Tesis de Máster

Study of an object recognition algorithm

Pierre Lassalle

Donostia - San Sebastián, Julio 2012

Universidad del País Vasco / Euskal Herriko Unibertsitatea
Departamento de Ciencia de la Computación
e Inteligencia Artificial

Director: Basilio Sierra

www.ccia-kzaa.ehu.es

Abstract

Graduate Department of

University of the Basque Country

2012

This project introduces an improvement of the vision capacity of the robot Robotino operating under ROS platform.

A method for recognizing object class using binary features has been developed. The proposed method performs a binary classification of the descriptors of each training image to characterize the appearance of the object class.

It presents the use of the binary descriptor based on the difference of gray intensity of the pixels in the image. It shows that binary features are suitable to represent object class in spite of the low resolution and the weak information concerning details of the object in the image.

It also introduces the use of a boosting method (Adaboost) of feature selection allowing to eliminate redundancies and noise in order to improve the performance of the classifier.

Finally, a kernel classifier SVM (Support Vector Machine) is trained with the available database and applied for predictions on new images.

One possible future work is to establish a visual servo-control that is to say the reaction of the robot to the detection of the object.

Key-words: binary descriptor, feature selection, supervised classification, object recognition

Acknowledgements

I wish to thank my supervisor Basilio Sierra, each of his advice was invaluable. I could talk without any constraints and in spite of the difficulty of the language, he listened to me and followed me during one year. I wish to thank the director of the Master, Yosu Yurramendi for his help and his support.

Contents

Chapter 1

Introduction

The new method proposed here is an object recognition algorithm in view of implementing in the robot Robotino.

This robot is equipped with a camera and works under ROS[1] platform. The scope is to develop an adequated recognition method with the available hardware. We need a process which does not require a lot of memory and computing time.

At the beginning of this project, our line of investigation was to use a naive Bayesian classification of SIFT descriptors extracted from images. This work turns out to be less effective for two main reasons. The first reason is the use of SIFT descriptor. Our context is not to recognize objects which presents exactly the same shape in the same scene. As a result using SIFT descriptors has proven the non-discrimination for variable object. What is interesting for this kind of descriptor is its invariance for scale and rotation but it is very sensitive to variations of the geometric shape of the object. The second reason is the high computing time and the large required memory.

Concerning our hardware available, we have a low-cost camera with a standard robot not presenting high performance of computing. Under these constraints recognizing an object seems to be difficult but over the past few years, it has been seen the use of binary

[1]Robot Operating System

descriptors. These descriptors are easy to calculate and the computing time is very low. Another quality is that in spite of the weak information of low-resolution image, the description of the object is relevant. About the classification, these recent past years it has been proven the effectiveness of kernel and boosting classifications. That is why in this project, we use these new notions and establish an algorithm combining them.
In this paper we will firstly extract features from training images in a database containing positive and negative samples using binary descriptors.
Then because of the high dimension of features, a boosting method will be introduced to have most relevant features.
Once the data treated, a SVM classifier with an appropriate kernel function is trained and used to predict a set of new vectors of observation.
Finally, we will discuss the results of the prediction and its performance.

Chapter 2

The learning database

To test our new algorithm, we have used the learning database[1] provided by [1]. We have formed two repertories: one for the learning step and the other for the test. The learning repertory contains 4500 images which represent pedestrians. They are called positive examples. It also contains 5000 images which represent an urban environment without pedestrians. They are called negative examples.
The test repertory contains 1000 positive examples and 1000 negative examples. The property of each image is the same:

- one color channel;
- size : 36*18 pixels

An example of such images is shown (cf 2.1). The urban environment is clearly variant and the resolution is very low. In a first view, detecting the presence of a pedestrian seems to be difficult. The manner to describe the information of the image must be robust despite its low quality.

