In Proceedings of 8th Workshop on Humanoid Soccer Robots
13th IEEE-RAS International Conference on Humanoid Robots (Humanoids), Atlanta, GA, 2013.

Humanoids Learning who are Teammates and who are Opponents

Vladimir Estivill-Castro
School of Information and
Communication Technology
Griffith University
Nathan, Brisbane 4111
Australia
Email: v.estivill-castro@griffith.edu.au

Jordi Radev
Departament de Tecnologies de la Informació i
les Comunicacions
Universitat Pompeu Fabra
Barcelona 08018
Spain
Email: jordi.radev@gmail.com

***Abstract*—RoboCup aims to progressively advance the research challenges in robotics by presenting a soccer tournament played by robots. One of the main aspects that shall fade is the need for pre-defined color coding; specially on what humanoids wear. However, most colors have rarely been modified over the years of the competition. The field remains green with white lines on it, the colors of the teams are predefined and the ball continues to have no pattern and is only orange. If colors of objects are no longer predetermined, a new parameter is needed for object recognition, and we propose to use shapes. However, computer-vision techniques for shape recognition are much more CPU-intensive than color recognition and perhaps they are unaffordable during the game. Our proposal here consists of identifying objects by their shape and extracting the colors from within these shapes. We identify the objects on the basis of their shape and learn autonomously the colors of the recognized objects. The chosen shape recognition algorithm is the Histogram of Oriented Gradients. This method has been proved to be capable to recognize complex objects and is widely used for pedestrian recognition. The complex objects we recognize are other humanoid robots (in particular Naos), similar in complexity for their anthropomorphic shape but less variable than human beings. This enables full learning of the environment colors in less than 1 minute on board of a Nao.**

I. Introduction

Once we place a humanoid robot in an environment, there would be more to come, and the natural question is, weather one humanoid robot would recognize the other. This is already the scenario at RoboCup in the humanoid leagues. The teams are composed of several robots; however, the competition rules still allow for recognition of teammates and opponents on the pre-defined color-coding specified by the competition's rules. This is significantly different from the spirit of the research at RoboCup where the ultimate aim is to enable robots to play alongside humans. This paper aims at revising the need for a fixed color-coded environment with predefined colors for objects like the ball, the field, the lines, the goals and also the teams' uniforms. The games played by humans do define that the teams wear different colors, but long are gone the days of black-and-white TV that demanded a dark top for one team and a clear top for the other; today's human soccer has even allowed the referee to use several colours, the ball can have colors and patterns and although most fields use white lines, white post and green grass, some games in artificial turf make surface color quite different. Playing at night and indoor soccer (fut-sal) allows even more variations (for the posts and the playing surface).

We study the Standard Platform League that uses Alderbaran's Nao. We suggest that the results reported in this paper are sufficient to remove the color constraints and enable

1) different team shirts (that is, teams should be able to wear a color of their choice),
2) allow different types of fields (different colors for lines and playing surface, different colors for balls, as well as goal posts).

In less than 1 minute of preparation, our automatic procedure can capture (learn) the environment's colors. Thus, teams can be introduced to the environment and their opponents before a match and using our approach, efficiently construct classifiers to segment the images the robot's will face in the up-coming match. Naturally, the technique could also run at occasional non-CPU intensive moments during a match (in the spirit of a background process amortizing its cost over time) and improve the learning, but we leave this aspect for further work.

We achieve this by using shape-based recognition. We can place the robot and the objects (mainly the ball, teammates and opponents) in introductory positions. We work in gray-scale images and progressively our approach learns the colors of the environment. The contrast of the surface near the central circle enables recognition of the color of the field-lines and the field itself, while the round shape enables the recognition of the ball. Techniques for this level are well-known in the computer vision literature, we will be using the Hough transform and fast clustering methods like k-means. Still, careful tailoring is used to actually obtain reliable samples from which to learn. But to recognize other humanoid robots we need to find them efficiently. the Histogram of Oriented Gradients (HOG) [5] is a solid technique widely used to recognize people in environments, and we demonstrate here it is effective in recognizing humanoids. The HOG uses as a classifiers Support Vector Machines. Once robots are found, we need to recognize their torso, in order to find samples of the team-shirt color. In our approach we will also be using machine learning algorithms. Nevertheless, we achieve the complete learning of a new color-coded environment completely on board of the Nao in less than 1 minute.

