Artif Life Robotics (2013) 18:76–82
DOI 10.1007/s10015-013-0102-4

ORIGINAL ARTICLE

A robotic software for intelligent applications

Mehmet Serdar Güzel · Yasin Hınıslıoğlu

Received: 12 March 2013 / Accepted: 7 August 2013 / Published online: 28 August 2013

Abstract This study addresses the development of a novel intelligent robotic software system which can control a low-cost five DOF robotic arm and allows the robot to be able to play Tic-Tac-Toe, a simple board game. The paper first aims to introduce proposed software and then details the application developed, including image processing, and decision making steps.

Keywords Robotic software · Robot arm · Tic-tac-toe · Plugin

1 Introduction

This paper introduces a robotic software tool, designed based on a low-cost commercial robot arm and supports simulator and online real mode controller. The architecture of the developed tool is flexible which allows developers to add new modules to software depending on the required problems. The main concern of this work is to introduce an intelligent application provided by IRSS. The software architecture allows a commercial low-cost robot arm to play a well-known board game. Board games are the games played on the board with certain pieces, which are moved across the board. It is a common knowledge that simple board games are considered to be the perfect entertainment for families because these games are known to provide fun to people of all ages. Some well-known board games (for example, chess, checkers, Tic-Tac-Toe) possess intense strategic value. Computer-based board game systems have been developed in the last 30 years [1]. In this study, a flexible and low-cost robotic system is introduced for playing tic-tac-toe against human opponents. Surveying the literature reveals that several systems and techniques have been improved so far for manipulators to play Tic-Tac-Toe. Manoel and Soares implemented a Java-based system which coordinates a set of LEGO robot to play the game of Tic-Tac-Toe [2]. Furthermore, Sungur and Halici proposed a system playing Tic-Tac-Toe game against a human opponent [3]. In addition, AI especially Neural Networks have been employed by researchers to play Tic-Tac-Toe. Sungur and Halici proposed a system working on optimizing neural networks for playing Tic-Tac-Toe. They primarily worked on Hopfield network, Boltzmann machine, and Gaussian machines. The performances of the models were compared through simulation. Siegel on the other hand adapted reinforcement learning as the learning method for the game [4].

Chellapilla and Fogel worked to describe efforts of hybridize neural and evolutionary computation to learn appropriate strategies in zero- and nonzero-sum games, including the Tic-Tac-Toe and Checkers [5].

Recently, Matuszek et al. developed a new manipulator system which is claimed to play board games against human opponents in non-idealized environments [6]. Furthermore, Rajani et al. proposed an algorithm using a Hamming Distance Classifier in Neural Networks to find the most optimal move to be made in the Tic-Tac-Toe problem such that the game always ends in a win or a draw [7].

This paper primarily presents a new flexible software, *IRSS*, having a flexible modular structure and supporting plug-ins. The software framework consists of three parts: kinematics, eye and, game modules. One of the main aspects of this paper is to highlight the intelligent, game-

M. S. Güzel (✉)
Department of Computer Engineering, Ankara University, Ankara, Turkey
e-mail: mguzel@ankara.edu.tr

Y. Hınıslıoğlu
Bilkent University, Ankara, Turkey
e-mail: yasinh@outlook.com

playing application of the proposed framework. This module in essence utilizes other modules and requires a physical robotic arm to perform the game. The eye module, responsible for image processing, provides input for the game and conducts the required image-based analyzes and procedures. The kinematics module, on the other hand, provides arm motion along the game board. The game module is the final module which first utilizes the eye module to capture and process images properly. It then employs an AI-based sub-module to determine the next possible location for the robot on the game board. The final movement of the robot along the board is provided by operating the kinematics module which moves the robot arm to any specific location on the board.