[1]available on the website http://www.science.uva.nl/research/isla/downloads/pedestrians/

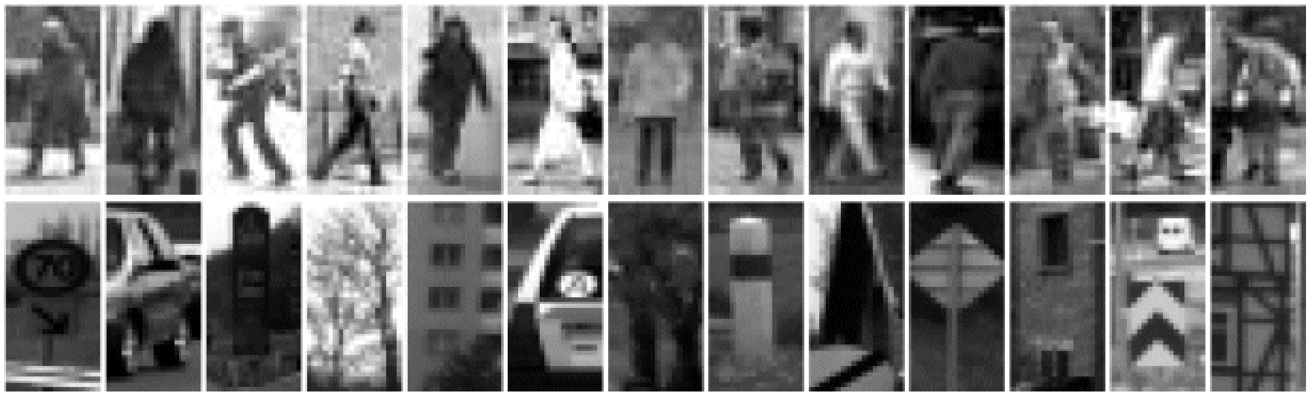

Figure 2.1: Images of the learning database

Chapter 3

The image processing library: OpenCV

For this work, we decided to choose OpenCV which is an open source computer vision library[1]. What is interesting is that library is written in C++ and runs under Linux. This is an optimized library which contains over 500 functions in various fields and the more important for us a sublibrary Machine Learning (MLL). This library is very useful for our project because it covers all the tasks needed.
What is more the robot Robotino of the laboratory runs under ROS platforms. This platforms uses many open source packages including the library OpenCV. Thus in a future work, it would be easy to implement our algorithm as a new package for ROS.

[1]see http://opensource.org

Chapter 4

Binary descriptors

4.1 Related work

These last years, new types of descriptors have appeared to answer to a specific constraint: working in real-time. Indeed nowadays we can see more and more embedded systems which use low-cost active camera to take video sequences of the environment. In this context, the followed recognition algorithm must be efficient. One of the difficulties is to realize the task in time. In this context binary descriptors are more appropriate. Their computational simplicity and weak computing time are the major advantages. What is more comparing two binary descriptors is very easy because the distance between them is very easy to calculate. Several binary descriptors have been built like the Local Binary Patterns (LBP)[2] and the Control Points[3].
For the LBP, the gray intensity of a central pixel is compared to those of its neighbors. The neighbor area is determined by the user. For the control points, the idea is to build two groups of points separated by a same distance K. The feature is equal to 1 if the gray intensity of all the pixels of the first group are greater than those of the second group, else the feature is equal to 0. Here the distance K must be chosen by the user.

4.2 Descriptor proposed

A global binary descriptor is used for this project. Its implementation is strongly inspired from the LBP introduced before.
The idea is to compare the gray intensity of pixels in images.
In our project, we decide, for a simple reason but also for covering the whole description of the image, to adopt a specific descriptor.
Precisely, we compare the gray intensity of each pixel of the image to all the others which belong to the same column and the same row. The binary value of each new feature respects the following rule:

$$1 \textbf{ if } I(p_1) \geq I(p_2)$$

$$0 \textbf{ else}$$

The image (cf 5.1) illustrates this rule. For example at the top of the image the intensity of the pixel p_1 representing the color of his hairs is greater than p_2 representing the color of the background in the same line so that the value of the corresponding feature is 1.

