Fast Object Detection by Regression in Robot Soccer

Susana Brandão*, Manuela Veloso, and João Paulo Costeira

Carnegie Mellon University - ECE department, USA
Carnegie Mellon University - CS department, USA
Instituto Superior Técnico - ECE department, Portugal
sbrandao@ece.cmu.edu, mmv@cs.cmu.edu, jpc@isr.ist.utl.pt

Abstract. Visual object detection in robot soccer is fundamental so the robots can act to accomplish their tasks. Current techniques rely on manually highly polished definitions of object models, that lead to accurate detection, but are quite often computationally inefficient. In this work, we contribute an efficient object detection through regression (ODR) method based on offline training. We build upon the observation that objects in robot soccer are of a well defined color and investigate an offline learning approach to model such objects. ODR consists of two main phases: (i) offline training, where the objects are automatically labeled offline by existing techniques, and (ii) online detection, where a given image is efficiently processed in real-time with the learned models. For each image, ODR determines whether the object is present and provides its position if so. We show comparing results with current techniques for precision and computational load.

Keywords: Real-time Perception, Computer Vision.

1 Introduction

In robot soccer, vision plays a crucial role on localization and actuation since both task rely on images to provide ground truth for landmarks and objects localization. One of the biggest challenges faced by robot soccer teams is to provide the robot with adequate models for each class of objects in the field. The current paper presents a highly efficient way of recognizing objects in this environment.

We address the problem of object detection in the RoboCup Standard Platform league that uses the humanoid NAO robots (www.aldebaran.com). In this league, the robots have access to images of high-resolution at a fast acquisition rate and have to process them without totally consuming the limited on-board robot computational resources, which are also needed for all other non-vision task functions. In this context, vision algorithms need to be not only highly reliable, but also computationally efficient and easy to extent to all the objects in the field.

* This work was partially supported by the Carnegie Mellon/Portugal Program managed by ICTI from Fundação para a Ciência e Tecnologia. Susana Brandão holds a fellowship from the Carnegie Mellon/Portugal Program, and she is with the Institute for Systems and Robotics (ISR), Instituto Superior Técnico (IST), Lisbon, Portugal, and with the Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA. The views and conclusions of this work are those of the authors only.

T. Röfer et al. (Eds.): RoboCup 2011, LNCS 7416, pp. 550–561, 2012.

Two widely used approaches for object detection in this domain are: (i) a scan-line based algorithm [8] that effectively reduces the size of the image to samples along vertically spaced-apart scanned lines, but relies on the manual definition of elaborated models of each object; and (ii) a run-length encoding region-based algorithm, CMVision [2], that effectively identifies colored blobs with objects, but is computationally expensive in large images. Other successful more focused approaches include the use of neural networks [6], circular Hough Transforms [4], and circle fitting [5].

Scan-line is a very thorough algorithm that relies mostly on human modeling of the several elements in the field. It creates color segments based on the scanning of just a few columns in the image. To compensate the information lost in the sampling, the algorithm uses human imposed priors on what the segments should be in the robot soccer environment. By requiring the use of a reduced set of lines, the algorithm is very fast. However, the modeling of each object in the field is quite time-consuming and the algorithm is not easily extendable to new objects.

CMVision relies on color segmentation to create blobs that will then be identified as objects. However, since blobs are created based on 4-connectedness, it requires color thresholding of almost all the pixels in the image. Afterward, blobs have to be sorted by color and size, and finally objects are detected based on how well the largest blobs of their respective color fit to a given model, which is again imposed by humans. All this process, albeit quite accurate and easier to extend to new objects than the previous, is computationally expensive.

In this work, we introduce a new object detection approach, ODR, (for Object Detection by Regression), that relies on the offline creation of statistical models for the relation between a whole image and the object position in that image. The use of statistical models enables ODR to use a sampled version of the image, which greatly reduces the online computational cost of the algorithm. Though more complex models could have been used, a further reduction on the computational load is obtained by using linear relations between object positions and images. State of the art offline detection algorithms, e.g. [3], use local statistics of the image to detect an object and thus require exhaustive search at all possible locations and scales, which is quite time consuming. By using the whole image, together with priors provided by the environment such as object colors, we are considering a *global* statistical representation of the image that avoids this exhaustive search. Furthermore, ODR is easy to extend since it relies on the also easy to extend CMVision algorithm to provide the object positions during the offline learning stage.

