

Acquiring Observation Models through Reverse Plan Monitoring

Sonia Chernova, Elisabeth Crawford, and Manuela Veloso

Computer Science Department,
Carnegie Mellon University,
Pittsburgh PA 15213, USA

Abstract. We present a general-purpose framework for updating a robot’s observation model within the context of planning and execution. Traditional plan execution relies on monitoring plan step transitions through accurate state observations obtained from sensory data. In order to gather meaningful state data from sensors, tedious and time-consuming calibration methods are often required. To address this problem we introduce *Reverse Monitoring*, a process of learning an observation model through the use of plans composed of scripted actions. The automatically acquired observation models allow the robot to adapt to changes in the environment and robustly execute arbitrary plans. We have fully implemented the method in our AIBO robots, and our empirical results demonstrate its effectiveness.

1 Introduction

In the traditional planning and execution scenario (e.g., [7]), a plan consists of a sequence of actions defined as a set of preconditions and a set of effects. When executing a plan action, the robot first checks if the preconditions of the step are satisfied in the current state. For instance, before attempting to execute the plan action “open door A,” the robot needs to be able to detect the action precondition “in front of door A.” Once the action is executed the robot can then verify the effect of the action (“door A open”) before deciding if the preconditions for the next action are met. These preconditions and effects are perceived through sensory data. Often the perceptions that the robot needs to make are complicated and thus the sensors must be very accurately calibrated.

In order to gather meaningful data from sensors, tedious and time-consuming calibration methods are often required. Calibration often needs to be performed for each sensor individually, and, in some cases, changes to the environment can make recalibration necessary. Traditionally, calibration processes are executed manually by a human expert until the desired behavior is achieved. We believe that many calibration tasks can be automated to a large degree, greatly reducing the level of human involvement without sacrificing the quality of the results. We propose a method to approach this problem that combines elements of calibration and plan execution.

An integrated robot planning and execution framework requires accurate state detection. However, as all actions have expected effects associated with them, we can also consider blindly executing a sequence of plan steps and then letting the robot associate the perceived state with the expected state. In this paper, we contribute a novel way of approaching plan execution - we *reverse* the perception role. Instead of expecting the robot to be able to accurately perceive the preconditions of actions, our approach, Reverse Monitoring, assumes that the robot knows the effects of the actions, can detect triggers of actions at a reduced level of detail, and can then associate its full perceived state with the effect of the actions. As we show in this paper, the robot is capable of refining its perception models to a higher level of detail through the execution of scripted plans. In this way, robots can autonomously acquire the accurate models they need to execute plans that require detailed sensing.

We demonstrate the effectiveness of our approach using the vision calibration system developed for the Sony AIBO robot. A typical vision calibration sequence on an AIBO robot requires a human to place the robot at various points in the domain to collect images of landmarks for calibration. The user must then hand-label each of the pixels in the images to classify it into one of the discrete color classes. Our proposed method allows the robot to collect and classify images with very little human assistance. This is accomplished by providing the robot with a plan that allows it to navigate by relying on sensing data in a reduced dimensionality space. As the robot proceeds with the plan and gathers information, its observation model is enriched, allowing the robot to sense in higher dimensions. When the execution of the plan is completed, the robot's sensors have been fully calibrated and it is then able to execute plans that require more detailed sensing.

We are not aware of any previous work combining the elements of plan execution and observation model refinement in a formal framework. Planning literature dealing with robots in the real world generally assumes a working sensor model and focuses mainly on issues of noisy sensors and partially observed states [4, 5, 7]. A large body of work relating to autonomous sensor calibration also exists, however these methods tend to focus on domain-specific or sensor-specific solutions [6, 9, 11, 12, 14]. We propose a general-purpose framework for updating a robot's observation model in the context of planning that is independent of the particular system or domain being used. The following sections present our approach in greater detail and present results demonstrating its effectiveness when applied to an existing robotic system.

2 Observation Models from Reverse Monitoring

2.1 Monitoring

A plan is a series of actions to be carried out in sequence. In classical planning actions have no duration; their effects are instantaneous and therefore the termination point of each action is clear. More realistic planning domains require the use of actions with durations. Durative actions typically represent a period

of time where certain conditions are true at the beginning and at the end of the action execution. The action terminates when the execution time runs out, so the termination point is easy to detect.

