Intentional Motion On-line Learning and Prediction

Dizan Alejandro Vasquez Govea, Thierry Fraichard, Olivier Aycard, Christian Laugier

Draft Manuscript for Review

Intentional Motion On-line Learning and Prediction

Journal:	*Machine Vision and Applications Journal*
Manuscript ID:	MVA-Apr-06-0093.R1
Manuscript Type:	Special Issue
Keywords:	video surveillance, human activity recognition, surveillance system, vision system, robot navigation

Dizan Vasquez · Thierry Fraichard · Olivier Aycard · Christian Laugier

Intentional Motion On-line Learning and Prediction

Received: date / Accepted: date

Abstract Predicting motion of humans, animals and other objects which move according to internal plans is a challenging problem. Most existing approaches operate in two stages: a) learning typical motion patterns by observing an environment and b) predicting future motion on the basis of the learned patterns. In existing techniques, learning is performed off-line, hence, it is impossible to refine the existing knowledge on the basis of the new observations obtained during the prediction phase.

We propose an approach which uses Hidden Markov Models to represent motion patterns. It is different from similar approaches because it is able to learn and predict in a concurrent fashion thanks to a novel approximate learning approach, based on the Growing Neural Gas algorithm, which estimates both HMM parameters and structure. The found structure has the property of being a planar graph, thus enabling exact inference in linear time with respect to the number of states in the model. Our experiments demonstrate that the technique works in real-time, and is able to produce sound long-term predictions of people motion.

Keywords Trajectory Prediction, Motion Models, Hidden Markov Models, Growing Neural Gas Algorithm

1 Introduction

Motion planning for dynamic environments is a very active research domain. Because the problem is NP-Hard [32], most of the research effort has been directed towards finding algorithms that are able to cope with this complexity. There is, however, another aspect of motion planning that is often overlooked despite its importance: motion planning algorithms need to know in advance the motion of the objects which populate the environment.

The problem is that, in most real applications, the future motion of the moving objects is a priori unknown, making it necessary to predict it on the basis of observations of the objects' past and present states. These observations are gathered using various sensors (*eg* radars, vision systems, etc.) which have limited precision and accuracy.

Until recently, most motion prediction techniques have been based on kinematic or dynamic models that describe how the state (*eg* position and velocity) of an object evolves over time when it is subject to a given control (*eg* acceleration) (cf. [40]). These approaches proceed by estimating the state, using techniques such as the Kalman Filter [21], and then applying the estimate to its motion equations in order to get state predictions.

Although these techniques are able to produce very good short-term predictions, their performance degrades quickly as they try to see further away in the future. This is especially true for humans, vehicles, robots, animals and the like, which are able to modify their trajectory according to factors (*eg* perception, internal state, intentions, etc.) which are not described by their kinematic or dynamic properties.

To address this issue, a different family of approaches has emerged recently. It is based on the idea that, for a given area, moving objects tend to follow typical motion patterns that depend on the objects' nature and the structure of the environment. Such approaches operate in two stages:

1. *Learning stage:* observe the moving objects in the workspace in order to determine the typical motion patterns.
2. *Prediction stage:* use the learnt typical motion patterns to predict the future motion of a given object.

Thus, learning consists in observing a given environment in order to construct a representation of every possible motion pattern. But, how long should we observe the environment in order to construct such a "pattern library"? Given the enormous number of possible patterns for all but the simplest environments, there is not a simple answer. This raises an important problem of existing learning techniques [36, 30, 3]: they use a "learn then predict" (fig. 1) approach, meaning that the system goes through a learning stage where it is presented with a set of observations (an example dataset) from which it builds its pattern models. Then, the models are "frozen" and the system goes into the prediction stage.

E-mail: Firstname.Lastname}@inrialpes.fr
INRIA Rhône-Alpes & Gravir-CNRS
655 av. de l'Europe. Montbonnot.
38334 St Ismier Cedex. France

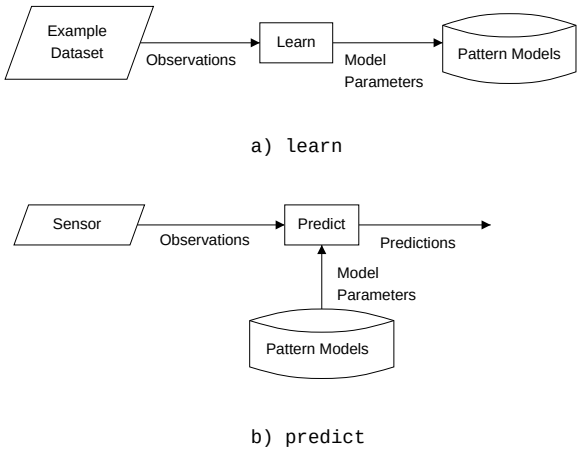


Fig. 1 Learn then predict

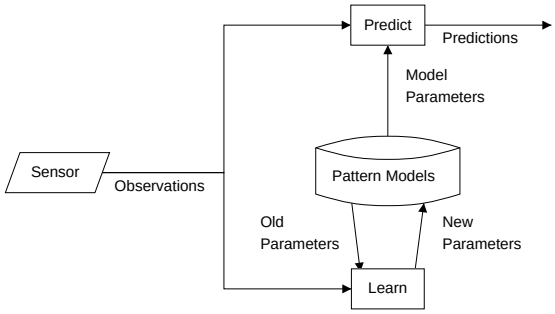


Fig. 2 Learn and Predict

The problem with this approach is that it makes the implicit assumption that all possible motion patterns are included in the example dataset, which, as we have shown, is a difficult condition to meet. In this paper we present an approach which, in contrast to the approaches mentioned above and discussed in detail in §2, works in a "learn and predict" fashion (fig. 2). That is, learning is an incremental process, which continuously refines knowledge on the basis of new observations which are also used for prediction. To the extent of our knowledge, this is the first learning based motion prediction technique in the literature to have this property.

Our work is based on Hidden Markov Models [31], a probabilistic framework used to describe dynamic systems which has been successfully applied to "learn then predict" approaches [30,3]. A Hidden Markov Model (HMM) may be viewed as a graph, where nodes represent states (*eg* places in the environment) and edges represent the transition probability of moving from one node to another in a single time step, the number of nodes and the existence of nodes determines the model's structure. The model also assumes that states are not directly observable but produce observations with a given probability.

Our approach is founded on the hypothesis that objects move in order to reach specific places of the environment called goals. Hence, motion patterns are represented using a different Hidden Markov Model (HMM) for every goal. Each such HMM describes how objects move to reach the corresponding goal. It is this set of HMMs that are used for prediction purposes.

Our learning approach, which is also the main contribution of this paper, is a novel approximate HMM learning technique based on the Growing Neural Gas algorithm [17]. It is able to process observations incrementally (*ie* one by one), in order to find the HMM's structure as well as to estimate the model's transition probabilities. Moreover, the algorithm is adaptive, meaning that it is able to insert or delete states or transitions in order to respond to changes in the underlying phenomenon. The same incrementality and adaptivity properties apply to goal identification, hence, it is able to create models for motion patterns that have just been discovered or to delete old ones when the corresponding patterns are no longer observed. The cost of learning for each iteration is linear with respect to the number of states, as is the inference (*ie* prediction). This enables real-time processing, which is indispensable for a "learn and predict" approach.

The paper is organised as follows: an overview of the related works is presented in §2. Section 3 is an introduction to Hidden Markov Models and the Growing Neural Gas algorithm. Our approach is presented in §4. In section 5, the theoretical aspects of our approach are discussed. Section 6 details our implementation of the algorithm. In section 7 we present the experiments we have carried out as well as the obtained results. Section 8 contains a discussion of future work. Finally, §9 presents our conclusions.

