Intentional Motion On-line Learning and Prediction

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

Intentional Motion On-line Learning and Prediction*

Dizan Vasquez, Thierry Fraichard, Olivier Aycard, and Christian Laugier

Inria Rhône-Alpes
http://emotion.inrialpes.fr

Summary. Motion prediction for objects which are able to decide their trajectory on the basis of a planning or decision process (e.g. humans and robots) is a challenging problem. Most existing approaches operate in two stages: a) learning, which consists in observing the environment in order to identify and model possible motion patterns or plans and b) prediction, which uses the learned plans in order to predict future motions. 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. This paper proposes a novel learning approach which represents plans as Hidden Markov Models and is able to estimate the parameters and structure of those models in an incremental fashion by using the Growing Neural Gas algorithm. Our experiments demonstrate that the technique works in real-time, is able to operate concurrently with prediction and that the resulting model produces long-term predictions.

1 Introduction and Related Work

In order to successfully interact with a dynamic environment, a person, a robot or any other autonomous entity needs to reason about how the objects which populate this environment are going to move in the future. However, this knowledge about the future is often unavailable *a priori*, hence it is necessary to resort to prediction: estimate future motion based on available knowledge about the object's present and past states. This explains the importance of prediction techniques for a number of research domains like motion planning, tele-surveillance and automatic traffic control [1, 2].

This work focuses on motion prediction for objects which are able to perform trajectories as a result of an internal motion planning process or decision mechanism (*e.g.* persons, animals and robots). It is assumed that such plans

* This work has been partially supported by a Conacyt scholarship. We also want to thank the support of the french CNRS Robea ParkNav and the Predit Mobivip projects.

are made *with the intention* to reach a specific goal, thus the name *intentional motion* which will be used hereafter to designate this kind of motion.

Assuming that the object's decision mechanism as well as all the relevant variables at every time step (*e.g.* internal state, sensorial input, etc.) are known, predicting its trajectory consists in replicating the planning process in order to find the intended trajectory. However, this assumption is not realistic. Neither the planning model nor the variables are known or observable (what is the decision mechanism of a human being?) and they must be inferred from observed motion before performing prediction. This leads to the following decomposition of the problem:

- *Learning.* Construct a plan representation based on observations.
- *Prediction.* Use the representation obtained during learning to estimate future states on the basis of present knowledge.

Thus, learning consists in observing a given environment in order to construct a representation of every possible plan for it. But, how long should we observe the environment in order to construct such a "plan library"? Given the enormous number of possible plans for all but the simplest environments, there is not a simple answer. This raises an important problem of existing learning techniques (e.g. [3, 4]): the use of a "learn then predict" 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 construct its plan models. Then, the plan library is "frozen" and the system goes into the prediction stage.

The problem with this approach is that it makes the assumption that all possible plans are included in the example dataset, which, as we have shown, is a difficult condition to meet. This paper addresses the problem by proposing a "learn and predict" approach which is able to learn in an incremental fashion (*ie* by continuously refining its knowledge on the basis of new observations used for prediction). To the extent of our knowledge, this is the first intentional motion prediction technique in the literature to have this property.

Learning techniques used by the "learn then predict" approaches are very diverse. For example in [5] plans are modeled as series of straight motion segments which are clustered together. In [3] and [6], typical behaviors are learned by clustering whole trajectories. In [7] Bui proposes Abstract Hidden Markov Models as a way to represent plans as hierarchies of probabilistic sub-plans or policies. Although the approach does not define an automatized learning mechanism, this has been done in [4] by using the Expectation-Maximization algorithm.

In this paper, we present an approach which is able to continuously learn from observations in an incremental fashion. It models plans as Hidden Markov Models (HMM)[8] augmented with a variable which indicates the goal that the plan intends to reach[2]. The learning algorithm is composed of

[2] An HMM is a stochastic finite-state automaton which models a process whose state evolves according to a transition probability at discrete time-steps. The

two modules: in the first one, the Growing Neural Gas algorithm [9] is used to estimate both the set of states in the model and the observation probabilities. The second module identifies goals and then uses a Maximum-Likelihood criterion to update the transition probability of the model. As mentioned above, the technique determines the number of goals and states in the model, thus learning the structure of the underlying HMM.