The paper is organized as follows. Section 2 presents the overall ODR approach with the description of the wide object detection problem to be solved. Section 3 presents the pre-processing required by ODR. Section 4 and Section 5 describe the ODR offline and online algorithms respectively. Section 6 shows experimental results that demonstrate the accuracy and effectiveness in object detection with a rich set of real images taken with a NAO robot, including truncated and occluded objects in noisy situations. We review the contributions of the work and conclude in Section 7.

2 Object Detection by Regression

ODR detects an object position in an image with a robust and computationally efficient algorithm. The algorithm leverages in the repeatability of the RoboCup environment to create a statistical model of the relation between an image and the position of a given object in that image. The statistical model can be learned offline and provides fast online detection by allowing image sampling. Furthermore, it can be extended to other objects without requiring extensive human modeling. As an output to the online stage, the algorithm returns an object position and a confidence on the presence of the object.

To relate images to object positions we use a linear model represented by a matrix W and affine vector $\mathbf{b}_0$. The model is learned offline using a set of training images, represented as a single matrix O. The images contain the object at different positions and this position is know and represented also as a single matrix P. To avoid over fitting to the training data and to provide filtering of noise, the training set is reduced by means of a principal component analysis. The linear model learned relates not the object positions P and the image training set matrix, O, but the object position and the matrix O_r that corresponds to the projection of O on the training set principal components, V. However, since the projection into principal components is a linear operator, the relation between the object position and the image will still be linear and thus efficient to use.

Clearly the relation between images and objects position is more complex than just linear. Typical images are cluttered, objects have different sizes, different colors and suffer from occlusions and pose changes. To allow such a simplification, we consider the existence of a pre-processing stage before ODR. There are no constrains to the type of pre-processing as long as it returns a version of the original image where each pixel is now assigned one of two values: 1 if the pixel is thought to belong to the object, 0 if not. To the set of pixels labeled as 1, we refer to as object hypothesis.

If the object hypotheses was errorless, e.i., all the pixel labeled as 1 belonged to the object and all the object pixels were assigned a value of 1, the object position could be recovered by simple centroid estimation. The centroid estimation is a linear operation over a normalized version of the pre-processed image such that the sum of the value of all the pixels in this image version is 1. This normalization allows to compute the centroid of images with objects with different sizes using the same set of coefficients. Since there is currently no algorithm capable of determining the object hypothesis without error, the pre-processed images will contain noise resulting from robot motion and from illumination.

The objective of ODR is to determine in a robust way the object position in the image, in spite of these errors. To decrease the impact of noise in the detection, we filter the images by projecting them in the principal components of a set of training images. These training images are representative of the set of all possible images containing the object at different positions and are the same as those used to create the statistical model. We also note that ODR could still return a position in images where an object is not present, but there is noise. Interestingly, we then include in ODR a method to estimate a belief in the results of the detection performed using the linear model, inspired by similar work in face classification [7]. During its online processing, ODR projects a given test image into the principal components computed using the training set. It then estimates the belief based on the distance between the image and its projection into the linear subspace generated by the principal components. It detects false positives by thresholding its belief.

ODR results are evaluated using real data acquired by a NAO robot while approaching a ball and a second robot. The data contains all the complexities to be expected from a robot moving. Objects are distorted by blur or/and are occluded and there are images that just contains noise. We present ODR performance using three different metrics: (i) the capacity to detect objects that are in the image, (ii) the capacity to classify an image on whether the object is present or not (iii) the time efficiency of the algorithm.

In Figure 1 we present a diagram for the whole ODR algorithm where we identify each task to be performed in the online and offline stage. Each stage follows three different steps that are closely related across stages. The first step consists in normalizing the images. The normalization accounts for changes in size and fits all images into the same distribution, which is a requirement for the principal component analysis. The second step consists in the projection into the principal components, which are learned in the offline stage and then used for belief estimation in the online stage. The third step differs more significantly between the offline and online stages. In the offline stage, we learn the linear relation between positions and image. In the online stage we use this relation to compute the object position.