2 Related Works

This section provides an overview of existing motion prediction approaches, which we have roughly classified in three categories: a) kinematic and dynamic approaches; b) discrete state probabilistic techniques; and c) clustering based techniques.

Finally, we also provide a short review of the literature on parameter and structure learning algorithms for Hidden Markov Models.

2.1 Kinematic and Dynamic approaches

Approaches based on the kinematic or dynamic properties of moving objects are often based on the Kalman Filter and Extended Kalman Filter's prediction step [9,20,22]. There are, however other approaches that also use dynamics for prediction: Chien and Koivo [11] proposed the use of a recursive autoregressive time series model whose parameters are estimated using the least mean squared error method. The approach proposed by [41] models motion using Hidden Markov Models to predict motion on the basis of a state

space which is relative to the object. A random walk is applied to predict motion in [28]. In [25] a clustering algorithm is used to predict motion on the basis of a fixed number of previous observations.

In general motion prediction approaches based only on the kinematic and dynamic properties of the objects are accurate in the short term, but fail to produce sound long-term prediction of human motion, mostly due to the fact that human motion depends on other factors than kinematic or dynamic constraints (e.g. plans, goals, perception, etc.).

2.2 Discrete State Probabilistic Techniques

This techniques perform some kind of state partition on the environment and then model motion as transition probabilities between states. One of the first examples of this kind of techniques was proposed by Tadokoro [35], the approach partitioned the environment into a 2D grid, transition probabilities between neighbor cells were assigned by a human expert. Kruse [24] advanced further by proposing a grid based approach which was able to automatically learn transition probabilities.

More recently Abstract Hidden Markov Models [8] have been proposed, this is a hierarchic framework that allows to reason about motion at different abstraction levels or resolutions. In [30] the Expectation-Maximization algorithm has been used to automatically learn its parameters.

2.3 Clustering Based Techniques

This approaches represent behaviors using trajectory prototypes which have been obtained through clustering of observed trajectories. An application of this kind of approaches to human motion was proposed in [23], where an ad-hoc clustering procedure was applied to find trajectory segments which were common between observed trajectories. Gaffney [19] proposed the use of mixture models to cluster trajectories, but did not apply them to human motion. A more standard clustering approach was proposed in [4,3] this approach uses the Expectation-Maximization algorithm to find trajectory models. An interesting feature of this approach is that the found clusters are then converted into Hidden Markov Models in order to perform inference. A similar approach based on pairwise clustering has been presented in [36].

In general, discrete-state and clustering based techniques have different strengths and weaknesses, the former trend to give less precise trajectory predictions than cluster based techniques and are, in general, more expensive in computation time, in the other hand, they are better suited to predict the state distribution probability at any given moment,and they are better able to represent certain unseen situations, like an object switching between behaviors.

2.4 Learning Hidden Markov Models

Since our approach is based on Hidden Markov Models, we will review existent work on Hidden Markov Model learning techniques. These techniques may be divided in two categories: parameter and structure learning.

The *de facto* standard technique for parameter learning is the Baum-Welch algorithm [2] which is a batch learning technique derived from the Expectation-Maximization (EM) algorithm [13]. An incremental approximation of Expectation-Maximization has been proposed in [29], although it does not guarantee convergence it often gives a good approximation [27]. Another approximative incremental approach which is also more general than EM has been proposed in [33].

The problem of structure learning is considered to be more difficult due to the huge search space. The restricted version of the problem consists in choosing the best topology knowing the number of states N, even supposing that the state is fully observable, this means choosing the best out of $O(2^{N^2 \ln N})$ possible directed acyclic graphs (DAG's) which is clearly unfeasible [27]. If the form of the graphs is restricted to trees, the Chow-Liu [12] algorithm may be used to find the optimal Maximum-Likelihood tree in $O(MN^2)$ where M is the number of observation sequences in the training set. The problem is even harder for HMM's, because the state is not observable. Moreover, in many cases even the number of states is unknown.

Due to all the problems mentioned above, approaches found in the literature are mostly heuristic, and is difficult to guarantee even local convergence. A popular approach is *model merging* [34] which starts with creating a state for every observation in the learning data set, and then, merges near states together. Other algorithm is stochastic optimization [16] which performs hill climbing search by choosing between stochastically selected structures. In [38] the search is performed using a genetic algorithm where individuals represent HMM structures. A different approach has been proposed in [7], which combines an entropic prior which favors low entropy (*ie* highly specific) models with trimming of weakly supported parameters and states.

Finally, it is worth mentioning again [3] in this context. It uses trajectory clustering to learn the structure of an HMM where cluster models are represented by disjoint components in the structure graph.

3 Theoretical Framework

This section provides a gentle introduction to the two main tools which are used by our approach. Readers which are already familiar with the Hidden Markov Models or the Growing Neural Gas algorithm may safely skip the corresponding sections and proceed directly to §4.

3.1 Hidden Markov Models

This is a concise discussion on Hidden Markov Models (HMM), the interested reader is referred to [31] for an excellent tutorial on the subject.

Hidden Markov Models are a type of Dynamic Bayesian Network [27] which is often used for the analysis of dynamic models using noisy sensors. They have applications in many different domains, such as speech recognition [31], genomics [15] and robotics [41].

An HMM may be viewed as a stochastic automaton which describes the temporal evolution of a process through a finite number of discrete states. The process progresses in discrete time steps, going from one state into another according to a given *transition probability*. It is assumed that the current state of the system is not directly observable, instead, it produces an output (i.e. observation) with a certain probability known as the *observation probability*.

3.1.1 Representation

An HMM describes the system using three stochastic variables: a) the present state s_t, b) the previous state s_{t-1}, and the current observation o_t, the rest of the model is defined by the following elements:

- The number of discrete states in the model N. A discrete state is denoted by its number $i : 1 \leq i \leq N$.
- The transition probability function, expressed by $P(s_t \mid s_{t-1})$. This probability is represented with a $N \times N$ *transition matrix* A where each element $a_{i,j}$ represents the probability of reaching the state j in the next time step given that the system was in state i.

$$a_{i,j} = P([s_t = j] \mid [s_{t-1} = i]) \tag{1}$$

- The observation probability function, expressed by $P(o_t \mid s_t)$. In general, for each state, the observation probability is represented by a Gaussian distribution[1].

$$P(o_t \mid [s_t = i]) = \mathbb{G}(o_t; \mu_i, \sigma_i) \tag{2}$$

  The set of all the Gaussians' parameters is denoted by $B = \{(\mu_1, \sigma_1), \cdots, (\mu_N, \sigma_N)\}$.
- The initial state distribution, expressed by $P(s_0)$. This represents our knowledge about the state of the system before receiving any observation and is denoted by Π. It is often represented by a uniform distribution $P([s_0 = i]) = \frac{1}{N}$ or by a $N \times 1$ matrix where $P([s_0 = i]) = \Pi_i$.

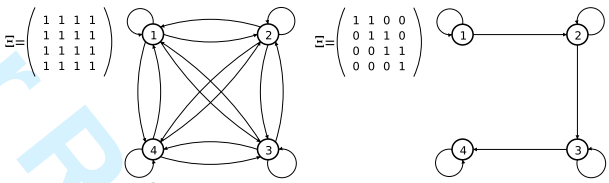
The three probability functions defined above form a Joint Probability Function (JPD) which encodes two conditional independence assumptions: a) knowing the state, observations do not depend on each other, and b) knowing the present state, the past and future states are mutually independent (eq. 3).

$$P(s_{t-1}, s_t, o_t) = P(s_{t-1})P(s_t \mid s_{t-1})P(o_t \mid s_t) \tag{3}$$

The parameters of these three probabilities are often denoted using the compact notation $\lambda = \{A, B, \Pi\}$, they are known together as the *model's parameters*, as opposed to the *model's structure*, which we will now discuss.

