Dynamic Bayesian Networks for semantic localization in robotics

Fernando Rubio*, M. Julia Flores*, Jesus Martinez-Gomez**, Ann E. Nicholson§

Abstract—In this paper we present a methodology that covers the following steps: (1) Image processing and discretization for creating feature-based scene descriptors. (2) Learning of static Bayesian networks and Naive Bayes classifier. (3) Learning of Dynamical Bayesian Networks. (4) Evaluation of the models. (5) Comparison. All this process has been tested in a real case: the KTH-IDOL2 (Image Database for rObot Local- ization) dataset for scene classification. Our experimental results show that BN models obtain good accuracy classification rates, using a small number of variables, and without applying extra knowledge in the discretization process (unsupervised).

Index Terms—Bayesian networks, Dynamic Bayesian networks, Robot localization, Machine learning, Visual Place Classification.

I. INTRODUCTION

In robotics, semantic localization refers to the task where the robot must report its location semantically with respect to objects or regions in the scene. For instance, SLAM (Simultaneous Localization And Mapping) is a popular technique to accurately localize the robot and simultaneously build a map of the environment [8]. However, in this paper we attempt to deal with the semantic localization instead of metric one. That is, we are interested in extracting semantic information (room categories) instead of locations (environment coordinates). Therefore, we tackle the semantic localization problem from the perspective of supervised learning, and we can denote it as visual place classification or categorization [37].

First works in this area were created between 2002 and 2006 [19], [30]. As a result of the interest of the research community for this problem, the Robot Vision challenge [24] was created in 2009. This challenge addresses the place classification problem, and since its origin, participants have proposed some interesting solutions [18].

This work explores the behavior of Bayesian Networks (BNs) in this classification task to determine the location where a robot is, with respect to the image this robot perceives. Bayesian networks can be understood as the key computer technology for dealing with probabilities in Artificial Intelligence [15]. Basically, they allow working with real problems, because BNs can naturally deal with the uncertainty of the real world, since its basis are on the Probability Theory and Bayes’ Theorem. In the late 80s efficient algorithms for inference (architectures) were designed, and during the last 15 years, many efforts have been done for learning Bayesian networks both from data, in collaboration with the expert (knowledge engineering), and even with combined methods [13]. Besides, it is nowadays really popular in many areas of application, which proves that they provide high performance for modeling, diagnosis, control and planning, industrial manufacture, natural resources, etc...

Besides, we have chosen this paradigm...
because it can easily be used for classification, and we can find many Bayesian networks classifiers in literature that have been successfully applied in real world [10]. Traditional Bayesian networks are static, but there is also the possibility to work with time-series domains with the use of dynamical Bayesian networks [20]. Since the data we are going to use in this paper deals with temporal information, we found very appealing to explore the distinct tools BNs formalism provides us.

In this article, we evaluated different proposals using as benchmark the KTH-IDOL2 dataset [17], which was the one proposed in the first edition of the Robot Vision task. This dataset was selected because, in addition to the semantic category, images are annotated with metric information. That would allow to evaluate our proposal for solving the metric localization problem, what sounds interesting for future research.

The rest of the article is organized as follows. Section II gives an overall introduction to Bayesian artificial intelligence, including the use of static and dynamic models. In Section II-C, the CAMML learning procedure is introduced, while the robot localization problem is stated in Section III. The dataset used for the experiments is described in Section IV, and the experimentation stage is depicted in Section V. Finally, conclusions are drawn in Section VI.

II. PRELIMINARIES: BAYESIAN ARTIFICIAL INTELLIGENCE

A. Bayesian Networks

Bayesian networks (BNs) are an increasingly popular paradigm for representing problems. A Bayesian network [12], [15] is a directed acyclic graph (DAG) whose nodes represent the random variables in the problem. A set of directed edges connect pairs of vertices, representing the direct dependencies (which are often causal connections) between variables. The set of nodes pointing to $X$ are called its parents, and is denoted $pa(X)$. The relationship between variables is quantified by conditional probabilities, usually tables (CPTs), which are associated with each node, namely $P(X|pa(X))$. The CPTs together compactly represent the full joint distribution.

The capability of a BN to express relationships, dependencies and independencies by its associated graph lies on its qualitative side. But these relationships are also modeled with a second element, quantitative, that forms a BN: probability distributions, as already introduced.

