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 (e.g., internal state and 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 (e.g., camera and laser scanner), and then using that model to predict further motion. Unfortunately, most current techniques use offline 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 (HMMs)—called growing hidden Markov models—which gives us the ability to incrementally learn both the parameters and the structure of the model.**

***Index Terms*—Hidden Markov models (HMMs), 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 parameterize. 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, to detect anomalous behavior or improve 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 offline, 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. Our approach extends further our previous work [3] by proposing a unified extension to hidden Markov models (HMMs) [4], which is a probabilistic framework that is very popular in the motion pattern learning literature (e.g., [1], [5], and [6]). This extension, named growing HMM (GHMM), enables incremental and online 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 on techniques based on HMMs. Section III presents GHMMs. 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

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.

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 a 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 (e.g., position and speed). This makes them

Manuscript received October 30, 2007; revised July 23, 2008 and November 20, 2008. First published May 5, 2009; current version published September 1, 2009. This work was supported in part by the European Commission under Contract BACS FP6- IST-027140 and in part by a Mexican National Council on Science and Technology scholarship. The Associate Editor for this paper was U. Nunes.

D. Vasquez is with the Autonomous Systems Laboratory, Swiss Federal Institute of Technology, 8092 Zurich, Switzerland (e-mail: vasquez@mavt.ethz.ch).

T. Fraichard and C. Laugier are with the e-Motion Project Team, French National Institute for Research in Computer Science and Control Rhône-Alpes, 38334 Grenoble, France, and also with the Laboratoire d'Informatique de Grenoble, 38041 Grenoble, France (e-mail: thierry.fraichard@inrialpes.fr; christian.laugier@inrialpes.fr).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TITS.2009.2020208

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.* [7]: they use coupled HMMs [8] to model interactions (e.g., 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] and [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 the 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 (i.e., trajectory prototypes) from an input data set consisting of raw trajectory data.

A representative example is the approach proposed by Bennewitz *et al.* [11], which uses expectation–maximization to perform the clustering. Other algorithms include hierarchical clustering [2], [12]–[14], graph cutting [15], and custom pairwise clustering algorithms [16].

To apply the obtained trajectory prototypes to perform tracking or motion prediction, a probabilistic framework is often used. This allows one to explicitly represent 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 in [18] and [19], which, by explicitly modeling time, obtained performance 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 depend on 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 (i.e., its goal). For example, Liao *et al.* [21] have used a hierarchical extension to HMMs—abstract HMMs (AHMMs) [22]—to learn and predict the motion of pedestrians in cities, where the three layers in the AHMM represented (top to bottom) goals, transportation modes, and the 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], which is a probabilistic planning technique that illustrates the connection between planning and motion prediction.

Other goal-based approaches include [25]–[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 toward the goal is modeled using overly simplistic mechanisms (e.g., linear interpolation for [25]), leading to unreliable physical-state estimations.

B. HMM-Based Approaches

In this section, we will focus on techniques based on HMMs 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 toward our application. The interested reader may refer to the papers of Juang *et al.* [28] and Rabiner [4] for a deeper introduction to the subject.

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 (e.g., places in the environment) and whose edges represent transitions between states. The system is supposed to be at a given state and to stochastically evolve 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 (i.e., observation) that is related to the actual state through an observation probability $P(O_t|S_t)$. Often, the initial state of the system is stochastically represented with a state prior $P(S_1)$.

HMM learning is composed of two subtasks.

- *Structure learning:* This subtask 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:* This subtask estimates the parameters for the three probability distributions (state prior, transition, and observation probabilities) from the data.

Different algorithms for structure and parameter learning exist in the literature, it is the choice of these algorithms that 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, and 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 the object's

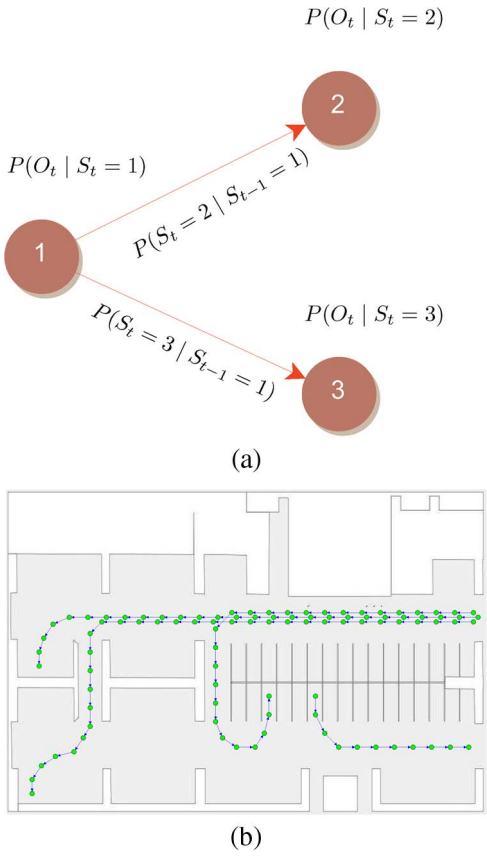


Fig. 1. (a) Basic three-state HMM. (b) HMM structure embedded in a parking lot (only a few motion patterns are displayed).

motion; and 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: 1) Typical motion patterns are represented with some sort of trajectory prototype; 2) structure learning is independent of parameter learning; and 3) learning is first performed offline, 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 differently behaves with respect to these points.

III. Growing Hidden Markov Models

In this section, we present our proposed extension to HMMs, i.e., GHMMs,[1] 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.

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

Our approach relies on three main assumptions.

1) We assume that input observation sequences correspond to complete examples (i.e., from beginning to end) of the whole process or system being modeled (e.g., in our application, this corresponds to complete pedestrian trajectories).
2) The evolution of the state of the modeled system or process is a continuous function.
3) 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 the 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 have basically adapted the incremental expectation–maximization approach proposed by Neal and Hinton [31] to deal with the 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 the *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:

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

The joint probability decomposition (JPD) for GHMMs is

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

[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 that 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 the parameters and structure change with every new observation sequence. Furthermore, notation $O_{1:t}$ will be used as a shortcut for the variable conjunction $O_1, O_2, \ldots, O_{t-1}O_t$.

