Book Title
Book Editors
IOS Press, 2003

Simple Form Recognition Using Bayesian Programming

Guy Ramel [a,1], Adriana Tapus [a], François Aspert [a] and Roland Siegwart [a]
[a] *Autonomous Systems Lab*
Swiss Federal Institute of Technology
Ecublens, Lausanne 1015, Switzerland

Abstract. The environment that surrounds us is very complex. Understanding and interpreting it is a very hard task. This paper proposes an approach allowing simple form recognition with a camera by using a probabilistic approach called Bayesian Programming. The main goal is to recognize several type of elemental features composing an image, such as local orientation of a contour, or corners. The Bayesian Program for feature recognition is presented and the learning stage explained. One approach has been validated through experiments.

Keywords. Computer Vision, Pattern recognition, Bayesian Programming, Humans Robots Interaction

1. Introduction

Nowadays, the ability to detect some particular shapes in an image is a crucial issue that take more and more importance. Indeed, in two-dimensional images, most of the time, important information about objects can be extracted from the particular shape of objects. For example, we can state that a door is composed of vertical lines, horizontal lines and four corners. This information can be considered as sufficient to recognize a door. This type of recognition can be taken as a basis for multiple industrial or security image processing applications (e. g. video surveillance, quality control). Therefore, there is a real need in detecting these features and in finding a way to classify them in a robust manner in order to be able to build stronger applications from this basic model.

Biological vision is an example of hierarchical based system: the part of the neocortex that treats the vision is composed of several layers named for example V1, V2, V4 and IT (enumeration non exhaustive). V1 is directly activated by the optic nerve and contains a population of neurons specialised in recognition of elementary features (primitives) such as lines with a particular orientation, corners or end of lines for example. Each layer is focused on more complex combinations of primitives. Finally, IT contains a population of neurons activated if specific objects are present in the field of vision (e.g. human face), and invariant to translation or rotation [2,3]. Even if this description is uncomplete, we show a brief overview of a cognitive vision system.

[1]Correspondence to: Guy Ramel, EPFL - STI - I2S Ű LSA, ME A3 434 (Bâtiment ME), Station 9, CH-1015 Lausanne. Tel.: +41 21 693 54 65; Fax: +41 21 693 78 07; E-mail: guy.ramel@epfl.ch.

As for human object recognition, top-down knowledge of objects from object recognition may be used to categorise objects in a real scene from primitive features. Some previous works use codebook of local appearances of a particular object category to recognise this one in real-world scenes [5]. Other approach described in [6] analyzes the appearance and the contour shape to classify objects. In [8], authors use Bayesian networks to describe a class of objects from primitive features given by the two first-derivative of the Gaussian basis function and by the 18-vector containing the responses to the basis filters of the first three derivatives at two scales.

All these considerations justify the development of a robust tool for recognition of simple forms, using some minimal information about the image (i.e. no a priori information can be taken). Furthermore, the user should be able to easily describe and modify the forms to recognize in the image. The approach should also be able to deal with the uncertainty and the possible range of variations that exists for each feature we want to detect and recognize. Another strong constraint is the computational efficiency.

Given all these constraints, we propose a new approach based on the Bayesian Programming formalism so as to recognise simple forms in a robust manner. This method, first described by Lebeltel in [4], is designed for robot programming using conditional probabilities. It addresses several interesting properties that can be applied to our form's recognition problem. One of the main strength is the fact that it is based on a supervised learning which can potentially enable the definition of any kind of form for recognition and the possibility of their modification in a very flexible way. Secondly, using the conditional probabilities allow some variations in the features to recognise. That permit a detection of form dealing with noise and other undesirable possible variations.

The rest of the paper is structured as follows. Section 2 briefly defines the Bayesian Programming formalism. Section 3 is dedicated to the probabilistic method used for the simple forms recognition. Experimental results are presented in Section 4. Finally, Section 5 draws conclusions and discusses further work.

