the state level. In order to compare approaches, we have assumed a deterministic transition probability where all the probability mass for the n^{th} time step is concentrated on the n^{th} point of a cluster:

$$P([S_{t+H} = i] \mid O_{1:t}, \phi_j) = \begin{cases} P(O_{1:t}|\phi_j) & \text{if } i \text{ is the } t+H\text{-th element of cluster } j \\ 0 & \text{otherwise} \end{cases} \qquad (18)$$

where $P(O_{1:t}|\phi_j)$ is the probability[8] that the observation sequence $O_{1:t}$ belongs to cluster j.

6.3 Leeds Data

In order to compare the performance on the Leeds environment, we proceeded by dividing the data into a learning data set (200 sequences) and a test data set (60 sequences). To evaluate how the model's size and accuracy evolve with respect to the size of the learning data set, we performed five experiments, giving 40, 80, 120, 160 and 200 input trajectories to the learning algorithms. In the case of the GHMM and to have a fair comparison, learning has been done on all the sequences in the learning data set prior to prediction. However, it should be noted that this is by no means a requirement for our approach since it is designed to learn and predict in a continuous fashion.

The parameters we have chosen for every algorithm are shown in table 1. Due to the difficulty to choose adequate parameters to compare the approaches, we have started by making an educated guess and then refining them by trial and error.

Algorithm	Parameters
EM	$\sigma = 7, K_0 = 15$
HFKM	$sampling\_step = 25, K_0 = 15$
GHMM	$\sigma^2_{pos} = 49, \sigma^2_{vel} = 0.8, \sigma^2_{goal} = 400, \tau = 9$

Table 1: Parameters for Leeds data.

[8] See (Hu et al. 2006) for a definition of this probability.

The meanings of the parameters that appear in the table and are not defined in the text are the following: K_0 is the initial number of clusters. For HFKM *sampling_step* means that in the first clustering step, learning sequences will be sampled by taking one out of *sampling_step* observations on them. Finally, for GHMMs, the covariance matrix is built as follows:

$$\Sigma = \begin{bmatrix} \sigma^2_{pos} & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma^2_{pos} & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma^2_{vel} & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma^2_{vel} & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma^2_{goal} & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma^2_{goal} \end{bmatrix} \tag{19}$$

6.3.1 Comparing Prediction Accuracy

Figure 12 shows the average prediction error as a function of the total number of trajectories in the learning data set. For every batch, full learning is performed and then the expected error is computed with Eq. (17) using the test data set as input.

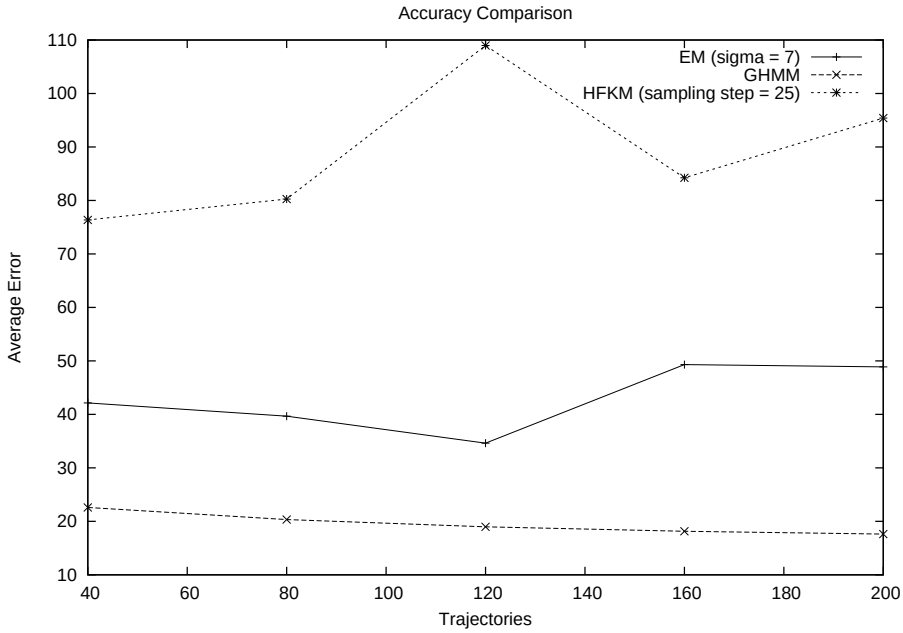


Figure 12: Leeds data: prediction accuracy.