2 Software design

The IRSS software is designed on a plugin-based structure. The architecture of the proposed software is shown in Fig. 1. The second level of the architecture illustrates plug-ins which allows researcher to develop their own screens or implement their own algorithms for further applications. The flowchart of the plugin level is illustrated in Fig. 2 in which the system searches through the possible installed plug-ins and runs all detected plug-ins. Each module of the tool is implemented inside this plugin-based structure. There are several screens developed to provide communication with users namely: Forward kinematics, Inverse kinematics, Chess, Checkers, Tic-Tac-Toe, and Eye. Forward and Inverse kinematics screen provides the motion control of the arm in both simulation and real-time mode.

The first level, user interface, provides the mathematical analysis and solution of the arm which is detailed in [8]. Chess and Checkers screens are reserved for future works so that associated plug-ins can easily be integrated to the software to run these games within the robot. Eye module

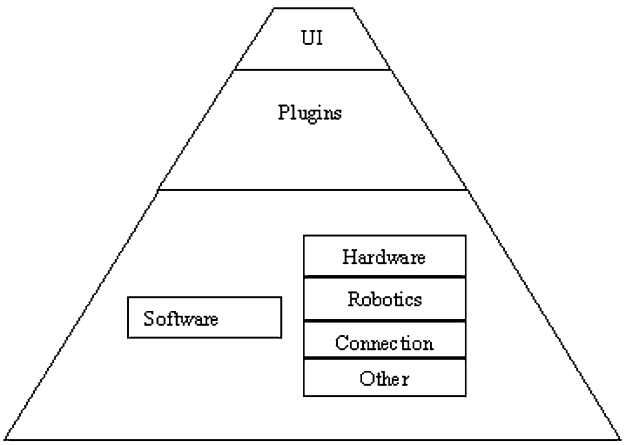

Fig. 1 The flowchart of the plugin-based software

is responsible for image capturing which in essence controls webcam and provides image processing for the grabbed frames through the camera. Tic-Tac-Toe module provides the robot to be able to play the game against any human opponents, as shown in Fig. 3 (see page 5). This part will be detailed in following sections. The tool is implemented with C# programming language in Visual Studio.Net platform. The Developed software tool supports two main operation modes, simulation and real-time control. Program accepts user commands in every mode. In the real-time application mode, System accepts user commands and sends output responses via control card to servomotors. The proposed software can control robot arm and camera with minimum parameter change in any environment in spite of illumination or other environmental problems. The second mode is simulation mode in which 2D model of the arm and game board are simulated.

3 System design

The physical system and the corresponding test environment are illustrated in Fig. 4. The physical system involves a Lynx-6 robot arm equipped with an electromagnet, a simple web camera, and a game board including metallic checker pieces. The arm has 5-DOF (Degree of Freedom) with a grip movement and is able to deliver fast, accurate and repeatable movements. The joints of the robot arm are shoulder rotation, shoulder back and forth, elbow, wrist up and down, wrist rotation, and gripper. Lynx-6 robot arm is inexpensive, flexible, and mainly designed for educational applications. Pulse-controlled local feedback servomotors having accuracy of 0.9 degrees per axis are used in all of the joints and grippers of the robot [9]. Having these motors reduces the complexity of feedback control. On the other hand with using of these motors it is possible to design microcontroller of FPGA-based embedded controller for robotic arm. SSC-32 servo card is used to control motors which provide the hardware interface between computer and the robot arm. It has a time resolution of 1 μs for accurate positioning and a dc motor control to generate extremely smooth moves. The card generated motion can be a speed controlled, time controlled or a combination (speed and time) motion. The motion of joints can be controlled either in sequential or parallel manner. It is also possible to add external sensors for advance applications to the overall architecture like cameras and IR sensors [9].