2. Bayesian Programming

Probabilistic methodologies and techniques offer possible solutions to the incompleteness and uncertainty problems when programming a robot. The basic programming resources are probability distributions. In the context of probabilistic method, the Bayesian Programming (BP) approach was originally proposed as a tool for robotic programming (see [4]), but nowadays used in a wider scope of applications ([7,9] shows some examples).

The Bayesian Programming formalism allows for using a unique notation and provides a structure to describe probabilistic knowledge and its use. The elements of a Bayesian Program are illustrated in Figure 1. A BP is divided in two parts: a description and a question.

2.1. Description

The purpose of a description is to specify an effective method to compute a joint distribution on a set of relevant variables $X_1, X_2, \ldots, X_n$, given a set of experimental data δ and a priori knowledge π. In the specification phase of the description, it is necessary to:

$$\text{Program}\begin{cases}\text{Description}\begin{cases}\text{Specification}(\pi)\begin{cases}\text{Pertinent Variables}\\ \text{Decomposition}\\ \text{Form}\begin{cases}\text{Parametric Forms}\\ \text{Program}\end{cases}\end{cases}\\ \text{Identification based on data}(\delta)\end{cases}\\ \text{Question}\end{cases}$$

Figure 1. *Structure of a Bayesian program.*

- Define a set of relevant variables $X_1, X_2, \ldots, X_n$, on which the joint distribution shall be defined
- Decompose the joint distribution into simpler terms, using the conjunction rule. The conditional independence rule can allow further simplification, and such a simplified decomposition of the joint distribution is called decomposition
- Define the forms for each term in the decomposition; i.e. each term is associated with either a parametric form, as a function, or to another Bayesian Program

2.2. Question

Given a description $\mathrm{P}(X1 \wedge X2 \wedge \ \wedge Xn \mid \delta \wedge \pi)$, a question is obtained by partitioning the variables $X_1, X_2, \ldots, X_n$ into three sets: Searched, Known and Unknown variables. A question is defined as the distribution $\mathrm{P}(Searched \mid Known \wedge \delta \wedge \pi)$. In order to answer this question, the following general inference is used:

$$\mathrm{P}(Searched \mid Known \wedge \delta \wedge \pi) = \frac{\sum_{Unknown} \mathrm{P}(Searched \wedge Unknown \wedge Known)}{\sum_{Unknown,Searched} \mathrm{P}(Searched \wedge Unknown \wedge Known)} \quad (1)$$

3. Form Recognition with Bayesian Programming

3.1. Primitives

In this work, we focus only on primitives features of low level. These ones are depicted in Figure 2. The primitives used are : vertical, horizontal, slash and backslash lines and four corners of different orientations. Several translated primitives of each type are added and used as Garbage Collector. Usage of garbage collector will be explained in the section 4.

3.2. Pertinent Variables

The choice of the pertinent variables is a crucial point in defining a Bayesian program since it will be the backbone of the program and the quality of our detection will highly depend on it. These variables must not only be relevant for describing the features in terms of attributes and characteristics, but also, not be too numerous in order to keep the decision calculation time reasonable.

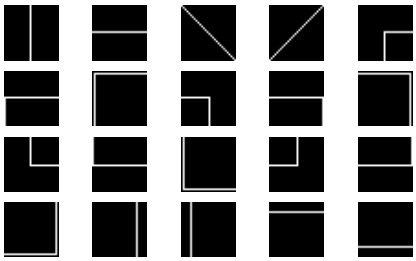

Figure 2. *Set of primitives with "real" primitives (vertical, horizontal, slash and backslash line and the four corners) and several translated primitives of each type used as Garbage Collector.*

Only one output variable is needed in our case. This is used to determine the type of the feature that have been recognized. This variable that is denoted by $Feat$. It is discrete and takes integer values over the range $[0 \dots N]$, where N corresponds to the number of different features that are searched in the image.