As it can be seen, the average error is much lower for GHMMs than for the other two approaches. Perhaps more surprising: while for GHMMs, the

error seems to decrease as more trajectories are used for learning, this is not the case of the other two algorithms. At first, we thought that this was due to initialization differences between runs, where different trajectories were used to initialize the clusters in each case. In order to discard this possibility, we have used the same initialization for all different runs without significant difference, as the figure shows. Our current explanation is that the fact of using a global optimization scheme may lead to very different clusterings depending on input data, sometimes with suboptimal results.

Another important factor, at least in the case of HFKM, seems to be that the model is "saturated" and is not able to perform further generalization on the basis of additional data. This leads us to what, in our eyes, is an important drawback of HFKM: it lacks some kind of scale parameter (like the covariance in EM and GHMM) to trade off a better accuracy for the cost of having a more complex model. This will become clearer on the model size comparison.

6.3.2 Comparing Model Size

The growth on model size with respect to the number of trajectories on the learning data set is displayed on fig. 13. As it can be seen in the figure, the size of the GHMM models is negligible with respect to the other two approaches. On the other hand, the model growth tends to stabilize for both GHMM and HFKM, while in the case of the EM approach, it jumps suddenly for 160 trajectories. As mentioned in § 6.3.1, this seems to indicate that both GHMM and HFKM have converged, but the behavior of EM is more difficult to explain.

The jump is explained by a number of "anomalous" trajectories, that appear between positions 120 and 160 in the learning data set. These trajectories force the creation of several clusters in the EM model that do not appear in the HFKM approach due to its cluster size threshold. In the case of GHMM, these trajectories also lead to a sudden growth of the model, but this is more difficult to perceive because of its small size.

Finally, in fig. 14, we plot the model size against the prediction error. This illustrates more explicitly the fact that, for the HFKM and EM algorithms, an increase in the model size does not necessarily lead to an improvement in the accuracy. As we will discuss in the following subsection, we have explored this further using simulated data.

6.4 INRIA Data

In the case of the simulated INRIA environment, we have proceeded in analogous fashion to what we have done with the Leeds data set, but this time, the learning data set contains 1000 sequences while the test data set contains

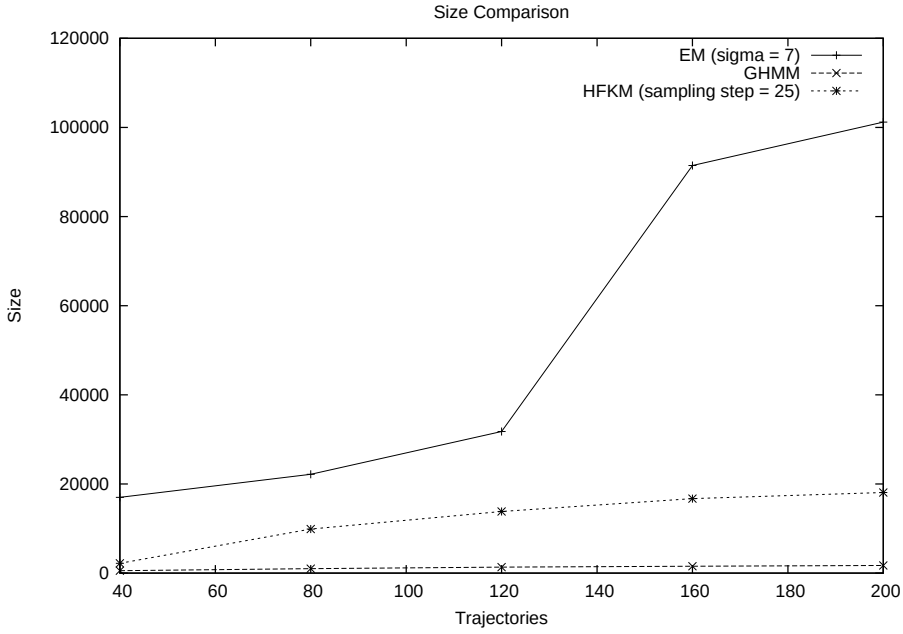


Figure 13: Leeds data: model size.