In the real world, the execution of a plan is dependent upon the ability to observe the state and detect state changes. Although action duration can be defined in terms of a specific time interval, most complex actions require some form of state monitoring in order to determine when the action is completed. Termination conditions are thus defined in terms of specific state features, such that the current state features must match the state features that symbolize the end of the action. We will refer to the process of tracking action transitions through the detection of state features as *Monitoring*.

One of the difficulties of this approach is that it relies heavily on accurate state detection, which can be very challenging when dealing with real-world domains. Robots acting in the real world must accurately detect a large number of states through a complex set of features. Common approaches to feature detection include thresholding, classification, and a number of other methods. Regardless of the feature detection method used, if a robot encounters a new situation or the environment changes causing the feature detection to fail in a previously known state, the robot must update its observation model in order to detect states correctly. For example, a robot that recognizes a particular door by the feature “Door is GREEN” will be unable to find the door if it is repainted in a different color. The robot’s observation model must be updated to use the new color for the state detection to work accurately.

Observation models of robots acting in real world domains frequently need to be updated since features of the environment often change without the robot’s knowledge. Corrections of the robot’s model are frequently done through manual calibration, which is often a tedious and repetitive task. To address this problem we introduce *Reverse Monitoring*, a process of learning an observation model through the use of plans composed of scripted actions.

2.2 Reverse Monitoring

We defined Monitoring as the process in plan execution of detecting state changes in order to determine whether the current action has been completed. Monitoring requires the robot to have a rich observation model in order to accurately detect the state. Reverse Monitoring is the process of updating the observation model through the execution of a plan consisting only of basic actions, which we refer to as *scripted actions*.

Scripted actions are actions with termination conditions that depend only on the state of the robot and that are independent of the state of the rest of the world. Examples are actions whose termination conditions are defined in terms of parameters such as time, number of steps, or the press of a button. These conditions do not require the robot to sense anything in the environment.

In Reverse Monitoring, a robot is provided with a plan and a specified state, which is true upon the termination of that plan. The process begins by executing a plan composed of a sequence of scripted actions. Upon termination, the robot

observes its state. Unlike normal Monitoring where the robot uses its observation model to detect various features and classify the state, in Reverse Monitoring the robot extracts new features from the environment and updates its observation model. For example, the previously mentioned robot that was looking for a green door, would execute a plan to navigate to the door and then observe the state, updating its model of the door to the new color.

People often rely on a process similar to the one described above to navigate in unfamiliar environments. One example is the task of navigating from a hotel to an unfamiliar conference building. The person may not be able to find the conference building because he would have no association between the building's name and any descriptive features. However, a basic set of directions can be used to navigate to the appropriate place and then learn what the building looks like. An example set of directions is: "Travel south for 2 blocks and then turn left. Travel another block. Stop. The conference center is on your left." Upon completing the specified actions blindly, the person arrives at the destination and creates an association between the surroundings and the target location.

The person may use some previous knowledge during the association task. For instance if the person has visited other conference centers, he may be able to use certain features to determine specifically which building he wants by distinguishing it from neighboring residential buildings. Once the association is made, the person is able to use the conference center as a feature in future plans.

An important detail of this approach is the issue of feature extraction. So far we have glossed over the topic by stating that the robot will extract various features from the environment in order to update its model. Specifically, we expect the robot is able to sense its surroundings, although at a reduced level of detail. For example, the person looking for a conference center knows that he is looking for a building and can see the various buildings around him, without yet knowing the names of the buildings. Similarly we will assume that part of the robot's plan includes a description of the types of features it should be looking for. This information is used to make a correct association between the state and the model object of interest.

Of course, there exists the possibility that the robot will become lost. Odometry errors and uncertainty in the environment can sometimes make it challenging for a robot to execute even a simple plan accurately. In some cases it may be possible for the robot to determine that it is lost if the features in the environment do not match the expected features specified in the plan. In other situations wrong associations between location and features will be made. Detailed analysis and discussion of what to do in such cases, is beyond the scope of this paper, but provides an interesting direction for further research.

2.3 Combining Monitoring and Reverse Monitoring

The use of regular Monitoring alone in planning allows for complex behaviors but makes the algorithm very sensitive to errors in state detection. The use of Reverse Monitoring alone supports only basic behaviors but allows the robot to update the observation model. We propose to combine the two approaches,

resulting in an algorithm that alternates between the two methods as needed. In cases where state detection is accurate and reliable, regular Monitoring is used to execute arbitrary plans as usual. When state detection breaks down, Reverse Monitoring is applied to update the model. The updated model is then used to make action termination decisions in more complex plans that use regular Monitoring.

