Internet of Things and Cloud Computing
2014; 2(1): 1-5
Published online May 30, 2014 (http://www.sciencepublishinggroup.com/j/iotcc)
doi: 10.11648/j.iotcc.20140201.11

A method to increase the computing speed of 2 on 2 soccer robots by implementing distributed computation in multi-processor architecture (experimental research)

Yahya Hassanzadeh-Nazarabadi[1], Sanaz Taheri-Boshrooyeh[2], Milad Bahrami[2], Danial Bahrami[3]

[1]Mobile Robots Department, Parse Lab of Robotics, Mashhad, Iran
[2]Computer Engineering Department, Ferdowsi University of Mashhad, Mashhad, Iran
[3]Semnan University, Semnan, IRAN

Email address:
ya_ha_na@ieee.org (Y. Hassanzadeh-Nazarabadi), staherib90@gmail.com (S. Taheri-Boshrooyeh), milad.1821370@gmail.com (Milad Bahrami), danial70.bahrami@gmail.com (Danial Bahrami)

To cite this article:
Yahya Hassanzadeh-Nazarabadi, Sanaz Taheri-Boshrooyeh, Milad Bahrami, Danial Bahrami. A Method to Increase the Computing Speed of 2 on 2 Soccer Robots by Implementing Distributed Computation in Multi-Processor Architecture (Experimental Research). *Internet of Things and Cloud Computing.* Vol. 2, No. 1, 2014, pp. 1-5. doi: 10.11648/j.iotcc.20140201.11

Abstract: Due to the competitive nature of soccer robots operations, execution speed of functions in critical actions like shooting, dribbling and positioning has an undeniable importance. In this paper the idea of implementing distributed functions for multi-processor computing in a multi-agent environment and also dividing computational load of functions among agents between teammates, is being presented for the first time. The practical implementation of this method resulted in high speed execution of the critical actions functions. Given that so far in this area of expertise, no research has been done, it is hoped that this technique can provide a practical approach.

Keywords: Bluetooth Connection, Queue Data Structure, AVR Micro Controller, Intelligent Agent – Multi Processor Architecture, I2C Communication

1. Introduction

Intelligent soccer robot has the ability to target and control the ball and playing a game with clear rules without human intervention. The main objective of soccer robot research is to design and implement a robot with the ability to compete with human players. In order to achieve this goal, different leagues such as humanoid robots, small and medium size robots and single and multi-agent robots have been created so far. The goal of the humanoid robots is to maintain balance and do human-like movements. In small size one, agent give orders to all other robots and manages the game. However, the goal of medium size soccer robots is individual processing and algorithm efficiency. 1-on-1 and 2-by-2 soccer robots are aimed at strengthening individual skills rather than group skills to move through within this. Single and multi-agents soccer robots that will be discussed in this paper perform their missions by operations such as positioning, identifying obstacles, goal targeting and processing data. The difference between multi-agent soccer robots (2-by-2) and single agents soccer robots (1-on-1) is an extra goalkeeper protecting the gate and cooperates with the other teammate in the necessary circumstances. These circumstances will be explained in more details later in this paper. A critical issue for each agent is to analysis data of input sensors, making decisions for the operation and coordinating with other teammates[1]. Due to the high computational load of the robots, response speed will be decreased. Proposed solution is to divide the decision making between two robots using intra-robot connections so that while one robot is handling heavy computations under a critical situation, it could share some of it's computations with the other robot, therefore overall performance and speed will increase.

With this objective in mind, in this paper, in section 2 we will describe the problem and discuss operations issues and limitations of 2-by-2 soccer robots. Section 3 introduces intelligent agents and multi-agent environments, section 4 will concludes the results of the pervious researches done in the related area, then in section 5 we will solve the problem of increasing the computing speed of 2-by-2 soccer robots, explain the implementation details and review the result. Finally in section 6 we will conclude this experimental research.

2. Description of Problem

In 2 by 2 soccer robots, each team consist two robots as teammates, one goal keeper and one player. The defense duty is shared by both of the players. In 2 by 2 soccer robots the winner is the team with positive goal difference[2]. However victory depends on computation speed and precision of performing of the decisions (precision of performing of the decisions are assumed equal). Decrease in computation speed is result of the long execution time of some of the critical functions. In this section the standard rules of 2 by 2 soccer robot will be discussed. These rules have been published by the international robocup organization which are used in all robocup competitions[3].