Thanks to this descriptor all the points are selected in the image so that we can conclude that the object is perfectly described. This descriptor is invariant to position and scale but it varies according to the luminance. The number of output features just for one image is $18 * C_2^{36} + 36 * C_2^{18} = 16848$.
The major drawback of this descriptor is the huge number of features. It is sure that among this large number of features there are redundancies and noise. This could affect the efficiency of the later classifier. In order to solve this trouble it seems necessary for us to operate a method of feature selection to reduce the useless information and to have a vector of observation containing only relevant features.

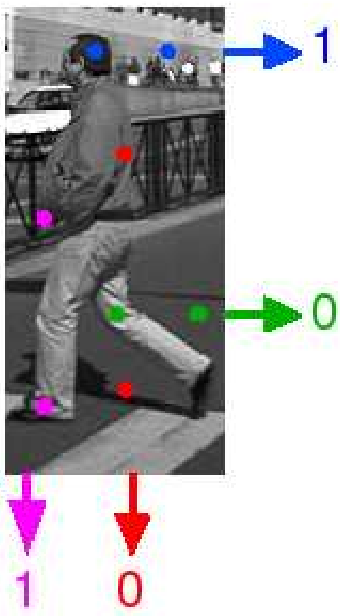

Figure 4.1: Binary descriptor used for the project

Chapter 5

Feature selection

5.1 Related work

The feature selection consists in selecting the best relevant features. We use feature selection because in some cases among the high number of features there are irrelevant and redundant variables which may hurt the classification performance of the classifier.
There exist filter methods to accomplish the selection of variables. One first method called univariate filter methods[4] consists in choosing the first k variables according to their strength of association with target-class. Various measures of association can be used like Fisher or the Pearson coefficient.
There is also Multivariate feature selection[5] where an indicator measures how predictive a group of features is and how many redundancy are.
Finally there exist Wrapper methods[6]. Let A,B,C be the predictors and M be the classifier. We want to predict the class label given the smallest possible subset of A,B,C, while achieving maximal performance. The drawback of this method is the dependence of the method of feature selection with the classifier selected.

5.2 Method proposed

The choice for the method is particularly difficult. It is necessary to have an independent method from the type of the later classifier. This algorithm has to be fast and does not require a lot of memory. Our aim is to have a reduced and relevant vector of observation for each image. The difficult task is resolved by the discovery of boosting methods[7] which answer to our objective. Even though the first task of this method is to classify, it can be used to select features and reduce dimension.

The principle of Adaboost[8] is to combine several weak classifiers to create a strong classifier. It is obvious that the more important the number of weak classifiers is, the more important the computing time and the required memory are.

For each vector of observation x_i we associate a weight $p(x_i)$ which varies according to the response of the algorithm at the previous iteration.

Let S be the set of learning composed of N examples with: $S : (x_i, y_i)_{i=1,..,N}$. x_i represents a vector of observation and y_i represents the label that is to say the class to which x_i belongs. Here we have two classes, so we define the class object as the positive class: $y_i = 1$ and the rest as the negative class: $y_i = -1$.

Let W_t be the weight vector at the iteration t. At the first iteration, the weight is the same for all the vectors: $W_0 = 1/M$ where M is the dimension of the vectors of observation.

For each iteration, a sample is drawn according to the weight (decreasing) and a weak classifier h_t is trained. Then the apparent error of h_t is calculated on the whole training data and the weights of each vector of observation are updated following the rule:

A constant α is calculated: $\alpha = ln(\frac{1-\epsilon_t}{\epsilon_t})$ where ϵ_t is the apparent error.

The weights are modified:

$$W_{t+1}(x_i) = \frac{W_{t+1}(x_i)}{Z_t} exp(-\alpha) \textbf{ if } h_t(x_i) = y_i$$

$$W_{t+1}(x_i) = \frac{W_{t+1}(x_i)}{Z_t} exp(\alpha) \textbf{ if } h_t(x_i) \neq y_i$$

Where Z_t is a normalization factor checking $\sum_{i=1}^{N} W_t(x_i) = 1$.

We record the weak classifier h_t and α and we move to the following iteration.