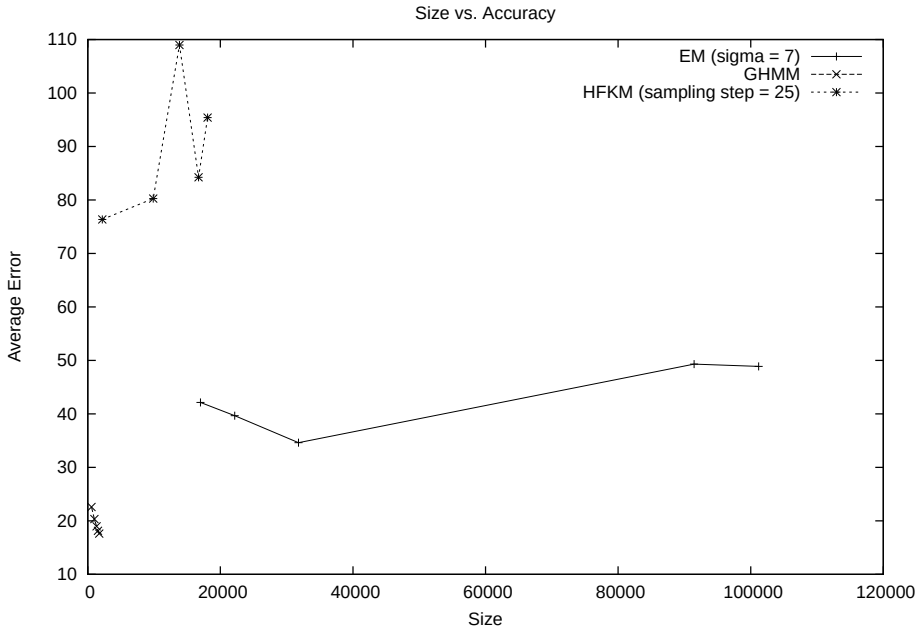


Figure 14: Leeds data: model size vs. prediction accuracy.

100. The learning batches for this data set contain 200, 400, 600, 800 and 1000 trajectories, respectively.

The parameters for every algorithm are shown in table 6.4. This time, however, we have included two different set of parameters per algorithm in order to show their influence on the model performance. The parameters have also been obtained by trial and error.

Algorithm	Parameters
EM (sigma = 0.5)	$\sigma = 0.5, K_0 = 15$
EM (sigma = 1)	$\sigma = 1, K_0 = 15$
HFKM (sampling step = 3)	$sampling\_step = 3, K_0 = 15$
HFKM (sampling step = 25)	$sampling\_step = 25, K_0 = 15$
GHMM (cov1)	$\sigma^2_{pos} = 2.25, \sigma^2_{vel} = 0.04, \sigma^2_{goal} = 16, \tau = 9$
GHMM (cov2)	$\sigma^2_{pos} = 1, \sigma^2_{vel} = 0.04, \sigma^2_{goal} = 16, \tau = 9$

Table 2: Parameters for the INRIA data.

6.4.1 Comparing Prediction Accuracy

In this data set, the accuracy of the GHMM approach is again better than that of the compared approaches, but this time the difference is not as big as for Leeds data. Moreover, the behavior of the other approaches, with the exception of HFKM with a sampling step of three, is this time much closer to that of the GHMM approach. This is explained by the absence of anomalous trajectories in the learning and test datasets, together with the existence of many similar example sequences per behavior.

We have not found the reason why HFKM with a sampling step of 3 behaves differently, our best explanation at this moment is that, as the number of points in a trajectory increases, the clustering starts to smooth out the distinctive part of every trajectory and to produce less clusters, which fail to accurately represent the underlying behaviors.

6.4.2 Comparing Model Size

Regarding the size of the model (fig. 16), the results obtained with the simulated data set are very different from Leeds. In this case the size of the models obtained with both GHMM parameterizations are similar to the other approaches, with the exception of EM (sigma = 1). On the other hand, once again the model size grows monotonically with the number of trajectories only for GHMMs, while for the other approaches it tends to oscillate. In fact for both HFKM parameterizations, the model size seems to decrease as the learning data set gets bigger.