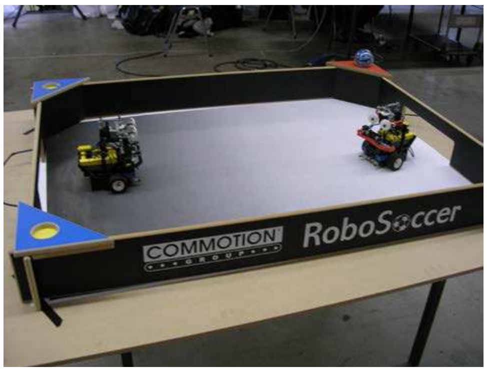

Figure 1. *Play field of 1 on 1 soccer robots.*

Field characteristics:

According to figure number 1, size of the play field is 183*122 cm. Around and at the back of the field are be surrounded by 14 cm tall black walls.

Width of each gate is 60 cm, gates are be placed in the center of the field widths. Height of each gate is 10 cm. A bar is on top of each gate to prevent robots to enter inside the gates. One of the gates is colored blue, while the other one is yellow. Outside of the gates are colored black [2].

Play field should offsets the effect of external infrared light and earth's magnetic field (optimal situation is not guaranteed), therefore the robots should be in a way that their different axles take high effect from infrared light to be able to detect the ball and take action to catch it[3].

Robot positions itself and find the ball by using ultrasonic and infrared sensors. There is an infrared transmitter with known frequency inside of the ball which is used by robots to find it [3].

The main problem in this paper is the heavy computation load on the robots after finding and catching the ball. This computation load occurs because of the calling and execution of critical operational functions like passing, shooting, dribbling and so on. Soccer robots can divide their computations load between other teammates, then when computations are done each agent combines the results and makes the necessary decision.

3. Introduction to Intelligent Agents a and Multi-Agent Environment

In artificial intelligence, an intelligent agent is an autonomous unit that uses sensors in order to understand the environment. This agent may also have actuators to pick, drop and move the objects. A multi-agent system consists of some number of intelligent agents that are related together. All the agents operate independently of each other. Such this system is used to solve problems that are hard or even impossible for a single agent system or an integrated system to solve[4].

In this problem the intelligent agent is the soccer robot and the multi-agent environment consists of two intelligent robots which are connected together by a wireless communication.

4. Studies of Previous Researches

By conducting a search on the IEEE article databases, it was concluded that so far no research has been explicitly done on the distributing computations of soccer robots. The reason for this anomaly is that the present studies on soccer robots are focused on optimizing the individual skills. Also there is no need for distributed computing in single-agent soccer robots, because in single agent environments there is only one agent. Of course, so many researches have been proposed in distributed computing in single agent systems. Most of these studies have been done at lower levels of the distribution, So that most of the researches focus on distributing computation between CPU cores of a single agent system or ultimately at the highest level, on distributing computation on different processors in a multi-processor system. But rarely there have been any research on distributing computations among different agents of a multi-agent system with aim to speed up the calculations or at least one that has been officially released.

As it was discussed in abstract and introduction of this article, the purpose of this research is accelerating the computations by implementing distributed computing in a multi-agent system. To this day, such distributions are often conducted with the following objectives:

- ✓ Positioning and path finding:
  In mobile robots such as firefighter robots, path finder robots and rescue robots, being aware of other agents positions and correct understanding of the environment have always been a critical issue. With this aim in mind, there have been many studies about information distribution to improve the agents understanding of the environment and other agent's position. These studies are mostly focused on distribution of information and almost are never on

distribution of computation, also aims of these studies have been based more on improving the awareness of agents, and they do not emphasis on computational speed [5]

- ✓ Task distribution in group operations:
  In multi-agent robots such as cleaner robots that are able to operate in groups, information distribution is done with the goal of dividing the environment into several parts and doing part of tasks assigned to that part [6]. For example, the cleaner robots can divide the house into several parts and clean each part with one robot. Therefore, the information exchanged between robots will be position's and mission's data [7]. For example, set the exact boundaries of each region, starting and finishing in one area and so on.

As mentioned at the beginning of this part, the major studies published on the context of distribution in multi-agent environment are about information distribution and task distribution. Moreover, no particular research about distribution of computation in multi-agent environment with the aim to increase the computation speed has been done.

5. Solution

In this solution operational functions are divided into two default categories, basic and advance functions.

Basic functions use less computation gain and are executed in the minimum number of clock pulse. They include the most basic calculations and commands such as operations are needed for passing, shooting, positioning, moving, dribbling and catching the ball. Basic functions can be transferred from one robot to another to be processed. These functions will not be computed by in a distributed manner. It means that it is only possible to run them in one processor. According to the agreement between robots, it would not be possible for a basic function to be divided into several smaller functions and be distributed across a network of several robots.

In contrast to the basic functions, advance functions will include higher level of computations and algorithms. These functions may consist some basic functions either. Due to the heavy nature of their processing load, Advance functions could be divided into several smaller parts and be distributed across a networks of robots. These functions could be consider as advance dribbling, catching the ball and blocking.