When the algorithm is finished, we have a strong classifier: $sign(H(x_i) = \sum_{i=1}^{T} \alpha_t h_t(x_i))$

An example of such a classification is shown figure 5.1.

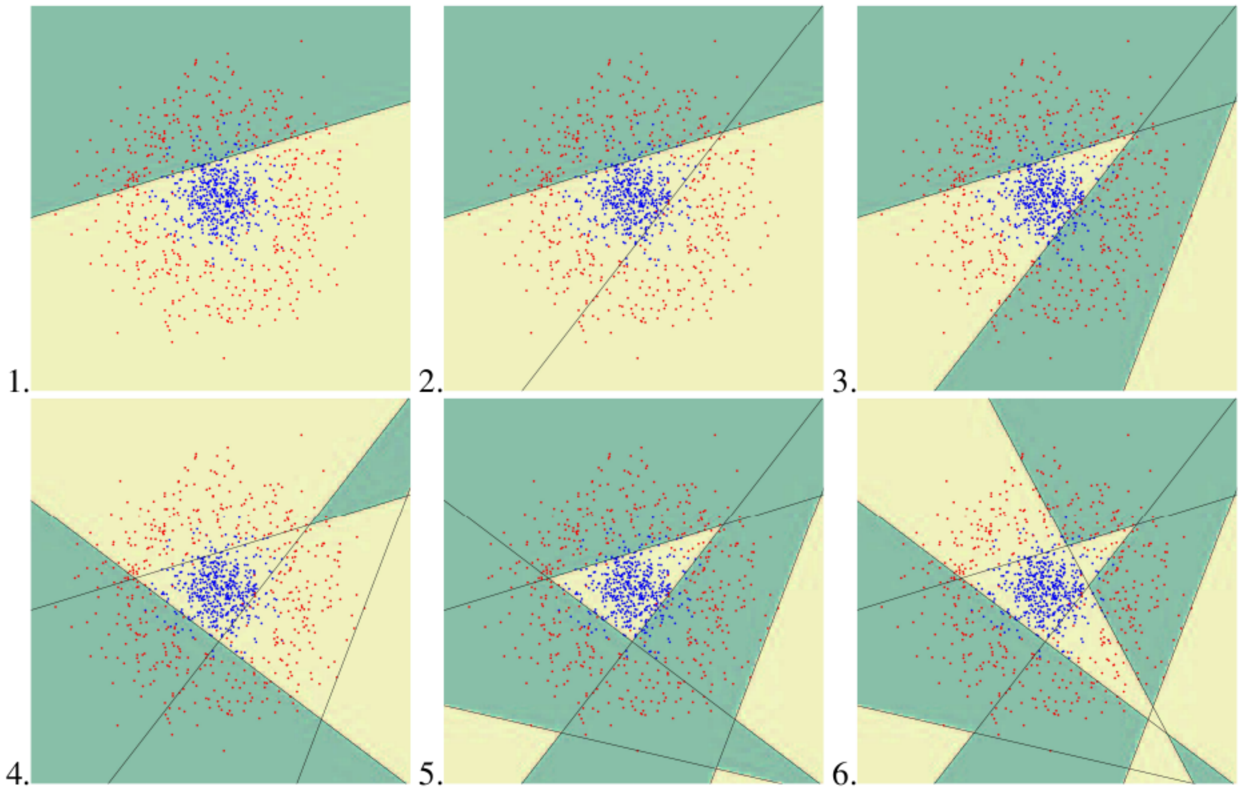

Figure 5.1: Adaboost classification

The new features of our vector of observation are the prediction of each weak classifier selected by the algorithm Adaboost.

The new expression of each new vector of observation is:

$$\{x_i = (h_1(x_i), h_2(x_i, ..., h_T(x_i)), i \in [1, .., N]\}$$

The scope for our project is to find a compromise between the number of weak classifiers and the constraints of computing time and memory. A low value does not describe correctly our object and the performance of the future classifier would be degraded. A high value would require a high computing time for a non-relevant improvement. The number 100 is experimentally chosen.