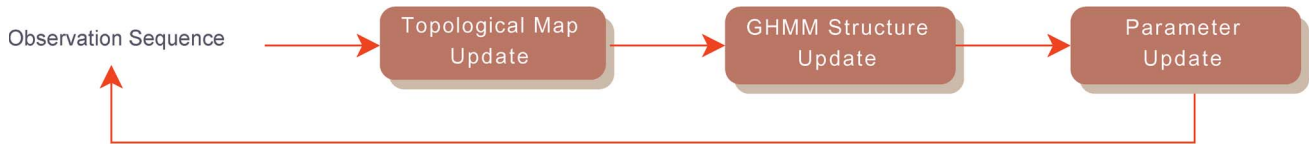


Fig. 2. Overview of the GHMM learning algorithm.

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, irrespective of the value of the time variable.

1) $P(S_0 = i) = \pi_i$. The state prior will be represented as a vector $\pi = \{\pi_1, \ldots, \pi_N\}$, where each element contains the prior probability for the corresponding discrete state.
2) $P([S_t = j]|[S_{t-1} = i]) = a_{i,j}$. 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.
3) $P(O_t|[S_t = i]) = \mathbf{G}(O_t;\ \mu_i, \Sigma)$. 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 $b = \{\Sigma, \mu_1, \ldots, \mu_N\}$.

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

In addition to 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 sections.

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 connecting contiguous nodes. Every node i has an associated vector w_i, corresponding to the region's centroid. The nodes are added and adapted to minimize the distortion of the model, i.e., the sum of the squared distances between the input (i.e., 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.

1) It minimizes the number of nodes while trying to keep the same average distance between neighbors.
2) It has linear time and memory complexity with respect to the number of nodes.

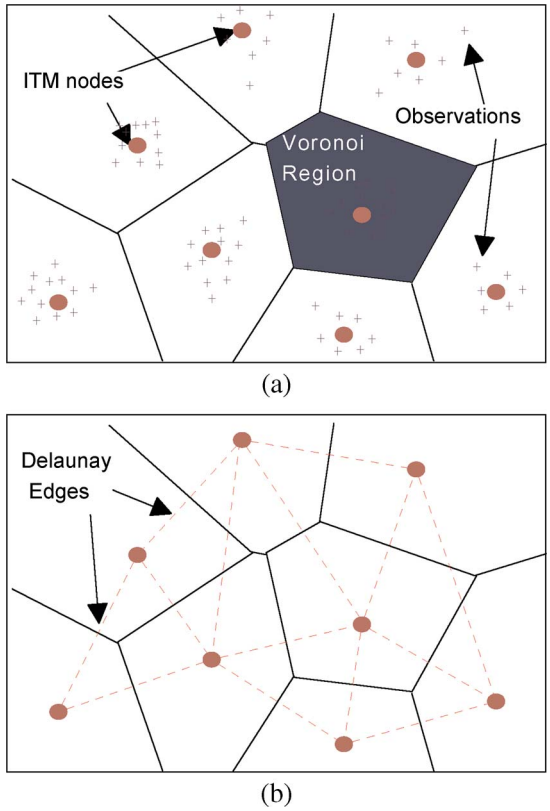


Fig. 3. Example ITM space decomposition. (a) Nodes and regions. (b) Edges.

3) Edges are a subset of the Delaunay triangulation, meaning that they can exist only between nodes representing adjacent Voronoi[4] regions (see Fig. 3).

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 the Mahalanobis distance with the same Σ as the observation probabilities.
2) **Weight adaptation**: Move w_b toward 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, and 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 and also delete any node that has no neighbors.
4) **Node adaptation**: 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 its distance from b is smaller than $(\tau/2)$.

[4]The Voronoi region associated with a node is defined by the set of all the points that 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 crucial part of the algorithm is that in addition to 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, however, 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 later 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 directly use it 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 up to one. This is corrected by a normalization step that takes place at the beginning of parameter update (cf. Section 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, these values will later be normalized to obtain true probabilities.
3) For every deleted node and edge in the topological map, assign a value of zero (i.e., 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 immediately performed after structure learning. The GHMM learning algorithm reestimates the parameters using an incremental version of the Baum–Welch technique based on the work of 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 and 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 Section 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 the structure update does not guarantee that the corresponding probabilities add up to one, as explained in Section 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 nonzero transition probability in the GHMM using

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

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

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

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

IV. Learning and Predicting Motion With GHMMs

Having presented GHMMs, this section focuses on 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 (i.e., 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* (i.e., goal) states, 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 (see Fig. 4).

A. Notation and Basic Assumptions

We assume that tracking data are available as a collection of observation sequences (i.e., trajectories). Every individual sequence $O^k_{1:T^k} = \{O_1, \ldots, O_{T^k}\}$ 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

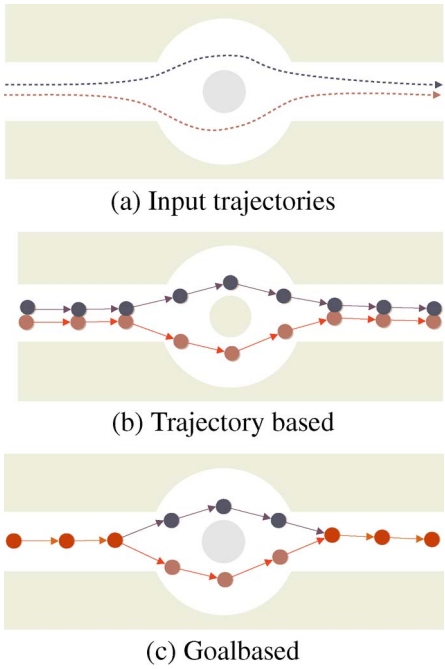


Fig. 4. Pattern representations generated from input data. (a) Input trajectories. (b) Trajectory-based prototypes. (c) Goal-based directed graph (the goal is the rightmost node).

positions (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 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

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

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 1-D environment in which objects can only move from point A to point B or *vice versa*, as depicted in Fig. 5(a).