By plotting the model size vs the accuracy (Fig. 17) we observe again that – with the exception of GHMMs – for a same parametrization, an increase in

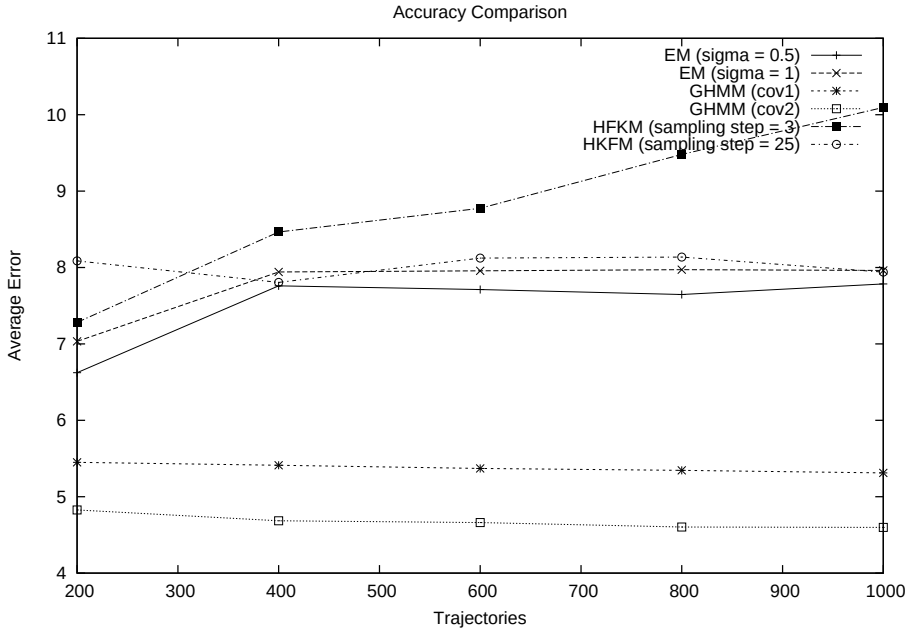


Figure 15: INRIA data: prediction accuracy.

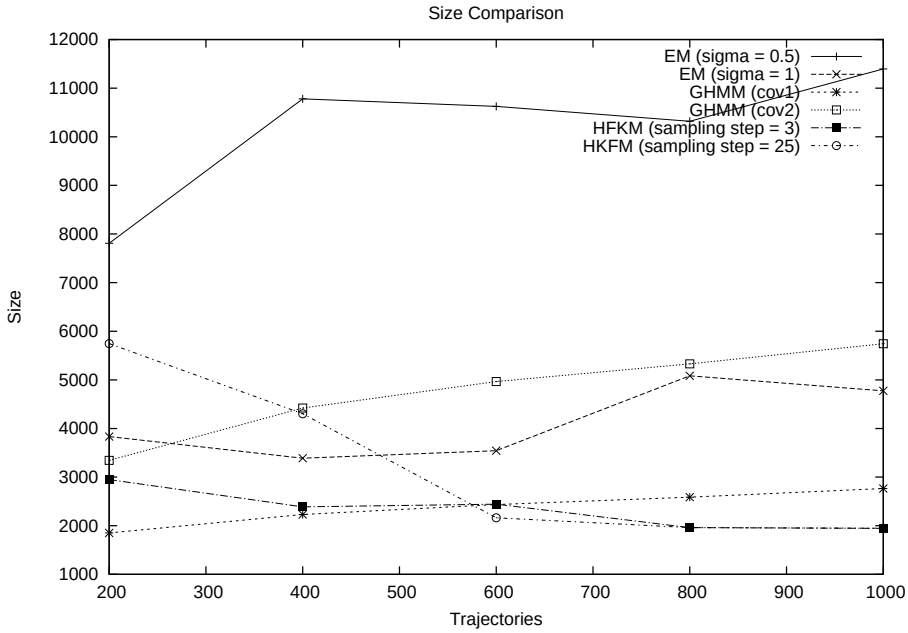


Figure 16: INRIA data: model size.

the model size is not directly translated into an improvement of the accuracy. It also shows that, between models of comparable size, those obtained with our approach exhibit consistently and significantly better accuracy.

This is also a good opportunity to recapitulate on the subject of choice of the model parameters. It is probable that, at least in the case of EM, reducing the covariance matrix could improve the accuracy to make it closer to what is obtained with GHMM, nevertheless, this would imply a huge increase in the model size, which is not comparable to that of the GHMM model. This seems to indicate that our model generalizes better, and that, in the case of EM the model is probably overfitting data.