This probabilistic information will be processed by Bayes' Theorem but taking advantage of the inherent partition of the variables’ families that BNs provide. BNs are primarily used for inference: evidence $e$ in the form of statements on the state of some variables is used to infer the posterior probabilities $P(X|e)$ for all remaining variables. Users can set the values of any combination of nodes in the network that they have observed. This evidence, $e$, propagates through the network, producing a new posterior probability distribution $P(X|e)$ for each variable in the network. There are a number of efficient exact and approximate inference algorithms for performing this probabilistic updating, providing a powerful combination of predictive, diagnostic and explanatory reasoning.

BNs can be used for classification or supervised learning. Supervised learning algorithms are trained on labeled examples, i.e., input where the desired output is known, so that the learned model can be used to classify new cases whose label is unknown, based on the rest of the attributes (also called predictive variables), so that we can generate a probable output for previously unseen inputs. As long as a BN present a Class variable, it can be used for classification. However, due to the complexity of learning algorithms, some simple classifiers, with a fixed structure can be used (less time for
structural learning), and where the CPTs size and complexity are under control. One of the most known and used BNs classifiers is Naive Bayes [9]. It makes the assumption that attributes are conditionally independent given the class, since the structure is that every attribute has as a unique single parent: the class variable. This assumption does not apply in many real problems, but this simplification has shown to work really well in many practical domains such as Computing (spam filters, for example), Medicine (diagnostic purposes), Economy (credit risk model) or Environmental Sciences (fish classification) [10].

B. Dynamic Bayesian Networks

Dynamic Bayesian networks (DBNs) are a long-established extension [15] to ordinary BNs that allow explicit modeling of changes over time (e.g. [14], [21], [23]). They have been used in a range of applications such as robot navigation and map learning [7], monitoring robot vehicles [22] and traffic monitoring in both [26] and the BATmobile project for monitoring an automated vehicle travelling on a freeway [11].

The general structure of a DBN is shown in Figure 1 (taken from [15]). In a DBN, for each domain variable \( X_i \), there is one node for each time step of interest \( X_i^T, X_i^{T+1}, X_i^{T+2} \), etc. Each time step is called a time-slice. The relationships between variables at successive time steps are represented by so-called temporal arcs, including relationships between (i) the same variable over time, \( X_i^T \rightarrow X_i^{T+1} \), and (ii) different variables over time, \( X_i^T \rightarrow X_j^{T+1} \).

Note that in the generic structure there are no arcs that span more than a single time step; this reflects the so-called Markov assumption that the state of the world at a particular time depends only on the previous state and any action taken in it. Given the typical restriction that both the structure and the CPTs are unchanging, a DBN can be specified very compactly.\(^1\)

C. CaMML learning

Bayesian networks provide an attractive framework for prediction, classification, diagnosis, tracking, descriptive purposes, etc... But the first important step is to get the appropriate model that represents the particular problem we deal with. This is not a trivial issue. One of the possibilities is to use expert elicitation, which in many cases is not possible and may need a complex and long process of knowledge engineering. Another possibility is to use an algorithm able to learn the model from data, that is Machine Learning techniques. In the last 15 years, learning of Bayesian networks has become one of the hot topics in the area [12], [15].

1) Learning Bayesian Networks: Assume our problem has \( n \) variables \( X_i \in V \), and a dataset to learn from, as a log file, formatted as a table. Each row in this data set is called a case and represents a record, concerning the particular value for every variable. Missing values can be treated by the BN learner or imputed by specific techniques. We have them available information to learn the model, and to use it later for future cases to predict, classify, etc...

Algorithms for learning BNs are to provide techniques for learning the DAG structure and also mechanisms for estimating the parameters of the complete network structure from data. There is one key limitation when

\(^1\)Some BN software packages (e.g. Hugin, GeNIe, Netica) provide a facility to specify a DBN compactly.
learning BNs from observational data only – there is usually no unique BN that represents the joint distribution. More formally, two BNs in the same statistical equivalence class (SEC) [3] can be parametrized to give an identical joint probability distribution. There is no way to distinguish between the two using only observational data (although they may be distinguished given experimental data). That is why many algorithms based on search techniques use the SEC space, which obviously, is also smaller and the search will be more efficient.

