Use of Fuzzy Histograms to Model the Spatial Distribution of Objects in Case-Based Reasoning

Alan Davoust, Michael W. Floyd, and Babak Esfandiari

Department of Systems and Computer Engineering
Carleton University
1125 Colonel By Drive
Ottawa, Ontario

Abstract. In the context of the RoboCup Simulation League, we describe a new representation of a software agent's visual perception ("scene"), well suited for case-based reasoning.

Most existing representations use either heterogeneous, manually selected features of the scene, or the raw list of visible objects, and use ad hoc similarity measures for CBR. Our representation is based on histograms of objects over a partition of the scene space. This method transforms a list of objects into an image-like representation with customizable granularity, and uses fuzzy logic to smoothen boundary effects of the partition. We also introduce a new similarity metric based on the Jaccard Coefficient, to compare scenes represented by such histograms.

We present our implementation of this approach in a case-based reasoning project, and experimental results showing highly efficient scene comparison.

Keywords: Case Based Reasoning, Fuzzy Histograms, Knowledge Representation, Soccer Simulation.

1 Introduction

The paradigm of case-based reasoning (CBR), where the decision-making process of an agent is based on a database of past situations, has been applied to a variety of artificial intelligence applications. A notable application is the problem of *modeling others from observation.*

An ongoing project at Carleton University is a CBR project in which an imitative software agent is trained to imitate the behavior of another target agent, based on the logs of this target agent's actions ([1], [2]). The agent of interest in this project is a simulated soccer-playing agent in the RoboCup Simulation League [3]. In this League, teams of autonomous agents play a simulated soccer game, interacting only with the game server. The server sends each player his perception of the playing field (the "*scene*"), described as a list of all the visible objects, with their coordinates w.r.t. the player. Objects include the ball, other players, lines, etc. This raw data could be compared to the result of a feature extraction in a real robot.

S. Bergler (Ed.): Canadian AI 2008, LNAI 5032, pp. 72–83, 2008.

Each agent (player) replies with its action (which may be *"dash"*, *"kick"*, or *"turn"*, with a parameter), and the server uses all the agents' actions to simulate another time step in the game, and so forth.

Emulating such an agent in a CBR approach requires forming a database of "cases" modeling the scene, and the agent's response action, which can be seen as the known solution to the problem. When presented with a new situation, the imitative agent's playing algorithm involves searching the case base for one or several similar scenes, and using these similar "cases" to decide on an action.

Initiatives to use CBR in the RoboCup context include [4], [5], [6] and [7]. In these other works, the authors selected particular features of the scenes, based on their intuition of what seemed important or not in a soccer game simulation. Features included lists of objects, counts of objects present in particular "strategic zones" of the playing field, and other indicators of the game situation, such as possession of the ball.

In earlier work with our imitative agent, [1] and [2], a scene was represented using a simple data translation of the server message, listing each object with its coordinates. This raw representation avoids the bias inherent in manually selecting domain-specific scene features, but does not support practical similarity metrics, which is a serious handicap for our real-time CBR problem.

We present here a new approach similar to the idea of occupancy grid maps [8] used in robotics. We model the scene by means of a 2-dimensional grid map, onto which we project the objects, using fuzzy logic to smoothly approximate their location in a discrete representation.

We obtain a fuzzy histogram of objects, which is an image-like representation of a scene, showing the spatial distribution of objects with a customizable granularity. We present a similarity metric based on the set-theoretic Jaccard Coefficient. We have implemented this scene representation in our imitative agent framework, and experimental results in this setting show that our approach compares favorably with previously presented approaches in terms of accuracy, while reducing the computational cost of comparisons by a factor of ten.

The rest of the paper is organized as follows: first, in Section 2, we define our approach with respect to related work, then in Section 3, we present our fuzzy histogram scene representation, and some distance and similarity metrics to compare such data structures. In Section 4, we give some practical details on our implementation of this approach in our CBR project, and finally we discuss our experimental results in Section 5, with some directions for future work.

