A bottom-up robot architecture based on learnt behaviors driven design

I. Herrero, C. Urdiales, J.M. Peula, and F. Sandoval

Dpt. Tecnología Electrónica, University of Málaga,
iherrero@uma.es,
WWW home page: www.grupoisis.uma.es

Abstract. In reactive layers of robotic architectures, behaviors should learn their operation from experience, following the trends of modern intelligence theories. A Case Based Reasoning (CBR) reactive layer could allow to achieve this goal but, as complexity of behaviors increases, the curse of dimensionality arises: a too high amount of cases in the behaviors casebases deteriorate response times so robot's reactiveness is finally too slow for a good performance. In this work we analyze this problem and propose some improvements in the traditional CBR structure and retrieval phase, at reactive level, to reduce the impact of scalability problems when facing complex behaviors design.

Keywords: Case based reasoning, reactive layer, learning architecture, robotics

1 Introduction

Hybrid architectures are, nowadays, the most successful approach to control systems in modern robots. Most hybrid systems have a planner layer, related to abstract knowledge and global objectives of the robot in the long-term, and a reactive layer, which decides the response of the robot at local short-term scale, and is composed by primitive behaviors. Diverse implementations differ in the organization of the layers; the tools used to build them; and the mechanisms to manage their interaction. However, behaviors at the reactive layer are usually designed at earlier stages of the architecture development in a fixed algorithmic way, with little or no adaptability to changes in environment and problem evolution. There are some evidences which suggests that these reactive components should be able to adapt by learning from the robot experiences. First, Ethology shows us that some animals behaviors, the so called reflex or innate, are acquired from birth; learned behaviors must be trained and developed by repetitions and experience; but there are also innate-with-memory behaviors which must be tuned by learning or experience: this is how a baby bee learns its hive look and how to navigate to and from it[11]. Furthermore, modern theories about human intelligence claim that even in a complex deliberative response, just a few neurons are involved, with a not very fast transmission of information. So human brain's calculation power is, a priori, orders of magnitude lower

than modern computers'. Instead, human intelligence appears to be funded on *learning from experience*, by storing everyday events, knowledge, and responses in a powerful data base imprinted on our brain, and retrieving them when in a similar situation[5]. This view of human intelligence has been assimilated to research in Artificial Intelligence(AI), especially in the field of Intelligent Robot Systems(IRS). The traditional (S)ense-(P)lan-(A)ct based paradigms, now include a new (L)earning primitive, to become a (S,P,A,L) set of primitives, from which new IRS architectures are being developed[18].

There are many approaches to robotic hybrid architectures which follow these learning guidelines [14][17][12], but most of them focus this learning on deliberative levels. As the human brain follows a "nest structure", its different areas should operate with the same principles no matter what are devoted to; they only differ in the nature of the information and the abstraction level of the concepts they manage, which defines their situation on the reasoning chain[5]. So, it seems reasonable to think about an IRS architecture in which different layers have equivalent operating mechanisms, at least at a functional and information processing level. In this sense, all the layers and sub-layers at every level should be based on some kind of learning by experience, *including the low reactive level*. Besides the improvement in flexibility and adaptability at this reactive level, the architecture would gain in homogeneity and scalability, specially if the same AI method is used for the layers development. In [13] we proposed a first approach to a CBR-learning based implementation of the reactive layers for a hybrid architecture, which was programmed to an AIBO-ERS7 robot. Despite the good performance of our previous work, it showed some problems related to the ad-hoc nature of the behaviors design, such as scalability problems with the increase of managed information, and retrieval time of the response. In this work, we suggest some improvements to solve these problems, mainly related to changes in the casebase structure and retrieval process. Furthermore, we propose a change of the philosophy inherent to our implementation, to switch from numeric to conceptual knowledge in order to be able to extend the model, in the future, to higher deliberative layers, following the aforementioned "nest structure" of the human intelligence.