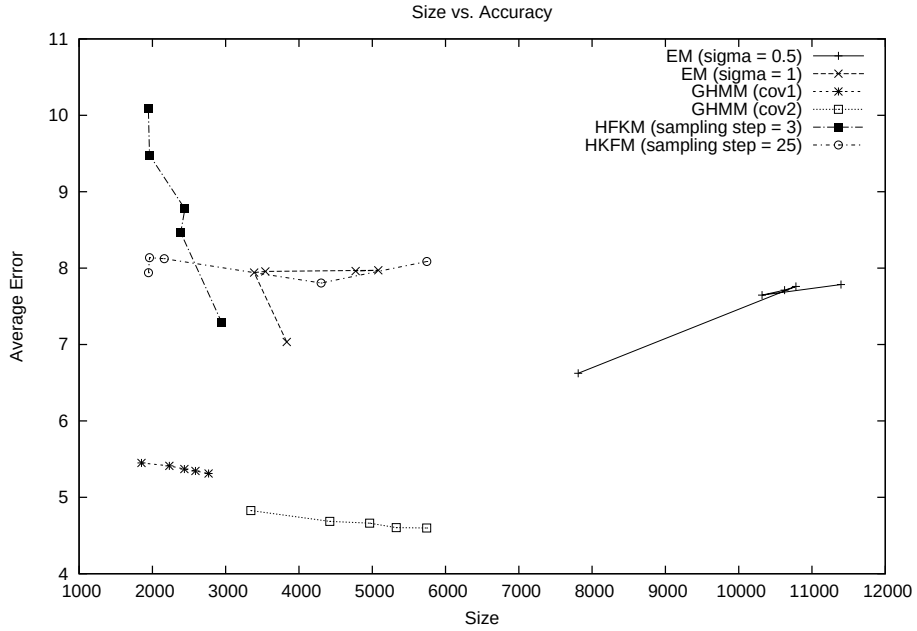


Figure 17: INRIA data: model size vs. prediction accuracy.

6.5 Evaluation Conclusion

Despite the good results that we have obtained so far, we consider that much more experimental work needs to be done before arriving to definitive conclusions. We consider at this moment that our approach performs better than the similar EM approach mainly because of two factors: a) it has a more expressive and compact structure and b) it learns the parameters of the observation and transition probabilities from data instead of relying in *a priori* knowledge. We consider the first factor to be more important than the second; it would be interesting, however, to apply HMM parameter learning to the EM approach

after trajectory clustering, to have a quantitative evaluation of the relative importance of both factors. It would be also interesting to do something similar in the case of HFKM, where an HMM is built from the obtained clusters.

7 Conclusions and Future Work

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. This improves over other HMM based techniques by implementing a model –even if rather crude– of the object's intentions.

We have performed extensive experimental validations of our approach and compared it against two state of the art techniques using real and synthetic data. In these experiments, our technique has performed consistently better than the other two approaches by providing more accurate predictions with smaller models. Even if further experiments would be necessary to reach definitive conclusions, we consider these results to be very encouraging. Moreover, our experiments also show that our approach is able to perform continuous learning and prediction in real-time even for large data sets.

We have implemented our approach as a C^{++} library and made it available together with some data sets in the following address: `http://svn.asl.ethz.ch/ghmm`.

Future work includes applying our approach to gesture recognition on video sequences using semi-supervised learning with human-labeled input sequences. A more challenging and open research direction would be to explore how our approach can be extended to embedded sensors, for example cameras or laser scanners mounted on vehicles.

A Forward, Backward and Observation Probabilities

For the sake of completeness, we include here the pseudo code for the computation of the forward and backward probabilities.

The probability of an observation sequence is computed from the forward probabilities using (20).

$$
\begin{aligned}
P(O_{1:T}|\lambda) &= \sum_{i=1}^{N} P(O_{1:T}, S_T = i|\lambda) \\
&= \sum_{i=1}^{N} \alpha_T(i)
\end{aligned}
\tag{20}
$$

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* $\alpha_t(i)$

B Computing Fuzzy K-Means Cluster Representations

The original paper of Hu et al. (2006) presents the following expression to compute the j^{th} cluster center $V_j(t)$:

$$
V_j(t+1) = V_j(t) + \frac{\sum_{l=1}^{M} R_{lj}(t) \cdot (X_l - V_j(t))}{\sum_{l=1}^{M} R_{lj}(t)} \tag{21}
$$

where M is the total number of trajectories, X_l is the l^{th} trajectory, and $R_{lj}(t)$ is the estimated membership of trajectory l to cluster j. However, in our experiments using this expression has lead to bad clustering results. The problem seems to be that (21) contains a typo since it does not take into account the right *fuzzification coefficient* (see, for example (Kolen and Hutcheson 2002)) whose value – judging from the other expressions in the paper – is two. The