4 Tic-tac-toe game module

Tic-Tac-Toe is a two-player board game which is sometimes referred to as a noughts and crosses, the game is

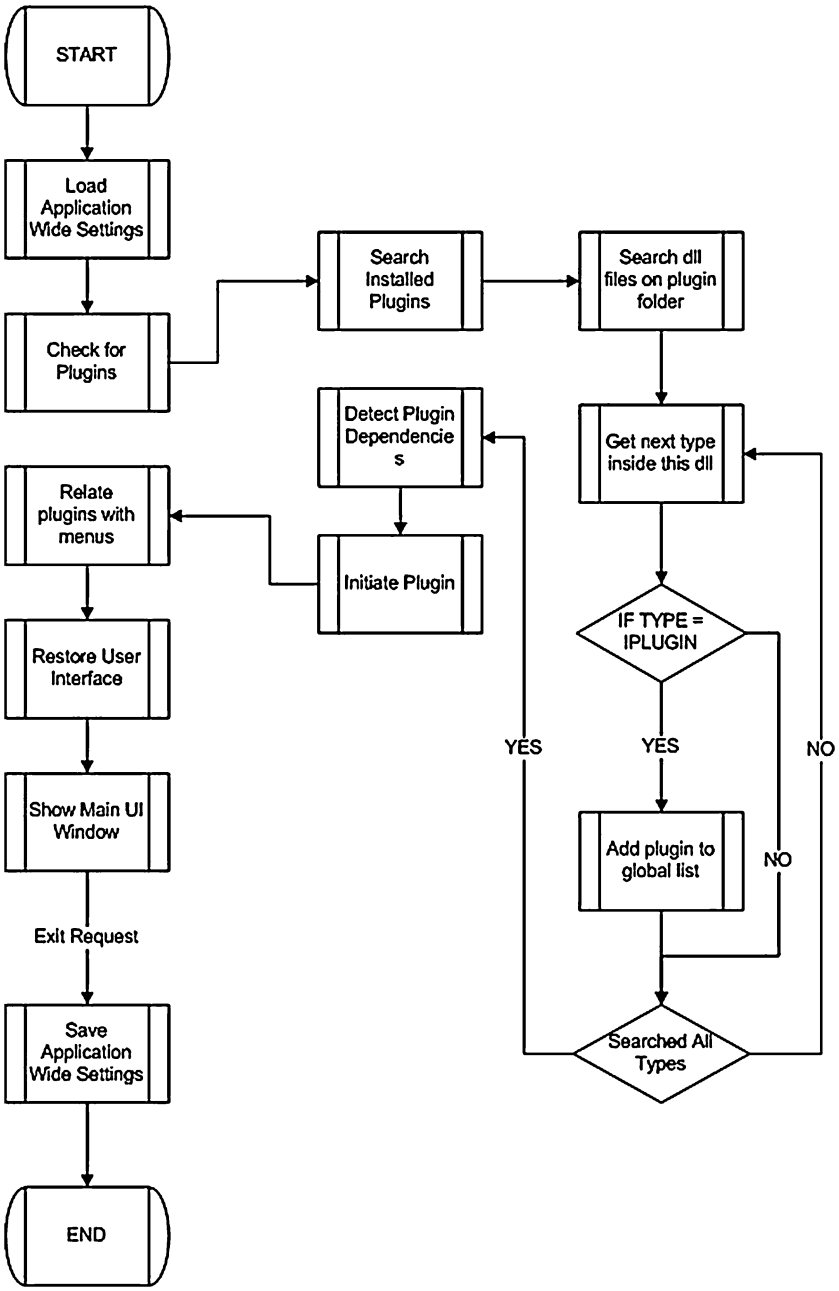

Fig. 2 The flowchart of the plugin-based software

played on a board consisting of 9 cells arranged as a 3 × 3 square, i.e. three rows and three columns. O and X, who take turns marking the spaces in this 3 × 3 grid, usually X going first. The player who succeeds in placing three respective marks in a horizontal, vertical or diagonal row wins the game. A typical game board is shown in Fig. 5.