An important issue to consider is where the scripted action plans come from for Reverse Monitoring. The presence of these plans implies that an expert of some sort is needed in order to guide the robot. Even in the human example, help is needed in order to acquire the directions to the destination. In this paper we assume that an expert is available and can provide a scripted action plan for getting from some known point to the destination. Another assumption we make here is that the robot’s designer selects when to transition between Monitoring and Reverse Monitoring.

Although these assumption may seem strong at first, they are quite practical in most cases when a robot is acting in the real world. In most cases the robot’s user knows when changes to the environment occur and the features of any new locations the robot needs to learn. User involvement required to produce these plans is usually trivial compared to the amount of input required for the alternative manual model update method, and plans can be reused an arbitrary number of times. However, it remains an important problem to develop methods in which these functions may be performed autonomously. Detecting when to switch between the two algorithms is a particularly compelling problem several aspects of which are already being studied in other areas [1, 10].

The following sections present our experimental results, which demonstrate how the combination of these two methods can lead to greater autonomy and reliability in a robotic system.

3 Experimental Domain

3.1 Problem Overview

We chose to demonstrate the effectiveness of the proposed approach using the Sony AIBO ERS7 robots (Figure 1(a)). The robot’s set task is navigation in the robot soccer domain using the on-board camera for sensory input. The robot’s vision system recognizes objects by their color and shape. Since an object’s color changes with the lighting in the environment, it often becomes necessary for the robot to update its model of what each particular color looks like. It is this task of vision color calibration that we will be focusing on for our experiment. Our aim is to show that by applying the Reverse Monitoring method, the robot is able to autonomously update its model and then return to regular plan execution using Monitoring, bypassing a tedious manual calibration process.

3.2 Robotic Platform and Domain

We used the commercially available Sony AIBO ERS-7 quadruped robot as the robotic platform. The robot is a complete autonomous system, with a 576 Mhz

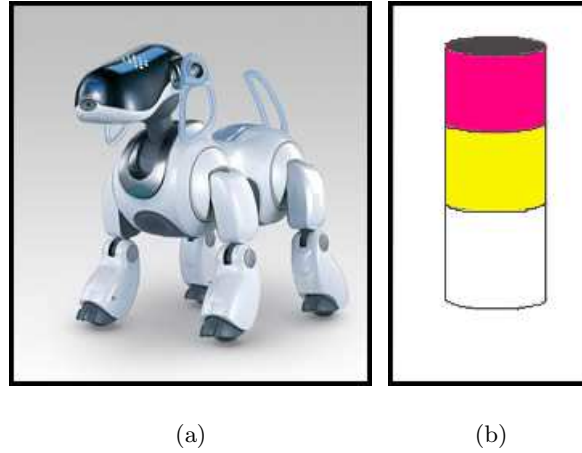


Fig. 1. The ERS7 robot (a), and a localization marker (b).

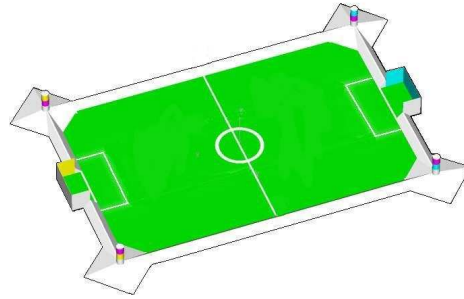


Fig. 2. The robot's environment.

MIPS processor, wireless communication and variety of on-board sensors. The main sensor, and the one we will be focusing on, is a color camera located in the head of the robot.

The robot's environment is the standard robot soccer field designed according to the rules of The RoboCup Federation [13]. The field is comprised of a green carpeted area surrounded by a white 10cm high wall, see Figure 2. Two goals are located on opposite sides of the field, and four uniquely colored markers are located at the corners. Each marker is 30cm tall with two 10cm-wide bands of color at the top, see Figure 1(b). Three colors are used to identify the markers and the goals - yellow, cyan and pink. The color combinations of the marker bands make it possible to uniquely identify each of the markers. The robot performs all navigation tasks by using the camera images to determine the relative location of the corner markers and triangulate its position on the field.