The structure of a model, Ξ, is the specification of its number of states N and the valid transitions between states. It may be visualized using the model's connectivity graph, where each vertex represents one state, vertices are joined by directed edges such that every strictly positive element of A ($a_{i,j} > 0$) will be represented by a directed edge from vertex i to vertex j. Hence, a matrix having only strictly positive elements corresponds to a fully connected, or *ergodic*, structure graph, while a sparse matrix corresponds to a graph having fewer edges. This is illustrated in fig. 3.

Structure is important because it affects the complexity of inference in the model, for example, filtering (state estimation) has complexity $O(N^2)$ for a fully connected structure graph, however, if the graph is sparse and each state has at most P predecessors, the complexity is $O(NP)$ [27]. Moreover, structure also influences the quality of inference [6,5].



(a) $O(N^2)$ edges (b) $O(N)$ edges

Fig. 3 Two examples of an order 4 structure graph.

3.1.2 Inference

HMM's are used to perform Bayesian inference: compute the probability distributions of unknown variables given the known ones. Like in other Dynamic Bayesian Networks, inference is often used in HMM's to perform filtering $P(s_t \mid o_{1:t})$, smoothing $P(s_t \mid o_{1:T})$ and prediction $P(s_{t+K} \mid o_{1:t})$. Other two probabilistic questions which are more specific to HMM's are: a) evaluating the probability of a sequence of observations, and b) finding the sequence of states that is more likely to correspond to a sequence of observations. This last question is answered using a dynamic programming technique known as the Viterbi Algorithm [37].

3.1.3 Learning

In order to perform inference with an HMM, it is necessary to define its structure Ξ as well as its parameters λ. But, how

[1] It is also common to represent it by a mixture of Gaussian, or, for discrete observations, by a table.

to choose these values for a particular application? The solution is to estimate (*ie* learn) these values from data.

Most approaches in the literature assume that the structure is known and only try to learn the model's parameters λ. The most widely used of these *parameter learning* approaches is the Baum-Welch algorithm which is an application of Expectation-Maximization [13].

Despite being an active research subject, *structure learning*, no structure learning algorithm has become standard, a review of existing techniques is presented in §2.

3.2 Growing Neural Gas

We will briefly introduce the Growing Neural Gas (GNG) algorithm, which is explained in detail in [17]. The Growing Neural Gas is an unsupervised competitive learning algorithm which may be applied to a variety of problems (e.g. vector quantization, topology learning, pattern classification and clustering). It processes input vectors and constructs a network of N elements called *units*, each of these units has an associated D-dimensional vector called its *weight* (w_i) and is linked to other units called its *neighbors* $\mathfrak{N}_i$. The algorithm starts with two units linked together, then, as new input vectors are processed, units and links may be added or deleted to the network.

The algorithm implicitly partitions the whole space into N *Voronoi regions* V_i which are defined by:

$$V_i = \{x \in \mathbb{R}^D : \|x - u_i\| \leq \|x - u_j\|, \forall j \neq i\} \quad (4)$$

We illustrate this in 2-dimensional space (fig. 4). Each unit occupies its own Voronoi region, given an input vector, the winning unit is the one having its weight vector in the same Voronoi region.

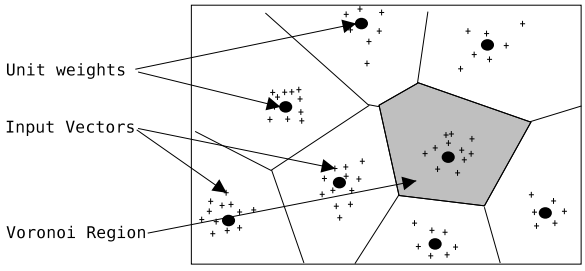


Fig. 4 Implicit partition: there are some 2-dimensional input vectors (crosses). The units' weights are represented by points and Voronoi regions are separated by boundary lines.

During learning, the units' weights are modified in order to minimize the *distortion* which is the mean distance between the winners and their corresponding input vectors. The algorithms also builds incrementally a topology of links which is a subset of the Delaunay triangulation of the units' weight vectors. This means that, in order to be linked together, two units must have a common border in the Voronoi region (see fig. 5).

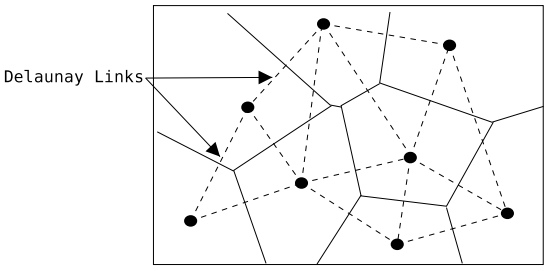


Fig. 5 The Delaunay triangulation for the previous example. Delaunay links are represented by dashed lines. Notice how the number of links corresponds to the number of borders for a given area.

Recapitulating, the GNG algorithm has the following properties:

1. No previous knowledge about the number of units N is required.
2. It is incremental: the model is updated by processing input in a one by one basis.
3. It is adaptive: units and / or links may be added or deleted to reflect changes in the underlying phenomenon.
4. It minimizes the distortion or quantization error: units' weights are placed in order to minimize their distance to input vectors.
5. Links are a subset of the Delaunay triangulation: they connect neighbor Voronoi cells.

But the main interest of GNGs lies in how this features interact, in other words, its ability to determine the number of discrete elements in which the space is partitioned as well as their respective weights and, at the same time, learning the related topology, all of it in an incremental fashion. Moreover, the memory footprint of the structure and the complexity the update algorithm are both linear with respect to the number of states.

On the other hand, these advantages come at a cost of using an approximate algorithm which does not have strict convergence properties. Nevertheless, we consider that this is an acceptable compromise in the context of our application.

4 Proposed Approach

The approach we propose consists of an algorithm which is able to continuously predict future motion based on a model which is constantly improved on the basis of observed motion. This *learn and predict* capability constitutes an advantage over existing techniques, making it unnecessary to have a learning dataset containing at least one example of every observable motion pattern.

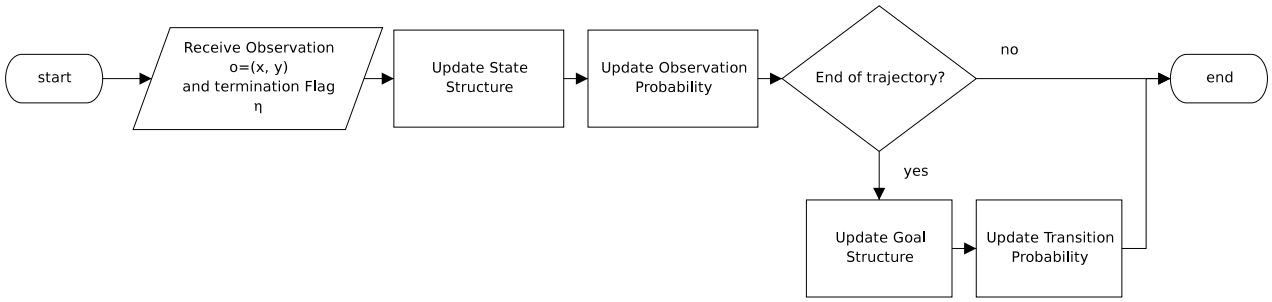


Fig. 6 Learning Overview

The input of our algorithm consists of position observations $o_t = (x_t, y_t)$ and a trajectory termination flag η_t which is set to one when the observation corresponds to the end of a trajectory (*ie* when the object has stopped moving for a long time or exited the environment) and is set to zero otherwise. At every time step, this input is used to compute a probabilistic estimate of the state with a lookahead of K timesteps ($P(s_{t+k} \mid o_t)$) as well as to update the model (fig. 2).