The rest of the paper is structured as follows: section 2 presents an overview of the problem. Section 3 discusses the details of our HMM-based probabilistic mode and describes how it is used to model plans. The details of the learning algorithm are presented in section 4. Section 5 discusses the experimental results. The paper ends by exposing our conclusions and explaining future research directions.

2 Problem Overview

This paper proposes an unsupervised learning algorithm which constructs plan representations by observing the motion of objects (e.g. pedestrians, vehicles, etc.) moving in a given environment. Plans are modelled as Hidden Markov Models augmented with a variable γ which is used to represent the particular goal that the object intends to reach.

The input of the learning algorithm is a continuous stream of observations $o_t = \{o_1, o_2, \cdots\}$ gathered through a tracking system. In order to keep notation simple, we will assume that no more than one object is observed at the same time, noting that the approach is easily generalizable to the multi-object case. It will also be assumed that the tracking system can determine when the object has stopped or exited the environment.

Every observation $o_t = (x_t, y_t, \eta_t)$ returned by the tracker consists of an estimate of the object's position[3] at time t and a binary variable η_t which indicates whether the object has reached the end of its trajectory ($\eta = 1$) or not ($\eta = 0$). A trajectory ends when the object stops moving or exits the environment.

Learning will consist in estimating the parameters of the slightly modified HMM which will be presented in the following section.

---

state of the process may only be observed through a noisy sensor, the probability that a measure provided by the sensor corresponds to a given state is known as the observation probability.

[3] Higher-dimensional observations (*ie* (x_t, y_t, x'_t, y'_t)) may also be used as input by the algorithm.

3 Probabilistic Model Definition

In order to develop our model, we will start from the HMM[4] joint probability distribution (JPD) for a single time-step, which may be written as follows:

$$p(q_t, q_{t-1}, o_t) = p(q_{t-1})p(q_t \mid q_{t-1})p(o_t \mid q_t) \tag{1}$$

Where q_{t-1} and q_t represent the state at time $t-1$ and t, respectively, and o_t represents the observation returned by the sensor at time t. The decomposition contains the three probabilities that define an HMM: a) the state prior, or belief state $p(q_{t-1})$; b) the transition probability $p(q_t \mid q_{t-1})$ and c) the observation probability $p(o_t \mid q_t)$.

In the context of this work. Discrete states will correspond to positions in the environment and transition probabilities will depend on the particular goal that an object intends to reach. In order to account for different goals, we will augment the HMM with a variable γ which is used to represent them:

$$p(q_t, q_{t-1}, o_t, \gamma) = p(q_{t-1})p(\gamma)p(q_t \mid \gamma, q_{t-1})p(o_t \mid q_t) \tag{2}$$

This JPD has been obtained from eq. 1 by making two additional conditional independence assumptions: a) The goal does not depend on the previous state $p(\gamma \mid q_{t-1}) = p(\gamma)$ and b) given the state, the observation is independent of the goal $p(o_t \mid q_t, \gamma) = p(o_t \mid q_t)$.

Due to the fact that γ is not time-dependent, this may be regarded as having a different Markov model for every value of γ, where all such models share the same observation probabilities and number of states. The idea is a simplified version (*ie* without the actions) of the probabilistic planning technique known as Markov Decision Processes.

Having defined a JPD, we will proceed to specify all the model's relevant variables as well as their respective domains:

- $N \in \mathbb{N}$: The total number of discrete states in the model. These states correspond to positions in the environment.
- $q_t, q_{t-1} \in [1, N]$: The object's states at time t and $t-1$, respectively.
- $o_t \in \mathbb{R}^2$: The object's state estimation returned by the sensor at time t. (*ie* the observation variable).
- $G \in \mathbb{N}$: The total number of goals in the model. The goals correspond to specific *places* in the environment (*ie* it may correspond to many discrete states).
- $\gamma \in [1, G]$: The goal that the object intends to reach.

Finally, we the representations we have chosen for the probability distributions:

[4] In this section, it is assumed that the reader is familiar with Hidden Markov Models. The interested reader is referred to [8] for an excellent tutorial on the subject.