II. THE CLASSIFIERS

The type of classifiers that we will learn are very efficient. In the literature of machine learning [1] they are known as decision lists. Decision lists can be learned using the PART algorithm [1]. We introduce them here briefly as they constitute the aim of the process of discovering the environment's colors and encode the representation of what is learned. Typically, the color attribute for a pixel in image is provided by three components labeled Y, U and V in a range [0,`precision`] (which is typically [0,255] when these components are one byte). The basic components of a decision list classifiers are characteristic functions defined by three intervals and a color class; one interval for each component. This elementary classifier can be encoded by its 3 intervals $[\min_Y, \max_Y]$, $[\min_U, \max_U]$, $[\min_V, \max_V]$ (or alternatively, by a set of 6 values $\{\min_Y, \max_Y, \min_U, \max_U, \min_V, \max_V\}$) and the color class identifier. The semantics of the elementary classifier is simple. If a pixel $p = (p_Y, p_U, p_V)$ satisfies $(\min_Y \leq p_Y \leq \max_Y) \wedge \min_U \leq p_U \leq \max_U \wedge (\min_V \leq p_V \leq \max_V)$, then the pixel p is assigned the color id of this elementary classifier; otherwise is left as unknown.

The 3 intervals of such an elementary classifier can also be encoded very succinctly in a 3 Boolean arrays, each array being the characteristic function that fires the rule. The Y array is such that $Y[i] = (\min_Y \leq i \leq \max_Y)$, for $i = 0, \ldots,$ `precision`. Similarly, $U[i] = (\min_U \leq i \leq \max_U)$, for $i = 0, \ldots,$ `precision` and $V[i] = (\min_V \leq i \leq \max_V)$, for $i = 0, \ldots,$ `precision`.

In a programming language like `C++`, then the elementary rule knows the color of the pixel $p = (p_Y, p_U, p_V)$ when `Y[p[y]] & U[p[u]] & V[p[v]]` evaluates to true. Note that this is the bit-wise **and** operation.

Decision lists are lists of these elementary classifiers. On a given pixel, the first elementary classifier is tested, and if it declares a color for the pixel, the ensemble of classifiers takes such decision as the color of the pixel; otherwise, the next rule is evaluated down the list. Once an elementary classifier determines it knows the color, the ensemble takes this as the decision. The ensemble may decide it can not determine a color if the list is exhausted without a decision. Although this may seem a complicated algorithm, it can be implemented in an extremely efficient way. A decision list of depth 32 elementary rules can be implemented in 3 integer arrays (with indexes in [0,`precision`] if the memory word for an integer is 32 bits) and an ensemble of 64 elementary rules is now quite feasible (with the integer representations of 64 bits). The three arrays are also `Y`, `U` and `V`. However, the characteristic function described early (the Boolean array) for one elementary rule is encoded in one bit position. The first rule in the ensemble goes in the least significant bit, the second rule in the next bit, and so on. The following code implement the algorithm in `C++`.

```
color= Y[p[y]] & U[p[u]] & V[p[v]];
color_id=0;
while (0 == color & 1)
     { color>>=1; color_id++;};
```

Decision list achieve very succinct representation and their classification accuracy is equal or better than decision trees, neural networks or support vector machines. They do not require integer arithmetic, multiplication or any other CPU-costly operation. They are several orders of magnitude faster at color-segmenting an image [2].

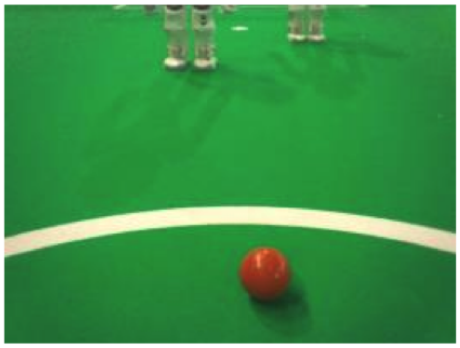


Fig. 1. A typical image used to learn the color of the surface of play, the color of line markings and the color of the ball.

Learning a decision list is a classical supervised learning setting, where we are to supply a large set of pixels for which the color is already known. The decision list learning algorithm can learn a two-class scenario (for example, it is a yellow pixel versus it is not a yellow pixel) or a classifier for several classes (lets say the colors of the environment). In fact, it is quite feasible to learn several classifiers, a generic multi-class classifier that is globally used during the match to analyze the image and find regions of interest. While later, specialized classifiers for the color of the ball are used in the region of the ball. The vision pipeline can select also a classifier based on context, for when the ball is close as opposed to when it is far. Suffice to say that constructing these classifiers becomes the realm of supervised learning. Thus the challenge is to find supervised examples, without human involvement acting as the supervisor.