2 CBR based reactive layer

Reactive layer relies on coupling sensors information and actuators response into a set of low level primitive modules or *behaviors*. Every behavior deals only with local, short-term information related to the goals or scope of the module, so it doesn't need a complex model of the environment[10]. Reactive Behaviors are very well suited to fast low-level decisions, so this layer is specially important in dynamic unstructured environments. These low level behaviors are combined later in a bottom-up way to produce more complex emergent ones. There are different methods to achieve this emergence, from combining or switching several behaviors[6], to using subsumption architectures[3]. But these methods usually depend on many parameters that need to be optimized for each specific prob-

lem, specially if different robots are used, as they depend on the robot kinematics and dynamics, sensor calibration, and even mechanical errors. For these reasons many reactive architectures and layers have included AI tools such as fuzzy-logic rules [2] or ANNs[9] to fine-tuning the proposed behaviors through experience. Developing analytical expressions to relate sensor information to motor commands may not be simple, as some reactive behaviors are hard to explain and may adapt better to one function or another depending on the situations. Fuzzy-logic methods define a set of rules to map sensor reading and motor actions, rules whose parameters can be adjusted during the robot execution. But it's difficult to accomplish a correct design of such rules, as they need a great knowledge about both fuzzy-logic and the problem domain. ANNs get over these obstacles as, once the structure of layers and neurons of the network has been chosen, the training process is transparent to the user. However, ANNs only provide us of a better or worse operating system, but are black boxes with no clues of what has gone wrong when they don't find a right solution[5]. On the other hand, Case Based Reasoning (CBR) combined with learning, could allow us to obtain a fully-working system, as well as to understand the learning process and the knowledge acquisition which is very useful to debug errors and malfunctioning of our system.

2.1 Experience Based Learning: CBR

Case-Based Reasoning is a reasoning, learning and adaptation technique to solve current problems by retrieving and adapting past experiences[1]. CBR systems rely on remembering old solutions given to situations similar to the current problem, and adapting them to fit it. The new problem together with its new solution can also be stored in the casebase,as a new case, to be used later. Thus, better solutions can be derived when faced against less experienced situations. As we can see, the philosophy of CBR is quite in tune with the model of intelligence as prediction, and the learning method to build it. A CBR system cycle to solve a new problem consists of four steps: (i) retrieve the most similar stored case to the new current case; (ii) adapt its solution to the new current case; (iii) evaluate the results of the proposed solution; (iv) learn from the new experience. Consequently, when creating a new CBR application, design decisions often concern: i) how to describe the problem to solve (ii) which is the casebase or case library structure; within a particular case structure; (iii) how retrieval process and similarity assessment between cases can be evaluated; (iv) how to adapt the old solution to solve the new current problem, (v) how to evaluate the success of the proposed solution, and (vi) what to learn and how to learn from solved problems (new experience gain).

CBR has been widely used in many experience learning frameworks in Robotics [7][8][15]. In [13], we proposed a reactive CBR-learning layer in which coupling of visual data and motion commands were learned by low level reactive behaviors. This way, the robot could know what to do from the perception it had at any moment. Such learning was done by supervised training, but from the own robot

experience too, so not only the goal of the behavior is learned, but also the environmental and the own robot physical conditions which affect the attainment of such goal. Although our method showed a good performance, its advantages were not evident because it was tested with fairly simple behaviors that could be implemented analytically.

In our new proposal we intend to systematize the process of definition of different behaviors when several visual elements (goals, rival and friend players,ball, field,..) are included in the CBR casebase which controls the behavior operation. This means that CBR must learn not only how every individual object influences the operation of the behavior, but how *their relationship* affects it, too. As there are more components in a CBR case, the number of required cases increases from nearly a hundred to thousands: scalability problems arise, as CBR cannot respond as quick as a reactive layer would need. Casebase size can be reduced by clustering or discretization which also implies a transition from a numerical representation of the visual information to a conceptual one. This way, concepts at low levels would combine to develop more abstract complex ones at higher level behaviors, which would emerge from the lower layer in a bottom-up integration. We propose some changes in the original structure and operation of the CBR casebase in order to further speed the process of recovering the solution at the module when a new problem or situation is presented to the behavior: i) from a flat database structure we have switched to a hybrid hierarchical-flat solution, defining the concept of "context" to traverse the tree hierarchy; ii) we have also split the CBR recovering phase in two sub-phases: one fast scene recognition by indexing objects; and the traditional search of the most similar case by a distance comparison.

3 CBR Behavior design

The scenario for our experiments is similar to the one described in [13], in which an AIBO ERS7 robot, programmed with the Tekkotsu software platform [16]. is used. We have developed a GUI to display the on-board camera image, and also guide the robot with a joystick. In this work, mainly visual information will be considered, but we will add a IR chest sensor too, to detect when the robot is in possession of the ball. The experiments will be run in an Robosoccer field in which, besides from the ball and two goals, a maximum of two opponent robots and a friend robot, could appear. The inclusion of all or only some of these elements in the CBR cases depend on the behavior definition and goals.

3.1 Case Definition

Case definition requires an input instance, related to the necessary knowledge for the behavior operation; a solution which defines the response of the behavior, either a concept or a motor action; and some measure of efficiency which is also related with the performance of the behavior in the fulfillment of its goal.

In our vision-based robot the input cannot be the whole image frame or a sub-sampled version, as its dimension would be proportional to pixel resolution. Using Principal Component Analysis(PCA), histograms, or color moments would help, but redundancy could affect negatively the performance of behaviors, as they are supposed to work only with local information. For this reason, qualitative techniques have been proposed as the better option for active vision applications. On the bad side, input feature selection in qualitative methods, depends heavily on the specific domain of the goal or problem to solve: final selected features should be representative, easy to obtain from image, and robust against artifacts such as changes in light conditions or occlusions. In our example, possible input features include descriptions of the objects found at each image. To obtain that description, images are codified in a HSV space and a fast segmentation algorithm is run to obtain different color blobs, represented by their centroid and area, plus possible clipping in case the object is at the boundary of the image; data is normalized to their extreme values. Besides objects description, the case includes a normalized PAN and NOD angle of the robot head, as it affects the relative positions of objects in a frame (see figure 2).

To reduce the amount of possible cases, input instance components are discretized/indexed; this process allows us to turn knowledge from numeric-algorithmic data to qualitative or conceptual information. The area of an object, for example, is not represented by its normalized number of pixels, but by a description such as "minimum area" or "medium area" which, besides of making easier to debug the "reasoning" process of the robot, allows to develop more complex concepts to describe the state of the environment and robot with respect to the goal of the behavior, i.e., "minimum area" and "high Y-centroid" would represent a "far object". Discretization is a key element for the definition of the input instance, and is also dependent of the goals and design of the behavior. At the moment a human expert heuristically decides the number of indexes and their boundaries for every component in the considered behavior. In the future would be desirable that such decisions were automatically taken by means of statistical analysis of the continuous input data obtained from the supervised learning phase.

In our original proposal the final output of a case was a single vector of instant motion, expressed by components of rotation, translation and strafe, which defined the robot trajectory until a new CBR consultation was done. This single vector output has a lot of inconveniences regarding a possible combination of several behaviors output in a similar way to the Potential Fields approach(PFAs), as is very prone to result in a null vector motion in legged robots. For this reason we use now a *chain of vectors* (figure 2), to represent a more complex movement pattern, but also easier to combine with other patterns from other low level behaviors. Size of the chain would depend on the desired granularity of the output but also on the reactiveness of the robot response as, with a too long chain, CBR consultations would be too sparse in time to obtain a fast enough reactive overall behavior. Output also includes additional values to represent special actions (shoot, dive,..) triggered in specific situations. Finally, an efficiency function at

the behavior module weights the utility of the output of a case, related to the goal of the behavior. Specific components of efficiency functions are behavior dependent, so they are usually defined by human experts, considering not only the objects and elements linked to the behavior, but also concepts as "softness" or "security" in the robot motion. Aspects related to efficiency function an case adaption will be addressed in future works.

3.2 Casebase structure and case learning

Casebase structure defines the organization of the cases at the database, and plays an important role in the retrieval phase of the CBR cycle. Stored cases are usually organized either on a flat memory or a hierarchical one:

- In a flat memory, cases are stored in a simple list, and the retrieval of a case is done by comparing the problem case with every case stored at the casebase, and retrieving the case best matching the problem description. This comparison is done using a distance or similarity function over the components of both the problem and each stored case. Despite the simplicity of this implementation and the easiness of addition of new cases, there are scalability problems when the number of stored cases grow up, as the recovery time is proportional to such amount.
- In hierarchical memories, cases are stored in the form of a graph or tree, in which every branch and leaf corresponds to a component of the case. Hierarchical memories provide a more efficient case retrieval, but the retrieval process could miss some adequate cases while searching, because once you have choose a path (depending on the content of the nodes), the cases you can access to are a subgroup of the total, thus making impossible to retrieve a case included in other area of the tree.

While in [13] we used a flat structure for the casebase of our behaviors, the increase in complexity of the cases at the new proposed behaviors, drove us to find another type of structure which allows to manage a great number of cases without being detrimental to the speed of retrieval phase. A purely hierarchical structure would suffer from the aforesaid problem of taking a wrong path which wouldn't allow to obtain the most suitable case. So, instead we have considered a compounded structure that mixes both viewpoints: our database is distributed over a tree in which final leafs are flat sub-casebases (see figure 3 for an example).

Traversing trough the tree is done by means of specific context information, which can be obtained from the problem presented to the behavior. This information is mutually exclusive, so there is no possibility of ending in a wrong leaf-casebase. This information context could be, for example, the presence of the own goal or the rival goal, which would be related to a very different set of cases, which could be separated in different leaf casebases. As we can access the right casebase very fast, we achieve a great improvement in casebase organization and retrieval with a minimum cost. Nevertheless, we must point out that is not always possible to find "context" components which allow to build the tree part of this mixed structure, so not all behaviors could enjoy this benefit.

Knowledge at the CBR casebase is obtained in a two stage procedure(figure 4). First, a *learning by observation* approach is taken to seed the casebase with a set of initial cases. In this training phase, a human trainer guides the robot using a conventional joystick to operate the desired behavior. The casebase is seeded from the cases learned during the different runs. The main advantage of this learning approach is that humans implicitly cope with kinematics and dynamics related to the robot, which are not easy to parameterize. So the robot, through this learning, can absorb them as well. Also, the intrinsic mechanisms related to the behavior are automatically incorporated to the robot knowledge without the necessity to fully understand their rules and parameters. This type of learning is typical in human education, as when a tennis trainer handle the arms of a pupil to show him how to play a forehand shot. Additionally, it's possible to manually add cases to the casebase, but a good understanding of the case components influence over the behavior operation is needed to do so. When a primary casebase has been obtained and the robot can run a basic operation of the behavior, a second stage of learning-by-own-experience is performed. The robot works in autonomous mode with no external control or supervision, retrieving from the behavior's casebase the most similar case to the current situation of the environment and robot state. This retrieval return not only the most similar stored case, but a measure of the similarity to the given problem. If the dissimilarity is over a threshold, it means that the robot is facing a new problem, so the output of the retrieved case should not be directly applied, but *adapted* instead. When adapted, output is applied and its performance is evaluated using a *efficiency function* which decides if the new case is good enough to be incorporated to the casebase.

3.3 Retrieval algorithm and similarity functions

One of the key aspect for the success of a CBR based reactive system depends on having a good retrieval algorithm which can recover the more suitable case to the presented problem, but also fast enough to allow a good response time. Flat casebases have a scalability problem because the problem must be compared to every case stored at the casebase so, when its size grows up too much, the response time can be too high for a good reactiveness of the module. Although discretization and/or clustering or the input instances can relieve this problem lowering the total amount of cases, complex behaviors could still end with a high number of stored cases. The proposed mixed tree-flat structure of the casebase contributes to improve the retrieval time, as total knowledge is distributed among the leaf casebases, reducing the number of cases to compare in the retrieval stage. Once in the right leaf-casebase, and to further improve the retrieval operation at the CBR casebase we have introduced a two stage similarity function. Traditional similarity functions quantify the similarity of two cases by computing the distance between each of its components in a multidimensional space, weighting every local distance according to the influence of the component on the case representation, and aggregating weighted component distances using different functions. The choice of the distance function is problem

dependent, and there is not a clear translation from one domain to other. An influential hypothesis has been that Euclidean distance is valid when stimulus dimensions are perceptually integral(such as the brightness and saturation of a color), whereas city-block distance is appropriate when stimulus dimensions are perceptually separable (such as the color and shape of an object)[4]. In our proposal we use a Manhattan normalized distance to compare problem and case similarity, but previously we do a pre-filtering stage in order to reduce the possible candidates to the recovered solution by means of a feature based distance, a *retrieval by objects in scene*. In this first stage, objects in scene are identified and a binary index with their presence(1) or not(0) is build. In our input instance, the descriptors of the elements related to the behavior are placed always in the same position, which is also used to represent the presence of the object in a string. Every case in the casebase has an associated index string representing the objects in the associated scene. When a problem is presented, the corresponding index string is generated and compared to the cases' ones using a Jaccard distance. A list with the most similar cases is generated, so the second retrieving stage carries out only on a subset of the casebase contents, thus improving the speed of the overall retrieving operation (figure 5).

4 Experiments and results

In this section we present an example of behavior design following the guidelines given above. The behavior is defined in the controlled framework of a Robosoccer application in which two goals, a ball, maximum of two rival players and one team player are considered. The goal of the proposed behavior will be reach the ball with the presence of other players in the field. This behavior is quite more complex than the original one of just reaching the ball alone, as it is not enough to aim the ball and walk straight to it; depending of the rivals and friend position more complex maneuvers must be achieved in order to, not only reach the ball, but circle an obstacle, block the rival and/or face open field when the ball is finally captured. The input instance is defined by the presence of ball and player objects; goals won't be included as we consider they don't affect the performance of this specific behavior. The case input includes also pan and nod of the robot head, as stated in section 3.1. As behavior is driven by visual information, a tracking function has been developed to automatically move the head in order to maintain on-board camera focused on objects which are significant to the behavior. Case output will be a chain of three temporally consecutive motion vectors. Learn by observation is achieved through several runs from different starting positions of our robot, ball and obstacles. In figure 6 we can see a set of diverse scenarios trained for the considered behavior, together with an example of a run experiment.

Cases are acquired at a rate of 4 for second, and intervals of discretization were chosen analyzing the evolution of the continuous data taken at the training phase. Initially we considered five equally spaced intervals or classes at each case component; but after an analysis of some runs one more class was added to the

Pan component to break symmetry in its description; and area of objects was finally categorized into seven classes with interval size increasing in a quadratic form. Categorization allowed to compress knowledge from about twenty five thousand cases to barely three thousands. Categorized cases with the same input and different output were grouped by a majority vote.

Robot performance in autonomous mode was almost perfect from the same trained situations, but also in a great variety of similar ones(figure 7). Nevertheless, a few scenarios that included input instances too different to the stored at the casebase showed a poor attainment, proving the necessity of a second stage of learn-by-own-experience case acquisition, as a complete supervised training at all possible situations is neither possible nor desirable. Also, categorization of the database allowed us to debug the execution of the experiments in a easy way.

In figure 8 we can see the improvement in retrieval time using our casebase structure and modified CBR retrieval method, with respect to the original flat casebase, under different situations. We gradually increased the complexity of the learned behavior by adding more objects and situations to the scenario, thus increasing the amount of knowledge needed at the behavior. We can see the evolution when space dimension of discretized case components grows, which increase also the dimensionality of possible cases at the behavior's casebase. We tested two discretization levels for the case components: a coarse one, as stated in the previous paragraph; and a fine one, in which we doubled the categories or discretization levels of each component, with respect to the coarse one. Retrieval time at the flat casebase is $O(cn)$, where c is the number of components of a case, and n the amount of cases at the casebase. Using our two stage procedure we can reduce the set of candidates to the second stage, thus improving the final retrieval time. The enhancement is proportional to the amount of cases in the sub-sets after stage one. As first stage filtering is a $O(n)$ operation, only situations with all cases in just one sub-set would show a worse performance. Increase in complexity of the behavior, with more objects and case components; and/or a finer-grained discretization, would magnify the advantages of our approach, as dimensionality of the space of possible cases would be greater.

5 Conclusions and future work

In this work we have presented an approach to design CBR-learning reactive behaviors for vision-based autonomous robots, when complexity of the target behavior increases, as there are more elements involved in the behavior's design, and scalability problems arise. This complexity affects the dimensionality of the possible cases which hold the knowledge of the behavior, and increases retrieval time which is directly related to this aspect. A good reactive behavior should respond quick enough to changes in environment and problem conditions, so a too-long retrieval time is unacceptable in any reactive layer. In order to solve these problems we have proposed a qualitative representation of the case components, by means of a indexation/discretization process which allow us to reduce

the maximum amount of possible cases at the CBR casebase, and also describe the behavior's scenario using abstract concepts instead of numbers. This description, besides making easier the design and debug process of the behavior, would allow us to build more and more complex concepts by aggregating simpler ones, in a "nest-structure" fashion, which agrees with modern theories or human intelligence structure and organization. We have verified the good performance of our previous work, under more tight conditions, with the development of an example of a complex behavior. This complex behavior allowed us to show how scalability problems could arise and compromise the good performance of the behavior reactiveness. We have proposed a new mixed hierarchical-flat casebase structure and a two stage case retrieval algorithm, in order to speed up the CBR operation and allow a good behavior response, despite the increasing of information inside the casebase. We have proved the validity of this scheme, whose benefits become more evident when the size of the casebase increases, as we have shown in different experiments.

Future work will focus on combination of different reactive behaviors to obtain an emergent complex one, which would both represent a higher level concept build from its compounding behaviors, but also an instruction of the actions the robot should take at this emergent behavior, derived from the corresponding actions at the low-level behaviors. Furthermore, we want to study more exhaustively the learning-by-experience stage, to improve new knowledge acquisition and casebase maintenance at the behaviors. In order to do so, we must develop better case adaption algorithms and adequate efficiency functions for the new adapted cases.

6 Acknowledgements

This work has been partially supported by the Spanish Ministerio de Educacion y Ciencia(MEC), Project n.TEC2011-29106 and Andalucia TECH:Campus of International Excellence.

References

1. A. Aamodt and E. Plaza. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI Communications*, 7(1):39–52, 1994.
2. E. Aguirre and A. González. Fuzzy behaviors for mobile robot navigation: design, coordination and fusion. *Int J Approx Reason*, 25(3):255 – 289, 2000.
3. R. Brooks. A robust layered control system for a mobile robot. *IEEE J Robot Automat*, 2(1):14 – 23, 1986.
4. W.R. Garner. *The processing of information and structure*. The Experimental Psychology Series. L. Erlbaum Ass.; Halsted Press, NYC, 1974.
5. J. Hawkins and S. Blakeslee. *On Intelligence*. Owl Books, 2004.
6. O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *IEEE 1985 Int Conf Robot*, volume 2, p. 500–505, Mar 1985.
7. M. Kruusmaa. Global navigation in dynamic environments using case-based reasoning. *Autonomous Robots*, 14(1):71 – 91, 2003.

8. H. Liu and H. Iba. *Genetic and Evolutionary Computation, GECCO 2004*, chapter Humanoid Robot Programming Based on CBR Augmented GP, p. 708–709. *Lect Notes Comput SC.*. Springer Berlin/Heidelberg, 2004.
9. K. H. Low, W. K. Leow, and M. H. Ang, Jr. A hybrid mobile robot architecture with integrated planning and control. In *Int J Conf Auton Agent Multi Ag (AAMAS '02)*, pages 219–226 , NY, USA, 2002.
10. Maja J. Mataric. *Interaction and Intelligent Behavior*. PhD thesis, Department of Electronic Engineering and Computer Sciencie, 1994.
11. R. Murphy. *Introduction to AI Robotics*. MIT Press, Cambridge, MA, USA, 2000.
12. J. C. Murray, H. R. Erwin, and S. Wermter. Robotic sound-source localization architecture using cross-correlation and recurrent neural networks. *Neural Networks*, 22(2):173–189, 2009.
13. J. M. Peula, C. Urdiales, I. Herrero, I. Sánchez-Tato, and F. Sandoval. Pure reactive behavior learning using case based reasoning for a vision based 4-legged robot. *Robot Auton Syst*, 57(6-7):688–699, June 2009.
14. C. Urdiales and A. Poncela and F. Sandoval. A CBR approach to behaviour-based navigation for an autonomous mobile robot. In *2007 IEEE Int Conf Robot*, Rome, Italy, 2007.
15. R. Ros, R.López De Mántaras, J. Arcos, and M. Veloso. Team playing behavior in robot soccer: A case-based reasoning approach. In Springer-Verlag, ed., *Lect Notes Comput SC., Proc. of the EICCBR 2007*, n. 4626, p. 46–60, 2007.
16. E. Tira-Thompson. *Tekkotsu: a rapid development framework for Robotics*. PhD thesis, Carnegie Mellon University, Pennsilvania, 2004.
17. M. Wang and J. N. K. Liu. Fuzzy logic-based real-time robot navigation in unknown environment with dead ends *Robot Auton Syst*, 56(7):625–643, 2008.
18. W. Xie, J. Ma, M. Yang, and Q. Zhang. Research on classification of intelligent robotic architecture. *Journal of Computers*, 7(2), 2012.

List of Figures

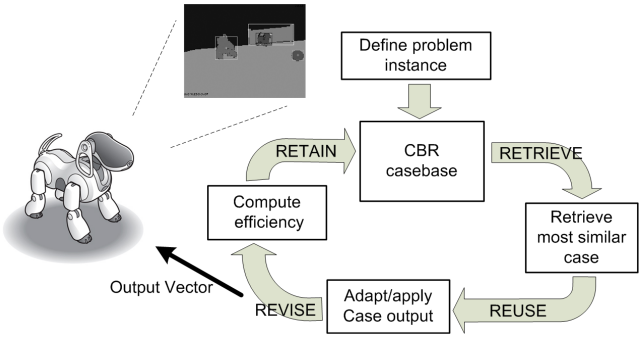


Fig. 1. CBR cycle

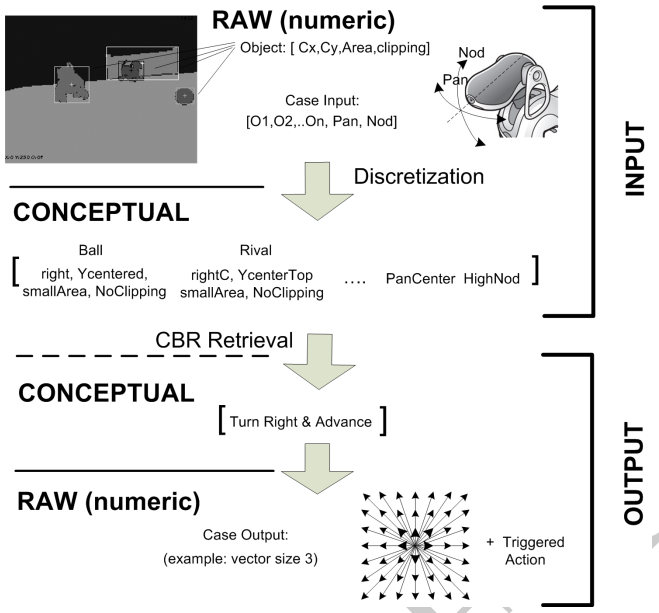


Fig. 2. Case definition: Input instance elements and output response

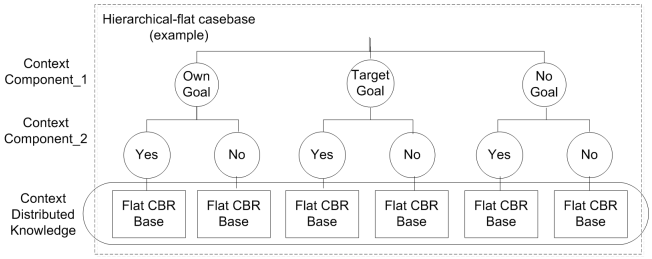
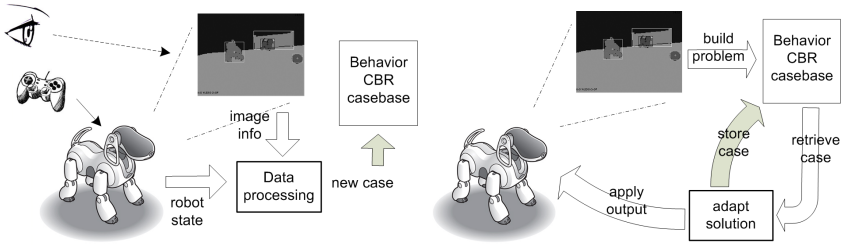


Fig. 3. Mixed Hierarchical-Flat casebase structure



(a) Learning by Observation (b) Learning by own-experience

Fig. 4. Two stage learning in our proposal framework

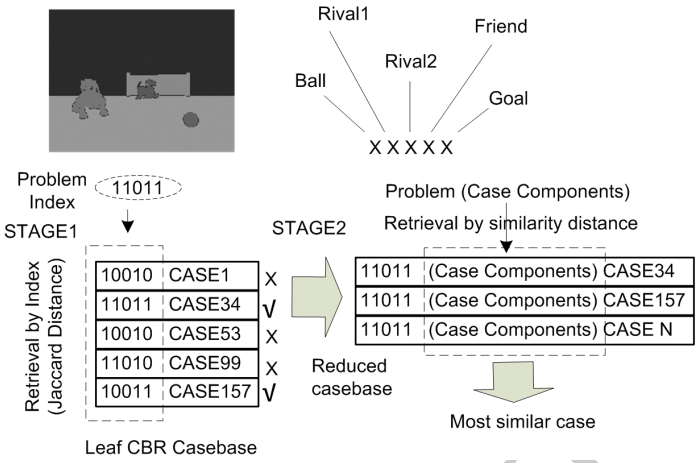


Fig. 5. Two stage retrieval operation at the CBR casebase

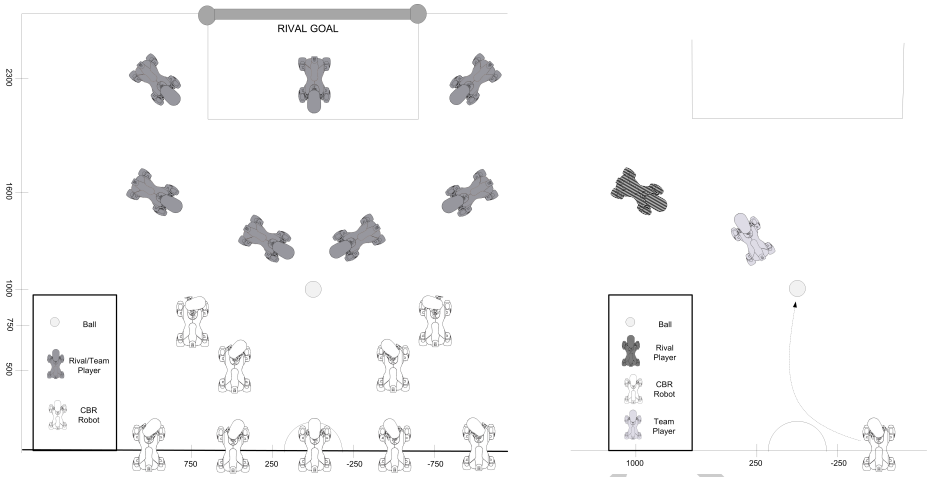
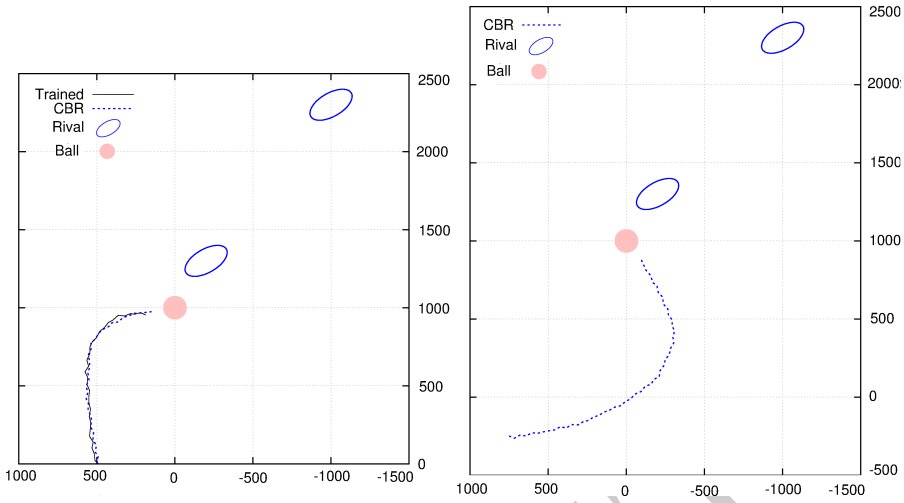


Fig. 6. Set of trained situations. Example of a learned run



(a) Trained situation (b) Untrained situation

Fig. 7. Performance example of designed behavior for both trained and untrained situations

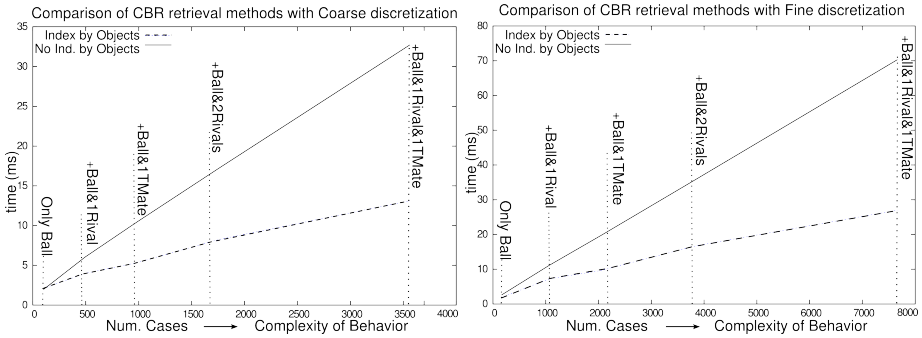


Fig. 8. Comparison of retrieval times using a two stage retrieval method