In our case, all low-level features are recognised inside a window, which one scans the entire image. As shown in the figure 3, this window is divided into several square zones. In each of these zones, the ratio between the number of white pixels (contour pixels) called $whitepix$ and the total number of pixels within a zone (called $totpix$) is employed for primitives detection. For each zone, this ratio is represented by a variable called X_i (i standing for a particular zone of a given feature) and is discrete over the interval $I_X = [0, 1]$. This can be expressed as :

$$X_i = \frac{whitepix}{totpix} \tag{2}$$

These variables corresponds to the $Known$ variable subset in our questioning inference . Therefore, the following joint distribution stands for our description of the problem:

$$P(Feat \otimes X_1 \wedge \ldots \wedge X_i \mid \delta \wedge \pi) \tag{3}$$

3.3. Decomposition

For this purpose, independence hypotheses need to be done in order to simplify the join decomposition. One can express this statement in the following way:
$P(X_1 \mid Feat \wedge \delta \wedge \pi) \perp \ldots \perp P(X_i \mid Feat \wedge \delta \wedge \pi)$.

Under this hypothesis, using the product and marginalization rule, the decomposition becomes:

$$\begin{aligned} P(Feat \wedge X_1 \wedge \ldots \wedge X_i \mid \delta \wedge \pi) = \\ P(Feat \mid \delta \wedge \pi) \times P(X_1 \mid Feat \wedge \delta \wedge \pi) \times \ldots \\ \ldots \times P(X_i \mid Feat \wedge \delta \wedge \pi) \end{aligned} \tag{4}$$

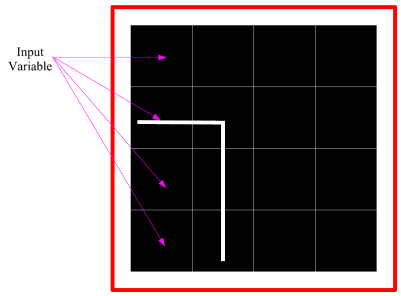

Figure 3. *This figure depict how is divided the windows on several square zones. Each of these ones correspond to a variable. One see inside the window one feature (an upper-right corner)*

3.4. Parametric forms

Since no a priori information about the distribution of the features is available, one assume $Feat$ to be uniformly distributed over $[0 \ldots N]$, i.e:

$$P(Feat = i) = \frac{1}{N} \quad \forall\, i \in \aleph, [0 \ldots N]$$

One assume that these variables follows a Gaussian probability law where the mean and the standard deviation is dependent on the particular zone and the feature corresponding to the variable.

$$P(X_i = x) = Gauss(\mu(zone, Feat), \sigma(zone, Feat)) \quad \forall\, i \in \Re, [0, N]$$

where $\mu(zone, Feat)$ and $\sigma(zone, Feat)$ will be computed during the learning.

This particular choice for the probability law seemed to be logical since a Gaussian law express the fact that for each primitive, each variable has a fixed value corresponding to the mean of the Gaussian. More the value will differ from the mean, the less probable it will be.

3.5. Identification

The identification process consists in determining the different free parameters of the previously chosen probability laws associated with the variables. In our case, we only need to determine parameters for the zone ratio variables (mean and standard deviation) since the $Feat$ variable is already completely determined. This is done by performing a learning over a defined data feature set.

3.6. Question

According to the equation 1 the Bayesian inference will therefore be of the following form :

$$P(Feat \mid X_1 \otimes X_2 \wedge \ldots \wedge X_i \wedge \delta \wedge \pi) \tag{5}$$

where i depends on the number of division used to describe the selected features.

Figure 4 resumes the Bayesian Programm used in this work.