We will first analyze what happens with trajectories that go from A to B. From our definition, the extended state is 2-D 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 Fig. 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 Fig. 5(c). In it, the left–right transition probabilities are high, the self-transition probabilities are much lower, and the right–left transitions probabilities are negligible (but never zero).

If we now train on the $B \rightarrow A$ trajectories, we will obtain something similar to the lower part of Fig. 5(d). The first thing to notice is that the extended state trajectories now go from $(5, -5)$ to $(-5, -5)$ and belong to a different 1-D manifold than the $A \rightarrow B$ 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 where there are 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 Section III-A. 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, enabling us to rewrite the observation probability as

$$P(O_t|S_t) = P\left(O'_t\, O''_t|S_t\right) = P\left(O'_t|S_t\right) P\left(O''_t|S_t\right) \tag{7}$$

and the whole JPD as

$$P\left(S_{t-1}\, S_t\, O'_t\, O''_t\right) = P(S_{t-1}) P(S_t|S_{t-1}) P\left(O'_t|S_t\right) P\left(O''_t|S_t\right). \tag{8}$$

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

$$P\left(O'_t|[S_t = i]\right) = \mathbf{G}\left(O'_t; \mu'_i, \Sigma'\right) \tag{9}$$

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

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 $P(O_t|S_t)$ is either a product of Gaussians or a product of a constant and a Gaussian, we may rewrite this

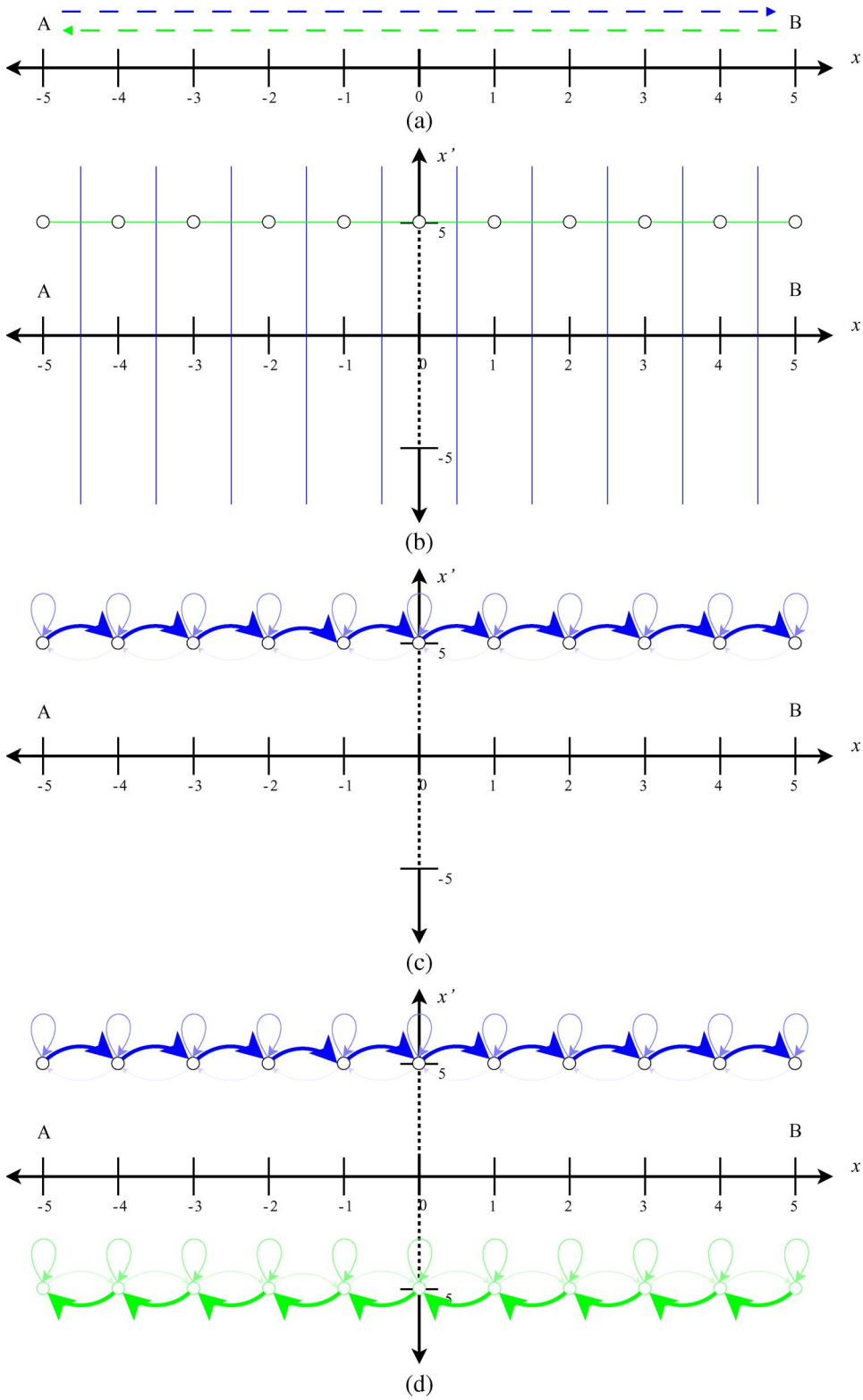


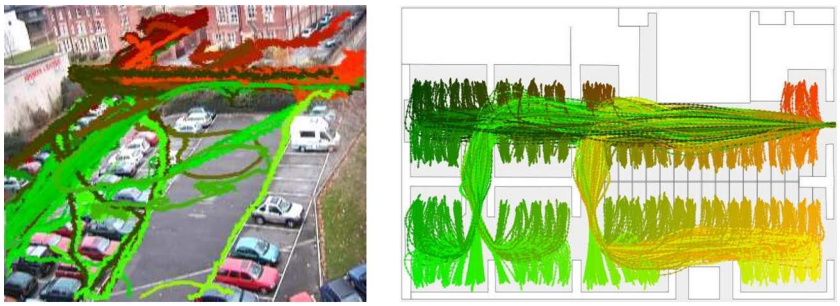
Fig. 5. Example of our motion learning algorithm in a 1-D environment. x and x' axes do not have the same scale. (a) Example 1-D environment. Objects move from A to B (top arrow) or from B to A (bottom arrow). (b) Extended observation sequence plot and Voronoi diagram. The Voronoi region's borders are depicted in blue, and Delaunay links are 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.