2 Related Work

Our CBR imitation framework for Robocup agents was initially presented in [1], and a number of optimizations were presented in [2]. In this previous work, the scene representation was a simple data translation of the server message, i.e., a list of objects with their position in polar coordinates. Comparing two scenes involved matching the objects in one scene with the objects in the other. It

was then possible to compare, for instance, the position of the ball in one scene with the position of the ball in the second scene, if the ball was present in both scenes. The cartesian distance between these two positions could be computed, and then the sum of distances for all pairs of objects across the scenes was used as a "distance" metric measuring the dissimilarity of the scenes.

The first problem with this representation is that is does not form an ordered set of features. At a given time the agent sees only a small fraction of the full list of objects present on the field; furthermore, objects may be interchangeable, e.g., two players of the same team, or two lines, cannot necessarily be recognized[1]. Matching the objects in this case may not be trivial. Lam *et al.* [1], as well as Karol *et al.* [5], who were also faced with the problem of interchangeable players in their own work, note that this object-matching problem accounts for the bulk of the computational cost in comparing two scenes.

As noted in the introduction, most other teams that have used CBR in the RoboCup context have tackled this problem by selecting particular features deemed important for soccer, to suit their particular needs. Our main motivation in this work is to propose a practical scene representation (in terms of comparison cost), while avoiding a biased selection of features by a human expert.

In a non-biased approach, we can draw a parallel with a robot navigating an unknown environment, in that the robot attempts to build a map of its environment. The typical example is a *grid occupancy map* [8], which is a grid in which each cell is assigned a probabilistic indication that an obstacle is found at this location. Such a representation also aims to project all the available information (e.g, from a sonar) on a feature map that represents the entire known environment.

Wendler et al. [4] used a representation of the pitch in the spirit of grid occupancy maps, noting the presence or absence of players in a segmented area around the ball. Other works have considered partitions of the playing field, e.g. [6], where the players in the "offensive zone" and "defensive zone" are counted, and make up two features of the scene representation.

In both cases this is used as only a small part of the representation. Furthermore, as noted in [1], such a segmentation induces boundary issues, artificially separating objects that may, in fact, be very near, or assigning to the same segment two relatively distant objects.

Our approach here is to use this general idea to draw a symbolic grid map, over which we place all the objects in the scene, and avoid the boundary issues by approximating an object's position using fuzzy logic.

3 Methodology

In this Section, we detail the method by which we obtain the data representation of a scene, and compare two scenes thus represented. We describe the method

[1] Depending on the context, it may be desirable to recognize them, or else to maintain a level of abstraction where they are interchangeable.

for the context of the RoboCup Simulation League, but it is general enough to be extended to other domains.

3.1 Position of an Object

The basis of our approach is a segmentation of the scene space (the visible part of the playing field), using a two-dimensional grid, as shown in Figure 1. We discuss the specific parameters of the grid in our implementation, in Section 4.1.

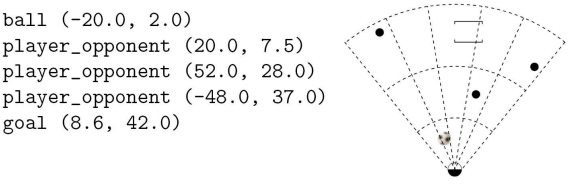

Fig. 1. An example scene described as a list of objects in polar coordinates, and segmented using a 3×5 grid

The position of an object in a segmented area can be smoothly approximated using fuzzy logic.

We consider each cell of the grid as a fuzzy set, rather than a conventional crisp set. This way, an object that is in the center of a cell might fully belong to it, whereas an object near the edge of the cell might have only a small membership in the cell. A logical approach is then to make the fuzzy sets corresponding to neighboring cells overlap, so that an object at the edge of one cell also belongs to the next cell. This way, as we move in the scene space, we smoothly transfer from one cell to the next, using this fuzzy membership function. Being able to smoothly transfer from one cell to the next makes the location of the boundaries much less important than in the crisp case.