4.1 Representation

Our approach is based on the hypothesis that objects move in order to reach specific places in the environment (*ie* goals). The idea is to identify all the possible goals. Then, for each of these goals, we construct a Hidden Markov Model (*ie* motion model) representing how an object should move in order to reach it.

It is assumed that transition probabilities depend on the goal to be reached (denoted by γ hereafter) and that structure and observation probabilities are independent of the goals, hence, they are the same for all motion models.

These assumptions lead to the following JPD:

$$P(s_{t-1}, s_t, o_t, \gamma) = P(s_{t-1})P(\gamma)P(s_t \mid s_{t-1}, \gamma)P(o_t \mid s_t) \quad (5)$$

So our model is defined by the following:

- The number of goals in the model G.
- The number of states in the model N.
- The transition probability function $P(s_t \mid \gamma, s_{t-1})$. This probability is represented with an unnormalized $G \times N \times N$ transition matrix[2] A where every element $a_{g,i,j}$ represents the number of times that a transition from state i to state j has been observed, given that the object is going to goal g.

$$a_{g,i,j} = P([s_t = j] \mid [\gamma = g][s_{t-1} = i]) \quad (6)$$

- The observation probability function $P(o_t \mid s_t)$. Is represented by Gaussian distributions. The set of all the Gaussians' parameters is denoted by $B = \{(\mu_1, \sigma_1), \cdots, (\mu_N, \sigma_N)\}$.
- The initial goal distribution $P(\gamma_0)$. Is considered to be a uniform distribution $P([s_0 = i]) = \frac{1}{G}, \forall i \in [1, N]$.
- The initial state distribution $P(s_0)$. Is considered to be a uniform distribution $P([s_0 = i]) = \frac{1}{N}, \forall i \in [1, N]$.

The set of *pattern models* shown in fig. 2 consists of structure and parameters. The structure is defined by the number of goals G, the number of states N, and the structure graph Ξ which is the same for all the A_g matrices.

The parameters consists of the transition table A, and the observation parameters B. Given that both $P(s_0)$ and $P(\gamma)$ are considered to be uniform, they do not require any extra parameter.

4.2 Learning

Learning is composed of several subprocesses, an overview is presented in fig. 6.

4.2.1 Updating State Structure

The input observation o_t is passed as an input vector to a GNG network $\mathfrak{G}_{state}$. Nodes of this network represent states, and links represent allowed transitions (*ie* each link represents two transitions, corresponding to the two possible directions). This network is used to update A as follows:

1. If a new unit has been inserted or deleted in $\mathfrak{G}_{state}$, a corresponding row and column are inserted to or deleted from all A_g matrices.
2. If a new link has been added or deleted from $\mathfrak{G}_{state}$, two corresponding transitions are added to or deleted from all A_g.

4.2.2 Updating Observation Probabilities

The observation probabilities are directly computed from $\mathfrak{G}_{state}$. Unit weights are used as mean values for the Gaussians' centers:

$$\mu_i = w_i, \forall w_i \in \mathfrak{G}_{state} \quad (7)$$

[2] This is considered equivalent to having G $N \times N$ matrices $A_g : 1 \leq g \leq G$

And σ_i is estimated from the average distance to unit's neighbors:

$$\sigma_i = \frac{C}{|\mathfrak{N}_i|} \sum_{j \in \mathfrak{N}_i} \|w_i - w_j\| \quad (8)$$

Where C is a weighting constant, in our experiments, we have set it to 0.5.

4.2.3 Updating Goal Structure

Goals are discovered by clustering together observations that correspond to trajectories endpoints, which is indicated by $\eta = 1$. Given that the number of goals is ignored, we will use another GNG structure $\mathfrak{G}_{goal}$ to perform the clustering. The number of clusters G corresponds to the number of units in $\mathfrak{G}_{goal}$. This means that, when a new unit is inserted or deleted in $\mathfrak{G}_{goal}$ a corresponding slice (*ie* matrix) is inserted to or deleted from A.

4.2.4 Updating Transition Probabilities

In this step, the entire observation sequence $\{o_1, \cdots, o_T\}$ for a trajectory is used, this means that it is necessary to store all past observations in an internal data structure.

In order to choose the transition probability table to update, the attained goal γ should be identified, this is done by choosing the goal which is closest to the last observation.

$$\gamma = \arg\min_i \|o_T - w_i\|, \forall w_i \in \mathfrak{G}_{goal} \quad (9)$$

Transition probabilities are then updated by applying maximum likelihood: we use the Viterbi algorithm to find the most likely sequence of states $\{s_1, \cdots, s_T\}$ and then, we increment the corresponding counters in A. This is just an approximation of the Baum-Welch estimation – which performs weighted counting according to the likelihood of every *possible* state sequence – nevertheless, it has been shown [34] that the results obtained with this approximation are comparable to those obtained Baum-Welch.

$$a_{\gamma,s_{t-1},s_t} = a_{\gamma,s_{t-1},s_t} + 1, \forall t : 1 \leq t \leq T-1 \quad (10)$$

A problem with this approach is that the transition counters of new links in the topology will have a much smaller value than those that correspond to older links, hence, old links will dominate. This is solved by multiplying transition weights by a *fading factor* f. In general this is similar to having a bounded counter with a maximum value of $\frac{1}{1-f}$.

$$a_{\gamma,s_t,i} = a_{\gamma,s_t,i} \times f, \forall t, i : 1 \leq t \leq T-1, i \in \mathfrak{N}_{s_t} \quad (11)$$

Instead of normalizing A, normalization is performed when computing the transition probability.

$$P([s_t = j] \mid [\gamma = g][s_{t-1} = i]) = \frac{a_{g,i,j}}{\sum_{k \in \mathfrak{N}_i} a_{g,i,k}} \quad (12)$$

Finally, the data structure that was used to store the observation sequence may be cleared.

4.3 Prediction

Prediction is performed in two steps. First, the belief state (*ie* the joint probability of the present state and goal) is updated using the input observation o_t:

$$P(\gamma, s_t \mid o_t) = \frac{1}{Z} P(o_t \mid s_t) \sum_{s_{t-1}} P(s_t \mid \gamma, s_{t-1}) P(\gamma, s_{t-1} \mid o_{t-1}) \quad (13)$$

Where $P(\gamma, s_{t-1} \mid o_{t-1})$ is the belief state calculated in the previous time step.

Then, motion state is predicted with a look-ahead of K time steps using:

$$P(\gamma, s_{t+K} \mid o_t) = \sum_{s_{t+K-1}} P(s_{t+K} \mid \gamma, s_{t+K-1}) P(\gamma, s_{t+K-1} \mid o_t) \quad (14)$$

Finally, the state is obtained by marginalizing over the goal:

$$P(s_{t+K} \mid o_t) = \sum_{\gamma} P(\gamma, s_{t+K} \mid o_t) \quad (15)$$

An alternative to state prediction is to predict the goal by marginalizing the state from the belief state:

$$P(\gamma \mid o_t) = \sum_{s_t} P(\gamma, s_t \mid o_t) \quad (16)$$

5 Analysis

This section we analyse the aspects of our approach which are different from conventional HMM techniques, explaining the rationale behind them as well as discussing their complexity. The subsection on structure learning with GNG is of particular interest, because it is central to our approach and constitutes our main contribution.