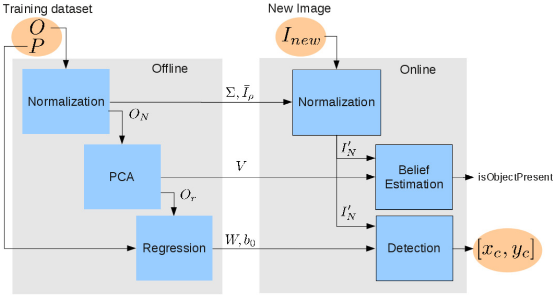
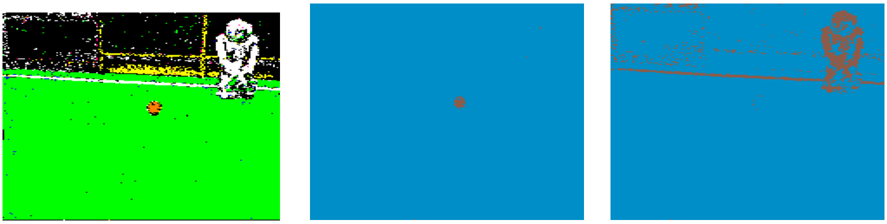


Fig. 1. Online and offline ODR

In the following sections we address the type of pre-processing required by ODR and explain in-depth each of the tasks that compose the offline and online stage.

3 Pre-processing

In the general case of object recognition, the pre-processing required by ODR can be the result of a segmentation or color threshold algorithm. In the case of humanoid robot soccer, color threshold is particularly appealing, since objects in the field are well defined by their color. Color thresholding is also a pre-processing stage in [2] and [8]. When the robot needs to detect the ball in those images, e.g., for kicking it into the goal, the object hypothesis for the ball in each image would be the set of all orange pixels. When the robot wants to identify other robots from a distance, the object hypothesis would be the set of white pixels. In Figure 2, we present an example of a thresholded image and the resulting object hypothesis for the ball and the robot.



(a) Thresholded Image (b) Obj. hypothesis for *ball* (c) Obj. hypothesis for *robot*

Fig. 2. Thresholded images and object hypothesis for *robot* and *ball*

After such pre-processing, an $N \times M$ image I with pixels indexed as $\{i, j\}$, can be represented as a binary vector in a NM space as $I' \in \{1, 0\}^{NM} : I'_k = 1$ if the pixel with indices $\{i, j : k = (i-1)M + j, 0 < j \leq M, 0 < i \leq N\}$ belongs to the object hypothesis and $I'_k = 0$ if not.

However, the images retrieved by the NAOs humanoid robots have a high-resolution that makes the color threshold of all the pixels a computationally heavy task. To overcome the computational burden, ODR only makes use of a subset of pixels. In a similar approach to [8], ODR scans the image at regular intervals, but it performs the scanning in the vector form of the image, I', not in the matricial form, I. We fix the sampling interval as $\varDelta$, and construct the image vector as $I'_s \in \{1, 0\}^D : I'_{s,k} = I'_{\varDelta k}$, where $D = NM/\varDelta$. To simplify notation, we drop the subscript s throughout the rest of the paper and, except when stated otherwise and without any loss of generality to the algorithm itself, refer to the sampled version of the image vector as the image vector in a $\mathbb{R}^D$ space.

4 ODR Offline Learning

In the offline stage, we have access to a large dataset of labeled, pre-processed and sampled images containing examples of different objects. The images corresponding to a specific object α are collected in an observation matrix O_α and the labels, i.e., the object positions, in a matrix P_α. If the aim was to have the robot identifying n objects, we would have n observation matrices, O_α and n position matrices P_α. Since each object is treated independently, we can again drop the subscript α and when we refer to observations matrix O, it is meant that the matrix only contains observations from a single object.

Each observations matrix, O, is constructed by assigning each pre-processed image to a row in the matrix. For example, if we had a set of L images , $I_1, ..., I_l, ..., I_L$, with N rows and M columns sampled at an interval of $\varDelta$ pixels, our observation matrix would be defined as:

$$O = \begin{bmatrix} I'_{1,1} & \cdots & I'_{1,1+\varDelta} & \cdots & I'_{1,n\varDelta+1} & \cdots & I'_{1,NM} \\ I'_{2,1} & \cdots & I'_{2,1+\varDelta} & \cdots & I'_{2,n\varDelta+1} & \cdots & I'_{2,NM} \\ \vdots & \dots & \vdots & \vdots & \dots & & \vdots \\ I'_{L,1} & \cdots & I'_{L,1+\varDelta} & \cdots & I'_{L,n\varDelta+1} & \cdots & I'_{L,NM} \end{bmatrix} \quad (1)$$

where $I'_{l,k} = 1$ if the pixel with coordinates $\{(n,m) \in \mathbb{N}^2 : k = (n-1)M + m\}$ belongs to image l object hypothesis.