Formally, our "grid" is a fuzzy partition of the *universe of discourse* of each coordinate, thus obtaining a two-dimensional fuzzy partition of the scene space. As defined in [9], a collection of fuzzy sets A_i is a fuzzy partition of a universe X if the so-called *orthogonality condition* (1) is satisfied:

$$\forall x \in X \sum_{i=1}^{n} \mu_{A_i}(x) = 1 \tag{1}$$

where μ_{A_i} is the membership function of the fuzzy set A_i.

This condition, which we need to apply when designing the membership functions of each grid cell, is consistent with the crisp approach of assigning each object to exactly one cell, i.e., making the grid a (crisp) partition of the scene. It ensures that all the objects are given equal importance, wherever they are

located. Following this approach, the position of each object in the scene is given by the membership value of this object in each cell of the grid.

3.2 Definition of a Histogram

Whether using fuzzy or crisp sets, we can represent the scene by listing the contents of each cell. More specifically, if several objects are interchangeable, then the "contents of a cell" can be reduced to the number of such objects that the cell contains.

Following this approach, we represent a collection of interchangeable objects using a histogram over the grid. In a way, this is an extension of the grid occupancy map defined by [8] to an environment where we identify a number of precisely positioned objects in the grid, rather than a boolean notion of "cell occupancy."

In order to adapt this representation to our context, in which not all objects are interchangeable, we can represent a scene by several histograms, one per object *category*. Categories include *ball*, *teammate_player*, *line*, etc.

Finally, using the fuzzy partition presented in the previous Section, we can extend the concept of a histogram to a fuzzy histogram, by taking the fuzzy cardinality of each cell. We define this fuzzy cardinality as the sum of the membership values of the objects contained in the cell.

According to this definition, if the grid contains a total of $n \times m$ cells, and we have a total of k different object categories, then a scene can be represented by a histogram, which is a matrix of dimension $k \times n \times m$. For practical purposes, we can also use an isomorphic representation, e.g, k vectors of dimension $n \times m$. Formally, let A_{ij} be the fuzzy set representing a cell (i, j) in the grid, and $\mu_{A_{ij}}$ its membership function. We define X_p to be the set of visible objects of type p. Using a matrix notation our full fuzzy histogram is a matrix H of dimension $k \times n \times m$, with elements h_{pij} such that:

$$h_{pij} = \sum_{x \in X_p} \mu_{A_{ij}}(x) \tag{2}$$

3.3 Similarity Metrics

In case-based reasoning, the agent, when presented with a new problem, needs to compare this problem with his database of reference "cases," and select one or several of the most similar cases in order to compute its solution regarding the new problem.

In our case of interest, this means that the key problem for our imitative agent is to compare two scenes. Our scene representation is isomorphic to a feature vector, and comparisons can be made using standard distance metrics for vectors in a space of dimension $k \times n \times m$ (or $n \times m$).

We propose here a similarity metric based on the set-theoretic Jaccard coefficient (also known as Tanimoto coefficient), which we expose for a histogram represented simply by a vector of length n, without loss of generality.

Swain and Ballard introduced in [10] a similarity metric for comparing histograms, called the histogram intersection, which they defined as follows:
Let $H(h_1, h_2, h_3 \cdots h_n)$ and $G(g_1, g_2 \cdots g_n)$ be two histograms.

The histogram intersection of H and G is:

$$inter(H, G) = \sum_{i=1}^{n} min(h_i, g_i) \tag{3}$$

They also proposed a normalized version, which they called the *normalized histogram intersection*. Here a sample H is compared to a reference G:

$$inter_{norm}(H, G) = \frac{\sum_{i=1}^{n} min(h_i, g_i)}{\sum_{i=1}^{n} g_i} \tag{4}$$

This similarity comparison was introduced to compare color histograms of images, and in that context, the normalizing factor is the number of pixels in the "reference" image. However, it is not symmetrical, which can be a problem.