III. THE ANALYSIS OF THE ENVIRONMENT

The first step of the over-all behavior that learns all the colors of the soccer match in the Standard Platform League is what we call the environment analysis (the color of the playing surface, the color of the line marking on the playing surface and the color of the ball).

The robot does need assistance in that is to be placed near inside the center circle or a position near line markings (however, we can see that actually this is just for the convenience of doing everything in less than 1-minute). The same could be achieved from any position in the side of the field, as the first two colors learned are the playing surface colors and the lines. Once this colors are learned, the robot can find and navigate to the center circle or any other place for later meeting other humanoids or have a good position where seeing most of the goal is to be expected. A typical image used to learn the line markings, the color of the playing surface and the color of the ball is shown in Fig. 1

Also, for the process to complete in one minute, a ball is placed close to the robot. The robot executes a behavior by which it looks down and grabs images. The images are processed in two phases as follows. In the first phase, images are converted to gray-scale, they pass a smoothing filter

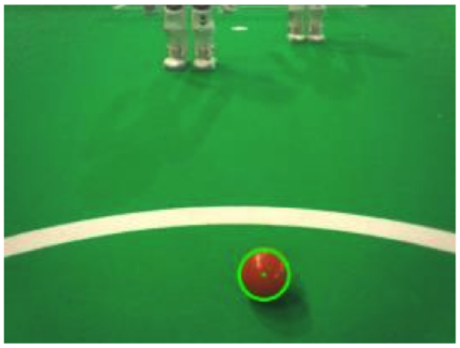
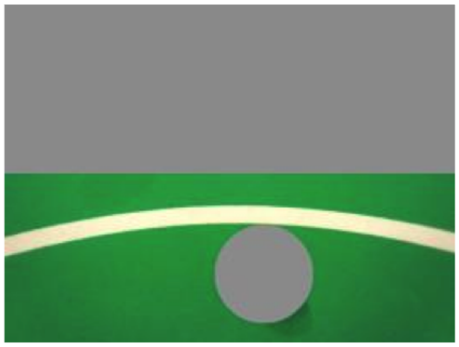


Fig. 2. A typical result of the Hough transform for finding circles of size similar to an expected nearby ball.

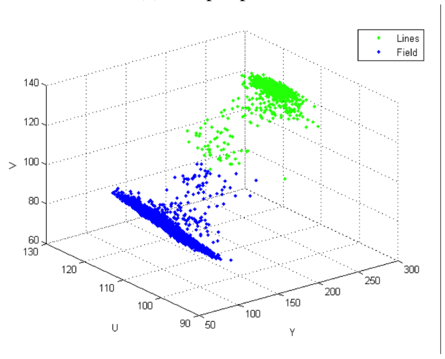
(actually the method works even if the playing surface has texture, like artificial grass or carpet with spots of several colors). A Hough transform to find circles (of the size of balls) is applied followed by circle clustering. Fig. 2 shows the typical result of the Hough transform. The clustering is on the circles found across several frames. Typically, sightings of a ball create circles very close to each other (occasionally a ball is missed and occasionally a false positive will occur). Nevertheless, once several consistent (very similar size and placement) circles and radius are found, this is recognized as average position and average size of the ball. For extracting the ball pixels and avoid pixels that are not belonging to the ball in case of a bigger detected radius than the real one a clustering procedure is ran starting from the center of the circle. On each iteration of the procedure an average of the components of the cluster pixels is calculated and all the neighbors of the already clustered pixels that are closer than a predefined constant are added to the cluster. The value of this predefined constant must be carefully chosen because in case it is too small only few pixels will be taken and many pixels that belong to the ball would be thrown away, on the other hand if the value is too large the cluster would expand around all the ball taking all the reflections as ball colors, which could ruin the accuracy of the classifier. This procedure keeps iterating until it converges, i.e. there is an iteration where no pixel is added to the cluster, then the first phase is over.