5.1 Learning HMM structure with GNG

As explained in §3.2, state learning uses a Growing Neural Gas network to learn both the total number of states N and the *a priori* state transition structure. The use of the algorithm is straightforward: Each observation received from the sensor system is processed as an input vector by the network. Network units correspond to states in the HMM and links between units correspond to allowed transitions between states.

The use of the GNG algorithm to discretize the space and find the network structure is justified by two rationale:

- Due to the fact that GNG minimizes the distortion, the weights of the network units are good representations of observations in the same cell.
- In order to move from one cell to another, an object should cross one of the cell's borders, which is equivalent to following a Delaunay link.

It should be noted, however, that this arguments only hold in cases where the HMM is being used as a discrete approximation of a phenomenon having a continuous state space, as is the case of motion.

The primary advantage of defining the structure this way is the reduction of the number of elements in the transition matrix. The Delaunay triangulation of the points represented by the unit's weights is a planar graph. This means that the number of links in it is $O(N)$ [14]. As we have mentioned in §3.2, GNG links are a subset of edges in the Delaunay triangulation, hence, they also define a planar graph. So we have effectively reduced the number of allowed transitions in the HMM structure from $O(N^2)$, for an ergodic model to $O(N)$. This is reflected in the cost of inference, since now only $O(N)$ operations are necessary to update the belief state and to perform one prediction step.

There is, however, a condition under which the use of GNG links may be too restrictive: if an object is moving too fast, it will pass through more than one cell in a single time step. In order to deal with this situation, the minimum length of a link l_{min} should be restricted with respect to the maximum speed V_{max} of observed objects and the sampling period of the sensor T:

$$l_{min} \geq V_{max} T \tag{17}$$

5.2 Using GNG to estimate Observation Probabilities

A major difference between our algorithm and Baum-Welch lies in the procedure used to compute the variance of observation probabilities. The procedure we propose is an heuristic method and does not find the best approximation to the variance, however, it allows to capture, at least qualitatively, the related uncertainties. Further work will focus on finding a more precise estimate of the variance.

Finally, it is important to mention that, since our algorithm is based on the GNG algorithm, it does not converge asymptotically, instead, its likelihood increases quickly, and then it tends to oscillate around a local maxima. On the other hand, it is precisely this behavior which is at the heart of the algorithm's adaptivity, and that permits it to estimate the number of discrete states in an incremental manner. Indeed, it is difficult to device an algorithm which converges and at the same time is able to learn newly observed motion patterns because both goals are somewhat contradictory. Since the latter ability is the main motivation behind our work, we have chosen to privilege adaptivity over convergence.

5.3 Prediction

Equations (13) and (14) are obtained by applying Bayes' rule and the conditional independence hypotheses that were used to define the JPD of the model (5).

Equation (13) is derived as follows:

$$P(\gamma, s_t \mid o_{1:t}) = \frac{1}{Z_0} \sum_{s_t - 1} P(\gamma, s_t, s_{t-1}, o_{1:t}) \tag{18a}$$

$$= \frac{1}{Z_0} \sum_{s_t - 1} P(o_t \mid o_{1:t-1}, \gamma, s_t, s_{t-1}) P(\gamma, s_t, s_{t-1}, o_{1:t-1}) \tag{18b}$$

$$= \frac{1}{Z_0} P(o_t \mid s_t) \sum_{s_t - 1} P(\gamma, s_t, s_{t-1}, o_{1:t-1}) \tag{18c}$$

$$= \frac{1}{Z_0} P(o_t \mid s_t) \sum_{s_t - 1} P(o_{1:t-1}) P(\gamma, s_{t-1} \mid o_{1:t-1}) P(s_t \mid \gamma, s_{t-1}) \tag{18d}$$

$$= \frac{1}{Z_1} P(o_t \mid s_t) \sum_{s_t - 1} P(\gamma, s_{t-1} \mid o_{1:t-1}) P(s_t \mid \gamma, s_{t-1}) \tag{18e}$$

In this derivation, passing from (18d) to (18e) is possible because $P(o_{1:t-1})$ is a constant, hence, it may be integrated into the denominator. As shown, the complexity of belief state update is $O(GN^2)$, but it is possible to reduce the complexity to $O(GN)$ by exploiting HMM structure and summing only over valid transitions[3]:

$$P(\gamma, s_t \mid o_{1:t}) = \frac{1}{Z_1} P(o_t \mid s_t) \sum_{i : s_t \in \Re_i} P(\gamma, [s_{t-1} = i] \mid o_{1:t-1}) \times P(s_t \mid \gamma, [s_{t-1} = i]) \tag{19}$$

State prediction (15) is derived in an analogous way:

$$P(\gamma, s_{t+K} \mid o_{1:t}) = \sum_{s_t - 1} P(\gamma, s_{t+K}, s_{t+K-1} \mid o_{1:t}) \tag{20a}$$

$$= \sum_{s_t - 1} P(\gamma, s_{t+K-1} \mid o_{1:t}) P(s_{t+K} \mid \gamma, s_{t+K-1}) \tag{20b}$$

[3] In reality, the complexity is $O(GND)$, where D is the degree of the structure graph, but in planar graphs D is constant.

Here the HMM structure may be again exploited to reduce complexity from $O(GN^2K)$ to $O(GNK)$:

$$P(\gamma, s_{t+K} \mid o_{1:t}) = \sum_{i:s_{t+K} \in \mathfrak{N}_i} P(\gamma, [s_{t+K-1} = i] \mid o_{1:t}) \times P(s_{t+K} \mid \gamma, [s_{t+K-1} = i]) \quad (21)$$

6 Implementation

We have implemented our approach using the C^{++} language on a Linux platform, this section explains our implementation decisions as well as the operational features of the algorithm.

6.1 GNG Parameters and Initialization

The GNG algorithm has six parameters, instead of calibrating them, we have used the same values that Fritzke has used in [18] for the state GNG[4]. In the case of the goal GNG, we have kept the same parameters, excepting for λ which has been set to 100 in order to accelerate goal learning.

Both GNGs have been initialized with two units having random weights constrained to lie within the environment's limits.

6.2 Data Structures

In order to exploit the sparse structure of both the GNG network and the transition matrices A_g, we have represented them using a list of neighbors for every state/unit. This allows to effectively perform sums as well as insert/delete operations.

6.3 Dealing with multiple objects

It is worth noting that, in all our experiments, we have assumed that there was only one object moving at the same time. On the other hand, our approach may be easily extended to handle more than one simultaneously moving object as long as interactions between objects are not modelled.

7 Experimental Results

7.1 Test environment

The environment we have chosen to validate our approach is the main lobby of our research institute. It features the main entrance to the building, a self-information directory post, the front-desk, a cafeteria area and a number of doors leading to various halls, rooms and auditoriums (Fig. 7). This environment is the heart of the institute, all the personnel passes through it at some point during the day for a reason or another (going in or out, coffee break, attending a lecture, etc). This environment is interesting because the motion patterns of the people is rich and the flow of people sufficient to ensure that it will be possible to gather a significant number of observations.

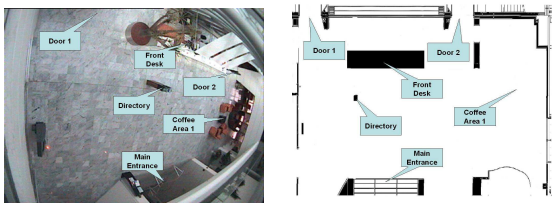


Fig. 7 Inria's main lobby: video view (left) and 2D map (right).