The label matrix, $P \in \mathbb{R}^{L \times 2}$, contains the coordinates of the object in terms of its centroid $\mathbf{p}_{l,c} = [x_{l,c}, y_{l,c}]$.

Our training datasets are composed of both synthetic images and real images captured by the robot while it searches and follows a ball. For the synthetic dataset, we simulate images containing a specific object plus random noise. In each image the object is placed in different positions and those position uniformly cover the whole image. The resulting images include random noise and occlusion in edges and corners. For each object, the synthetic dataset is composed of 768 images. Examples can be found in Figure 3 for the ball example. The robot collected data includes balls in different parts of the image, but the sampling is not thorough. The robot is acting according to the ball position and keeps the ball, and the close by objects approximately in the center of the image. The resulting dataset contains fewer examples of ball on the edges of the image. However, real images introduce the variability on the object shape, pose and illumination that the robot will experience during run-time.

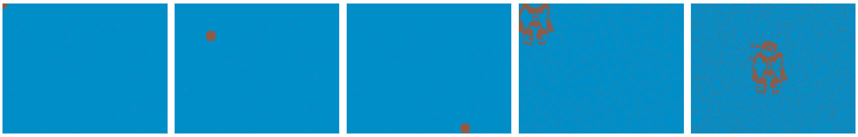
From the total of 856 real images we have for Ball and the 204 for Robot, we have occlusion on the image edges (Figure 4(e)) and by other objects (Figure 4(c)). We also have several examples of motion blur (Figure 4(d)) and of random noise (Figure 4(f)) captured by the robot while searching for the ball in the environment. All the real images were labeled automatically using CMVision.

All the offline tasks will be based on the observation matrix O constructed using the training dataset and in the position matrix P. The main output of the online stage is the set of the linear regression coefficients, W and $\mathbf{b}_0$.

Normalization

There are two stages of normalization: the first enable us to deal with objects of different sizes, the second for standardizing the data before performing the PCA. To deal with objects of different sizes, we first need to normalize all the images I': $I'_\rho = I' / \sum_k I'_k$. The observation matrix composed of the normalized images is represented as O_ρ.

For the principal components analysis we compute the observations sample covariance matrix ([1]) and it is a best practice to normalize the values for each pixel so that they follow a unitary zero mean Gaussian distribution. $C(O_\rho) = \Sigma^{-1}(O_\rho - \bar{O}_\rho)^T(O_\rho - \bar{O}_\rho)\Sigma^{-1}$ where O_ρ is the observation matrix corresponding to I_ρ, $\bar{O}_\rho$ is a matrix whose lines are all equal and correspond to the mean of each pixel over all the dataset $\bar{I}_\rho$, and



(a) (b) (c) (d) (e)

Fig. 3. Examples of synthetic images used in training. Images include objects in different positions (Figures 3(a)-3(c) for the ball; Figures 3(d)-3(e) for the robot), occlusion in image borders (Figures 3(a) and 3(d)) and noise (Figures 3(a)-3(e)).

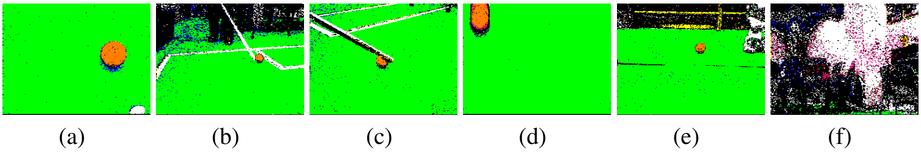


Fig. 4. Examples of real images used for training and testing. Examples include objects of different sizes (Figures 4(a)- 4(e)), motion blur (Figure 4(c)) , occlusions (Figure 4(c) and 4(e)) and random noise (Figure 4(f)).

Σ is a diagonal matrix, whose elements, σ_{ii}, are the standard deviation of the pixel i over O_ρ.

The mean and standard deviation estimated over the training dataset, will be used in the online stage, where each new image is normalized to fit the same distribution.