BN structural learning algorithms can be classified into constraint-based and metric-based. Constraint-based methods (e.g., PC [29], RAI [38]) use information about conditional independencies gained by performing statistical significance tests on the data. Metric-based methods (e.g., K2 [5], CaMML [36]) search for a BN to minimize or maximize a metric; many different metrics have been used, (e.g. K2 uses the BDe metric, CaMML uses an MML metric; see [15, Ch 9]). Metric-based BN structural learners also vary in the search method used and in what is returned from the search; some learners (e.g., K2) return a DAG, others (e.g., GES [4]) learn only the SEC.

2) CaMML: a tool for learning BNs: CaMML\(^2\) attempts to learn the best causal structure to account for the data, using a minimum message length (MML) metric with a two-phase search, simulated annealing followed by Markov Chain Monte Carlo (MCMC) search, over the model space. Both MML [34], [35] and the better known MDL [27] are inspired by information theory, and make a trade-off between prior probability (model complexity) and goodness of fit. With both, the problem becomes one of encoding both the model and the data, and the best model is then one that minimizes the message length for that encoding.

The full details of MML encoding are not required for this paper, but we can write the relationship between the message length, the model and the data given the model as:

\[
\text{msgLen} \propto - \log(P(\text{Model})) - \log(P(\text{Data}|\text{Model})).
\]

The CaMML metric is a combination of the message of the MML encoding of the BN, incorporating three parts: (1) the network structure, (2) the parameters given this structure, and (3) the data given the network structure and these parameters.

In contrast to other metric learners that use a uniform prior over DAGs or SECs for their search, CaMML uses a uniform prior over Totally Ordered Models (TOMs). A TOM is a DAG specified at a somewhat deeper level; it can be thought of as a DAG together with a total ordering of its variables. Just as an SEC is a set of DAGs, a DAG is a set of TOMs.

CaMML also differs from other learners in using Metropolis sampling to estimate a distribution over the model space. CaMML builds a hierarchy of models. It samples TOM space moving from TOM to TOM with probabilistic pressure applied by the MML metric. Every time a TOM is visited a visit to that TOM’s DAG and SEC is recorded. CaMML also records a visit to the DAG’s “clean” representative — that is the DAG with all spurious arcs removed — and to that clean DAG’s SEC. SECs are also joined in a process analogous to cleaning.

3) Learning Dynamical Bayesian Networks: Learning Dynamical Bayesian networks is also possible [20], even though, if the structure gets more complex, the same happens with the learning algorithms and their possibilities, which increase also enormously. Nowadays, there is a small and growing literature on the subject (e.g., [1], [28], [31]). However, this is an issue still under development, there are many algorithms for specific cases, but the most known tools do not provide algorithms to learn DBNs specifically.

Recently, an extension of CaMML to learn

\[^2\text{This software is downloadable from https://github.com/rodneyodonnell/CaMML/ [Last accessed on 27th March 2014].}\]
DBNs has been included. Main related issues can be found in the presentation by Black et. al [2]. Basically, this extension will divide the process of learning a DBN in these three steps:

1) Learn an order of variables within each time slice, assuming this order is identical in all time slices.

2) Learn the intraslice arcs for the given variable order. In Fig. 1 we see those arcs refers to every time-slice \( t = i \), those connecting \( X^t = i \), that is in the dotted framebox. DBNs assume they are the same for every slice.

3) Learn the temporal (or interslice) arcs. In Fig. 1 we see those arcs refers to arcs between slices \( t = i \) (previous) and \( t = i + 1 \) (next), those connecting variables of the previous instant to the following one.

If static CaMML used the space of TOMs for searching the graphical structure for a BN, for DBNs authors have defined DTOMs (Dynamic TOMS), which is essentially a TOM plus an \( N \times N \) binary matrix specifying the presence/absence of arcs between each of the \( N \) nodes in the first time slice and the \( N \) nodes in the second time slice. Thus, it is required a structure metric to specify the code length for encoding a DTOM (Dynamic TOM). The first time slice parameters are learned from the data directly (given the intraslice network structure learned during the search). The Metropolis search has then been adapted to the dynamical environment, in learning a DBN, the original CaMML mutation operations remain, however, they are used for modifying the variable order and the intraslice arcs of a DBN only. As such, additional mutation types were required for modifying the intraslice arcs: temporal arc change, double temporal arc change and cross arc change.