The testing of our approach has been done in two stages. First, we have used observation data coming from a software simulating the trajectories of people in the main lobby. Then we have used live observations coming from a visual tracking system.

The interest of using simulated data is that it allows to evaluate our approach in controlled conditions for which it is possible for instance to predefine the number of motion patterns.

In both cases, we have gathered a significant number of observations. We have presented 1000 trajectories to the learning algorithms in order to build an initial model, before starting to measure the results. Then, prediction performed has been measured using another 300 trajectories. The test results obtained with simulated data (resp. real data) are presented in section 7.2 (resp. 7.3).

7.2 Simulated Observations

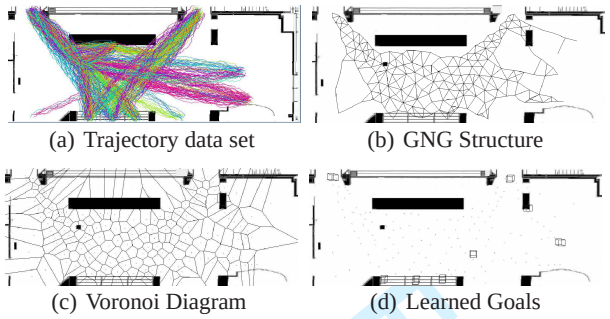
7.2.1 Getting the Observations

The simulating system we have developed relies upon a number of control points representing "'places of interest" of the environment such as the doors, the front-desk, etc. Based upon this set of control points, a set of 32 typical motions patterns has been defined. Each motion pattern consists in a sequence of control points to be traversed. An observed trajectory is computed in the following way: first, a motion pattern is randomly chosen. This motion pattern provides a set of control points. Then, *goal points* corresponding to each of these control points are randomly generated using two-dimensional Gaussian distributions whose mean value are the control points. Finally, the motion between two goal points in the sequence is simulated using discrete, even-size steps in a direction drawn from a Gaussian distribution whose mean value is the direction of the next goal point. Switching from one goal point to the next is done when the distance to the current goal point is below a predefined threshold.

[4] The parameters were: $\lambda = 300$, $\varepsilon_b = 0.05$, $\varepsilon_n = 0.0006$, $\alpha = 0.5$, $\beta = 0.0005$ and $a_{max} = 88$

We have generated the 1300 trajectories that were required for our experiments. an image of trajectories in the data set are presented in fig. 8.

7.2.2 Learning Results



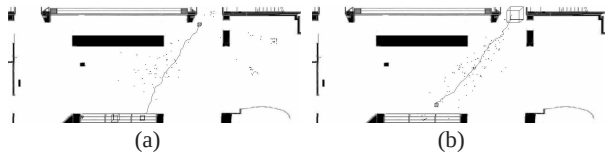
(a) Trajectory data set (b) GNG Structure

(c) Voronoi Diagram (d) Learned Goals

Fig. 8 Overview of the learning process (simulated data).

We have run our algorithm against the simulated data set 8(a). The algorithm took about 2 minutes to process the 1000 initial trajectories, which contained a total of 58,262 observations meaning an average processing frame rate of about 480 observations per second. As a result of the learning process, a structure having 196 states has been found (figs. 8(b) and 8(c)). Also a total of eight goals were identified, as it may be seen in fig. 8(d) the detected goals seem to correspond to many of the interesting points shown in fig. 7.

7.2.3 Prediction Results



(a) (b)

Fig. 9 Prediction Examples (simulated data).

Figure 9 illustrates the prediction process. The figures show the current position of the object as a small cube and – for reference – the "real" future trajectory of the object as a solid line. The estimated goals are marked by cubes whose size vary with their estimated probability. Finally, the state probability for $t+3$ seconds has been illustrated with particles, where a higher concentration of particles indicates a higher probability of being in that area three seconds after the prediction has been made.

In order to test prediction performance, we measure the distance between the predicted goal and the real final destination of the object (fig. 10). For each of the 300 trajectories

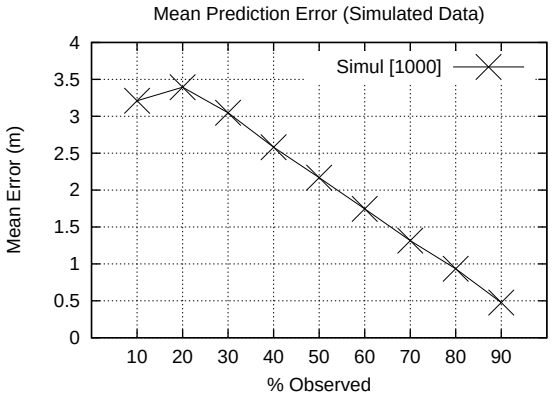


Fig. 10 Prediction Error (simulated data).

we measure this distance when 10% of the total trajectory has been seen, then, we do the same for 20% of the total trajectory and so on until 90%.

We think that the obtained results are quite good. Even when only 30% of the total trajectory is used to predict, the mean error is of about 3 meters which is relatively low with respect to the size of the environment and the distance between goals.

7.3 Real Observations

7.3.1 Getting the observations

To gather observations about the motions performed in the test environment, we use the visual tracking system proposed in [10]. This system detects and tracks objects moving in the images of a video stream. The information collected (position and size of the moving object, etc.) is then projected into the 2D map of the environment. The data flow of the overall tracking system is depicted in Fig. 11. It features the detector-tracker, a module to correct the distortion of the video camera, and a final module to project the information in the 2D map of the floor. These modules are detailed in the upcoming paragraphs.

*Camera and Tracker.*A single wide-angle camera mounted over one of the lobby's corners is used. The camera is directly connected to the host computer. The tracker processes raw data coming from the camera and outputs data consisting of sets of observations, (*ie* frames), that the tracker sends at regular time steps. Every observation consists of an identification number (ID), the x and y coordinates of the moving object's gravity centre in the image coordinate system, the width and height of the object's bounding box and the orientation of this bounding box. A trajectory is a sequence of regularly sampled observations consisting of the target's id, centre of gravity, width, height and orientation.

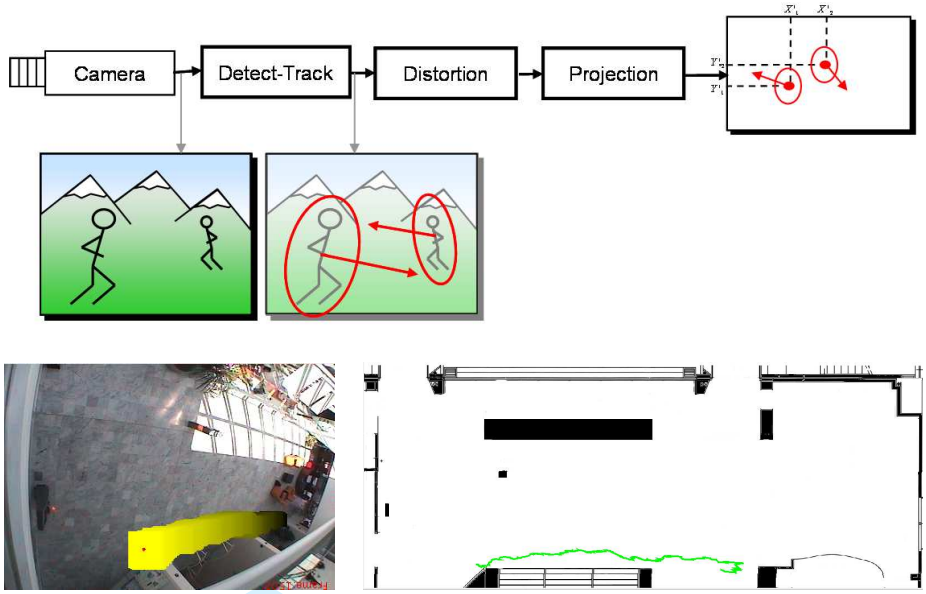


Fig. 11 Architecture of the visual tracking system (top). Example of a motion observed in the image (bottom-left), and its projection in the 2D map of the environment (bottom-right).