Principal Components Analysis

The principal components, V, correspond to the sample covariance matrix, $C(O_\rho)$, eigenvectors. The components form an orthogonal set of synthetic images that span the subspace of images with the same object in different positions and with different sizes.

Examples of the principal components obtained using the ball dataset are represented in Figure 5. The examples highlight the hierarchy in resolution of the principal components: the first components, which contain more information, have lower spatial frequency. We can thus reduce the dimensionality on our datasets by projecting the images into the first c components of this new basis, as seen in eq. (2). The effect will be equivalent to filtering in the spatial frequencies domain.

$$O_r = O_\rho V_c^T \qquad (2)$$

The number of components used affects the regression results. If ODR uses a large number of components, noise is added to the object model. Furthermore, the number of coefficient to be estimated during regression increases and over-fitting to noise becomes a possibility. If too few components are used, the reduced dataset observation matrix O_ρ may not have enough information to provide good regression results. In particular, all the small examples of the object may be filtered out. Deciding on the number of components to keep for the regression depends on the relative size of the smaller object we want to be able to identify and of the type of noise we expect to find during the online stage.

By training regression models using different number of components and computing the mean detection error per image in an independent dataset which reflects our expectations for the online stage, we can choose a priori the best number of components to use. The impact on precision of the number of components is illustrated in Figure 6(a) for the ball example. In this case, the mean error per image becomes constant after the use of 200 components, but the variance starts to increase. In the remaining of the experiments in this paper, we use only the first 200 components for the ball. For the robot, since it is considerably larger than the ball, we only need to use 15 components to estimate its position.

Regression
After the dataset dimensionality reduction, ODR performs a linear regression between the reduced images $I'_{r,l}$ and the known object positions $\mathbf{p}_l = [x_{c,l}, y_{c,l}]$. The result of the linear regression is the set of coefficients $W_r = (w_{r,x}, w_{r,y})$ and $\mathbf{b}_0 = [b_{0x}, b_{0y}]$, which solve the linear least squares problem in eq. (3) and are given by eq. (4) ([1]).

$$\min_{W_r} \|P - \tilde{O}_r \tilde{W}_r\| \tag{3}$$

$$\tilde{W}_r = (\tilde{O}_r^T \tilde{O}_r)^{-1} \tilde{O}_r^T P, \tag{4}$$

where P is the matrix which row l is the position vector $\mathbf{p}_l$ corresponding to image l, $\tilde{O}_r = (\mathbf{1}, O_r)$ and $\mathbf{1}$ is a column vector with ones that allow us to incorporate the affine bias term in $\tilde{W}_r = [\mathbf{b}_0^T, W]$. The set of coefficients in W and $\mathbf{b}_0$ are the main output of the offline stage.

5 ODR Online Testing

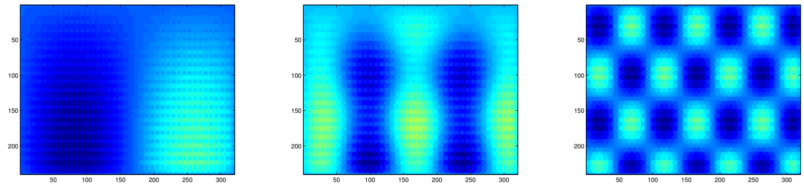
To find the object position in a new image, I_{new}, the robot needs to normalize the image vector following the same steps as in the offline stage. First the image is converted into a probability distribution, $I_{\rho,new}$. Then is normalized so it falls in the unitary zero mean Gaussian distribution estimated in the offline process: $I_{N,new} = (I_{\rho,new} - \bar{I}'_\rho)\Sigma^{-1}$.

Detection corresponds to the application of the linear model learned in the offline stage using equation eq. (5).

$$\mathbf{p}_{new} = \tilde{I}_{N,new}\tilde{W}_r = \mathbf{I}_{N,new}V^T W_r + \mathbf{b}_0 \tag{5}$$

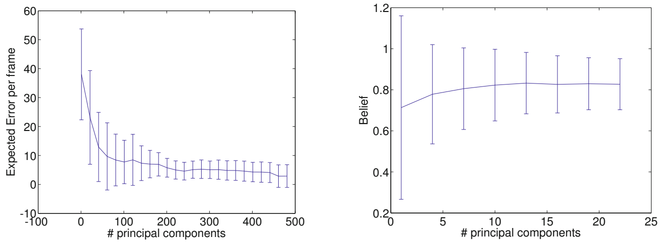
where $\mathbf{p}_{new}$ is the object position in the new image.