3.3 The Vision System

We will give a brief overview of the robot’s vision system, which allows the robot to sense the environment. The vision system is responsible for converting the raw YUV camera images into information about relevant objects that the robot sees [2, 3]. This task is accomplished in two major stages. First each pixel in the 208x160 color image is classified into one of the predefined color classes using a color lookup table. Then, the shapes and sizes of symbolic color regions are examined to find objects of interest in the image. This entire process runs in real-time at approximately 30 frames per second.

For the vision method to work, it is vital to have a color lookup table that will map raw YUV pixel values to their color classes. This mapping is the final product of the calibration method. Since color is the only distinguishing characteristic between different landmarks in this domain, it is vital that the color threshold mapping be accurate in order to distinguish between them.

The traditional calibration method typically requires a human to place the robot at various points in the domain to collect images of landmarks for calibration. The user must then hand-label each of the pixels in the images to classify it into one of the discrete color classes. This tedious process results in the robot being able to recognize certain colors and then use that information in combination with object shape to recognize objects. In the following section we will describe how we have applied the Reverse Monitoring method to automate this process.

4 Reverse Monitoring for Landmark Modeling

Our goal in this domain is for the robot to navigate in the environment using the markers as features in order to determine its location. When the observation model is available and the robot is able to accurately map pixel values in the image to their appropriate color label, the navigation task can easily be accomplished through regular Monitoring. However, if the lighting conditions are changed and the color mapping is no longer accurate, the robot’s ability to navigate deteriorates until in the worst case it has no color information at all.

Using the Reverse Monitoring method, the robot’s color model can be updated to use the new color values. Given a scripted action plan, the robot is able to blindly navigate to the defined points on the field, record the state at that point, and use the features of that state, i.e. pixel color values, to update the observation model. This method allows the robot to collect and analyze the images with very little human assistance. Once this is complete, the robot is able to execute arbitrary plans in the domain.

4.1 Scripted Plan Execution

The key steps of the scripted plan are outlined in Table 1. The robot is directed on a path that visits each of the landmarks to be modeled. Once the images for

-
1. Robot is at start point, P_0 .
The robot is placed at the preset start position.
 2. For $i = 1$ to N , for N landmarks that the robot will model:
 - (a) Execute scripted step S_i by walking to point P_i using a series of scripted actions
The robot walks using odometry (and not sensory data) to traverse the preset distance.
 - (b) At point P_i , collect image data for landmark L_i
The robot automatically perturbs its position and angle to get images from several vantage points.
 - (c) Learn observation models from acquired image data.
-

Table 1. Scripted Plan for Landmark Model Acquisition.

a particular landmark are obtained, they are analyzed to extract the object’s features, mainly the pixel colors of the marker bands. Once the full plan has been executed, the robot has a complete observation model, which allows it to execute arbitrary navigation plans in the domain.

As discussed earlier, the scripted plan includes a description of the key features the robot should focus on. In this particular case, the robot is told the physical size and shape of the marker. This information is not sufficient for accurate navigation in the robot environment since the only distinguishing characteristic of the landmarks in our domain is color. However, it is enough to locate the marker and the bands in the image (see the next section for more details). Since the plan specifies exactly which marker the robot is near, the colors can be appropriately assigned to the marker bands.

4.2 Image Analysis

Several standard vision algorithms were applied to extract the marker features from the images. This process replaces the method traditionally used in

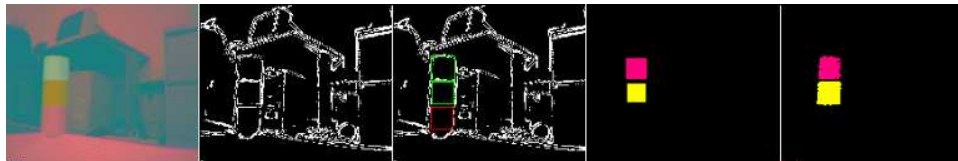


Fig. 3. The image analysis pipeline: original YUV image; edge image, the marker detected; the coloring of the detected marker; the original image color segmented with the updated thresholds.

this system of a human labeling the marker regions by hand. Figure 3 shows an example of an image at various stages in the image processing pipeline. The Sobel edge filter is first applied to the raw image, resulting in the edge image seen in the second stage in Figure 3. This step reduces the dimensionality of the observations from a multiple color image to a binary edge image containing only color transition information. A Hough transform [8] is then used to locate the two colored bands of the marker (see image 3 of Figure3). The bands of the marker are then labeled with the appropriate colors, while the rest of the image is marked in black to symbolize the background, or non-marker, region (see image 4 of Figure3). The achieved color mapping is then used to generate an updated lookup table, mapping pixel color values in the raw image into the symbolic color space. The final image in Fig. 3 shows the original image after it has been color segmented using the newly learned mapping.