Adapting this measurement to make it symmetrical has been done in various ways in the image-processing community.

We propose here to normalize the histogram intersection by a factor that can be seen as the histogram *union*. By extending Swain and Ballard's histogram intersection notion to a corresponding[2] union notion, we obtain a histogram union defined as follows:

$$union(H, G) = \sum_{i=1}^{n} max(h_i, g_i) \tag{5}$$

In the case of sets the Jaccard Coefficient is defined as the norm of the intersection of the sets over the norm of their union :

$$j(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{6}$$

Extending this formula to histograms, we can normalize Swain and Ballard's histogram intersection by the histogram union. We then obtain the following similarity metric, which we will refer to here as the Histogram Jaccard Coefficient[3].

$$j_{hist}(H, G) = \frac{\sum_{i=1}^{n} min(h_i, g_i)}{\sum_{i=1}^{n} max(h_i, g_i)} \tag{7}$$

Note that these definitions have been presented without concern as to whether the histograms and sets were fuzzy or crisp, but here we assume that the fuzzy partition we defined in Section 3.1 allows us to extend the rest of the framework, implicitly defined in a "crisp" context, to the fuzzy case.

[2] By analogy with the correspondence between T-norms and S-Norms for intersection and union of fuzzy sets (see [9] for details).

[3] This metric is very similar to various extensions of the Jaccard Coefficient used in market basket data analysis, and in information retrieval (see, e.g. [11], and [12]).

In this work, we choose to compare scenes by comparing each of the k pairs of fuzzy histograms that represent the scenes, and consider the final result to be a weighted sum of those partial results.

The weights, meant to reflect the importance of each object type in the similarity evaluation, are discussed in Section 4.2.

4 Implementation

The scene representation presented in the previous Section is a general definition, which can be customized by selecting particular membership functions, and defining particular categories of objects for a specific application. In this Section we describe our specific implementation of this approach in the CBR framework defined in [1].

4.1 Membership Functions

The main parameters for designing our fuzzy partition of the scene space are the following:

- *Dimension of the grid.* The dimension of the full scene representation will be $k \times n \times m$, where k is the number of object categories, and $n \times m$ is the grid size. These parameters will fully determine the processing time of the case comparison, and the memory occupation of the case base.

  We present here results with the following grid sizes: 3×5, 4×6, and 5×8, and we have $k = 7$ types of objects. These sizes give a total dimension of 105, 168, and 280, respectively, for the full feature vector representing the scene.
- *Boundaries of the cells.* As noted previously, fuzzifying the boundaries of the cells makes their exact location less important, as they do not introduce any "drastic" separation of the data points. The location of objects in the scene is expressed in polar coordinates, and in our implementation we simply spaced the cells equally over the central-nearer area of the visible field. Cells defining the extremities of the visible space (sides and most distant points) are simply defined by their boundaries nearer to the origin (lower distance bound, higher bound of the angle for the extreme-left, lower bound of the angle for the extreme-right).
- *Membership functions.* The membership functions are constrained by the orthogonality condition, and by the boundaries of each cell. In order to avoid boundary effects, we use continuous membership functions. Simple functions allowing straightforward orthogonality include triangular membership functions, and functions based on sin^2 (sinus squared). In our implementation, membership functions for the angle coordinate are based on sin^2, whereas membership functions for the radial coordinate are triangular. In both cases, the membership function for the extremities of the grid are constant, in the region where they do not overlap with the neighboring cell.

Our 1-dimensional membership functions for a 3×5 grid are shown in Fig. 2.