We can measure the degree of accuracy of this description by using eq. (6) to project the image in the PCA basis and re-project it back into the images space. The resulting image, I'_{rep}, corresponds to an image with coordinates in the original space of I'_N, but in the subspace generated by the principal components. If I'_N is well described by the components, the I'_N and I'_{rep} should be very similar and the angle θ formed between



(a) 1st component (b) 10th component (c) 34th component

Fig. 5. Examples of principal components corresponding to a dataset composed of the synthetic dataset for the ball



(a) Position error for the ball for different number of of components

(b) Belief estimation for the ball example

Fig. 6. Impact of the number of principal components used in detection and belief estimation in the image case

the two vectors should be very close to 1. We use the cosine of the angle θ, eq. (7), as a proxy for our belief in the existence of the object in the image.

$$I'_{rep} = V^T I_r = V^T V I'_N \tag{6}$$

$$\cos\left(\theta_{object}(I_N)\right) = \frac{I'_N \cdot I'_{rep}}{\|I'_N\| \|I'_{rep}\|} \tag{7}$$

Due to the number of multiplications required, the belief estimation can be very time - consuming. To reduce the computational load, we change equation eq. (6) and choose carefully the number of principal components to be used at this stage.

We change equation eq. (6) by, instead of re-projecting the whole image back into the regular image coordinates, re-projecting only the pixels that belong to the object hypothesis. We are not comparing all the pixels of I'_N and I'_{rep}, but only the fraction that should had been correctly reconstructed.

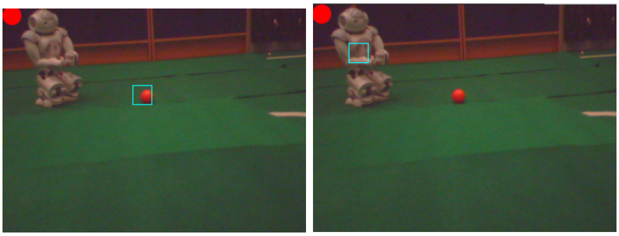
Furthermore, we note that the number of principal components used at this stage can differ from the number of components used for regression. To determine the minimum number of components required for belief estimation, we compute the mean and standard deviation of belief in the independent dataset previously used for estimating the position error. For the ball example we present the results in Figure 6(b). The mean belief increases with the inclusion of more components, but becomes approximately constant after the inclusion of at least 10 principal components. Thus, by fixing the number of components to 10 we retain most of the information needed to assert the presence of the object. The number of components to be used depends on the object level of detail. While for a ball, just 10 components are good enough for describing the object, the same is not true for the robot. Albeit larger than the ball, the robot has a more detailed shape and thus required more high order components to be represented.

The selection of the decision threshold depends on the number of components used to estimate the belief. Using 10 components and considering only the case of ball detection a threshold of 0.65 takes into account all the examples inside the error bars as exemplified in the plot in Figure 6(b).

6 Experimental Results

In this paper we analyze the results of ODR using elements of the RoboCup environment, such as the ball and other robots. In particular we evaluate the algorithm in three different dimensions. First the capacity of object detection knowing that the object is in the image. Second, the capacity of identifying if the object is in the image or not. Third the time efficiency of the algorithm. We separate the problem of object detection accuracy from the problem of belief estimation in our result presentation. This is motivated by the possibility of changing either one of these parts of the algorithm without affecting the other. Also, since they solve different problems, one provides a position while the other classifies an image, we use different metrics to express results.

The capacity for detecting the object is measured by the percentage of correctly detected objects over the total number of objects that were given to identify. I.e., over all the set of testing images, we only consider those that contained the object. For the ball example, ODR detects correctly 92% of all the ball examples in the dataset. As for to the identification of the second robot, ODR identifies the position of 87.5% of all the robots. These results include changes of pose and occlusion. Examples of detection are provided in Figure 7



(a) Ball detection (b) Robot detection

Fig. 7. Examples of ball and robot detection

The capacity for classifying each image according to whether the object is present or not given a belief score is illustrated by the three usual metrics used to evaluate classification algorithms: the precision, the recall and the average precision. The precision evaluates the capacity to differentiate between two classes and is given by the percentage of true positives over the total number of positives. The recall evaluates the capacity of identifying the objects from the desired class and is given by the percentage of true positives over the total number of true examples in the dataset. Both precision and recall metrics depend on the classification criterion. In ODR, the criterion corresponds to a threshold in the belief. Only images with belief higher than the threshold are classified as having the object. The average precision attempts to reconcile precision and recall by considering both metrics at different threshold values. The average precision itself is computed by measuring the area under a precision recall curve, where each point in the curve corresponds to a precision and a recall computed at the same threshold.