probability as a single Gaussian

$$P\left(O_t|[S_t=i]\right)=\frac{1}{Z}\mathbf{G}(O_t;\mu_i,\Sigma) \tag{11}$$

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

$$\Sigma=\begin{bmatrix}\Sigma' & 0\\ 0 & \Sigma''\end{bmatrix} \tag{12}$$



(a) (b)

Fig. 6. Data sets. (a) Real data. (b) Synthetic data.

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[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 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 reestimated using

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

where $P(S_{t-1}|O_{1:t-1})$ 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

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

Sometimes, we are interested in knowing the probability of the continuous-state probability distribution—as opposed to the discrete space, which is 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:

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

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

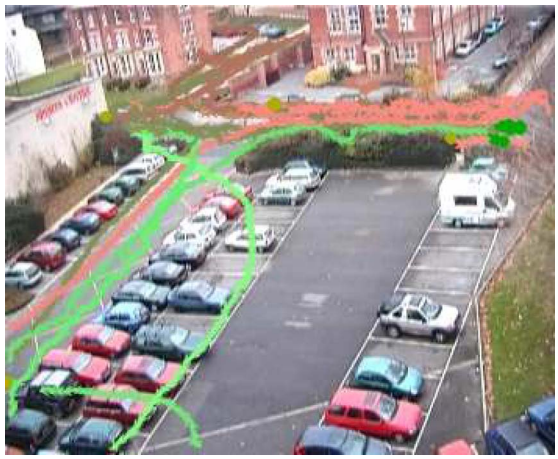


Fig. 7. Projection on the image of the learned structure after 100 trajectories have been processed (Leeds data set).

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 (see Fig. 6): 1) real data obtained through a visual tracker in a parking environment, wherein tracking errors have been corrected by humans when necessary by inspecting the original video sequence, as described in [32], and 2) synthetic data, which is generated by a simulator. As mentioned in Section IV, we have included velocity information in the simulation and accordingly extended the state and observation vectors. This allows, for example, inferring 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. 7 shows the resulting structure after applying the learning step for 100 trajectories of the Leeds data set.

Fig. 8 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.

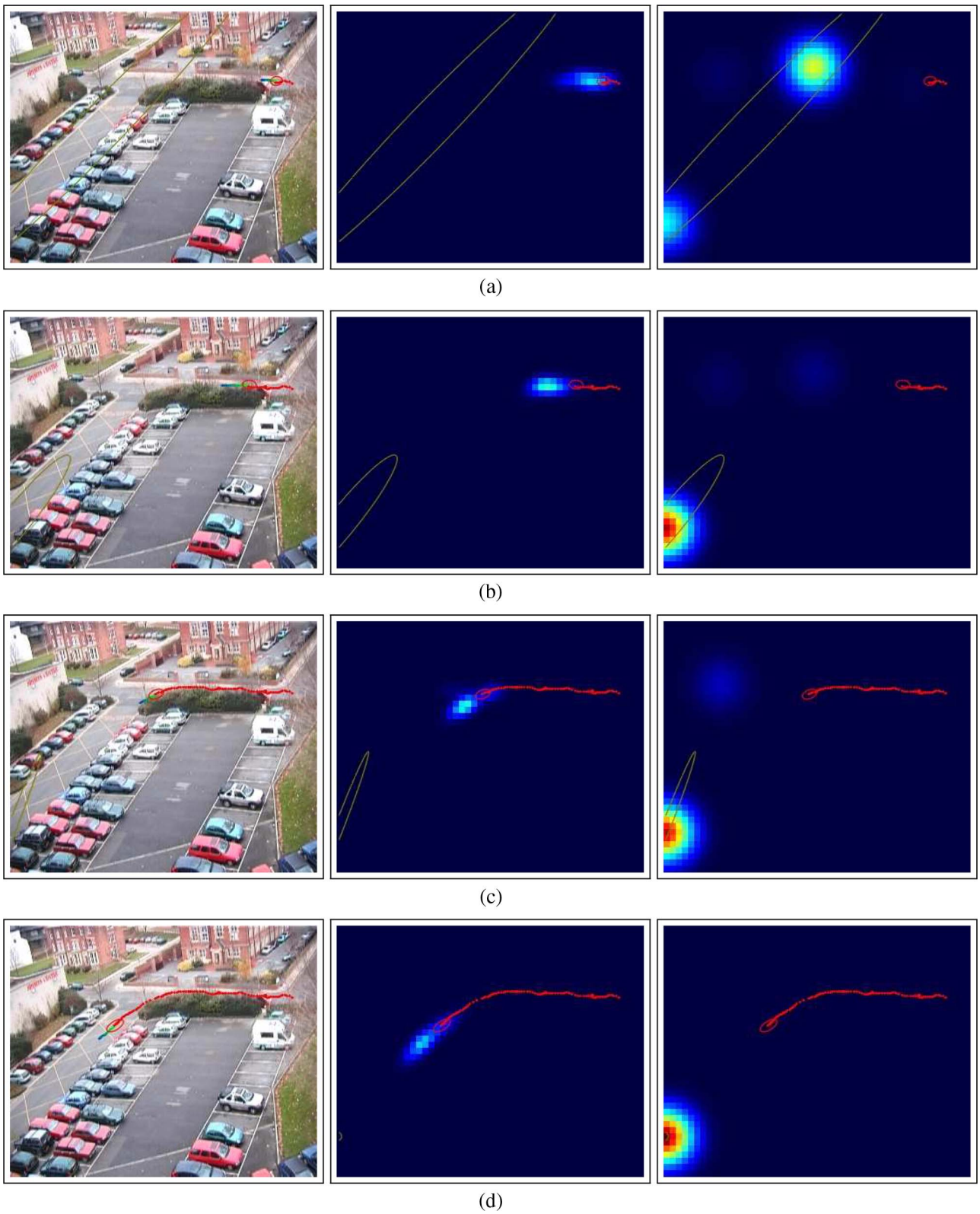


Fig. 8. Example of a sequence of predictions for an obstacle moving in the Leeds environment. See Section V-A for details. (a) $t = 10$. (b) $t = 30$. (c) $t = 82$. (d) $t = 110$.