*Distortion correction and homographic projection.*Due to the use of a wide-angle lens, the image is subject to heavy distortion, which must be corrected before projecting the image into the world coordinate system. We have used four coefficient distortion correction as described in [39].

The corrected target gravity centres are multiplied by a precalculated homography matrix in order to project them into the world plane. It is worth noting that the target's centre most often corresponds to a point which is located higher than the floor level, thus an error is introduced by projecting it into the floor. However, as the error is consistent for targets of similar height, we have decided that it is acceptable at this stage of our work.

*Data Association.*Due to the fact that we have an environment where there are multiple objects moving at the same time, the tracker does not always keep objects ID's, which is a requirement for our approach. Hence, in order to improve ID keeping, we post-process projected data applying the Joint Probabilistic Data Association (JPDA) algorithm [1] the fact of applying the algorithm on the world coordinate system helps to calibrate and improve the results of the algorithm.

7.3.2 Learning Results

Figure 12(a) shows the real data set that we have obtained. It is important to notice that trajectories in the data set do not cover the entire environment. This happens because of clipping due to the distortion correction algorithm we have used.

The algorithm took about 70 seconds to process the 36,444 observations of the first 1000 trajectories in the data set for an average of about 520 observations per second. The found

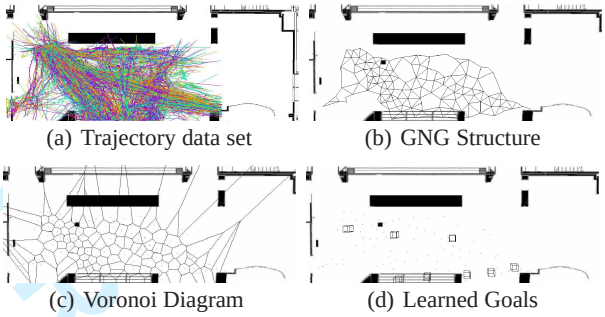


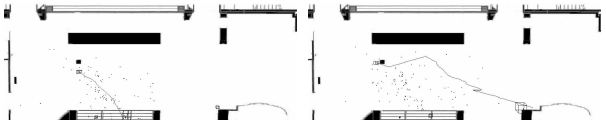
Fig. 12 Overview of the learning process (real data).

structure (figs. 12(b) and 12(c)) consisted of 123 states. The algorithm has detected eight goals, shown in fig. 12(d). In this case, some of the goals do not correspond to interesting points of the environments. This happens because of two factors: a) clipping reduced the environment an creates some fake entry and exit points, and b) the tracking system sometimes losses the identifier of an object resulting in single trajectories being broken down in several smaller trajectories.

7.3.3 Prediction Results

Figure 13 illustrates prediction based on real data. The example in the right is especially interesting because, although state prediction does not seem to correspond to the real trajectory, the goal has been correctly predicted.

In fig. 14 we present the results of our prediction performance measure (see §7.2.3). Here, the results are even better than for simulated data, which may be surprising at first. The reason is again clipping, which reduces the effective size



(a) (b)

Fig. 13 Prediction Examples (real data).

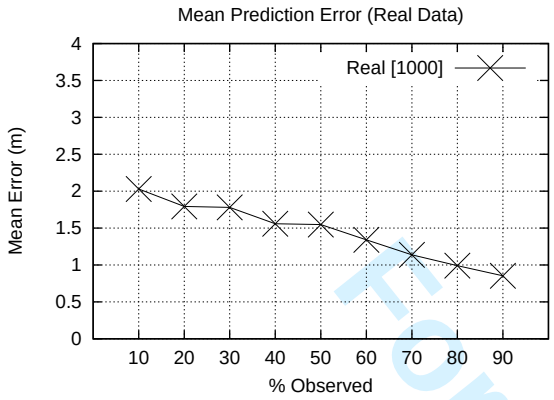


Fig. 14 Prediction Error (real data).

of the environment, thus bringing goals closer together. In the other hand, the noisy nature of real data and is reflected by the slower convergence of real data when compared with simulated one.

8 Future Work

We have started to work on the application of our approach to a different setting: the ParkView experimental platform, which is able to track a car moving in a parking lot (fig. 15). In our first experiments we have experienced a problem with GNG: parking places get underrepresented due to the relatively low likelihood of observing motion in them compared to observing motion in the parking lanes or corridors. This has motivated us to try a different algorithm: Grow When Required [26] which seems to build better representations of areas having low observation likelihood.

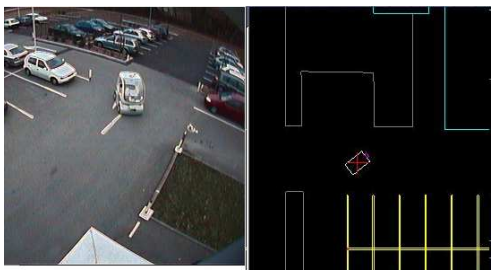


Fig. 15 The ParkView platform: *left*) camera view of the Cycab experimental car moving in the parking lot of the laboratory; *right*) the Cycab as detected on the tracking system.

In the medium term, a number of lines of work are being considered: a) including velocity and object size in the space representation; b) modelling of semi-dynamic objects such as doors which may be either open or closed; c) the extension of the algorithm to learn hierarchical plan models such as Abstract Hidden Markov Models [8].

A longer term research project would be to model interactions between moving objects such as collision avoidance or pursuit behaviors.

9 Conclusions

In this document, we have presented a method for learning motion patterns from observations and, at the same time, use the learned patterns to predict future motion. Our approach makes the hypothesis that objects move in order to reach specific places (*ie* goals) in the environment and represents motion using a Hidden Markov Model for every goal.

The main contribution of this paper is the application of the Growing Neural Gas algorithm in order to enable goal identification as well as on-line structure and parameter learning of the Hidden Markov Models' used to represent patterns. The found HMM structure is a planar graph, which allows exact inference with a computation cost which is linear with respect to the number of states. Thanks to this, we have been able to implement a "learn and predict" approach, thus allowing the continuous improvement of existent knowledge on the basis of new observations. To the best of our knowledge no other technique in the literature is able to do that.

The technique has been implemented and applied to both real and simulated data. The experiments show that the learned model may be used to efficiently predict the intended goal of an object. Moreover, this is performed in real time.

Acknowledgements This work has been partially supported by a Conacyt scholarship. We also want to thank the support of the CNRS Robea ParkNav and the Lafmi NavDyn Projects.