- $p(q_t \mid \gamma, q_{t-1})$: Table, it will be further described in §4.3.
- $p(o_t \mid [q_t = i])$: Gaussian $\mathcal{G}(\mu_i, \sigma_i)$.
- $p(q_0)$: Uniform $\mathcal{U}_N = \frac{1}{N}$. This probability is used to initialize the belief state for a new trajectory.
- $p(\gamma)$: Uniform $\mathcal{U}_G = \frac{1}{G}$. This probability is used to initialize the goal's belief for a new trajectory.

4 Parameter Learning algorithm

On the basis of the model specification presented in §3 it is possible to define a learning algorithm which consists in estimating parameters from observations. Having defined the priors as uniform distributions, this leaves four parameters to be estimated: the transition and observation probabilities, and the values for N and G. It is worth noting that, by learning both N and G, the proposed technique is able to learn the structure of the model. This is an significant departure from existent techniques [3, 4], which depend on values fixed *a priori*.

Assuming that, for every observed trajectory the associated goal is known, learning may be performed using the Baum-Welch algorithm [10] which is a specialization of Expectation-Maximization [11] and has become the standard learning technique for HMM's. However, it has two problems which prevent its application to our particular problem: a) it is not incremental and b) it needs to know the number of states to be learned *a priori*. The first problem may be solved by using incremental variants of the algorithm [12, 13], but the second one is more difficult to solve and is not a trivial task. Moreover, we want to deal with the general case, where goals are not known beforehand and should be identified.

The approach proposed in this paper takes a different approach by splitting the problem in three tasks:

1. *State GNG*. The observation probability $p(o_t \mid q_t)$ and the number of states N are estimated using the Growing Neural Gas algorithm (§4.1).
2. *Goal GNG*. Another instance of GNG is used to estimate the number of goals G as well as their position.
3. *Viterbi Counting*. The Viterbi algorithm [14] is used to perform a maximum likelihood (ML) estimation of the transition probability $p(q_t \mid \gamma, q_{t-1})$. This estimation uses the outputs of tasks 1 and 2 (§4.3).

The rest of the section provides the details of the tree learning tasks.

4.1 Learning discrete states and observation probabilities

The observation probability for a given state $p(o_t \mid [q_t = i])$ is defined as a gaussian. Therefore, the learning algorithm should estimate the mean value μ_i and standard deviation σ_i for the N states.

This rises the question of the "correct" value for N, which is an important question. The state space is continuous, when it is mapped to a finite set of discrete values an error is introduced in the representation. The number of states allows to trade off accuracy and computational efficiency. By incrementing the value of N the approximation error – also known as distortion – is reduced at the expense of additional calculation burden.

There is another way of reducing the distortion: discrete states may be placed in such a way that the mean distance between them and observed data is minimized. This is known as Vector Quantization [15].

Our approach uses a Vector Quantization algorithm known as Growing Neural Gas (GNG) [9] in order to estimate the number of discrete states of the model as well as the mean values and standard deviation of the observation probabilities. This algorithm has been chosen between many different approaches existent in the literature [15, 16, 17, 18] due to its following properties:

- It is fast. The costliest operation is of $O(N)$. This can be further optimized by using a hierarchical structure like an r-tree[19].
- The number of states is not fixed. New states are added and deleted as observations arrive.
- It is incremental. This makes it suitable to process continuous streams of observations.

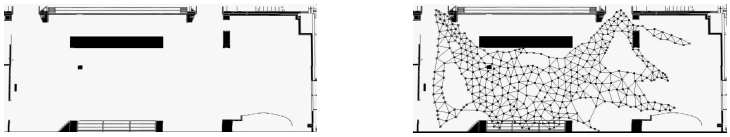
The algorithm processes observation on a one by one basis. It produces a graph, where nodes representing discrete states are explicitly linked to their closest neighbors (the graph is a subset of the Delaunay triangulation). Every node i is associated to a vector μ_i known as the centroid.

The application of this structure to estimate the required parameters is straightforward: state information $\{x_t, y_t\}$ contained in each observation is used as an input for a GNG. The resulting set of nodes represents discrete states whose centroids are the mean values of the observation probabilities. The standard deviation σ_i for state i is approximated by averaging the half length of the links emanating from the corresponding node

Insertion of new states is no longer allowed when $\arg\min \sigma_i$ is less than a given threshold. This restrains the algorithm from discretizing the space below the sensor's precision. Thresholding may be regarded as defining a minimum cell size, which is similar to conventional grid approaches but with two important advantages: a) the location of the cells is not fixed *a priori* and b) only relevant cells are represented. The latter advantage depends on the ratio between the existing positions (*ie* the full grid) those which are actually visited by objects. Usually, this advantage becomes more important as the state dimension grows due to motion constraints which apply to the object (*e.g.* acceleration and speed limits, unaccessible areas, etc.).