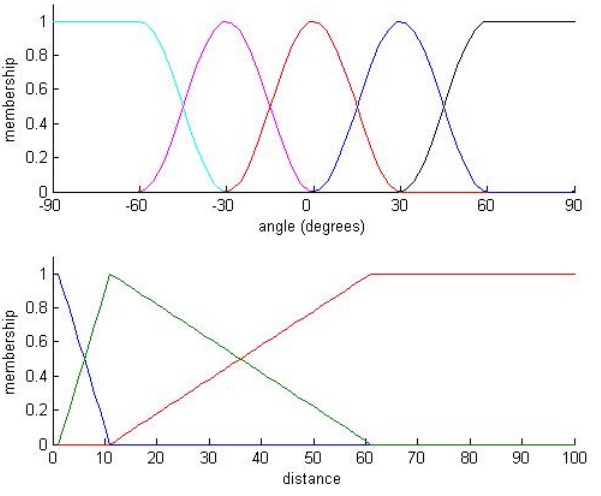

Fig. 2. Fuzzy partition of the polar angle (top) and radius (bottom)

4.2 Case-Based Reasoning Algorithm

We have integrated our scene representation in the framework defined in [1], and in this Section, we briefly describe the CBR algorithms adapted to our scene representation.

- **Setup:** We first set the general parameters of the framework: the size of the grid used to discretize the scene, and the membership functions to be used for the cells of this grid.
- **Preprocessing algorithm:** Using a log file, we compute a data structure that the agent will use as its case base. For each line in the log file, we create a scene using our fuzzy histograms representation, and attach the response action (also recorded in the log). The scene and response action compose a case. The cases are then aggregated as a single case base.
- **Real-time CBR algorithm:** When presented with a new scene, the agent creates the fuzzy histogram representation of this scene, and matches it to its case base. The agent selects the most similar scene, and outputs the action that was associated with that scene.

5 Evaluation

5.1 Evaluation Criteria

In this paper, we have put forward several characteristics of our scene representation, which we consider make it more adequate for real-time case-based reasoning.

First of all, our representation is a simplified representation of the original list of objects, in which we replace exact coordinates by a fuzzy histogram showing the spatial distribution of the objects, so we need to evaluate the new representation to ensure that this loss of information is acceptable. In our imitative agent framework [1], we evaluate this by comparing the imitative performance of the agent using our scene representation with the results of the original studies ([1] and [2]) which use the list of objects as a scene representation.

The main benefit of our representation is to provide a feature vector representation, which allows for much more computationally efficient scene comparisons. Comparisons are a key issue since our agent needs to meet real-time constraints. We report here the processing time of our agent, showing a significant improvement.

Thirdly, keeping in mind the essential assumption of the CBR paradigm, namely that "similar problems have similar solutions," we evaluate the effect of scene segmentation on the distinguishability of scenes, i.e. whether using our scene representation we can still tell apart two scenes that were originally different when originally described as lists of objects. Clearly, if two scenes associated to different actions become indistinguishable, then we have similar problems with *different solutions*, and this will set a theoretical upper bound for our performance in any CBR application.

From a significant number of logs, we counted how many scenes had a representation identical to that of another scene, and compared whether the actions were similar or not. Similar actions meant the same action label and statistically similar parameters, i.e the two parameter values differ by less than the standard deviation of the parameter[4].

Finally, as the Jaccard coefficient is reputed to be more "robust" in the data-mining literature, we compared it with the Euclidean distance for our experiments, analyzing how different dimensions for the scene representation impacted the imitative accuracy of the agent.

5.2 Experimental Results

As the absolute performance of the agent depends on many factors, we have simply compared our results with those reported by Floyd *et al.* in [2]. As a target agent for our imitation experiments, we used an existing RoboCup agent called Krislet [13], a stateless agent that exhibits a simple decision-tree type behaviour, looking for the ball, running after it, and kicking it towards the goal.

The imitative performance of an agent was obtained by cross validation, using three independent sets of logs from the same target agent. We used a first set of logs as a training set, i.e., we preprocessed those logs and used them as a case base for the agent, and we used a second set of logs as a validation set. We tested each pair-wise combination of log files and computed the average results.