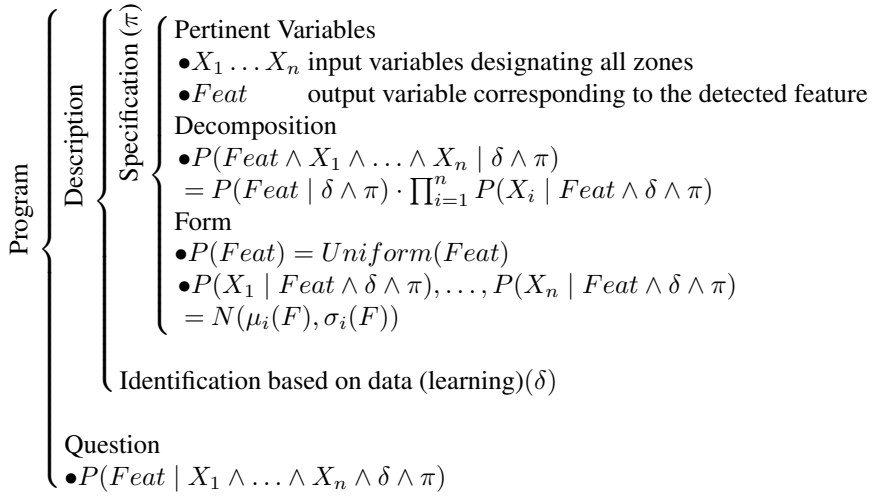

Figure 4. *Structure of our Bayesian program for primitives recognition.*

4. Experimental Results

In order to validate our Bayesian programming based approach for visual primitives detection, some experiments were conducted.

The first step is a pre-process used to obtain a black and white image, by applying a simple method of thresholding. This is followed by a gaussian blurring filter in order to smooth the image and remove the noise. Finally, edges in the resulting image are detected using the well known Canny edge detector [1]. After this pre-processing step, our method for simple forms recognition can be applied.

The second step of experiments is to determine the parameters of Gaussian distributions for our variables. This is made by calculating the mean and the standard deviation iteratively from a set of examples. The learning set contains examples of all kind of primitives with variations in orientation and in position with respect to the center of the window. Nevertheless, the positions stay close to the centre, because if primitives are learned in all the surface of the windows, all variables will take a similar value for all primitive, and recognition capacity will be strongly affected. In order to avoid this, one add garbage to the set of primitive. These garbage collectors allow capturing all features outside the correct interval.

The third step is to use the Bayesian program to recognise primitives similar to those learn. To do that, the program scans the entire image with a window and analyse the window at each step. Since the windows can contain several primitives at the same times a new intermediary process is needed. Blob detection is realised so as to detect all features in order to process them separately and contingently to remove too small feature to correspond to a primitive.

4.1. Results

To validate our program, a first test was realised on the learning set itself, with different configurations of variables and sizes for the windows. Hence, we can choose the best configuration for the real tests with features not included in the learning set. The Table 1 depict the results of this test. It can be noticed that the best configuration is for a size of 12×12 pixels, and 36 variables (6×6 variables made from 2×2 pixels). The best results obtained with the most big number of variables is explained by the fact that we don't take into account the configuration of black and white pixels to evaluate a variable, but only the number of black and white pixels. So, the fewer variables are, the more confusion between primitives is.

	Set A			Set B		
Nb pixels	12×12		8×8	12×12		8×8
Nb variables	36	16		36	16	
Mean	83.6%	74.1%	68.4%	85.9%	77.4%	70.0%

Table 1. *Results for the validation of the method and the determination of the best type of window.*

Finally, Table 2 shows some results realised with a window of 12×12 pixels, and 36 variables, and with an image composed of several lines drawn in several orientations, and with corners of different angles described in figure 5.

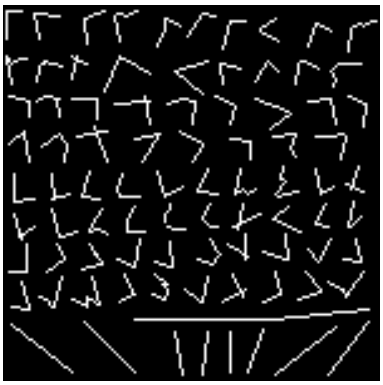

Figure 5. *Example of testing image after the pre-processing.*

Table 2 shows a percentage of correct recognition bigger than 68% with a average of 81%. Another remark is that the sum of percentages for columns are not equal to zero. This is a result of the misclassification between the core-primitives and the garbage collector which are not showed in this table. Indeed, we consider that a garbage detection is not pertinent for statistics since it depends of the position of the primitive and not only of the type of primitive.