References

1. Bar-Shalom, Y., Fortmann, T.E.: Tracking and data association. No. 179 in Mathematics in science and engineering. Academic Press, Boston;London (1988)
2. Baum, L.E.: An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes. In: O. Shisha (ed.) Inequalities III: Proceedings of the Third Symposium on Inequalities, pp. 1–8. Academic Press, University of California, Los Angeles (1972)
3. Bennewitz, M., Burgard, W., Cielniak, G., Thrun, S.: Learning motion patterns of people for compliant robot motion. Internationl Journal of Robotics Research **24**(1), 31–48 (2005)
4. Bennewitz, M., Burgard, W., Thrun, S.: Learning motion patterns of persons for mobile service robots. In: Proceedings of the IEEE Int. Conf. On Robotics and Automation, pp. 3601–3606. Washington, USA (2002)
5. Binsztok, H., Artières, T.: Learning model structure from data: an application to on-line handwriting. Electronic Letter on Computer Vision and Image Analysis **5**(2) (2005)

6. Brand, M.: Structure learning in conditional probability models via an entropic prior and parameter extinction. Tech. rep., MERL a Mitsubishi Electric Research Laboratory (1998)
7. Brand, M.: Structure learning in conditional probability models via an entropic prior and parameter extinction. Neural Computation **11**(5), 1155–1182 (1999)
8. Bui, H., Venkatesh, S., West, G.: Policy recognition in the abstract hidden markov models. Journal of Artificial Intelligence Research **17**, 451–499 (2002)
9. Burlina, P., DeMenthon, D., Davis, L.S.: Navigation with uncertainty: Reaching a goal in a high collision risk region. In: Proceedings of the 1992 IEEE Int. Conf. on Robotics and Automation. Nice, France (1992)
10. Caporossi, A., Hall, D., Reignier, P., Crowley, J.: Robust visual tracking from dynamic control of processing. In: International Workshop on Performance Evaluation of Tracking and Surveillance, pp. 23–31. Prague, Czech Republic (2004)
11. Chien, Y., Koivo, A.: Visual feedback to predict obstacle motion for on-line collision-free trajectory planning of cylindrical robots. IEEE/RSJ International Workshop on Intellingent Robots and Systems pp. 612–618 (1989)
12. Chow, C., Liu, C.: Approximating discrete probability distributions with dependence trees. IEEE Transactions on Information Theory **IT-14**(3) (1968)
13. Dempster A.and Laird, N., , Rubin, D.: Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society **9**(1), 1–38 (1977). Series B
14. Diestel, R.: Graph Theory, 3^d edn. Graduate Texts in Mathematics. Springer-Verlag, Heidelberg, New York (2005)
15. Durbin, R., Eddy, S., Krogh, A., Mitchison, G.: Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids. Cambridge University Press, Cambridge (1998)
16. Freitag, D., McCallum, A.: Information extraction with hmm structures learned by stochastic optimization. In: Proc. of the Seventeenth Nat. Conf. on Artificial Intelligence and Twelfth Conf. on Innovative Applications of Artificial Intelligence, pp. 584–589. AAAI Press / The MIT Press, Austin, Texas, USA (2000)
17. Fritzke, B.: A growing neural gas network learns topologies. Advances in Neural Information Processing Systems (1995)
18. Fritzke, B.: Some competitive learning methods (1998). Paper Draft. URL ftp://ftp.neuroinformatik.ruhr-uni-bochum.de/pub/software/NN/DemoGNG/sclm.ps.gz
19. Gaffney, S., Smyth, P.: Trajectory clustering with mixtures of regression models. Tech. Rep. 99-15, University of Californa, Irvine (1999)
20. Han, K., Veloso, M.: Physical model based multi-objects tracking and prediction in robosoccer. In: Working notes of the AAAI 1997 Fall Symposium on Model-directed Autonomous Systems. MIT, Boston (1997)
21. Kalman, R.: A new approach to linear filtering and prediction problems. Trans. Am. Soc. Mech. Eng., Series D, Journal of Basic Engineering **82**, 35–45 (1960)
22. Kawase, T., Tsurunosono, H., Ehara, N., Sasao, I.: Two-stage kalman estimator using advanced circular prediction for maneuvering target tracking. In: Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing, pp. 2453–2456. Seattle, WA (US) (1998)
23. Kruse, E., Gusche, R., Wahl, F.M.: Acquisition of statistical motion patterns in dynamic environments and their application to mobile robot motion planning. In: Proceedings of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, pp. 713–717. Grenoble, France (1997)
24. Kruse, E., Gutsche, R., Wahl, F.: Estimation of collision probabilities in dynamic environments for path planning with minimum collision probability. In: Proceedings of IROS, pp. 1288–1295 (1996)
25. Liu, P.X., Meng, M., Hu, C.: Online data-driven fuzzy clustering with applications to real-time robotic tracking. In: Proceedings of the IEEE Int. Conf. On Robotics and Automation. New Orleans, LO (USA) (2004)
26. Marsland, S., Shapiro, J., Nehmzow, U.: A self-organizing network that grows when required. Neural Networks (2002)
27. Murphy, K.P.: Dynamic bayesian networks: Representation, inference and learning. Ph.D. thesis, University of California, Berkeley (USA) (2002)
28. Nam, Y.S., Lee, B.H., Kim, M.S.: View-time based moving obstacle avoidance using stochastic prediction of obstacle motion. In: Proceedings of the 1996 IEEE Int. Conf. on Robotics and Automation. Minneapolis, USA (1996)
29. Neal, R.M., Hinton, G.E.: A new view of the EM algorithm that justifies incremental, sparse and other variants. In: M.I. Jordan (ed.) Learning in Graphical Models, pp. 355–368. Kluwer Academic Publishers (1998)
30. Osentoski, S., Manfredi, V., Mahadevan, S.: Learning hierarchical models of activity. In: IEEE/RSJ International Conference on Intelligent Robots and Systems. Sendai, Japan (2004)
31. Rabiner, L.R.: A tutorial on hidden markov models and selected applications in speech recognition. Readings in speech recognition pp. 267–296 (1990)
32. Reif, J., Sharir, M.: Motion planning in the presence of moving obstacles. In: Symp. on the Foundations of Computer Science, pp. 144–154. Portland (OR) USA (1985)
33. Singer, Y., Warmuth, M.K.: Training algorithms for hidden markov models using entropy based distance functions. In: Advances in Neural Information Processing Systems 9, NIPS, pp. 641–647. MIT Press, Denver, CO (USA) December 2-5, 1996 (1996)
34. Stolcke, A., Omohundro, S.: Hidden markov model induction by bayesian model merging. In: S.J. Hanson, J.D. Cowan, C.L. Giles (eds.) Advances in Neural Information Processing Systems, vol. 5, pp. 11–18. Morgan Kaufmann, San Mateo, CA, Denver, USA (1993)
35. Tadokoro, S., Hayashi, M., Manabe, Y., Nakami, Y., Takamori, T.: Motion planner of mobile robots which avoid moving human obstacles on the basis of stochastic prediction. In: IEEE International Conference on Systems, Man and Cybernetics, pp. 3286–3291 (1995)
36. Vasquez, D., Fraichard, T.: Motion prediction for moving objects: a statistical approach. In: Proc. of the IEEE Int. Conf. on Robotics and Automation, pp. 3931–3936. New Orleans, LA (US) (2004)
37. Viterbi, A.J.: Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. IEEE Transactions on Information Theory **IT-13**(2), 260–269 (1967)
38. Won, K.J., Prügel-Bennet, A., Krogh, A.: Training hmm structure with genetic algorithm for biological sequence analysis. Bioinformatics **20**(18), 3613–3619 (2004)
39. Zhang, Z.: A flexible new technique for camera calibration. IEEE Transactions on Pattern Analysis and Machine Intelligence **22**(11), 1330:1334 (2000)
40. Zhu, Q.: A stochastic algorithm for obstacle motion prediction in visual guidance of robot motion. The IEEE International Conference on Systems Engineering pp. 216–219 (1990)
41. Zhu, Q.: Hidden markov model for dynamic obstacle avoidance of mobile robot navigation. IEEE Transacions on Robotics and Automation **7**, 390–397 (1991)