Algorithm 2: *Backward_probabilities*$(O_{1:T}, \lambda)$

input :

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

output : Backward probabilities $\beta_t(i)$

```
1  begin
2      for i = 1 to N do
3          β_T(i) = 1
4      end
5      for t = T − 1 down to 1 do
6          for i = 1 to N do
7              β_t(i) = Σ_{j=1}^{N} P([S_{t+1} = j]|[S_t = i]) P(O_{t+1}|[S_{t+1} = j]) β_{t+1}(j)
8          end
9      end
10 end
11 return all β_t(i)
```

corrected equation has produced much better results, it is:

$$V_j(t+1) = V_j(t) + \frac{\sum_{l=1}^{M} R_{lj}^2(t) \cdot (X_l - V_j(t))}{\sum_{l=1}^{M} R_{lj}^2(t)} \tag{22}$$

References

Maren Bennewitz, Wolfram Burgard, and Sebastian Thrun. Learning motion patterns of persons for mobile service robots. In *Proc. of the IEEE Int. Conf. On Robotics and Automation*, pages 3601–3606, Washington, USA, 2002.

Maren Bennewitz, Wolfram Burgard, Grzegorz Cielniak, and Sebastian Thrun. Learning motion patterns of people for compliant robot motion. *Internationl Journal of Robotics Research*, 24(1):31–48, January 2005.

Matthiew Brand, Nuria Oliver, and Alex Pentland. Coupled hidden markov models for complex action recognition. In *In Proc. of the 1997 Conf. on Computer Vision and Pattern Recognition*, pages 994–999, San Juan (PR), 1997.

Allison Bruce and Geoffrey Gordon. Better motion prediction for people-tracking. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, New Orleans, US, April 2004.

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

Dan Buzan, Stan Sclaroff, and George Kollios. Extraction and clustering of motion trajectories on video. In *Proc. of the Int. Conf. on Pattern Recognition*, Cambridge, UK, 2004.

Sylvain Calinon and Aude Billard. Incremental learning of gestures by imitation in a humanoid robot. In *Proc. of the ACM/IEEE conf. on Human-Robot Interaction*, pages 255–262, 2007.

Hannah Dee and David Hogg. Detecting inexplicable behaviour. In *In Proceedings of the British Machine Vision Conference*, pages 49–55, Kingston (UK), 2004.

Hannah-Mary Dee. *Explaining Visible Behaviour*. PhD thesis, University of Leeds, 2005.

Kevin R. Dixon, John M. Dolan, and Pradeep K. Khosla. Predictive robot programming: Theoretical and experimental analysis. *Int. Journal of Robotics Research*, 23(9):955–973, September 2004.

Amalia F. Foka and Panos E. Trahanias. Predictive autonomous robot navigation. In *In Proc of the IEEE/RSJ Int. Conf. on Intelligent Robots and System*, volume 1, pages 490–495, Lausanne (CH), October 2002.

Shaogang Gong and Tao Xiang. Recognition of group activities using dynamic probabilistic networks. In *In Proceedings of the Ninth IEEE International Conference on Computer Vision*, volume 2, pages 742–749, Washington (US), 2003.

Anthony Hoogs and A.G. Amitha Perera. Video activity recognition in the real world. In *Proc. of the Twenty-Third AAAI Conf. on Artificial Intelligence*, pages 1551–1554, Chicago (US), 2008.

Ronald A. Howard. *Dynamic Programming and Markov Process*. MIT Press, Cambridge (US), 1960.

Weiming Hu, Dan Xie, Tan Tieniu, and Steven Maybank. Learning activity patterns using fuzzy self-organizing neural network. *IEEE Trans. on Systems, Man and Cybernetics*, 34(3):1618–1626, June 2004.

Weiming Hu, Xejuan Xiao, Zhoyu Fu, Dan Xie, Tieniu Tan, and Stephen Maybank. A system for learning statistical motion patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(9):1450–1464, September 2006.

Jan Jockusch and Helge Ritter. An instantaneous topological map for correlated stimuli. In *In Proc. of the International Joint Conference on Neural Networks*, volume 1, pages 529–534, Washington (US), July 1999.