Another difficulty is to interpret why a given primitive is better recognised than another one. Some configurations of features inside the window can be ambiguous, and in order to build statistics, we have to choose if the Bayesian program got the wrong class

	VL	HL	SL	BL	UL	UR	LL	LR
VL	89%		7%		1%	3%	3%	2%
HL		98%	1%	1%				1%
SL	1%		68%			3%	4%	
BL				76%			1%	3%
UL	1%		1%	13%	77%	4%		1%
UR	3%		5%		1%	80%	3%	2%
LL	4%	1%	1%		1%	6%	82%	8%
LR	1%		1%	7%		1%	5%	81%
Nb measures	129	104	82	106	206	246	205	217

Table 2. *Results with real image. In this table, meanings of abbreviation are: VL = vertical line; HL = horizontal line; SL = slash line; BL = back slash line; UL = upper-left corner; UR = upper-right corner; LL = lower-left corner; LR = lower-right corner.*

for the feature or not. This act can depend on the variation of the visual appreciation. Moreover, some primitives can be more similar to the class belonging to the garbage collector than to another one. This last fact can strongly depend on the set choose for the learning.

Moreover, our scanning scheme allows one feature to be detected multiple times. This fact can increase the quality of the detection. Indeed, if the same feature is detected N times, it is not absolutely necessary that the detector recognises the right feature N times. We can imagine that a majority of correct decisions for each feature should be sufficient for a future processing. From this point of view, the importance of detection relies no longer only on the ground-truth results but also on the limitation of false positives for each features which if the necessary condition to state with high probability that a feature detected is of the right type. Taking into account all these requirements, it can be noticed that Table 2 shows good results with an average of good detection equal to 81%. This high capability of avoiding confusion between features is also very important for the robustness.

5. Conclusion and Future Work

This paper presented a new method for visual primitive's recognition by using the Bayesian programming methodology. This work took place in the context of perception and interpretation of the environment using a probabilistic method. From the experiments, we conclude that the presented approach is practical and robust with a global result of 81% of correct matching. This result can be improved by using the following property of the scanning of the image: a primitive is analysed in several position inside the window. Thus one can eliminate the cases where the primitive is recognised as a class belong to garbage collector.

In the context complex objects recognition, a future work will be to realise a hierarchical Bayesian program to recognise more complexes objects composed of several primitives.

References

[1] J. Canny, *A Computational Approach to Edge Detection*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol 8, No. 6, Nov 1986.

[2] David Hubel and Torsten Wiesel, *Receptive fields of single neurones in the cat's striate cortex*, Journal of Physiology, Vol 148, pages 574-91, 1959

[3] David Hubel and Torsten Wiesel, *Receptive fields, binocular interaction and functional architecture in the cat's visual cortex*, Journal of Physiology, Vol 160, pages 106-54, 1962

[4] Olivier Lebeltel, *Programmation Bayésienne des Robots*, INP Grenoble, 1999

[5] Bastian Leibe and Bernt Schiele, *Interleaved Object Categorization and Segmentation*, in British Machine Vision Conference (BMVCŠ03), Norwich , UK, Sept. 2003

[6] Bastian Leibe, Bernt Schiele: *Analyzing Appearance and Contour Based Methods for Object Categorization*. CVPR (2) 2003: 409-415

[7] K. Mekhnacha, E. Mazer, and P. Bessière, *A robotic CAD system using a Bayesian framwork*, In Proceedings of the IEEE-RSJ International Conference on Intelligent Robots and Systems, Takamatsu (JP), Best Paper Award, 2000

[8] Justus H. Piater and Roderic A. Grupen, *Feature Learning for Recognition With Bayesian Networks*, In Proceedings of Fifteenth International Conference on Pattern Recognition (ICPR 2000), Barcelona, Spain, 2000

[9] A. Tapus and G. Ramel and L. Dobler and R. Siegwart, *Topology Learning and Recognition using Bayesian Programming for Mobile Robot Navigation*, Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Sendai, Japan, 2004