In each row, the left image shows an actual picture of the parking lot featuring different overlays, as shown in Fig. 9(a): 1) the current and previous observations, depicted as red dots; 2) the current state estimation approximated by a Gaussian indicated with a red ellipse; 3) the current goal estimation also approximated by a Gaussian, represented by a golden ellipse; and 4) the mean value of the predicted states for different time horizons going from $H = 1$ to $H = 15$, where H represents the

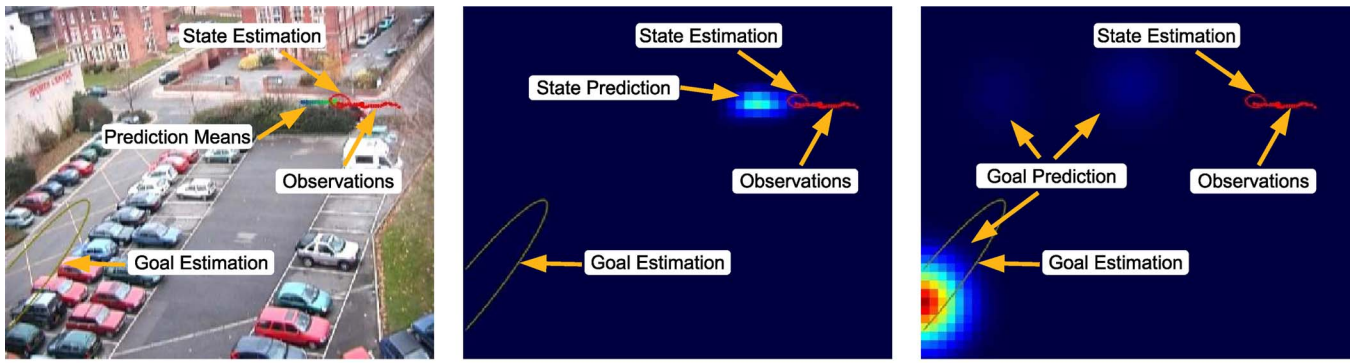


Fig. 9. Explanation of Fig. 8; see Section V-A as well. (a) Left column. (b) Center column. (c) Right column.

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 2-D projections in the image space of the state and goal estimations. As depicted in Fig. 9(b) and (c), they display—in addition to the previously mentioned overlays—the probability distribution for the predicted position for $H = 15$ and the 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 (15) to the cells of a regular grid. Since the augmented state is 4-D, 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 (15) to the cells of a regular grid, much like that 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 (i.e., 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 quickly becomes associated with a vehicles' goal, and by $t = 82$, the only two goals with a significant probability correspond to the vehicles' destinations.

It is also noteworthy 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 very slowly move with respect to the size of the image, particularly 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}

$$\langle E \rangle = \frac{1}{K} \sum_{k=1}^{K} \frac{1}{T^k - H} \sum_{t=1}^{T^k - H} \sum_{i \in \mathcal{S}} P\left([S_{t+H} = i] | O_{1:t}^k\right) \times \left\| O_{t+H}^k - \mu_i \right\|^{1/2}. \quad (16)$$

C. Model Size versus Prediction Accuracy

Fig. 10(a) and (c) shows 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 previously been observed or that significantly differ from all the rest.

D. Model Size versus Processing Time

Fig. 10(b) and (d) plots 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 may be expected, time seems to linearly depend on the 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 10-s trajectory requires about slightly more than 1 s to be learned.

E. Modeling Motion With Cycles

An interesting question is to see if and how our approach could deal with motion patterns containing cycles. As an