Chapter 6

SVM: Supervised Classification

6.1 Related work

Two categories of statistical learning must be distinguished: Supervised and non-supervised methods. In this project, we have access to a huge database where images are labeled so that we decide to use a supervised classifier.
Supervised methods permits to evaluate the distribution of a quantity based on linked values without measuring it straightly. The supervised learning takes in account two kinds of variables: X which are input variables and Y which are the prediction of the input variables.
There are two kinds of supervised classification:

– the generative models using a discriminant or quadratic analysis, it is the case of the Naive Bayes;

– the discriminative models using decision trees or K-Nearest Neighbors, it is the case of the SVM or the RVM[1];

[1]Relevance Vector Machine

6.2 Learning algorithm

The use of the kernel classifier SVM[9] is justified because it introduces a major advantage: it permits to obtain non-parametric classification rule. It only depends on a reduced number of support vectors among all the training vectors. This is very interesting because we have a huge number of training vectors and we are limited regarding the memory.
In this space of dimension 100, we suppose there exists a separating hyper-plane between positive and negative vectors. The goal is not to find only one but finding that which pass right through the middle of the two classes. Like this, if an example is not correctly described during the learning step and presents variation its classification will not be different if its distance to the hyper-plane is high. Thus we have to find an hyper-plane whose minimal distance between the two classes must be maximal. This distance is called the margin (figure 6.1).

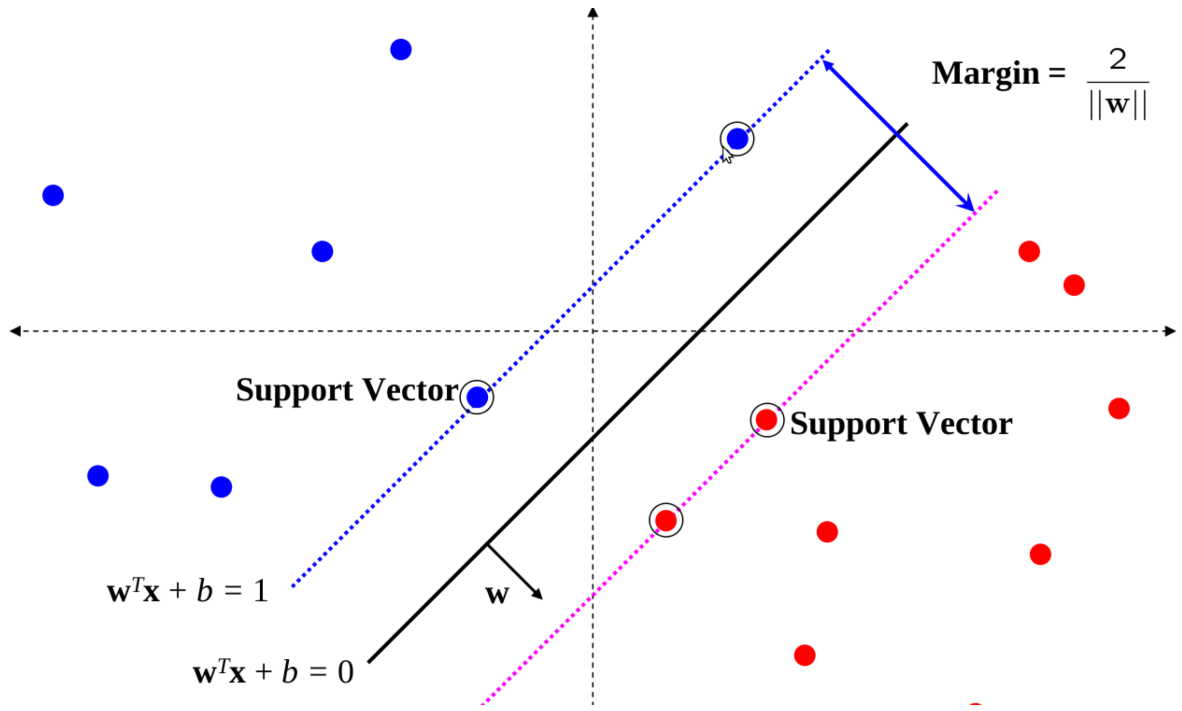
Margin = $\frac{2}{||\mathbf{w}||}$
Support Vector
Support Vector
$\mathbf{w}^T\mathbf{x} + b = 1$
$\mathbf{w}^T\mathbf{x} + b = 0$
$\mathbf{w}$