In the second phase, everything above a horizon calculation (about the top half of the image) is ignored and pixels are separated into two classes. Those that belong to a circle slightly bigger than the expected size of the ball versus pixels outside the circle. Those pixels outside the circle are clustered by their color into two clusters by k-means, since these are laying-surface pixels or line-marking pixels. They then become training instances for the decision-list classifier about the line-markings and the playing surface. Fig. 3 shows the input pixels into the clustering with k-means, and the result of k-means with $k = 2$ on these pixels (which separates mainly into training samples for line-markings and training samples for the color of the surface of play).

Fig. 4 shows the flow diagram for the environment analysis. The procedure starts by enlarging the cluster of pixels of the color of interest and we present it here as enlarging the blob/patch of a color to something close to its boundary is also used to recognize the team's colors on the shirts they



(a) Sample pixels



(b) Clustering result

Fig. 3. The result of clustering into two clusters pixels below the horizon that are not in the circle of the ball (plus some margin).

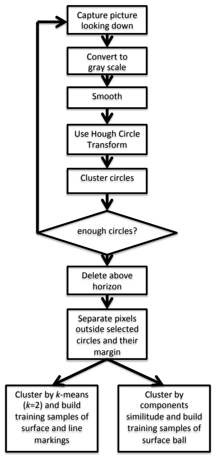

Capture picture looking down
Convert to gray scale
Smooth
Use Hough Circle Transform
Cluster circles
enough circles?
Delete above horizon
Separate pixels outside selected circles and their margin
Cluster by k-means (k=2) and build training samples of surface and line markings
Cluster by components similitude and build training samples of surface ball

Fig. 4. The process of learning the SPL playing environment.

wear. Fig. 5 shows the flow diagram of the procedure to grow a blob of a certain color. Figure 6 Illustrates the progress of the algorithm that enlarges the set of samples to be forwarded for training as pixels of the color patch for the ball.

IV. Learning the colors other humanoids wear

We now describe the procedure to find other humanoids in the field, and then use the findings to obtain samples of their team shirt in order to construct decision list classifiers for recognizing teammates and opponents. Again, the process here is efficient, but not efficient enough to run for every frame during game play. The purpose of learning the team shirts

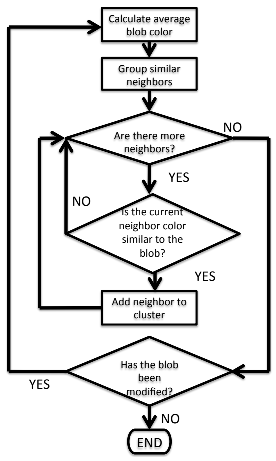
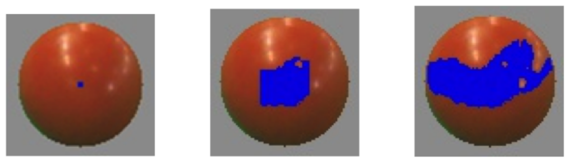

Calculate average blob color
Group similar neighbors
Are there more neighbors?
NO
YES
NO
Is the current neighbor color similar to the blob?
YES
Add neighbor to cluster
Has the blob been modified?
YES
NO
END

Fig. 5. The region-growing process that enlarges the number of samples of a color patch or blob.



(a) One iteration, 9 pixels (b) 13 iterations, 579 pixels (c) 76 iterations, 2,546 pixels

Fig. 6. The enlarging of the set of pixels that are recognized as pixels of the ball color.

colors is that, from there on the recognition of teammates and opponents during the match can be performed every frame using color-segmentation. But again, if desired, the process described here can be run on spare CPU cycles during the game to improve the classifiers.

For this part two Naos are placed in front of the Nao that is learning, one closer than the other. The closer Nao can be selected as a player of the same team and the farther Nao will then be recognized as the opponent.

As we announced in the introduction, the first fundamental technique used here is the Histogram of Gradients (HOG) which uses support vector machines as the classifier after features are extracted on images. The Histogram of Gradients generates feature descriptors (similar to the popular SIFT algorithm); however, the HOG technique has been found to be more reliable in identifying human (people) in images [3]. In fact, the HOG technique has been used on board of robots to follow a human [4]. Our work here demonstrates that the HOG technique is also very effective to locate where in an image is a humanoid robot. The HOG creates a training set for the support vector machine, and the support vector machine performs the actual identification of a box that frames the object of interest in an image.

The fundamental steps of HOG are as follows.