As previously mentioned, each element of the application is implemented inside plugin. Simulator, Forward kinematics (FK), Inverse kinematics (IK), Eye, and Tic-Tac-Toe are implemented as plug-ins and can be modified without recompiling the main software which only entails the recompiling of the desired plugin. Each plugin is allowed to access other plugin data. For instance, data results of IK module can be accessed by FK module and vice versa. Data results of both modules can also be accessed from Simulator plugin. In addition to these, each plugin connects robotic hardware using serial connection plugin which can easily be altered with wireless or other type of connections. This flexibility makes this application an extremely powerful tool for robotics researchers so that developers can implement their own features without considering the whole system. The software architecture consists of several subsections and several engines, as previously mentioned in Sect. 2. There are four main

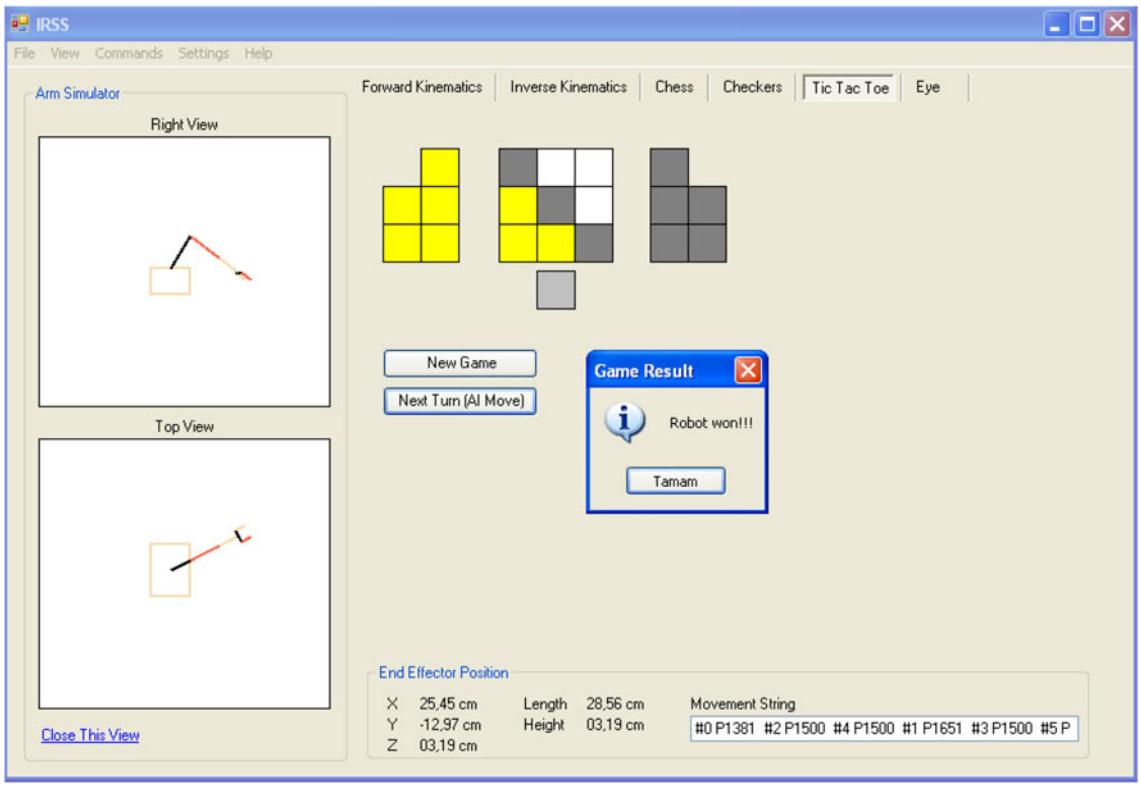

Fig. 3 The main screen demonstrating Tic-Tac-Toe game

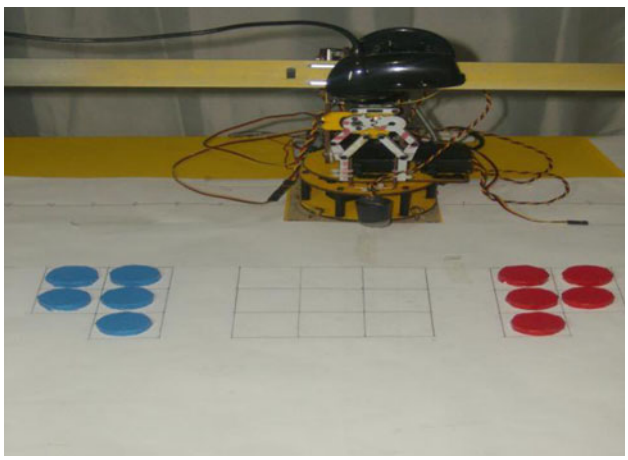

Fig. 4 Main view of the system

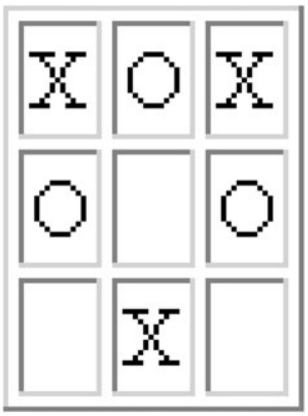

Fig. 5 Tic-Tac-Toe game board

engines which are improved within this software tool, these are Kinematic, Simulator, Eye, and AI engines.

4.1 Kinematics engine

It is responsible with the solutions of forward and inverse kinematics equations and the calculations related to joints and Cartesian coordinates. In forward kinematics, the length of each link and the angle of each joint are given and the position of any point in the work space of the robot is calculated; whereas, the length of each link and the position of the point in work space are given and the angle of each joint is calculated for the inverse kinematics engine.

4.2 Simulator engine

It is responsible with the interpretation of the robot model, construction of internal data structures to represent the

robot and manipulating the joint angles, as specified by simulation commands and the robot's kinematics.

4.3 AI engine

This engine provides a min–max tree algorithm for the primary control of the robot to determine the next best move.

4.4 Eye engine

This engine is directly related to control webcam and to provide image processing method for the grabbed frames through the camera.

The game module plugin employs all of these plug-ins. The inverse kinematics module was previously discussed in [7], moving the robot arm along the board. Eye module and AI modules will be detailed in the following sections, respectively.