An example of the use of GNG is presented in fig. 1. The environment is a simulator of the laboratory's entry hall. It contains a number of places which may constitute motion goals (*e.g.* the stairways in the bottom or the

two doors in the top of figure 1a). Fig. 1b presents the state of the GNG structure after processing 1000 trajectories.



a) Sample Environment b) GNG Network

Fig. 1. Using GNG to represent discrete states in the laboratory entry hall.

4.2 Identifying Goals

The problem of automatically identifying the goals that an object seeks to reach using only observation data is very difficult since these goals are often related to information which is not present in this data (*e.g.* the presence of a billboard).

The approach taken here aims to identify goals based on a simple hypothesis: when an object stops moving (or exits the environment) it is because it has reached its goal. This leads to a simple goal identification scheme: every observation o_t having $\eta_t = 1$ (end observation) is sent to a GNG structure which clusters this information together into high-level goals.

The nodes of the resulting GNG graph corresponds to goals[5]. The graph itself may be used to identify the goal that corresponds to a given end-state observation:

$$\gamma = \min \arg_i \|(x_t, y_t) - \mu_i^g\|, \text{for } \eta_t = 1 \tag{3}$$

4.3 Learning transition probabilities

Transition probabilities are updated once a complete trajectory is available, this means that all non-end observations are stored until an observation having $\eta = 1$ is received, then, expression 3 is used to compute the attained goal g. For every observation in the trajectory o_t, the Viterbi algorithm is used in order to find q_t given the past state $q_{t-1} = i$ (which has been estimated in the previous iteration)[6]:

[5] notations μ_i^s and μ_i^g will be used henceforth in order to distinguish between state and goal GNG's

[6] This implies iterating through the domain of q_t, meaning that the update step has cost $O(N)$.

$$q_t = \max\arg_j \left\{ p([q_t = j] \mid [\gamma = g], [q_{t-1} = i]) p(o_t \mid [q_t = j]) \right\}$$

The obtained values for g, i and j are then used as indices to update a transition count matrix A on a maximum-likelihood criterion:

$$A[g, i, j] \leftarrow A[g, i, j] + 1 \tag{4}$$

If the observation correspond to the first step of a trajectory only the current state is estimated using:

$$q_0 = \max\arg_i \{ p(o_0 \mid [q_0 = i]) \} \tag{5}$$

Transition probabilities are calculated using:

$$p([q_t = j] \mid [\gamma = g], [q_{t-1} = i]) = \frac{\mathcal{A}[g, i, j]}{\sum_h \mathcal{A}[g, i, h]} \tag{6}$$

Finally, when N or G change due to additions or deletion on the corresponding GNG structures the corresponding columns and rows are simply inserted or deleted accordingly, this is possible due to the fact that we are storing counts instead of probabilities in the transition matrix.

5 Experimental Results

In order to validate it, the proposed approach has been applied to the prediction of pedestrian motion in the entry hall of the Inria laboratory, which is a rectangular area of approximately 8 x 20 meters. As it may be seen in fig.2, the environment consists mostly of an open area without much structure.

We have performed experiments with both real and simulated data. Real data has been gathered through a vision system which tracks people using a single camera having wide-angle lenses. The system projects observations from the camera plan to the floor plan. It is worth noting that, due to the projection process, the system ends up covering only about 60% of the total area.

Simulated data consists of noisy trajectories between predefined sequences of control points.

5.1 Evaluation Criterion

In order to perform prediction, the probability $p(\gamma \mid o_0, \cdots, o_t)$ has been estimated using a particle filter with a resampling step [20] to integrate new observations.