1) The image is transformed to gray-scale.
2) The image is divided into cells.
3) Gradients for each direction within the cells are quantified
4) Cells are grouped into blocks

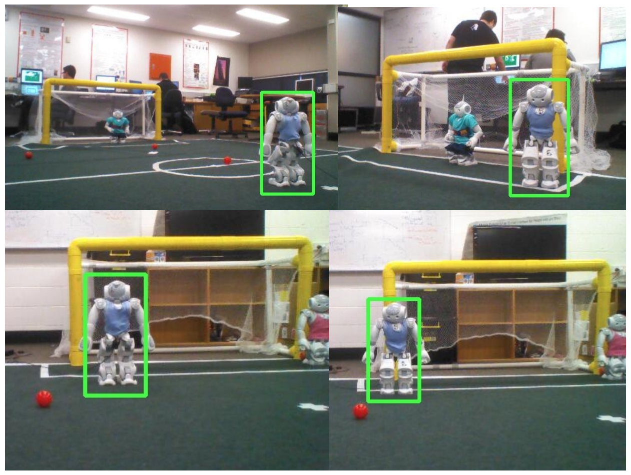


Fig. 7. HOG Technique localizes the humanoid robot on the image by its shape.

5) A window (box) is defined for detection.

Using the full resolution of the Nao camera (1280×960 pixels) the process took around 11 seconds for each image processing, but we found that changing the resolution to VGA (640×480 pixels) the time was improved (down to 3 seconds) and the accuracy remained the same. The best parameters we found to achieve the recognition of Nao robots from another Nao are as follows.

1) The window size was set to 64×128 pixels.
2) The block size was set to 16×16 pixels.
3) The block stride was set to 8 pixels in each dimension (left to right of the image and top to down).
4) The cell size was set to 8×8 pixels.
5) The number of bins in the histogram was set to 9.

Thus, the feature vector dimension of a training sample is as follows.

- There are 4 positions that a block of 16 pixels fits horizontally along the windows of 64 pixels, and the stride puts another 3 blocks (each of these in between the others).
- There are 8 positions that a block of 16 pixels fits vertically along the windows of 128 pixels, and the stride puts another 7 blocks (each of these in between the others).
- There are 4 cells of size 8×8 in each block of size 16×16.
- The gradients are placed in a histogram of 9 bins.

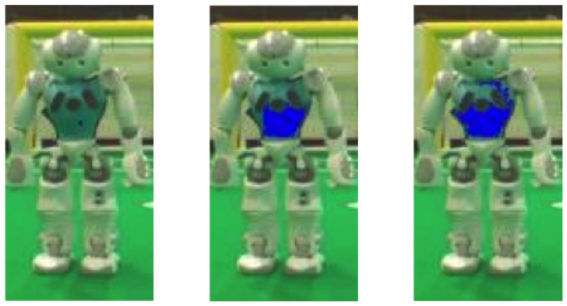
In total, the dimension is

$$(4+3) \times (8+7) \times (4 \times 9) = 3,780.$$

To create a training set for the shape-based detection of Naos using HOG and support vector machines we used 512 as positive examples images where we presented a 64×128 box of pixels framing a Nao. However, these were in fact 256 images as we doubled the samples by flipping the images horizontally.

TABLE I. RESULTS OF 6-FOLD CROSS VALIDATION TO ASSES ACCURACY OF HUMANOID ROBOT DETECTION.

True positives	452
False positives	3
True negatives	2769
False negatives	60
Average precision	99.33%
Average recall	88.26%

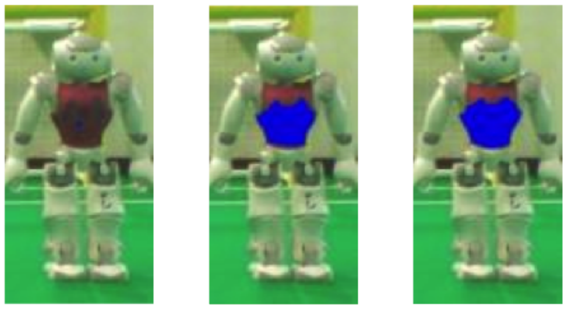


(a) One iteration, 9 pixels (b) 18 iteration, 995 pixels (c) 53 iterations, 137 pixels

Fig. 8. The progression of adding pixels that become training samples for the color of the uniform of nearest robot.

We also presented 2,772 framing boxes as negative examples taken randomly from images that were flipped vertically in order to avoid taking a Nao image as negative example by mistake. Once the training set is used to build a classifier the same training set is tested with the classifier and all the false positives are added as negative instances, then a new classifier is trained and this one is the one used for detecting Naos. This is the same procedure followed by the original HOG paper [5]. This setting achieved high accuracy in recognizing other Naos.