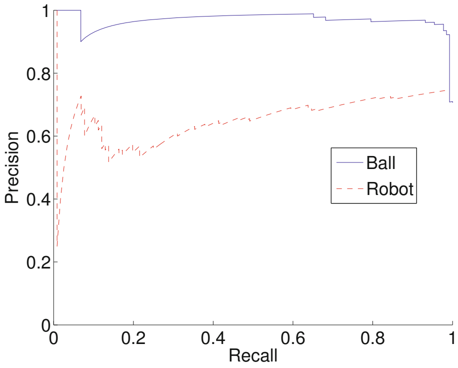


Fig. 8. Precision recall curves for both objects

In the graphic of Figure 8 we present the precision recall curves for the ball and robot. The average precision for the ball is 0.96 while for the robot is 0.67. We compare the performance of both time and accuracy of ODR with respect to other methods. In particular we compare them against CMVision, which was used as the ground truth in training. For the comparisons, we use both methods offline, running each one 1000 times per frame in a Pentium 4 at 3.20GHz. The processing for CMVision includes thresholding, blob formation and ball detection, while ODR includes only thresholding and ball detection.

Results for processing time are presented in Figure 9. ODR achieves an average processing time lower than that obtained by CMVision in both cases.

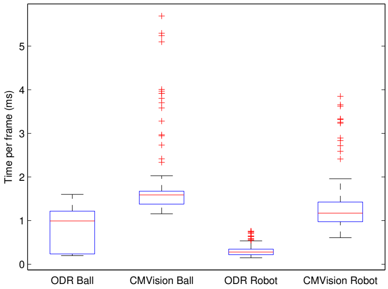


Fig. 9. Comparison of processing times using both ODR and CMVision

7 Conclusions

In this paper, we have contributed a novel object detection by regression approach. Results, which were obtained for the RoboCup case study, show that our learning of offline linear object models for object detection and position provides a fast and robust online performance. ODR is best applicable in general, if the environment and the specific objects to detect and if object presence hypotheses can be pre-computed.

The context of the RoboCup robot soccer is particularly adequate for ODR, as the field and the objects are known ahead, and processing of their color provides a simple prior for the object hypotheses. Our learning models effectively capture the online object images, even given the extreme variations of the images in real game situations. The learned models by ODR are based on a small number of pixels, leading therefore to fast online image processing. Experimental results show that such sampling did not adversely affect position detection precision, which is close to par with the state of the art algorithms when the object is present in the image. Furthermore ODR is significantly more efficient.

ODR is also able to identify if the object is absent in the image based on the learned models. The number of principal components used by ODR affects the ability to reconstruct the image and hence, to identify the absence of the object. The higher the number of principal components, the more precisely the absence of the object is detected, but also the higher the computational cost. In our experiments, we favored a low computational cost, and ODR was still able to successfully identify the absence of the objects with the needed accuracy. In general, setting the tradeoff between the number of principal components used online and the computational cost will depend on the needs of the domain.

References

1. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer (2007)
2. Bruce, J., Balch, T., Veloso, M.: Fast and Inexpensive Color Segmentation for Interactive Robots. In: IROS 2000 (2000)
3. Felzenszwalb, P., Mcallester, D., Ramanan, D.: A discriminatively trained, multiscale, deformable part model. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2008 (2008)
4. Hashemi, E., MRL Team: MRL Team Description Standard Platform League (2010)
5. Hester, T., Quinlan, M., et al.: TT-UT Austin Villa 2009: Naos Across Texas (2009)
6. Nieuwenhuisen, M., Behnke, S.: NimbRo SPL 2010 Team Description (2010)
7. Turk, M.A., Pentland, A.P.: Face recognition using eigenfaces. In: ICCV 1991 (1991)
8. Röfer, T.: B-Human Team: B-Human Team Report and Code Release (2010), `http://www.b-human.de/en/publications`