Advance functions can perform the operations despite the coordinator changes. It means that these functions could be transferred either completely or partially to another platform in order to the execution. By default each robot has two processors. As is shown in figure number 2, processors are named as processor 1 and processor 2. Processor 2 is the robots main processor which has the task of scheduling and performing the functions of the robot. The main operations of robot (including its operating systems execution) will be run on this processor. In this experiment, we choose ATMega 32 as the processor 1 and 2. However, In the experiment, there was not any kind of special operating systems. Sample codes was ran using Codevision AVR.

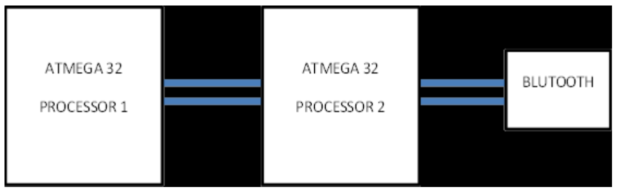

Figure 2. *Connection between robots processors and Bluetooth connection.*

As is shown in figure 2, processor 1 and 2 are connected together using I2C communication. Processor 2 and the other robots are connected to each other using Bluetooth connection. All the communications between the two robots are via this Bluetooth connection.

Before the computing, each robot determines if the function is advance or basic. If it is an advance function it is distributable. First the basic functions will be recognize by the processor. Then, Basic functions will be put in a queue in order due to maintain some priority. We will refer to this queue as the work queue.

Afterward processor 2 will check the status of itself and other processors by sending some request due to ask them giving ACK. After receiving the ACK which shows that the processor is not busy right now, Processor 2 will send a function to whichever one that is not busy.

The robot also stores list of its tasks that it has to do in a queue will refer it as the task queue. If both processors of robot be busy and at least one of the processors be involve in a task that still a function of it be in the task queue, it will select the function in the task queue and send it to the other robot. In the other word, due to deadlock prevention, processor 2 sometimes makes an exception and changes the priority of the functions in the work queue according to the status of task queue.

The other robot receives the function and adds it in its work queue. It also adds the task of the function into its task queue. When a function becomes the head of a queue, robot (processor 2) will pop it up and computes and executes the function and send the results back to the other transmitter robot.

For example, suppose the robot is in a one on one position with the opponent goalkeeper. One on one is an advance function including three basic functions, move, positioning and kick. In this situation one on one function will be in task queue and the three other functions will be in work queue. After processor 1 and 2 of player robot are engaged in processing, it will send the third function (Here for example kick) to the it's teammate (goalkeeper)'s second processor. Processor 2 of the goalkeeper will decide about execution turn and place (processor 1 or 2). In this example the kick function happens only if the goalkeeper confirms it. Due to the long distance between ball and goalkeeper in a strike situation, it cannot see the ball, so it doesn't perform the defend task and would be in the idle mode until the next ball observation. Therefore, at this point, the goalkeeper has a low computation load and can accept and compute the receiving functions.

To calculate the speed improvment between the distributed situation and single processor situation several experiment was conducted in radio silence. Radio silence refers to a situation that there would be no connection between robots, both robots would operate as a blind single system who couldn't observe it's teammate except via it's sensors, but even in that situation, robot couldn't make connection to it's teammate. Because of radio silence robots weren't able to send functions to one another. Also, in order to prevent the inter-robot communication and performing the functions in the distributed manner (even in a sigle robot) processor 1 of robots was disabled.

By using tic() and toc() functions at the start and the end of advance functions their runtime was calculated. This experiment was repeated twenty times. Each time robots played a five minutes game. Afterwards radio silence was aborted. Connection between robots was established and also processor 1 of robots was activated again. Experiment was repeated again with the same length and frequently in the distributed manner and advance functions run time was calculated. Outcome was that multiprocessor distributed computing in a multi agent environment speeded up one on one function by 7.9 times, advance dribble function by 8.4 times, and advance catch the ball function by 3.3 times.

The disadvantages of this method are revealed when the receiver robot processors are both busy. It will execute the sent function with delay or even reject it if its queues are full. This situation can even lead to increase in the advance functions runtime when compared to single processor systems. We came to the conclusion that this situation only occurs during the defense time. To avoid this situation, functions related to defense were no longer considered advance functions and were processed only in single processor mode. Future research of the authors of this article will be to solve this problem. Also, this method will suffers from the single point of failure and by somehow even deadlock. Consider a situation in which one robot sent some function to the other one and wait to get the results. If this situation happens a blocking manner, the sender has to block to wait, until receiving some results from the other one. At this moment, if the receiver falls down or faces some problems, the sender will be in a situation such as a deadlock, freezing and waiting for ever. This situation could be improved by using non-blocking approach or using a time out. Waiting for a fix amount of time, then checking the connection, resend the query or do it by itself would be such solutions to this problem.