5 Experimental Results

To test the performance of the algorithm, we evaluate the accuracy of the robot’s model and its ability to navigate in the environment. The robot’s performance is compared to its normal performance under the traditional system, where the observation model is manually calibrated by the user. The following sections describe the results for model and navigation accuracy.

5.1 Model Accuracy

The robot’s observation model was tested by evaluating the accuracies of the color class mapping and the marker detection. Specifically, we analyzed:

- the percent of pixels accurately labeled by the marker detection as marker regions
- the percent of pixels accurately mapped by the model to their correct color class
- the accuracy of the marker distance estimates generated by the model

The accuracy of the marker detection during the image analysis step is very important to the success of the algorithm. The more pixels that are labeled with their correct color class during training, the more accurate the model will be at detecting markers in new images. Incorrectly labeled pixels will lead to poor performance. As a result, our marker detection algorithm is designed to be conservative, preferring not to label anything in the image if it is not sure of the marker’s location instead of labeling the wrong part of the image. The labeling also avoids the edges of the marker in order to avoid classifying parts of the background as marker colors. Overall, the automatic classification did not label markers in 17% of the images due to low confidence about their location. In the remaining 83% of the images the algorithm labeled an average of 9% fewer pixels than a human. Figure 4 shows several sample images in the marker detection stage.

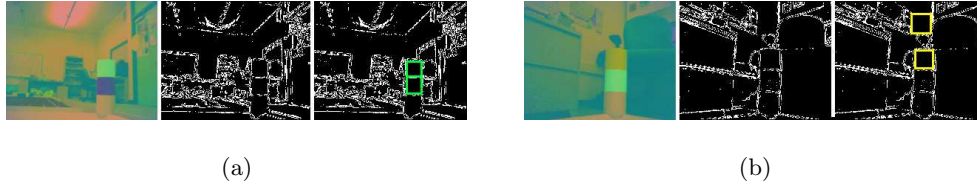


Fig. 4. Marker identification in robot camera images. Green boxes mark the boundaries of regions classified as marker bands. Yellow boxes mark boundaries of candidate regions in images where no marker was found.

The color lookup table generated from the automatically labeled images was compared to a lookup table generated from a human labeling of the same images. Both color lookup tables were applied to a set of previously unlabeled test images. On average, automatic calibration accurately labeled 89% of the marker pixels, while manual calibration resulted in 98% accuracy compared to a manual labeling of the test images. Figure 5 shows a sample image of a marker classified using the two lookup tables.

Finally, we evaluated the effect of the decreased number of labeled pixels on the marker distance estimates. After classifying the images, each one was used to calculate the location of the displayed marker relative to the robot. We found that the 9% decrease in the number of labeled pixels did not have a significant impact on the distance estimates, causing an average of 3mm difference in the distance measurements (0.6% of the actual marker distance). We therefore conclude that the model generated through automatic labeling is comparable to the manually generated model.

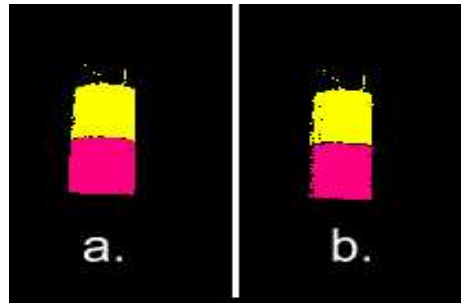


Fig. 5. Image of a marker after it has been color segmented using manual (a) and automatic (b) calibrations.

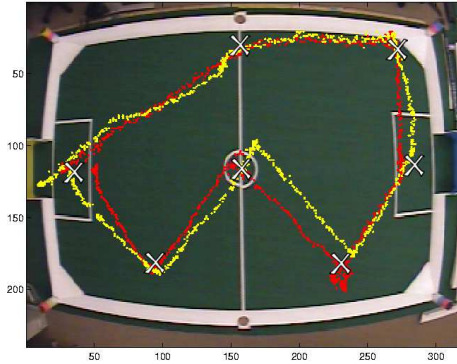


Fig. 6. The path taken by the robot while executing the navigation task as seen from an overhead camera. Trial of the robot using manual calibration shown in yellow, trial of automatic calibration shown in red.