III. MOBILE ROBOT LOCALIZATION

The generic problem of localization consists in answering the question Where am I? [16]. That is, a robot has to estimate its location within the environment given an specific environment representation. The localization problem can be tackled from two points of view: metric localization and semantic localization. The first type is related to the estimation of \( <x, y, \theta> \) locations and assumes the use of a map as environment representation. On the other hand, semantic localization describes the environment using semantic labels instead of coordinates. There are several types of labels that can be used to define the environment, but scene categories are one of the most common. Using this approach, semantic terms as "corridor" or "kitchen" are directly associated to environment locations.

In this work, we assume indoor environments and visual images as only sensor information. Therefore, the semantic localization problem can be seen as a indoor scene classification problem. We opted for the RobotVision@ImageCLEF 2009 challenge [25] as benchmark for our study since metric and semantic localization problems are managed together. That allows us to test our proposal for solving both problems using the same benchmark. In order to perform a deeper study, we used the original dataset from the RobotVision@ImageCLEF sequences were extracted: IDOL2.

IV. DATASET

The KTH-IDOL2 dataset [17] consists of 24 sequences of images acquired using two mobile robot platforms. These sequences were acquired within an indoor environment consisting of five rooms of different functionality (one-person office, two-persons office, corridor, kitchen, and printer area) under three illumination conditions (in cloudy weather, in sunny weather, and at night) across a span of 6 months. Figure 2 shows the map of the environment used for acquisition and the semantic categories.
Fig. 2. Map of the IDOL2 environment.

Each image is labeled with the $<x, y, \theta>$ position from it was acquired from but also the semantic label of the scene. These two sources of information are used in this article to extract from each image topological and semantic information. Some sample images for each one of the five semantic categories are shown in Fig. 3.

Fig. 3. Semantic labels used in the IDOL2 dataset.

V. EXPERIMENTATION

In this work, we evaluate the use of DBNs for mobile robot localization. Concretely, we assume the use of CaMML Bayesian learning and indoor environments. In order to perform such evaluation, there are several steps to be carried out. Firstly, source images have to be processed to extract visual features from them. Then, image descriptors must be created from these features. After generating the image descriptors, a discretization step has to be performed to allow this information to be used as input for the learning algorithm. Finally, several BN classifiers can be generated and tested.

A. Feature extraction and descriptor generation

Feature extraction transforms the input images into a set of characteristics named features. Features are expected to extract relevant information to perform a desired task. Due to global features perform better than local ones for scene recognition [33], we selected a feature from this type. In this evaluation, we used a global feature based on the use of a histogram build from gradient information. This feature is named HOG (Histogram of Gradients [6]) and extracts a histogram that can be directly used as image descriptor.

HOG-based image descriptors encode image information using continuous data. That is, we select an initial set of key-points in the image (using edge detection), and for each one of these key-points, we compute the angle of highest invariance. Taking into account that angles range from 0 to 360, we obtain a 360-sized histogram whose entries represent the frequency of points in the image that present their highest invariance at gradient $g_i$. Therefore, we start from 360 variables encoding continuous data (frequencies).

B. Variable reduction and discretization

In order to use the HOG descriptors as input data in a CaMML learning procedure, it is necessary to perform a preliminary discretization step. In addition to convert continuous values to discrete ones, we opted
to reduce the large number of initial variables. That is, the original 360 variables are reduced to $n$ by merging them. This is done by using wider sets in the HOG histogram construction. Given that histogram variables represent frequencies, reducing their number has an important effect over their values. This reduction helps to create manageable input data for posterior learning stages.

Regarding the discretization, continuous values are translated into 4 different categories. These categories represent low frequency, low-medium frequency, medium-high frequency and high frequency. The limits for these categories will depend on $n$ (the number of variables) using the next rules:

- Low category: $[0, 1/2n)$
- Medium/Low category: $[1/2n, 1/n)$
- Medium/High category: $[1/n, 2/n)$
- High category: $[2/n, 1]

There are 5 different $n$ values that are evaluated in this article: 5, 10, 20, 50, and 100. The $n$ selection will affect the generalization/specialization for the HOG descriptors. The use of large $n$ values involves the generation of specific descriptor that can cause over-fitting when training the classifier. On the other hand, small $n$ generate too general models incapable to differentiate between classes.