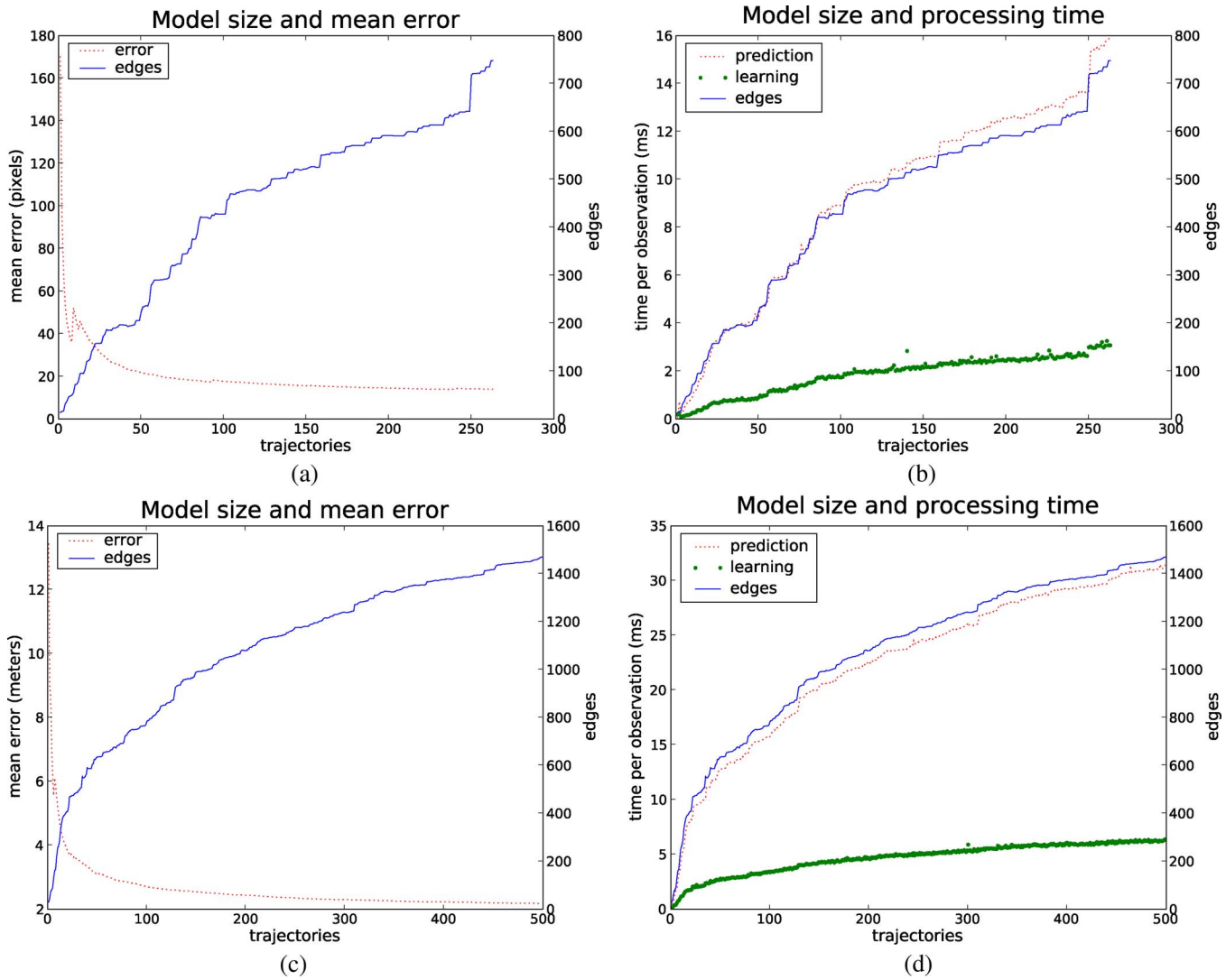


Fig. 10. Error and computation times. First row: Real data. Bottom row: Synthetic data.

example, let us imagine an 800-m track running event, where runners have to go twice around the racetrack to finish the race. It is evident that the situations at 395 and 795 m 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 5 m. 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 this situation, resulting in equiprobable predictions of stopping and continuing at both 395 and 795 m.

Although this example may seem contrived, this situation frequently arises 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 always turns to the right.

Of course, including the *time* variable in observations implies augmenting the observation covariance matrix to

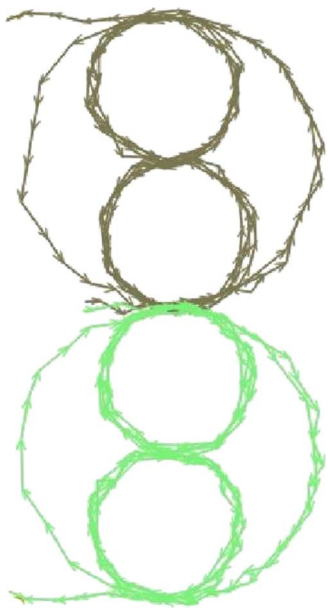


Fig. 11. Examples of cyclic trajectories. One trajectory per motion pattern is shown; the upper (gray) and lower (green) trajectories correspond to left and right turns, respectively. See Section V-E for further details.