Many indicators could describe the accuracy of the imitation. However, as this aspect is secondary to our discussion, here we only report the overall accuracy

[4] Calculated from all the instances of the action throughout the logs.

Table 1. Accuracy and processing time of the imitative agent using list of objects representation, Histograms, Fuzzy Histograms

Representation	List of objects	Histograms		Fuzzy Histograms	
dimension	N/A	3×5	5×8	3×5	5×8
Accuracy	68.4	67.9	82.1	70.7	80.4
Processing time (ms per scene)	27	2.5	5.4	2.5	5.4

of the imitative agent in discriminating between the three possible actions, i.e *kick*, *dash*, and *turn*, without considering the parameter value. Hence a positive match is a test scene for which the imitative agent selects the same action as the original agent.

Table 1 lists the results of the imitative agent in terms of predictive accuracy and computation time (average time for each scene comparison). The similarity metric used here is the Histograms Jaccard Coefficient, and the case base comprises 3000 scenes, as in the experiments of [1] and [2]. It clearly appears that our histogram-based approach can easily attain the accuracy of the original "list of objects approach," while being computable in significantly less time. Our interpretation of this result is that the loss of information resulting from the fuzzy histogram representation is not a problem, and on the contrary we could conjecture that it actually improves the representation by generalizing the data. In this particular evaluation, the use of fuzzy logic does not make a clear difference. Varying the dimension of the representation causes a near-linear increase in processing time, but the method remains highly efficient.

However, as noted in the previous Section, the scene segmentation causes a loss of information, which places an impassable upper bound on the potential accuracy of the non-fuzzy logic approach. Assuming we optimized some parameters of our CBR agent, as in [2], we still could not surpass a certain level of accuracy.

Table 2. Indistinguishability of scenes, with and without fuzzy logic

Number of indistinguishable[a] scenes (total 9264)

Grid Dimension	3×3	3×5	4×6	5×8
Non-fuzzy	5309	4697	4773	3075
Fuzzy	354	328	221	209

Number of indistinguishable scenes with different actions

Non-fuzzy	2320	1660	1712	578
Fuzzy	34	6	4	4

[a] We refer to a scene as indistinguishable if there exists at least one distinct scene that has the same representation

This limitation is shown in Table 2. Here, from our full list of working logs, we counted every scene which is not unique (in terms of histogram representation), and for these non-unique scenes, how many were identical to another scene but associated to a different action, i.e. a "similar problem with a different solution." As expected, the use of fuzzy logic clearly removes this overgeneralisation. The few indistinguishable scenes in the fuzzy case are due to the constant value used for the membership functions of the extremity grid cells (see Fig. 2).

Finally, in order to evaluate the robustness of the similarity metrics, we varied the grid size from 3×5 to 5×8, i.e. the total dimension of the representation varying from 105 to 280 dimensions. The accuracy of the imitation is plotted against the total dimension of the representation in Fig. 3. The Histograms Jaccard Coefficient clearly scales better than the Euclidean distance measure.

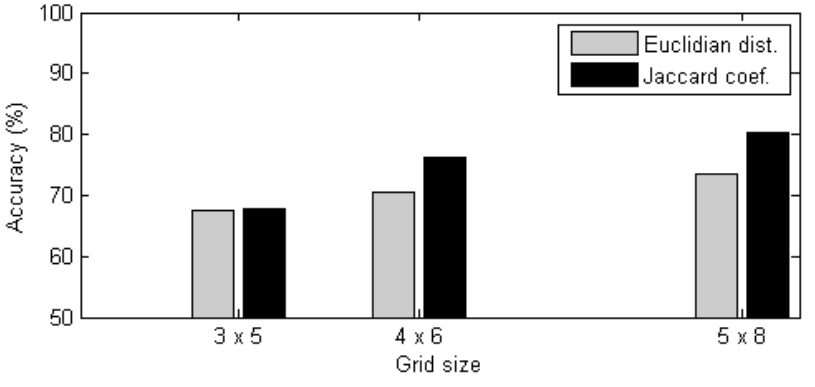

Fig. 3. Accuracy obtained using the Jaccard Coefficient Similarity, and Euclidian distance, for different grid dimensions. (note accuracy range represented 50-100%)