Fig. 4 shows a sample discretization for an input image and four $n$ values. It can be observed how $n=100$ obtains a fine-grain representation of the original histogram, while $n=5$ simplifies it in excess. Regarding the class labels, red color is used to represent bins (angle ranges) with low frequency. Orange and yellow color represent medium/low and medium/high frequency, respectively. Finally, blue color denotes high frequency. The four processed histograms are shown using the same scale. That allows to point out how similar values are associated to different labels. Concretely, it can be seen that last bin presents similar values for $n=20$ (0.16) and $n=10$ (0.18). However, we obtain label high for $n=20$ and medium/high for $n=10$. That occurs because high label is associated to values greater than $(2/n)$, that is, 0.1 and 0.2 with 20 and 10 values respectively.

C. Classification: $n$ value selection

We used the first 12 sequences of the KTH-IDOL2 dataset (those acquired with Dumbo robot, shown in Fig. 5) as benchmark for all our evaluations. Therefore, we have four sequences acquired under cloudy conditions (cloudy1-cloudy4), four with night conditions (night1-night4), and another four acquired under sunny lighting conditions(sunny1-sunny4). The first two first sequences (for each lighting conditions) are used for training (e.g. sunny1 and sunny2), the third is used for validation (e.g. sunny3), and the fourth one for test (e.g. sunny4). We used in all our tests the accuracy on the room category classification as evolution measure. Let us remember that KTH-IDOL2 images are labeled with the semantic category of the room where such image was acquired: one-person office, two-persons office, corridor, kitchen or printer area.

The first set of experiments has as objective the selection of the most appropriate value for $n$. In order to achieve this goal, we evaluated five different values using the following procedure. Firstly, we trained both a Naive Bayes classifier and a Bayesian Network (CaMML learning procedure) using as input the two training sequences. This process was repeated for each one of the lighting conditions. Finally, we evaluated the obtained models using the three proposed validation sequences (night3, sunny3 and cloudy3). Therefore, we generated 30 differ-

---

**Fig. 5. Original sequences distribution in the KTH-IDOL2 dataset**
ent classifiers and obtained 90 classification rates.

The obtained results for the CaMML Bayesian Network and the Naive Bayes Classifier are shown in Tables I and II respectively. Due to memory problems, we could not generate the CaMML Bayesian Network with \(n=100\) and the Sunny training sequence. That happened because Sunny sequences are notoriously larger than the others.

In order to graphically present how the value of \(n\) affects to the classification rate, we computed the mean accuracy for all its values. We performed this step for the CaMML Bayesian network, the Naive Bayes classifier, and both combined. These results are presented in Fig. 6, where we can observe how the accuracy starts decreasing when using \(n\) values higher than 50. However, the accuracy increase from \(n=20\) to \(n=50\) is not as significant as the increase for the descriptor size (more than two times greater). Therefore, we decided to select 10 and 20 as optimal \(n\) values, and use them for the following experimentation stage.

When the number of variables remain low (5 and 10), BNs seem to behave slightly better than NB with respect to the accuracy values. This tendency switches as long as the number of variables increases, being more evident when \(n = 100\). It could be surprising, since one would expect that a more complex model, able of catching any kind of (in)dependencies between variables, should give better results always, and probably more clearly with more variables. However, after analyzing the problem, this result seems reasonable. When learning a BN model, we need to learn the structure, and then the parameters (Conditional Probability Tables, named CPTs). For every variable, \(X_i, P(X|\text{pa}(X))\) has to be estimated. These estimations are normally based on statistical
information, taken from the training data. When every variable \((X_i \in \{X_1, \ldots, X_n\})\) has a single parent, as in Naive Bayes, the complexity of the table is \(|\Omega_{X_i}| \times |\Omega_{\text{Class}}|\), where \(\Omega_V\) is the set of all possible states and \(||\) operator denotes the number of elements in this set. So, in our case this CPT we’ll have \(4 \times 5 = 20\) entries. This makes easier to compute the CPTs based on estimations. On the contrary, when the BN has no structural restrictions, we can for example find that a variable \(X_j\) has a set of 5 parents (one of them can be the class or not). This would produce a table with \(4^5 \times 5 = 1280\) entries. This makes that the estimations will be less reliable (because we don’t have unlimited input data for learning), and it could happen that we find some combinations not found in data. For future work, we may study other BN-classifiers called Semi-Naive, which have a simple structure but allow certain relationships among attributes [10].