The performance of the algorithm has been evaluated by measuring the difference between the predicted and effective final destination. The first n

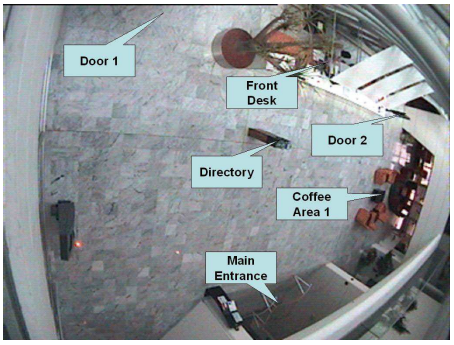


Fig. 2. The INRIA entry hall

observations of a trajectory were used to predict the most probable goal $g = \max \arg_i p([\gamma = i] \mid o_0, \cdots, o_n)$. The global estimation error is calculated as the the average of all the distance between the goal such obtained and the real end of the trajectory.

5.2 Results

We have run our experiments using datasets of 600 trajectories both for real and simulated data. The algorithm was initialized by processing 500 trajectories before starting to record the results. The the remaining 100 trajectories were processed as follows: for every trajectory, the predicted goal is estimated using 10% of its length, then 20% and so on until 90%. This allows us to measure how new knowledge improves prediction.

The results obtained for both simulated and real data are presented in fig. 3.

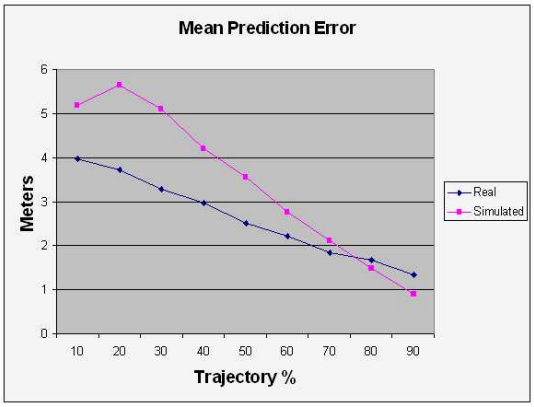


Fig. 3. Experimental results

It may be seen that both error curves decrease in near linear fashion with respect to the known fraction of the trajectories, we regard this as an encouraging result, particularly in the case of real data, which was very noisy due to the tracker's inability to adequately track the object's motion during all of its motion. However, we think that faster (non-linear) convergence rates are achievable, in particular by using a more efficient goal discovery mechanism.

It may be surprising to find that real data seems to perform better than simulated data. The reason is that the simulator produces trajectories which cover the entire entry hall, while, as we have explained above, real data is gathered only in a fraction of the environment.

It is important to mention that all the results presented here are preliminary. For example, the chosen performance measure should be improved to take into account situations where a trajectory passes through more than one goal, in this case, the system will probably predict that the object intends to reach these "intermediate" goals (which we think is fine), but, as they do not correspond to the trajectory's final position, the resulting prediction error will be high.

5.3 Real-time

In our experiments, prediction has been performed simultaneously with learning and graphic display for the test data set. Our unoptimized implementation of the technique runs on a 2 Mhz Athlon PC at a frequency of $20-60$ Hz. Even if we consider this as adequate for most real-time systems involving pedestrians, the system may be further optimized at the code level or by using a more efficient technique for searching the winner in the GNG structure, for example (§4.1).

6 Conclusions

In this paper, we proposed a method for learning motion plans from observations. Our approach represent plans as Hidden Markov Models. Learning consists of three modules: a) the Growing Neural Gas algorithm is used to estimate the total number of states N as well as the observation probability distribution; b) another GNG structure is used to estimate the number and positions of goals in the environment, and c) the Viterbi algorithm is used to perform a Maximum-Likelihood estimation of the transition probability.

The main contribution of this technique is that it follows 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. A second important contribution consists on the fact that, by learning the number of states and the number of goals, this technique is able to learn the structure of the model, this distinguishes our work from techniques with fix this values *a priori*.

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.

7 Future Work

The approach presented in this paper is a first approximation to the problem. In the short term, our goal is to test the approach in a different setting: the ParkView experimental platform, which is able to track a car moving in a parking lot (fig. 4).

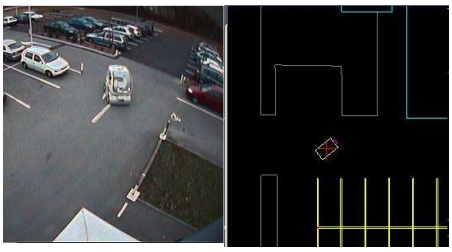


Fig. 4. The ParkView platform: *left*) camera view of the Cycab experimental car moving in the parking lot of the laboratory; *right*) the Cycab as detected on the tracking system.