It would be clear that this method, by itself, is deadlock free, since an advance function is always a major action which only would be depended on basic functions. Since no basic function couldn't distribute between several processors, no advance function would be waiting for the other one.

6. Software Implementation

To exchange computations between two robots we used predefined functions, func( ) and result( ). The func() function is responsible for transferring computations of a basic function to the another robot. The func() is defined as follows:

void func (name, ID, parameter1 = 0, parameter2= 0, parameter3= 0)

The first argument of this function is the basic function name that one robot wants to send to another one. Sender ID is the second argument. Sender ID should be a unique ID representing the transmitter robot. It could be a string, binary code, dynamic code or something like this. Third to fifth arguments are the basic function input arguments. func() function after receiving the input parameters and packing them sends them to the other robots.

As was mentioned before, each robot should has a unique ID. This is due to network security and prevents interfering interaction between an outsider and the robots and engaging the processors with worthless information. Obviously IDs are completely confidential between teammates. Due to prevent the tapping into the communication between two teammates in order to hacking the robots, overflowing them with some dummy calculation or even make them doing something against their team(like as own goal), a list of per assigned random ID used for this communication. These random ID are pairs, such as (ID1,ID2). Each pair could only one time be used. Both robots know all the pairs. When a robot wants to use the func() in order to send a basic function to the other teammate, It selects one of the unused ID pairs, and use the ID1 part in the func() function. When the other robot receives the request, it first checks the ID1 and search that could it find ID1 in an unused pair? if "yes" then it would use the ID2 of that pair as it's ID for this transaction. If it couldn't find that ID1, It will simply just ignore the message as a spam.

When a robot receives the Computational data, it will call the related basic function then the basic function will perform the computations. Afterwards the receiver robot will sent the results of the basic functions to the result () function. This function packs the data and returns it to the sender robot. Structure of the result () function is exactly like the func() function, but this function only has the three first arguments of the func() function. First and second arguments are the basic function name and the robots name. The third argument is the result of the basic function. When a robot wants to send the packed data of func() function, it will send the //func string to the other robot before sending the actual data. This string will tell the other robot that what it is about to receive is a function for computation and it should send back the result of it. Receiver robot will send the //result string to the other robot when it wants to send back the result. This string will tell the other robot that it is about to receive the result of computations.

7. Conclusion

In this paper, first we introduced 2 on 2 soccer robots, their tasks and missions. The importance of speed in the calculation of the robot were discussed, afterwards we introduced dividing the functions computations by using distributed implementation of multi-processor computing. Conditions, requirements, and implementation details were discussed.

Generally, this method is on average of 5 to 6 times faster than single processor mode, which until now has been unprecedented. The main disadvantage is the inefficiency when both robots being involved in a high computational load. While implementing this method, it was found that this condition occurs when the team is defending. Defense function was no longer considered as an advance function and was processed only in single processor mode. It is hoped that in future research this particular case can be executed in multi-processor architecture as well.

References

[1] Hassanzadeh Nazarabadi Y, Saghlatoon H, Sharif Shazileh A., "A method to create the most accurate goal targeting in 1 on 1 soccer robots", in 5th international conference on the Evaluation of Novel Approaches to Software Engineering, Athens, Greece Greece, 2010.

[2] Robocup Organization, "Jounior Soccer Robots Rules", Robocup, Mexico City, Mexico,2012

[3] Iranian Robocup Organization, "Jounior Soccer Robots Rules", Robocup Iran Open, Tehran, Iran, 2012

[4] Russell S, Norving P, Artificial intelligence: A Modern Approach, by Prentice Hall, 3rd edition, 2002

[5] de Melo, L.F.D.J., A.E. ; Lopes, G.M.G. ; Rosario, J.M. , Mobile robots and wheelchairs control navigation design using virtual simulator tools, in Industrial Electronics (ISIE), 2012 IEEE International Symposium on 2012, IEEE: Hangzhou, China.

[6] Guruprasad, K. R. D., P. (2012). Distributed Voronoi partitioning for multi-robot systems with limited range sensors. In Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on Vilamoura, Algarve [Portugal], IEEE: 3546- 3552.

[7] Pavone, M. A., A. ; Frazzoli, E. ; Bullo, F. (2011). "Distributed Algorithms for Environment Partitioning in Mobile Robotic Networks." Automatic Control, IEEE Transactions on 56(8): 1834- 1848.