Figure 6.1: Hyper-plan, margin and support vectors

The hyper-plane is characterized by the following equation:

$$arg \max_{w_0,w} \min_k (||x - x_k|| : x \in R^N, w^T x + w_0 = 0)$$

For this hyper-plane, the value of the margin is $1/||w||$. Maximizing the margin reverts to minimize $||w||$.

A quadratic optimization method is used to determine this margin. It turns out that the solution depends on the number of training observations.

Sometimes because of the dimension of the training vectors, it can be impossible to find an hyper-plane. It turns out that the higher the description space is, the higher the probability to find an hyper-plane is. It is necessary to use non-linear internal function allowing to move to a bigger space description (figure 6.1).

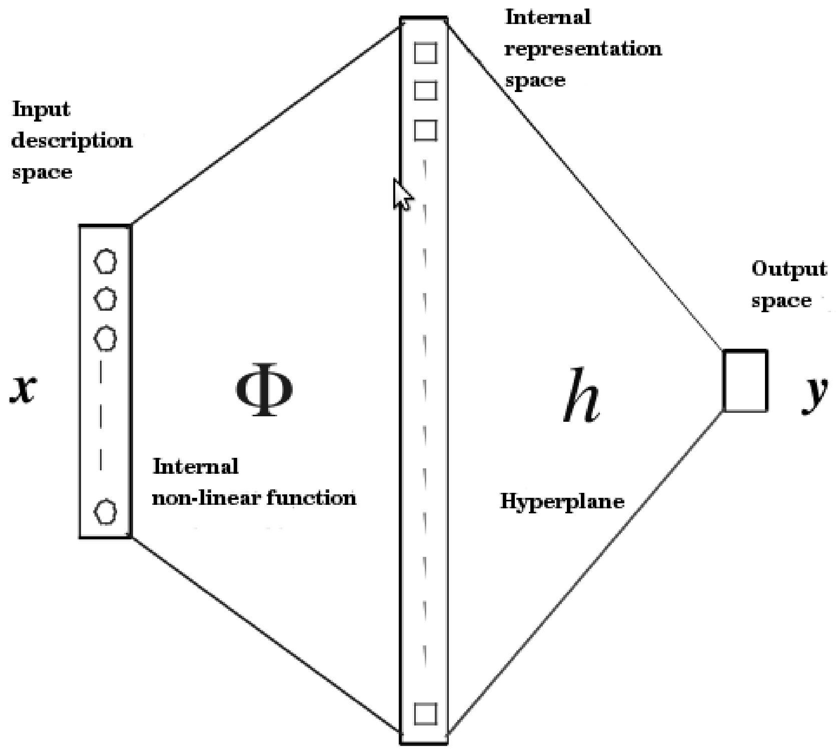

Figure 6.2: Internal re-description

Finally, a last point has to be clarified. The optimization method needs to calculate scalar product in the internal re-description space to find the hyper-plane. It is obvious that calculating a scalar product is complicated in such a space. That is why bilinear symmetric functions which realizes implicitly a product scalar in a space of high dimension are used because they are easier to calculate. These functions are called kernel functions. They become the only parameter that has to be fixed by the user when he uses the classifier SVM. The table (figure 6.1) sums up the essential kernel functions used for the classifier :

Kernel function	**mathematics expression**
polynomial	$K(x, y) = (x.y + c)^n$
RBF(radial basis function)	$K(x, y) = exp(\frac{-\|\|x-y\|\|^2}{2\sigma^2})$
sigmoid	$K(x, y) = tanh(a(x.y) - b))$

Table 6.1: Kernel functions most used

For the polynomial and RBF functions, the value of c and σ^2 are fixed by the user. For the sigmoid function, the user must be careful selecting the value a and b because the theorem of Mercer is checked only for certain values.