Neil Johnson and David Hogg. Learning the distribution of object trajectories for event recognition. In *Proc. of the British Machine Vision Conference*, volume 2, pages 583–592, September 1995.

Bing-Hwang Juang, Stephen E. Levinson, and M. Mohan Sondhi. Maximum likelihood estimation for multivariate mixture observations of markov chains. *IEEE Transactions on Information Theory*, 32(2):307–309, March 1986.

Imran Junejo, Omar Javed, and Mubarak Shah. Multi feature path modeling for video surveillance. In *Proc. of the 17th conference of the International Conference on Pattern Recognition (ICPR)*, pages 716–719, 2004.

John F. Kolen and Tim Hutcheson. Reducing the complexity of the fuzzy c-means algorithm. *IEEE Transactions on Fuzzy Systems*, 10(2):263–267, April 2002.

Dana Kulic, Wataru Takano, and Yoshihiko Nakamura. Incremental learning, clustering and hierarchy formation of whole body motion patterns using adaptive hidden markov chains. *Int. Journal of Robotics Research*, 27(7):761–784, July 2008.

Pat Langley. Order effects in incremental learning. In P. Reimman and H. Spada, editors, *Learning in Humans and Machines: Towards an Interdisciplinary Learning Science*. Elsevier, 1995.

Lin Liao, Dieter Fox, and Henry Kautz. Learning and inferring transportation routines. In *Proc. of the National Conf. on Artificial Intelligence AAAI-04*, 2004.

DR Magee. Tracking multiple vehicles using foreground, background and shape models. *Image and Vision Computing*, 22:143–155, 2004.

Dimitrios Makris and Tim Ellis. Finding paths in video sequences. In *Proc. of the British Machine Vision Conference*, pages 263–272, 2001.

Dimitrios Makris and Tim Ellis. Spatial and probabilistic modelling of pedestrian behavior. In *Proc. of the British Machine Vision Conference*, pages 557–566, Cardiff, UK, 2002.

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

Nuria M. Oliver, Barbara Rosario, and Alex P. Pentland. A bayesian computer vision system for modeling human interactions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):831–843, August 2000.

Sarah Osentoski, Victoria Manfredi, and Sridhar Mahadevan. Learning hierarchical models of activity. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Sendai, Japan, 2004.

Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Readings in speech recognition*, pages 267–296, 1990.

Gideon Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, 1978.

Chris Stauffer and Eric Grimson. Learning patterns of activity using real-time tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):747–757, August 2000.

Andreas Stolcke and Stephen Omohundro. Hidden markov model induction by bayesian model merging. In Stephen José Hanson, Jack D. Cowan, and C. Lee Giles, editors, *Advances in Neural Information Processing Systems*, volume 5, pages 11–18, Denver, USA, 1993. Morgan Kaufmann, San Mateo, CA.

Neil Sumpter and Andrew Bulpitt. Learning spatio-temporal patterns for predicting object behaviour. *Image and Vision Computing*, 18(9):697–704, 2000.

Dizan Vasquez. *Incremental Learning for Motion Prediction of Pedestrians and Vehicles*. PhD thesis, Institut National Polytechnique de Grenoble, Grenoble, FR, February 2007.

Dizan Vasquez and Thierry Fraichard. Motion prediction for moving objects: a statistical approach. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 3931–3936, New Orleans, LA (US), April 2004.

Dizan Vasquez and Thierry Fraichard. Intentional motion on-line learning and prediction. In *Field and Service Robotics*, Port Douglas, Australia, July 2005.

Dizan Vasquez, Thierry Fraichard, and Christian Laugier. Incremental learning of statistical motion patterns with growing hidden markov models. In *13th International Symposium of Robotics Research*, Hiroshima (JP), November 2007.

Michael Walter, Alexandra Psarrow, and Shaogang Gong. Learning prior and observation augmented density models for behaviour recognition. In *Proc. of the British Machine Vision Conference*, pages 23–32, 1999.

Xiaogang Wang, Kinh Tieu, and Eric Grimson. Learning semantic models by trajectory analysis. Technical report, Massachussets Institute of Technology, 2006.

Tao Xiang and Shaogang Gong. Beyond tracking: Modelling activity and understanding behaviour. *International Journal of Computer Vision*, 2006.

Xuanli Lisa Xie and Gerardo Beni. A validity measure for fuzzy clustering. *IEEE Transactions on Pattern Analysis and Macine Intelligence*, 13(8):841–847, August 1991.