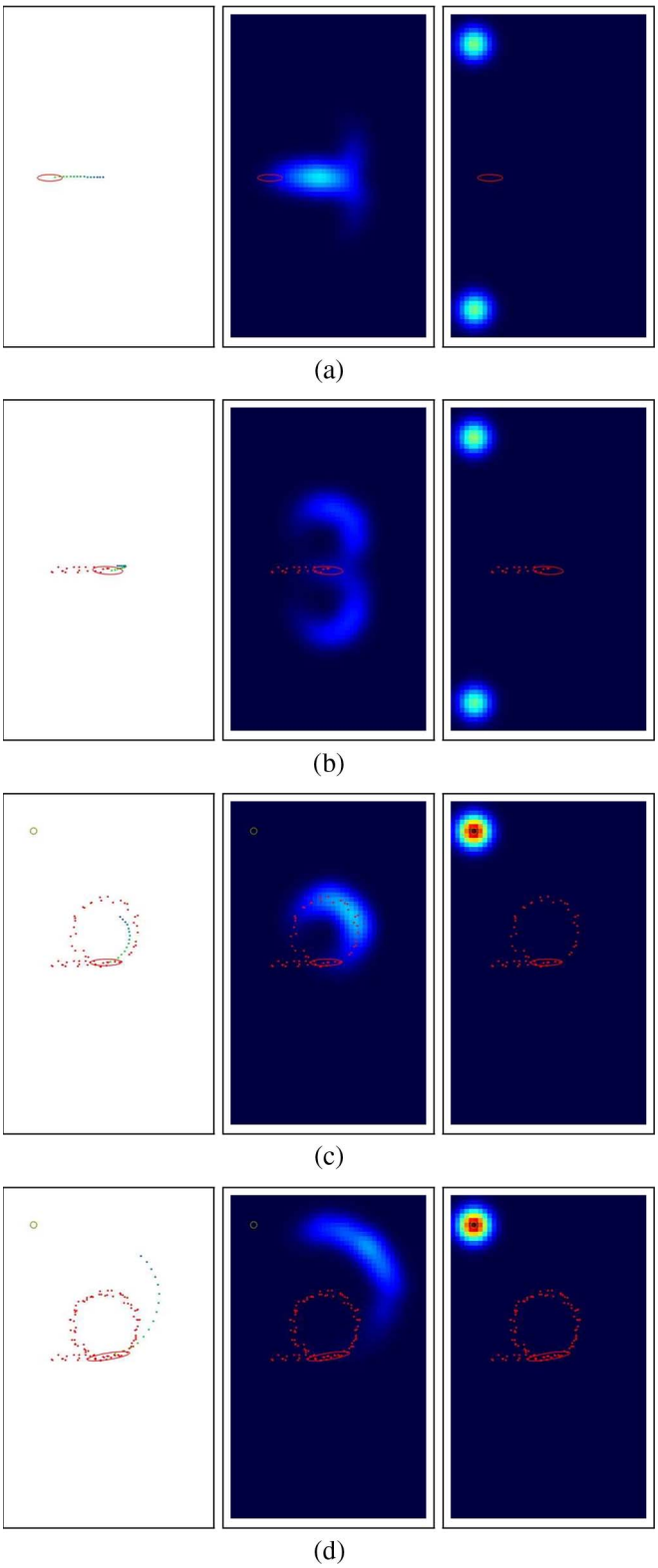


Fig. 12. Prediction example for the cycle data set. (a) $t = 1$. (b) $t = 20$. (c) $t = 60$. (d) $t = 100$.

include the time variable. Without going into the parameter details, Fig. 12 illustrates our algorithm working in this data set.

Notice how the object's positions for $t = 20$, $t = 60$, and $t = 100$ are very similar. Nevertheless, the predictions are clearly different. At $t = 20$, the object has not even turned; hence, the estimated probability for both goals are 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 easily be dealt with by modifying the covariance matrix.

VI. Conclusion

We have developed a novel extension to HMMs that is able to incrementally learn both the model's parameters and structure. We have applied this extension to vehicle and pedestrian motion by defining an augmented state that 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 on performing an 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 pseudocode for the computation of the forward (Algorithm 1) and backward (Algorithm 2) probabilities.

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**
$\alpha_t(j) = [\sum_{i=1}^{N} \alpha_t(i)P([S_t = j]|[S_{t-1} = i])]P(O_t|[S_t = j])$
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* $\beta_t(i)$

The joint observation probability (i.e., the probability of an observation sequence given the model) is computed from the forward probabilities using

$$P(O_{1:T}|\lambda) = \sum_{i=1}^{N} P(O_{1:T}, S_T = i|\lambda) = \sum_{i=1}^{N} \alpha_T(i). \quad (17)$$

ACKNOWLEDGMENT

The author's would like to thank Prof. R. Siegwart, Prof. M. Devy, and Prof. W. Burgard for their insightful comments on this paper and Dr. H. Dee and the University of Leeds for kindly sharing their experimental data. 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.

REFERENCES

[1] M. Bennewitz, W. Burgard, G. Cielniak, and S. Thrun, "Learning motion patterns of people for compliant robot motion," *Int. J. Robot. Res.*, vol. 24, no. 1, pp. 31–48, Jan. 2005.
[2] W. Hu, X. Xiao, Z. Fu, D. Xie, T. Tan, and S. Maybank, "A system for learning statistical motion patterns," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 9, pp. 1450–1464, Sep. 2006.
[3] D. Vasquez and T. Fraichard, "Intentional motion on-line learning and prediction," in *Proc. Conf. Field Service Robot.*, Port Douglas, Australia, Jul. 2005.
[4] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," in *Readings in Speech Recognition*. San Mateo, CA: Morgan Kaufmann, 1990, pp. 267–296.
[5] M. Walter, A. Psarrow, and S. Gong, "Learning prior and observation augmented density models for behaviour recognition," in *Proc. Brit. Mach. Vis. Conf.*, 1999, pp. 23–32.
[6] D. Makris and T. Ellis, "Spatial and probabilistic modelling of pedestrian behavior," in *Proc. Brit. Mach. Vis. Conf.*, Cardiff, U.K., 2002, pp. 557–566.
[7] N. M. Oliver, B. Rosario, and A. P. Pentland, "A Bayesian computer vision system for modeling human interactions," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 831–843, Aug. 2000.
[8] M. Brand, N. Oliver, and A. Pentland, "Coupled hidden Markov models for complex action recognition," in *Proc. Conf. Comput. Vis. Pattern Recog.*, San Juan, Puerto Rico, 1997, pp. 994–999.
[9] S. Gong and T. Xiang, "Recognition of group activities using dynamic probabilistic networks," in *Proc. 9th IEEE Int. Conf. Comput. Vis.*, Washington, DC, 2003, vol. 2, pp. 742–749.
[10] T. Xiang and S. Gong, "Beyond tracking: Modelling activity and understanding behaviour," *Int. J. Comput. Vis.*, vol. 67, no. 1, pp. 21–51, Apr. 2006.
[11] M. Bennewitz, W. Burgard, and S. Thrun, "Learning motion patterns of persons for mobile service robots," in *Proc. IEEE Int. Conf. Robot. Autom.*, Washington, DC, 2002, pp. 3601–3606.
[12] D. Makris and T. Ellis, "Finding paths in video sequences," in *Proc. Brit. Mach. Vis. Conf.*, 2001, pp. 263–272.
[13] D. Buzan, S. Sclaroff, and G. Kollios, "Extraction and clustering of motion trajectories on video," in *Proc. Int. Conf. Pattern Recog.*, Cambridge, U.K., 2004, pp. 521–524.
[14] D. Vasquez and T. Fraichard, "Motion prediction for moving objects: A statistical approach," in *Proc. IEEE Int. Conf. Robot. Autom.*, New Orleans, LA, Apr. 2004, pp. 3931–3936.
[15] I. Junejo, O. Javed, and M. Shah, "Multi feature path modeling for video surveillance," in *Proc. 17th Conf. ICPR*, 2004, pp. 716–719.
[16] X. Wang, K. Tieu, and E. Grimson, "Learning semantic models by trajectory analysis," Mass. Inst. Technol., Cambridge, MA, 2006. Tech. Rep.
[17] N. Johnson and D. Hogg, "Learning the distribution of object trajectories for event recognition," in *Proc. Brit. Mach. Vis. Conf.*, Sep. 1995, vol. 2, pp. 583–592.
[18] N. Sumpter and A. Bulpitt, "Learning spatio-temporal patterns for predicting object behaviour," *Image Vis. Comput.*, vol. 18, no. 9, pp. 697–704, Jun. 2000.
[19] W. Hu, D. Xie, T. Tieniu, and S. Maybank, "Learning activity patterns using fuzzy self-organizing neural network," *IEEE Trans. Syst., Man, Cybern.*, vol. 34, no. 3, pp. 1618–1626, Jun. 2004.
[20] C. Stauffer and E. Grimson, "Learning patterns of activity using real-time tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 747–757, Aug. 2000.
[21] L. Liao, D. Fox, and H. Kautz, "Learning and inferring transportation routines," in *Proc. Nat. Conf. Artif. Intell. AAAI*, 2004, pp. 348–353.
[22] H. Bui, S. Venkatesh, and G. West, "Policy recognition in the abstract hidden Markov models," *J. Artif. Intell. Res.*, vol. 17, pp. 451–499, 2002.
[23] S. Osentoski, V. Manfredi, and S. Mahadevan, "Learning hierarchical models of activity," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sendai, Japan, 2004, pp. 891–896.
[24] R. A. Howard, *Dynamic Programming and Markov Process*. Cambridge, MA: MIT Press, 1960.
[25] A. F. Foka and P. E. Trahanias, "Predictive autonomous robot navigation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Lausanne, Switzerland, Oct. 2002, vol. 1, pp. 490–495.
[26] A. Bruce and G. Gordon, "Better motion prediction for people-tracking," in *Proc. IEEE Int. Conf. Robot. Autom.*, New Orleans, LA, Apr. 2004.
[27] H. Dee and D. Hogg, "Detecting inexplicable behaviour," in *Proc. Brit. Mach. Vis. Conf.*, Kingston, U.K., 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 Trans. Inf. Theory*, vol. IT-32, no. 2, pp. 307–309, Mar. 1986.
[29] D. Vasquez, "Incremental learning for motion prediction of pedestrians and vehicles," Ph.D. dissertation, Inst. Nat. Polytechnique de Grenoble, Grenoble, France, Feb. 2007.
[30] J. Jockusch and H. Ritter, "An instantaneous topological map for correlated stimuli," in *Proc. Int. Joint Conf. Neural Netw.*, Washington, DC, Jul. 1999, vol. 1, 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. Norwell, MA: Kluwer, 1998, pp. 355–368.
[32] H.-M. Dee, "Explaining visible behaviour," Ph.D. dissertation, Univ. Leeds, Leeds, U.K., 2005.