The choice of one of these functions with their parameters is experimental. It is the major default of this classifier because finding the best choice may be difficult. For our project, it turns out that the polynomial function with $c = 1$ and $n = 3$ gives the best results but it could be completely different from another context.

Chapter 7

Evaluation measures

In order to estimate the performance of the classifier and to discuss about the obtained results, we choose the recall and precision curve. The prediction error of a classifier is either a false positive FP or a false negative FN. RP represents the right positive.

The indicator Precision evaluates the capacity of the classifier to associate each vector of observation with the correct class. Its expression is:

$$Precision = \frac{RP}{RP + FP}$$

The indicator Recall evaluates the capacity of the classifier to find positive examples. Its expression is:

$$Recall = \frac{RP}{RP + FN}$$

Both indices must tend toward 1 for a better performance but of course it is impossible to reach exactly one. In order to obtain a general indice of the effectiveness of our classifier, a variable called F_1 is computed:

$$F_1 = \frac{2 * Precision * Recall}{Precision + Recall}$$

The more this indice tends to 1, the more effective is the classifier.

To test the performance of our classifier, we have trained the classifier with a variable number of training vectors containing an equal number of positive and negative examples.

Then we have tested our trained classifier with 2000 new images representing the two classes. The results of these predictions are shown in the table (cf 7.1)

Number of training vectors	**Recall**	**Precision**	F_1
200	67%	89%	76.4%
400	70%	86%	77.1%
1000	73%	83%	77.6%
2000	**74%**	**82.9%**	**78.2%**
4000	76%	79%	77.4%
6000	77%	77%	77%
8000	76%	76%	76%

Table 7.1: Precision and Recall

The index Recall tends to improve with the number of training vectors until a certain limit. The index Precision is quite good and tends to decrease a little when this number increases. We can see that the good compromise is 2000 training vectors.

Chapter 8

Conclusion

At the end of this project a quite effective recognition algorithm has been established. It is based on a effective binary descriptors, a pertinent feature selection and a fast and effective classification. It respects all the constraints regarding the computing time (230 μs for a complete prediction of a new image) and the memory imposed. What is more its implementation in the robot Robotino would be done easily because its operating system ROS takes into account the library OpenCV and the language C++ which are techniques used to develop our algorithm.

This work represents an intelligent treatment of the images taken by the camera of the robot. Indeed, we have created an algorithm capable of interpreting images coming from the scene. For a future work what would be interesting is to elaborate a visual planning algorithm which would control the behavior of the robot according to the presence or not of the object and its localization.

To conclude, the results on the effectiveness of this algorithm for standard robot turn out to be interesting and permit to open a new line of investigation on the object.

Bibliography

[1] S. Munder and D. M. Gavrila. An Experimental Study on Pedestrian Classification. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 28, no. 11, pp.1863-1868, November 2006

[2] Timo Ojala, Matti Pietikinen, and Topi Menp. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. IEEE Transactions on Pattern Analysis and Machine Intelligence, 24(7) :971987, July 2002.

[3] Yotam Abramson, Bruno Steux, and Hicham Ghorayeb. Yef (yet even faster) real-time object detection. In International Workshop on Automatic Learning and Real-Time, pages 513, Siegen, Germany, 7 September 2005.

[4] Isabelle Guyon and Andre Elisseeff. An introduction to Variable and Feature Selection. Journal of Machine Learning Research 3: 1157-1182, 2003.

[5] Mark A. Hall, Correlation-based Feature Selection for Discrete and Numeric class Machine Learning.

[6] Yvan Saeys, Inaki Inza and Pedro Larranaga. A review of feature selection techniques in bioinformatics.

[7] Kenji Kira and Larry A. Rendell, The feature Selection Problem: Tradional Methods and a New Algorithm (1992)

[8] Freund and Schaipre. Experiments with a new Boosting Algorithm, 2001.

[9] Antoine Cornuejols. Une nouvelle methode d'apprentissage: Les SVM. Separateurs vaste marge, 2002.