D. Classification: dynamic models

In this subsection we study how to integrate the temporal continuity of images sequences into the classifiers. Concretely, we compared the results when using dynamic Bayesian classifiers instead of static ones.

For this purpose, we generated a single training sequence by merging all the previous ones (sunny1, sunny2, cloudy1, cloudy2, night1, and night2). Moreover, we also created just one test sequence using all the proposed ones (sunny4, cloudy4, and night4). This process is shown in Figure 7.

The obtained static classifiers were then compared with the generation of dynamic ones. Our first trial consisted in using the CaMML learning procedure to directly generate a dynamic Bayesian Network. The obtained results were terrible, due to the dynamic network classified all the images with

![Training Sequence and Test Sequence](image-url)
the same class. That is, if the first image was classified as Corridor, all complete training sequence was classified as Corridor. This happened because there are less than 1% of room transitions in the training sequences. Therefore, if it is assumed that the previous class is available, the use of this class is expected to obtain higher than 99% classification rates.

In view of direct learning obtains dynamic networks that are not capable of detecting room changes, we proposed two transition models. The first one assumed a medium probability (40%) of remaining in the same room for two consecutive frames, while the second one used a more conservative approach. The class transitions for both approaches are shown in Table IV. Method shown in [32] is the basis for the combination of parents CPTs into one table.

The obtained results are shown in Table V, and the difference with respect to the static models is graphically presented Fig. 8. We can see how dynamical BNs outperform static BNs (which is logical since the problem had a dynamical component), but this tendency is not clearly exposed for the Naive Bayes model (see Fig. 8). These results show us how CaMML learning provides more realistic representations of the problem to solve, due to the fact that it uses less restrictive learning models. Another interesting remark is the comparison between Model A and B, being the second more conservative than the first, but with enough weight for the transitions to make the model work properly. Remember that using a too conservative distribution would provide poorly models as we explained above. It seems that Naive Bayes is less sensitive to this change of model than BN, because of its inherent simplicity. This has sense since the CPTs involved in a NB are much more controlled, with the same complexity, while in a BN the changes in the class CPT can affect more variables.
Bayes classifiers obtain too specific models (overfitting), in contrast to the capacity of generalization proved by Bayesian Networks. Furthermore, Bayesian Networks showed as more appropriate than Naive Bayes for adding them the dynamic behavior. Regarding the discretization, we have exposed the trade-off between generalization (small \( n \) values) and specialization (large \( n \) values). As can be drawn from results, intermediate \( n \) values (as 10 or 20) resulted in satisfactory models capable of classifying scenes in real-time.

VI. CONCLUSIONS AND FUTURE WORK

We have presented a procedure for using Bayesian classifiers to solve the problem of semantic localization. This procedure includes an unsupervised discretization step optimal for coping with histograms as input, because most of the visual features extracted from images present such structure. The article included novel Bayesian learning techniques as CaMML, and also the use of dynamic Bayesian networks to cope with the temporal continuity of the dataset sequences.

Based on the experiments, we can conclude that we can achieve quite reasonable results by means of our proposal. These results were achieved using a reduced number of predicting variables (10 and 20), which makes the classifier capable of working in real time. Moreover, the dynamic Bayesian networks showed as an appropriate solution for integrating sequences of images.

This work can be extended in many different ways. We have in mind to include new visual features, as SIFT-dense descriptors. This inclusion would allow to evaluate how Bayesian approaches cope with multiple data fusion. We also would like to evaluate our proposal using topological information as input instead of semantic one. This could be done using the topological annotations that are included in the KTH-IDOL 2 dataset. A comparison between our approach and the use of Support Vector Machines is also considered.

ACKNOWLEDGMENT
This work has been partially funded by FEDER funds and the Spanish Government (MICINN) through project TIN2010-20900-C04-03, and by the Interconecta Programme 2011 project ITC-20111030 ADAPTA.

REFERENCES