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 [7].

References

1. D. Koller, J. Weber, T. Huang, J. Malik, G. Ogasawarea, B. Rao, and S. Russell, "Towards robust automatic traffic scene analysis in real-time," in *Proceedings of the 33rd Conference on Decision and Control*, Lake Buena Vista, FL (USA), December 1994, pp. 3776–3781.
2. K. Kyriakopoulos and G. Saridis, "An integrated collision prediction and avoidance scheme for mobile robots in non-stationary environments," in *Proceedings of the IEEE Int. Conf. on Robotics and Automation*, Nice, France, May 1992, pp. 194–199.
3. M. Bennewitz, W. Burgard, and S. Thrun, "Learning motion patterns of persons for mobile service robots," in *Proceedings of the IEEE Int. Conf. On Robotics and Automation*, Washington, USA, 2002, pp. 3601–3606.

4. S. Osentoski, V. Manfredi, and S. Mahadevan, "Learning hierarchical models of activity," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sendai, Japan, 2004.
5. E. Kruse, R. Gusche, and F. M. Wahl, "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*, Grenoble, France, 1997, pp. 713–717.
6. D. Vasquez and T. Fraichard, "Motion prediction for moving objects: a statistical approach," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, New Orleans, LA (US), apr 2004, pp. 3931–3936.
7. H. Bui, S. Venkatesh, and G. West, "Policy recognition in the abstract hidden markov models," *Journal of Artificial Intelligence Research*, vol. 17, pp. 451–499, 2002. [Online]. Available: citeseer.ist.psu.edu/bui02policy.html
8. L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Readings in speech recognition*, pp. 267–296, 1990.
9. B. Fritzke, "A growing neural gas network learns topologies," *Advances in Neural Information Processing Systems*, 1995.
10. L. Baum and T.Petrie, "Statistical inference for probabilistic functions of finite state markov chains," *Annals of Mathematical Statistics*, no. 37, 1966.
11. N. Dempster, A.and Laird, , and D. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society*, vol. 9, no. 1, pp. 1–38, 1977, series B.
12. Y. Singer and M. K. Warmuth, "Training algorithms for hidden markov models using entropy based distance functions." in *Advances in Neural Information Processing Systems 9, NIPS*. Denver, CO (USA) December 2-5, 1996: MIT Press, 1996, pp. 641–647.
13. R. M. Neal and G. E. Hinton, "A new view of the EM algorithm that justifies incremental, sparse and other variants," in *Learning in Graphical Models*, M. I. Jordan, Ed. Kluwer Academic Publishers, 1998, pp. 355–368.
14. A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Transactions on Information Theory*, vol. IT-13, no. 2, pp. 260–269, April 1967.
15. Y. Linde, A. Buzo, and R. Gray, "An algorithm for vector quantizer design," *IEEE Transactions on Communications*, vol. COM-28, pp. 84–95, 1980.
16. T. Kohonen, *Self-Organizing Maps*, ser. Springer Series in Information Sciences. Berlin, Heidelberg: Springer, 1995, vol. 30, (Second Extended Edition 1997).
17. M. Martinetz and K. J. Schulten, "A "neural-gas" network learns topologies," in *Proceedings of International Conference on Articial Neural Networks*, T. Kohonen, K. Mkisara, O. Simula, and e. J. Kangas, Eds., vol. I, North-Holland, Amsterdam, 1991, pp. 397–402.
18. G. Carpenter, S. Grossberg, and D. Rosen, "Fuzzy art: An adaptive resonance algorithm for rapid, stable classification of analog patterns," in *Proc. Int. Joint Conf. Neural Networks*, vol. II, Seattle, USA, 1991, pp. 411–420.
19. A. Guttman, "R-trees: A dynamic index structure for spatial searching." in *SIGMOD Conference*, 1984, pp. 47–57.
20. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for on-line non-linear/non-gaussian bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174–188, feb 2002. [Online]. Available: citeseer.ist.psu.edu/article/arulampalam01tutorial.html