4.4.1 Eye module

Image capturing is provided by a low-cost web camera in this study. The camera takes record of the game board continuously and a simple but efficient image processing methods are employed to recognize the position of the game pieces. Many techniques have been improved recently for detecting game pieces on the game boards. Most of them are very complex to design and hard to implement. Once the complexity of a vision-based algorithm increases, the consumed time for the process will be increased dramatically, which is one of the worst situation for a real-time system. Accordingly, a simple detection algorithm is utilized to overcome the detection problem of the board and game pieces improved for the vision system. The algorithm basically employs image subtraction method which is one of most well-known arithmetic operations used in image processing operations [10]. An example of image processing step used in the game is illustrated in Fig. 6 (see page 5). The algorithm requires recording all possible movements during the game before the system has been started. Accordingly, all possible movements are stored in the image databases. Once the human opponent makes a move, the current situation of game board is captured by the Eye module. This snapshot image is matched with templates in the database via subtracting operation. The steps of the algorithm and explanations are shown below from a to g.

a. The images of all positions on the game board are stored on the image database at jpeg format.
b. Current state of the game board is captured by the camera at jpeg format.
c. The current image is subtracted from all template images stored in the database, respectively.
d. A threshold value is first selected and then a global variable (gv) is assigned to determine the best match.
e. Current image is subtracted from the first template image stored in the database. Gv is increased for each pixel difference which is more than the threshold value.
f. Step e is performed until the current image is compared with all the template images in the database.
g. Finally, the smallest value of gv is selected with a simple sorting algorithm. The smallest one presents the distribution of the game pieces which will be the main input for the AI engine.

The calibration procedure was carried out only once at deployment phase to construct the image database. In this proposed algorithm, both gray and colored images are tested. According to the result of these preliminary tests, it can be stated that colored images are more reliable.