We evaluated the accuracy of classification in over 3,000 images by k-fold cross validation with $k = 6$. Table I shows that our accuracy is above 99% and recall is 9 out of 10. Once both Naos are discovered in the image, the procedure to recognize a blob of a color to create the sample set for constructing decision list classifiers is as described in the previous section. Fig. 8 shows the collection of pixels for the shirt of the nearest robot. Analogously, Fig. 9 shows the gathering of pixels for the far robot.



(a) One iteration, 9 pixels (b) 18 iteration, 885 pixels (c) 108 iterations, 2,679 pixels

Fig. 9. The progression of gathering pixels that will become samples for learning the color of the uniform of the far robot.

TABLE II. RESULTS OF 6-FOLD CROSS VALIDATION TO ASSES ACCURACY OF GOAL CORNER DETECTION.

True positives	187
False positives	61
True negatives	64,406
False negatives	41
Average precision	75.55%
Average recall	82.57%

V. LEARNING THE COLOR OF GOAL-POSTS.

The final procedure is the recognition of goals in order to learn the color of the posts that make the goal. While there are several algorithms to recognize lines (RANSACK, or the Hough Transform [3, Chpater 4]), we found that it was in fact better to recognize the corner shapes in an image that happen when a post meets the crossbar of a goal. And also, despite that there are algorithms dedicate to find corners (like the Harris detector [3, Chaper 4]) we found that the HOG technique again was more effective (in the trade-off of accuracy vs learning time). However, in this case, the shapes to be located in the image are less complex (we are after the inverse-L-shape of the post meeting the cross-bar, and not a kneeling, walking or standing Nao). Thus, the parameters for the HOG are simpler.

1) The framing-window size is 16×16.
2) The block size is 8×8.
3) The block stride is 4 in the horizontal direction and also in the vertical direction.
4) The cell size 4×4.
5) The number of bins for the histogram is 9.

Thus, the feature vectors in this case are much smaller. The dimension is $(2 + 1) \times (2 + 1) \times 4 \times 9 = 324$.

We train the corresponding support vector machine with 228 positive examples provided (from only 114 images as each is flipped horizontally to create a symmetric positive example). The number of negative examples is 64,467 and they have been picked by framing randomly parts of images and flipping them vertically to avoid getting a goal corner by mistake. Table II shows that our accuracy is above 75% and recall is 8 out of 10.

Again, the blog/patch forming algorithm of Section III is used to enlarge the goal-post pixel samples to use in the training sets of decision-list classifiers.

VI. INTEGRATION AND IMPLEMENTATION

The implementation of the complete procedure is culminated by the construction of a decision-list classifier that recognizes all the colors for which sample pixels have been obtained. We highlight that the actual execution of the PART-learning algorithm [6] occurs on board of the Nao[1].

However, some house-keeping needs to be performed before the decision-list classifier is invoked. This includes merging all the samples of the different colors into one large training set, and most importantly, balancing the classes. It is rather common that several thousands of pixels labeled as playing surface color are obtained, while only a few hundred

[1]We used Alderbaran's provision of a virtual machine, installed Java and used the Weka package to place the learning procedure on the Nao itself

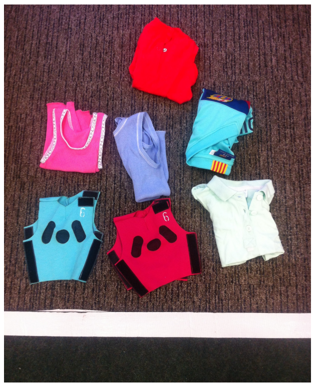


Fig. 10. A set of different shirts with different colors which were learned successfully. The two surfaces in this image were also tested as playing surfaces and the learning was successful.

samples are obtained for the post, and even less for important objects like the ball or the jersey's of the teammates and the opponents. We used the Weka algorithm for class balancing. The last step is to actually put in place the decision list classifier into the vision module and let the robot use it in actual play (chasing the ball, localizing, and recognizing teammates and opponents).

The results of this process is shown on a video (http://youtu.be/DEMaRopZSrQ) recorded at the RoboCup-2013 venue and the colors learned are the colors of the competition. Robots are located at different distances, and in under a minute, colors are learned for all objects in order to recognize them using color segmentation by decision lists.

VII. Conclusion

The results are remarkable from a qualitative perspective therefore it has been proved to be possible to learn the colors related to a soccer match in less than a minute. Note that earlier, the HOG was considered infeasible for the Nao robots [7]; thus, the approach took advantage of the colors team-marker belt, requiring previous knowledge of team colors [7]. Similarly, knowledge that lines and robots are white is used to find lines, and to find robots in those white patches that do not pass the test as lines [9]. The main classifier in such proposal [9] is artificial neural networks. One virtue of this method is that estimations of distance to the other robot are obtained [9]. We do not require the lines to be white, but we do learn from images with a Nao.

We tested the environment analysis phase on many playing surfaces, with different balls and with different team shirts and the method proved to be effective. Fig. 10 shows different shirts/jerseys and two other playing surfaces that were successfully learned. Fig. 11 shows different colored balls that were also successfully learned.

The smoothing contributed remarkably well at detecting separately the colors that constitute the surface from the color of the lines and the ball. It even achieved good results on non-homogeneous floors and helped the Hough transform to detect the ball more accurately.

The recognition of other Naos is also very robust, detecting

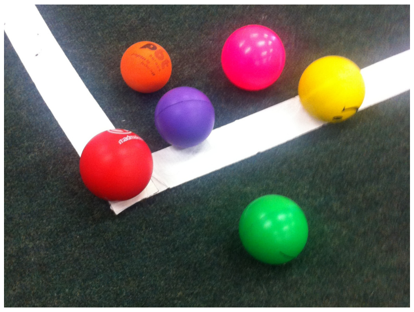


Fig. 11. A set of different balls with different colors which were learned successfully.

Naos in different position, like kneeling or walking as the training set included Naos in several different positions.

The process has proved to be capable to be executed in less than a minute with satisfactory results being executed entirely on the Nao autonomously.

We are aware that a process to avoid calibration and use a shape-vision approach to vision during the game exists [8].

Acknowledgment

The authors would like to thank all members of the MiPal RoboCup Team and the anonymous referees for many helpful remarks.

References

[1] I. Witten and E. Frank, *Data Mining — Practical Machine Learning Tools and Technologies with JAVA implementations*. San Mateo, CA: Morgan Kaufmann Publishers, 2000.

[2] V. Estivill-Castro and N. Lovell, "Improved object recognition - the robocup 4-legged league," in *Intelligent Data Engineering and Automated Learning, 4th International Conference, IDEAL 2003*, ser. Lecture Notes in Computer Science, J. Liu, Y. ming Cheung, and H. Yin, Eds., vol. 2690. Hong Kong, China: Springer, March 21-23 2003, pp. 1123–1130.

[3] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to Autonomous Mobile Robots*, 2nd ed. Cambridge, MA: MIT Press, 2011.

[4] W. Pairo, J. Ruiz-del Solar, R. Verschae, M. Correa, and P. Loncomilla, "Person following by mobile robots: analysis of visual and range traking methods and technologies," in *Proceedings of 17th RoboCup International Symposium*, ser. Lecture Notes in Computer Science. Eindhoven, Netherlands: Springer, June 2013, to appear.

[5] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2005)*. San Diego, CA, USA: IEEE Computer Society, 20-26 June 2005, pp. 886–893.

[6] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: an update," *SIGKDD Explorations*, vol. 11, no. 1, pp. 10–18, 2009.

[7] A. Fabisch, T. Laue, and T. Röfer, "Robot recognition and modeling in the in the roboCup standard plarform league," in *5th Workshop on Humanoid Soccer Robots at Humanoids*, 2010, http://www.humanoidsoccer.org/ws10/program.html.

[8] T. Reinhardt, "Kalibrierungsfreie bildverarbeitungsalgorithmen zur echtzeitfähigen objekterkennung im roboterfußball," 2011, in German http://thomas-reinhardt.de/Vision/.

[9] S. Metzler, M. Nieuwenhuisen, and S. Behnke, "Learning visual obstacle detection using color histogram features," in *RoboCup 2011: Robot Soccer World Cup XV —papers from the 15th Annual RoboCup International Symposium*, ser. Lecture Notes in Computer Science, T. Röfer, N. M. Mayer, J. Savage, and U. Saranli, Eds., vol. 7416. Istanbul, Turkey: Springer, July] 2011, pp. 149–161.