6 Conclusion

We have presented a method to represent the visual perception of an agent, based on a segmentation of the visible space. The distribution of objects is represented using a histogram over the segments, for each object category. Fuzzy logic allows for the smooth spreading of the count of objects over neighbouring segments according to the actual position of the objects, and thus limits boundary effects.

This representation supports practical similarity metrics, including a robust metrics based on the set-theoretic Jaccard Coefficient, and the efficient computation of the similarity metrics make this representation highly suitable for case-based reasoning.

Furthermore, this feature vector representation opens doors for the application of other machine learning techniques while avoiding the need of *a priori* manual feature selection. In our RoboCup imitation framework, we intend to build on this work by exploring automated feature selection methods.

References

1. Lam, K., Esfandiari, B., Tudino, D.: A scene-based imitation framework for Robocup clients. In: MOO Modeling Others from Observation, AAAI workshop (2006)
2. Floyd, M., Esfandiari, B., Lam, K.: A Case-based Reasoning Approach to Imitating RoboCup Players. In: Proceedings of FLAIRS-2008, Florida AI Research Symposium (to appear, 2008)
3. Robocup, `http://www.robocup.org`
4. Wendler, J., Lenz, M.: CBR for Dynamic Situation Assessment in an Agent-Oriented Setting. In: Aha, D., Daniels, J.J. (eds.) Proc. AAAI 1998 Workshop on Case Based Reasoning Integrations, Madison, USA (1998)
5. Karol, A., Nebel, B., Stanton, C., Williams, M.: Case Based Game Play in the RoboCup Four-Legged League Part I The Theoretical Model. In: Polani, D., Browning, B., Bonarini, A., Yoshida, K. (eds.) RoboCup 2003. LNCS (LNAI), vol. 3020, pp. 739–747. Springer, Heidelberg (2004)
6. Marling, C., Tomko, M., Gillen, M., Alexander, D., Chelberg, D.: Case-based reasoning for planning and world modeling in the robocup small size league. In: IJCAI Workshop on Issues in Designing Physical Agents for Dynamic Real-Time Environments (2003)
7. Ros, R., Veloso, M., López de Mántaras, R., Sierra, C., Arcos, J.L.: Retrieving and Reusing Game Plays for Robot Soccer. In: Roth-Berghofer, T.R., Göker, M.H., Güvenir, H.A. (eds.) ECCBR 2006. LNCS (LNAI), vol. 4106, pp. 47–61. Springer, Heidelberg (2006)
8. Moravec, H.P., Elfes, A.: High resolution maps from wide angle sonar. In: Proc. IEEE Int. Conf. Robotics and Automation, pp. 116–121 (1985)
9. Dubois, D., Prade, H.: Fuzzy Sets and Systems, theory and applications. Academic Press, New York (1980)
10. Swain, M., Ballard, D.: Color indexing. International Journal of Computer Vision 7(1), 11–32 (1991)
11. Strehl, A., Ghosh, J.: Value-based customer grouping from large retail data-sets. In: Proceedings of the SPIE Conference on Data Mining and Knowledge Discovery, Orlando, Florida, April 24-25, vol. 4057, pp. 33–42. SPIE (2000)
12. Haveliwala, T., Gionis, A., Klein, D., Indyk, P.: Similarity Search on the Web: Evaluation and Scalability Considerations, Stanford Technical Report (2000)
13. Langner, K.: The Krislet Java Client (1999), `http://www.ida.liu.se/~frehe/RoboCup/Libs/libsv5xx.html`