5.2 Navigation Accuracy

The robot's navigation abilities were tested to determine the effects of the generated model on task execution. The robot was presented with a task of navigating to and stopping at seven set points on the field. The only information given to the robot was the sequence of points that it should visit. The only input used by the robot were the marker observations.

The robot's performance was evaluated based on the accuracy with which it was able to reach the set points, as well as the directness of the path from point to point. Figure 6 shows the trajectories the robot took when executing the task using the automatically generated (red) and manually generated (yellow) models. The target points are marked in the image with a white 'X'. The robot performed equally well using both models, stopping an average of 8.7cm off target using manual calibration and 6.3cm off target with the automatic calibration. The path taken between the points was direct for both trials with a few exceptions.

We consider both results to be good, and the difference in trajectories not significant due to the unavoidable noise of the environment. This experiment indicates that the automatically generated observation model results in performance that is comparable to that of manual calibration methods.

6 Conclusion

In this paper we present a general-purpose framework for updating a robot's observation model in the context of planning. By applying this framework, which we call Reverse Monitoring, a robot is able to update its model through the execution of a simple scripted plan. This allows the robot to adapt to changes in the environment and results in a more robust planning system. This method replaces the manual sensor calibration stage that, although not part of traditional planning, is a fundamental part of any robotic system that planning may

be implemented on. We demonstrated the effectiveness of this approach by replacing one of the key calibration stages in an existing robotic system without experiencing any degradation in performance.

References

1. M. Basseville and I. Nikiforov. *Detection of Abrupt Change - Theory and Application* PrenticeHall, Englewood Cliffs, N.J., 1993.
2. J. Bruce, T. Balch, and M. Veloso, *CMVision* (<http://www.coral.cs.cmu.edu/cmvision>).
3. J. Bruce, T. Balch, and M. Veloso, "Fast and inexpensive color image segmentation for interactive robots", In the Proceedings of IROS-2000, 2000.
4. Fernandez, J. and Simmons, R., "Robust Execution Monitoring for Navigation Plans", Proceedings of the Conference on Intelligent Robots and Systems, 1998.
5. Fichtner, M., Gromann, A., and Thielscher, M., "Intelligent execution monitoring in dynamic environments", *Fundamenta Informaticae*, Volume 57, pp. 371-392, 2003.
6. Graefe, V., "Object- and Behavior-oriented Stereo Vision for Robust and Adaptive Robot Control", In the Proceedings of the International Symposium on Microsystems, Intelligent Materials, and Robots, Sendai, 1995.
7. K. Z. Haigh and M. Veloso, "Planning, Execution and Learning in a Robotic Agent", In the Proceedings of the Fourth International Conference on Artificial Intelligence Planning Systems, 1998.
8. P.V.C. Hough, "Machine Analysis of Bubble Chamber Pictures", In the Proceedings of the International Conference on High Energy Accelerators and Instrumentation, CERN, 1959.
9. Jüngel, M., Hoffmann, J. and Lözsch, M., "A Real-Time Auto-Adjusting Vision System for Robotic Soccer", In the Proceedings of the 7th International Workshop on RoboCup 2003.
10. S. Lenser, "On-line Robot Adaptation to Environmental Change", PhD thesis CMU-CS-05-165, Carnegie Mellon University, 2005.
11. Livyatan, H., Yaniv, Z., Joskowicz, L., "Robust Automatic C-Arm Calibration for Fluoroscopy-Based Navigation: A Practical Approach", *Lecture Notes in Computer Science*, Volume 2489, pp. 60-68, 2002.
12. Mayer, G., Utz, H., and Kraetzschmar, G., "Towards Autonomous Vision Self-Calibration for Soccer Robots", In the Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 2002.
13. RoboCup Four-Legged Robot League Rules (<http://www.tzi.de/4legged/bin/view/Website/WebHome>).
14. Zrimec, T., and Wyatt, A., "Learning to Recognize Objects - Toward Automatic Calibration of Colour Vision for Sony Robots", In the Proceedings of the Machine Learning in Computer Vision Workshop, ICML 2002.