Dizan Vasquez received the Ph.D. degree in computer graphics, computer vision, and robotics from the Institut National Polytechnique de Grenoble, Grenoble, France, in February 2007.

He is a Postoctoral Fellow with the Autonomous Systems Laboratory, Swiss Federal Institute of Technology, Zurich, Switzerland. His main research interest is the automatic construction of motion models and maps from sensor data. Other research interests include computer vision, computational geometry, and self-organizing systems.

Dr. Vasquez is the recipient of the 2007 EURON Georges Giralt award for the best European thesis on robotics.

Thierry Fraichard received the Ph.D. degree in computer science, with a dissertation on motion planning for a nonholonomic mobile in a dynamic workspace, and the Habilitation Ã Diriger des Recherches (Accreditation to Supervise Research) degree for his work entitled "Contributions to motion planning," from the Institut National Polytechnique de Grenoble, Grenoble, France, in April 1992 and March 2006, respectively

From September 2007 to August 2008, he was a Visiting Professor with the Autonomous Systems Laboratory, Swiss Federal Institute of Technology, Zurich, Switzerland. From December 1993 to November 1994, he was a Postdoctoral Fellow with the Manipulation Laboratory, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA. From November 2000 to January 2001 and then again in November 2001, he was a Tan Chin Tuan Fellow with the Intelligent Systems Laboratory, Nanyang Technological University, Singapore. From September to December 2002, he was a Japan Society for the Promotion of Science Fellow with the Distributed Adaptive Research Unit, Riken Institute, Tokyo, Japan. He is currently a Research Fellow with the e-Motion Project Team, French National Institute for Research in Computer Science and Control Rhône-Alpes, Grenoble, France. His research focuses on motion autonomy for vehicles, with a special emphasis on safe navigation, motion planning (for nonholonomic systems, in dynamic workspaces, and in the presence of uncertainty), prediction of the future motion of moving objects, and the design of control architectures for autonomous vehicles.

Christian Laugier received the Ph.D. and "State Doctor" degrees in computer science from Grenoble University, Grenoble, France, in 1976, and 1987, respectively.

He is currently a Research Director with the French National Institute for Research in Computer Science and Control Rhône-Alpes, Grenoble, where he is also the Scientific Leader of the e-Motion Project Team. He is also the Deputy Director of the Laboratoire d'Informatique de Grenoble. He is a Coeditor of several books, e.g., the recent Springer Tracts in Advanced Robotics books on autonomous navigation in dynamic environments and probabilistic reasoning and decision making in sensory-motors systems. He is also a Coeditor for several Special Issues of scientific journals, including the *International Journal of Robotics Research*, *Advanced Robotics*, *Journal of Field Robotics*, *RIA*, and *International Journal of Vehicle Autonomous Systems*. In addition to his research and teaching activities, he participated in the start-up of four industrial companies in the fields of robotics, computer vision, computer graphics, and Bayesian programming. His current research interests are mainly in the areas of motion autonomy, intelligent vehicles, and probabilistic robotics.

Dr. Laugier is a member of several scientific national and international committees, including the Administrative Committee of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) and the IEEE Technical Committee on Intelligent Transportation Systems and Autonomous Vehicles. He has been the General Chair or the Program Chair of several international conferences, including IROS'97, IROS'02, IROS'08, and the Sixth International Conference on Field and Service Robotics. He was a Coeditor of a Special Issue of the IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS.