Impossibles Architecture for Aibo Four-Legged Soccer in RoboCup 2007

Hamid Reza Vaezi Joze, Kianoosh Mokhtarian, Nima Asadi
Ali Kamali, Mahdi Safarnejad, Navid Zalghadr, Hossein Kaffash
Department of Computer Engineering
Sharif University of Technology

Abstract

AIBO four-legged soccer league is considered as one of the most popular test-beds for Artificial Intelligence (AI) and Robotics. This paper presents Impossibles AIBO 4-Legged main architecture for RoboCup 2007 which is going to be held in Atlanta, USA. This architecture includes different modules such as World Model (WM) module, Vision, Action (Decision Making and Motion Controller), Communication, and Localization modules. These modules are explained briefly in this paper. The whole parts of our architecture are being implemented from scratch when we start work on Aibo league in previous year and in this year we mainly start working an improving different module using scientific approaches and also working on new tools for simplifying the process of developing code on robots. "Impossbiles" is the first Iranian team (unfortunately the last team) participating in AIBO 4-legged competitions. There were no 4-legged Soccer local competitions in Iran; hence, we have not been able to participate in such contests; however, we believe that "Impossibles" strong background in RoboCup, our previous experience and Team Report prove our endeavor in Artificial Intelligence and Robotics Laboratory (AIRL). Some of our scientific results have been published in [5],[6],[7].

1. Introduction

As a research group, *Impossibles* team has been set up in Artificial Intelligence and Robotics Laboratory (AIRL) of Computer Science and Engineering Department at Sharif University of Technology since March 2004. Research Areas of *Impossibles* were categorized into three groups including Artificial Intelligence (Machine Learning, Multi-Agent Systems, and Reasoning), Theoretical Computer Science (Algorithms, and Data Structures), Soft Computing (Fuzzy Theory, and Genetic Algorithms).

RoboCup's interesting features attracted us to begin implementation of our ideas in Rescue Simulation Environment (RSE) to participate in RoboCup2005 in Osaka [4]. So it was our first participation in such international competitions. Having coded from scratch, we applied our new ideas. Consequently, *Impossibles* got world championship in Rescue Simulation League in Osaka 2005.

As Second RoboCup experience we start working on 4-legged Soccer which have more serious robotic and multi agent environment. Participating in RoboCup2006 in Bremen in the first year of working on Sony Dogs (AIBO), the result is not as we want and we dropped in the first round. However defeat is a bridge to win and we learnt many things from Bremen. The most important one is the necessity of out of robot software to adjust parameters of algorithms in new environment and also developing suitable debug routine to reduce debug time.

In order to overcome these problems we start developing two softwares: (i) Aibo Controller which enables us to do most of the process in a computer which connects to the Aibo by wireless connection. These processes could be Color Segmentation, Object Detection and Localization, (ii) Aibo Simulator which is a high level simulator to test strategies and high-level decisions. In this paper, the *Impossbiles* AIBO architecture and its subsystems are explained briefly. For more detail about each subsystem refer to our technical report [3].

2. Architecture

Our previous experience in Multi-Agent System (MAS) architecture design in Simulation league environment and last year experiment led us to World Model Based Architecture (WMBA). We employ it as our basic design architecture for concurrently-running objects of Open-R SDK. WMBA contains three major tasks that are done independently in following subsystems:

1. Sensing Subsystem
2. Communication Subsystem
3. Action Subsystem
4. Debugging Subsystem

These subsystems are run repeatedly with different frequencies. They are also managed in such a way that objectives are achieved and constraints are convinced. The main constraint of the AIBO robots is the limited resources such as CPU. The last subsystem which is new from previous architecture is in charge

of gathering appropriate information from other subsystems to communicate Aibo Controller software which will be described later.

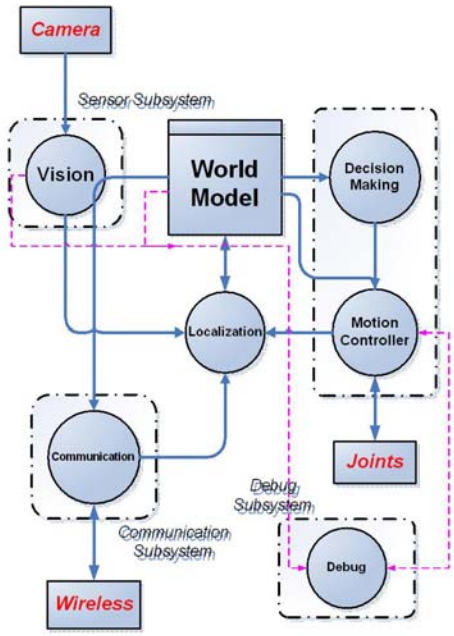

Figure 1: World Model Based Architecture

Figure 1 demonstrates the World Model Based Architecture. As in DFD diagrams, subsystems are denoted by dotted rectangles, data flow is shown as arrows, and processes are shown via circles.

To make Decision Making module completely separate from other parts, we use a non Open-R abstract class called *AbstractPlayer*. There are two pure virtual functions in this abstract class. The first one is *sense* which is called every time robot gets some information. The second virtual function is *decisionMaking* which is called in a specific period of time to decide about new actions. The result of this function could be any action such as walk, shoot, look or other action. There are some actions such as standing back to normal position in the case of falling down and also some actions that we called them Blocking Skills will be done completely so *decisionMaking* is not called during these actions (we will discuss about them in more detail in motion section).

3. Decision Making

Since there is no central processing, Decision Making (DM) is done via distributed manner. Each player decided for itself based on the information from its sensors, from other teammates and from game status. After deciding, each robot informs other teammates about its decisions so that they can update their status and strategy.

a. Architecture

DM gets needed information from the sensors (including vision), motion controller and other robots (via communication). Collected data is analyzed for choosing team strategy first (described later). After this step, each player's role, position and action can be decided. Actions include shooting the ball, walking to a specified coordinate, finding the ball and grabbing the ball. These information are shared among all players, for example, when the position of the ball is determined by on of the players, it informs other players so that they no longer need to search for the ball.

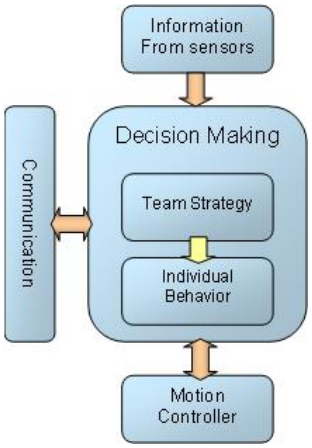

Figure 2: architecture of DM module

b. Team Strategy Database

Overall team strategy is determined by a fuzzy function which inputs are the remaining time, current score, and opponent strength. By using this information and the output of the fuzzy function, team strategy could be complete offence or complete defense.

Due to CPU usage problem in AIBO robots, Impossibles AIBO robot uses a team strategy database to avoid CPU consumption. Team Strategy Database (TSD) contains pre-calculated and analyzed values which can be used for team strategy. Besides, TSD also contains proper positions for each player. So when overall team strategy changed, players can find their new positions from TSD quickly without any calculation.

c. Individual Behaviors

Individual behaviors are affected by overall team strategy, ball ownership and information from sensors and vision. After analyzing these data, each player decides for itself. These individual behaviors consists looking, walking, grabing the ball and shooting which are described in [3].

4. Vision

Vision problem for these robots which refer to recognize type and location of surrounding objects is described as follows:

- Input: (i) The output of robot camera in the form of 30 pictures per second, with the size of 208x160 pixels and in the YCrCb color space consisting color distortion and noise. (ii) The value of robot's joints through which the value of camera's pan, tilt and roll can be obtained.
- Output: (i) Robot's distances and angles in relation to the ground's fix location by which the robot estimates its position in the field. (ii) Robot's distances and angles in

relation to moving objects which determine their position relative to the robot.

The first work carried out on the image received from robot's camera is the correction of distortion existed in the color of image pixel far from the center of image. Then the color of each pixel in the corrected image is attributed to one of the predefined color class (green, white and et. al) or to a class allocated to unknown color to specify existing color areas in the image.

Then, existing blobs of each color (connected component of image's point with the same color) are formed for the color of existing objects in the image, and based on the color, size and density of existing objects in the image is recognized. Also, by using a method provided for obtaining the three dimensional coordinates of each pixel in the image in the outside space relative to itself, position of each object relative to the robot is calculated. Specially, ground's lines are among of important objects in the soccer field which are distinguished in a separate manner with seeking green-white edges. Finally, having determined the position of objects relative to the robot, the position of robot in the field is calculated through the objects having a fixed place in the environment.

Figure 3 display the architecture of robot's real-time vision system which is based on our research in [6]. Also this system require offline processing regulating some provided algorithms' parameters for various setting prior to the application of software on the robot.

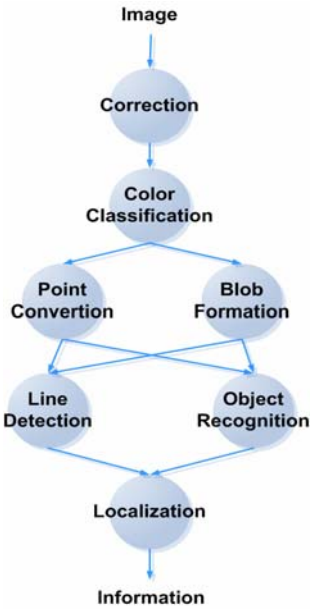

Figure 3: architecture of real-time vision system of robot

5. Localization

We use a localization algorithm to localize soccer player robots in the field which is called piecewise Linear Probability Distribution Localization (PLPDL) [5] that besides designing for low-performance and low-memory machine can localize robots in the uncertain and dynamic situation. The approach is based on Markov localization [9] which localize robot probabilistically. Our approach inherits from Markov localization the ability to localize a robot under global uncertainty. The main idea of this approach is separating probability density functions of random variables which determine robot position. Using piecewise linear functions to approximate probability distribution functions of these random variables would help our approach to be fast and inexpensive which is suitable for real-time processing of 4-legged soccer robots. We describe PLPDL and its application in Aibo robots in [5]. There is tow main update in PLPDL which are come from other modules.

- ***Movement Update:*** We consider X a random variable and its probability density related to x position of robot and ΔX as a random variable of movement of robot in x dimension and its probability density. So the new value for X will be $X + \Delta X$. In this way the corresponding density function for X is obtained. We know from probability theory that if X,Y,Z are random variables and Z=X+Y so probability density of Z could be conclude by convolving probability density of X and Y. And also other random variables will be updated independently using motion data that should be in the form of different probability density function for each random variable.
- ***Sensor Update:*** Sensor is usually return to the vision module in robot. However it could be any other sensor for localizing mobile robot. If we suppose sensor data return a probability density function for each variable such as X and p that is the belief of correctness of these data. Now we should create a new probability density for X by the following formula using previous density of X and sensor data in the form of new density function and our belief of its correctness:

$$Fx_{New} = Fx_{old} \times (1 - p) + Fx_{Sensor} \times p$$

6. Motion

We divide usage of motion skills of Aibo robot in two categories: Blocking Skills and Non-Blocking Skills. While performing a non-blocking skill, Decision Making (DM) module can make new decisions, if necessary, and send an interrupt to Motion Controller module. Thus, Motion Controller will halt the previous action and start performing new commands. Walk and rotate are included in non-blocking skills. On the other hands Blocking Skills are smallest consecutive segment motions which are not needed to be interrupted such as different kind of shoot and blocking ball by goalie.

a. Blocking Action

Blocking Skills are smallest consecutive segment motions which are not needed to be interrupted. So Decision Making should use them in suitable situation. These actions contain consecutive value of joints which can be utilized from a static look up table that is not change during the game. We use our structure to store these date in files, therefore we have separate file for each blocking skill. These actions contain all type of shoots, block by goalie and stopping the ball.

b. Non-Blocking Action

The main non-blocking action is walking. We modeled the walk of a robot as movement of Aibo's paws on the perimeter of an ellipse. In this model, movement of the left front leg is synchronized with movement of the right rear leg. This is true for the right front leg and the left rear leg too. This resembles natural behavior of animals in their normal movements.

To form the ellipse based on which the robot plans to move, walk needs number of parameters. These parameters are as follows:

- Semi-major axis of the ellipse, with the symbolic name of *a*
- Semi-minor axis of the ellipse, with the symbolic name of *b*
- Coordination $(x_{0,} y_{0,} z_0)$ of the center of the ellipse.

7. Tool

Running compiled code on AIBO is time-consuming and inefficient in many cases. Thus, lack of a debugging software and a simulator for testing and debugging purposes was felt and the need was obvious.

Therefore, Impossibles spent months on writing tools which let us debug the codes running on Aibo platform and simulate the codes without having robots available. These tools consist of AIBO Controller and AIBO Simulator.

c. AIBO Controller

This software is divided into two units cooperating with each other. A *DEBUG* module written to memory stick (which can be companied with other soccer modules and run in parallel with them) and a client software running on the PC.

DEBUG module sends collected data to the client program. Data consist of all internal states of other soccer or non-soccer modules (provided for other challenges) in addition to other internal representations of the robot (including joint data, images, body sensor data, world states, perceptions, etc.)

On the other side, the client application receives the data and makes it possible to visualize and analyze them via different internal implemented algorithms and other user defined ones. This will give us opportunity to run vast of algorithms and methods in different fields (Image Processing, Motion, etc.) based on data collected.

On the other direction, since reaching to the state which revealed bugs in a nondeterministic and real environment is somehow impossible, setting parameters to return back to the desired state, can be found in our controller application.

Channel between *DEBUG* module and the client program is a wireless connection. Results from experimental statistical analysis of protocols in a wireless connection [8] led us to choose packet sizes and protocols in a way to achieve higher throughput.

To conclude, a list of features provided by *AIBO Controller* is followed:

- *Visualization* and *analysis* of data, especially intermediate representations of other modules running in the robot
- Turning *on* or *off* different algorithms and parts of the code for debugging purposes
- *Modification* of robot's state and parameters to algorithms
- Run different algorithms solely on the collected data on the client side instead of the robot itself.

d. AIBO Simulator

Writing codes to memory stick after each change and waiting for the robot to load modules, all occur for several times and all slow down the process of code development. Thus, a simulator was developed by *Impossibles* to simulate the codes on the PC instead of the Aibo itself.

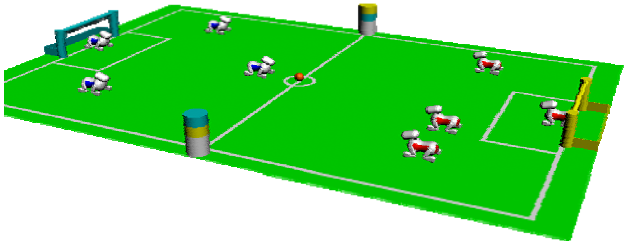

Figure 4: Snapshot of AIBO Simulator

Architecture of our simulator is shown in **Error! Reference source not found.**. It consists of three separate modules with well defined interfaces.

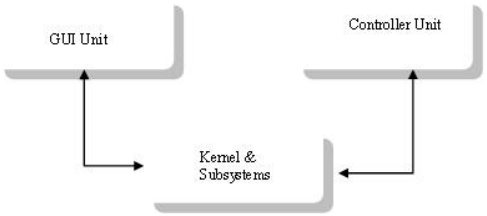

Figure 5: AIBO Simulator's architecture

i. Controller Unit

Currently, the code developed for robots can not be used in simulator directly. Some changes have to be performed on the codes before use.

Modules developed for the simulator should implement a shared interface called *AbstractAIBOModule* in order to have the authority of running inside the kernel.

AbstractAIBOModule provides some ports for communication purposes –among subsystems of the kernel and user implemented modules– and some abstract methods like what Open-R objects should implement. Ports should be configured to send and receive data from kernel defined ports and these configurations can be done in a file like *stub.cfg*.

After preparation steps, the code can be compiled with regular gcc compiler and linked to the simulator. Then by

running the simulator, simulation of the code is possible and results can be seen graphically.

ii. Kernels & Subsystems

Inside the simulator is the kernels and other subsystems which do the main task of simulation. The structure is shown in Figure 6.

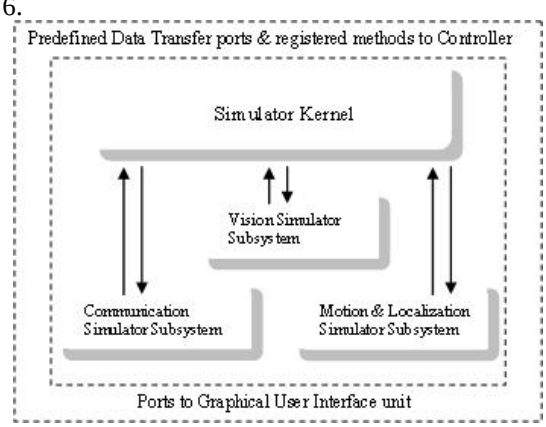

Figure 6: Kernel & Subsystem's structure

Kernel is designed to control other subsystems, execute commands from Controller and models the environment as inputs for GUI Unit. Kernel breaks high level commands received from Controller to low level commands and passes them to corresponding subsystem. On the other direction Kernel collects data of each frame from different subsystems and report them to Controller unit through ports.

Modeling the environment is also a responsibility of kernel. All functions for drawing models are a part of kernel. By using OpenGL, modeling a 3D environment is possible and suitable in many aspects.

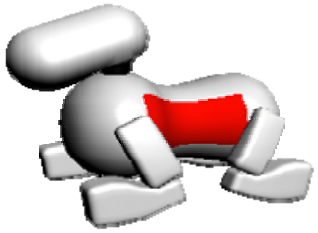

Figure 7: AIBO model in our simulator

Vision Simulator Subsystem prepares virtual images for the simulated robots. Since the simulation is done in a 3D environment, making virtual images is possible and each robot can have its vision module run with these data as inputs. For better results, lighting conditions can be adjusted at starting point.

Communication Simulator Subsystem is responsible of communicational links among robots. This subsystem is given all realistic network parameters at first (which contains Bit Error Rate, possibility of data loss and the condition in which the virtual wireless system should work), then it simulates all connections and data transmission among all robots by message passing techniques.

There is a *Motion and Localization Simulator Subsystem* (MLSS) which receives data from the kernel and moves the robots to the place where they are expected to be. Based on the fact that even movement of robots might be influenced with errors and this will result in unexpected locations, the MLSS will apply error functions. Therefore, as what happens in a real world, robots might not be exactly in the place they were expected to be. This will make simulation resembles the real world more.

iii. Graphical User Interface

Our simulator interface includes an editor in which scene files are defined. This increases the flexibility of environment modeling. As a result, we can place our robots in an environment rather than soccer field.

iv. Future Works on Simulator

In the current simulator, the codes developed for robots have to undergone changes in order to be used for simulation. Thus, one of the ambitions is to change the architecture of our simulator by removing the Controller unit and adding a Code Emulator. By emulating the code, we can use robot codes without any changes applied.

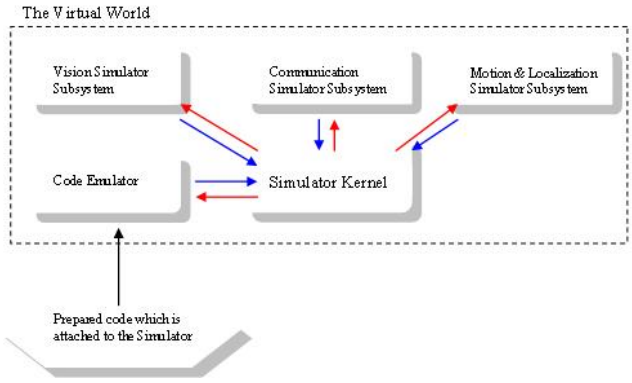

Figure 8: Simulator's structure in the future

Figure 8 shows the structure of our simulator in the future.

Connecting the real robots to the simulator and make it possible for them to play the role of virtual robots is also another ambition of the team. With such tool, we can configure the simulator to judge contests among virtual robots and real physical robots in a soccer game.

Another long term plan is to generalize the idea of simulation to other robots. This idea takes long time to be completed and might need extra part to be added to current structure.

8. Conclusion

In this paper, the *Impossbiles* AIBO architecture and its subsystems were explained briefly. Additionally, we presented *Impossibles* research group's backgrounds in RoboCup, and our competitive and research objectives. As the first Iranian team has experience of participating in RoboCup2006 AIBO 4-legged

League in Bremen, we intend to be ranked as one of the top teams in RoboCup2007 besides our scientific results.

References

[1] H. R. Vaezi Joze, S. Aliari Zounoz, S. Rahbar, M. Valipour, A. Fathi, *Impossibles Aibo Four-Legged Team Description Paper RoboCup 2006*, RoboCup 2006, June 2006.

[2] J. Habibi, S. Aliari Zounoz, H. R. Vaezi Joze, S. Rahbar, M. Valipour, A. Fathi, K. Mokhtarian, A. Kamali, *Impossibles Aibo Four-Legged RoboCup 2006 Report*, RoboCup2006, Bremen, Germany, June 2006. http://ce.sharif.edu/~impossibles/soccer_2006/impossibles_report_2006.pdf

[3] J. Habibi, H. R. Vaezi Joze, K. Mokhtarian, N. Asadi, A. Kamali, M. Safarnejad, N. Zolghadr, H. Kaffash *Impossibles Aibo Four-Legged RoboCup 2007 Report*, http://ce.sharif.edu/~impossibles/soccer_2007/impossibles_report_2007.pdf

[4] J. Habibi, A. Fathi, S. Hassanpour, B. Sadjadi, M. Ghodsi, H. R. Vaezi Joze, M. Valipour, Moahammad Reza Ghodsi, Alireza Fathi, *"Impossibles Team Description"*, RoboCup, Osaka, Japan, 2005.

[5] H. R. Vaezi Joze, J. Habibi, S. Rahbar, *piecewise linear probability distribution localization: fast and inexpensive mobile robot localization*, (to be submitted)

[6] K. Mokhtarian, H. R. Vaezi Joze, J. Habibi, *An Inexpensive Approach for Real-Time Vision on Four-legged Footballer Robots*, In Proc. 12th International CSI Computer Conference, Feb 2007.

[7] H. R. Vaezi Joze, M. Ghodsi, *Design and Implementation of soccer software for Aibo robots*, B.S. thesis, Computer Engineering Department at Sharif University of Technology, 2006.

[8] G. Xylomenos, G. C. Polyzos, *TCP and UDP Performance over a Wireless LAN* , Proc. of IEEE INFOCOM, pp. 439-446, 1999.

[9] D. Fox, *Markov Localization: A Probabilistic Framework for Mobile Robot Localization and Navigation*, Institute of Computer Science III, University of Bonn, Germany, Doctoral Thesis, December 1998.