4.4.2 AI module

This module is responsible for integrating the intelligence to the proposed system. This module employs Min–Max algorithm for decision making which is quite useful for especially Tic-Tac-Toe game. The Min–Max algorithm is applied in zero-sum games, such as tic-tac-toe, checkers, chess, go, and so on. All these games have at least one thing in common, they describes a situation in which a participant's gain or loss is exactly balanced by the losses or gains of the other participant(s). Also, they can be described by a set of rules and premises. With them, it is possible to know from a given point in the game, what are the next available moves. So, they also share other characteristics, they are 'full information games' [11]. Each player knows everything about the possible moves of the opponent. The best way of explaining the algorithm is to employ a search tree. Search trees are a way to represent searches. A representation of a

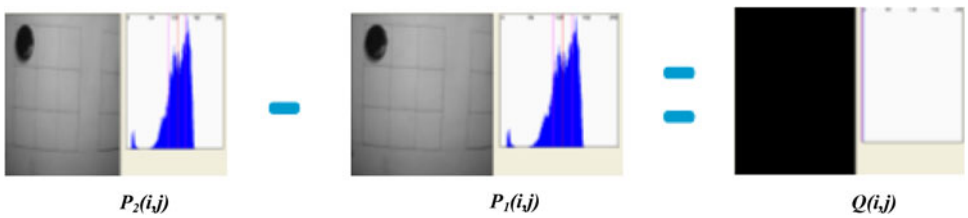

Fig. 6 Image subtraction operation

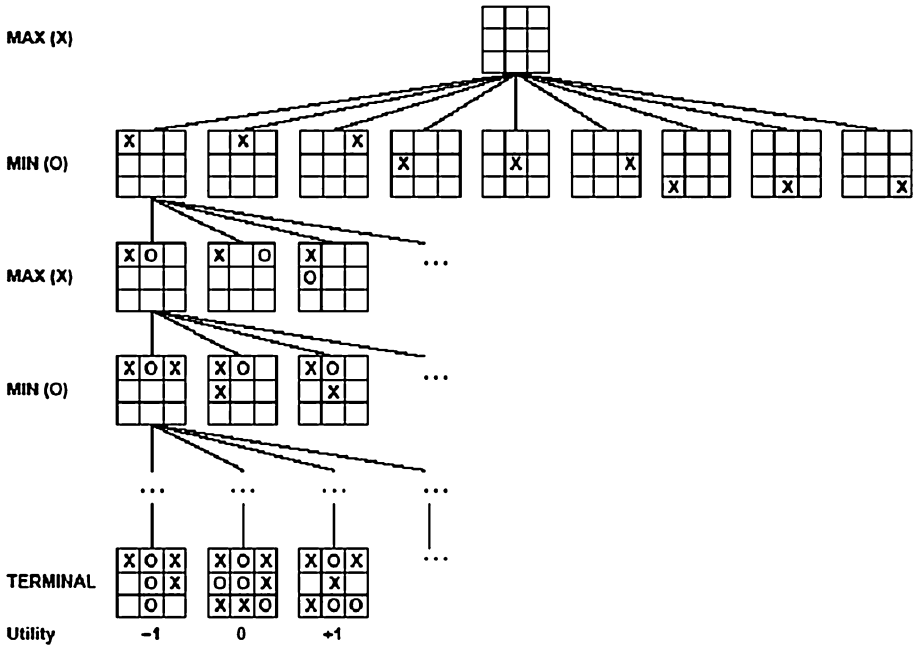

Fig. 7 Min–Max search tree

Table 1 Experimental results

Game number (5), R(AI), H(Human)			
R won	H won	Draw	Percentage of winning (%)
9	0	6	60
13	0	2	86
7	0	8	47
12	0	3	80
10	0	5	66
8	0	7	53

search tree is illustrated in Fig. 7. The squares are known as nodes. There are two players involved, MAX and MIN. A search tree is generated, depth-first, starting with the current game position up to the end game position. Then, the final game position is evaluated from MAX's point of view, as shown in Fig. 7 (see page 6). Afterward, the inner node values of the tree are filled bottom-up with the evaluated values. The nodes that belong to the MAX player receive the maximum value of its children. The nodes for the MIN player will select the minimum value of its children The MAX player will try to select the move with highest value in the end. But the MIN player also has something to say about it and he will try to select the moves that are better to him, thus minimizing MAX's outcome [12, 13].

5 Discussion and conclusions

The system was evaluated in two different ways one of which measured the performance of AI engine. In this

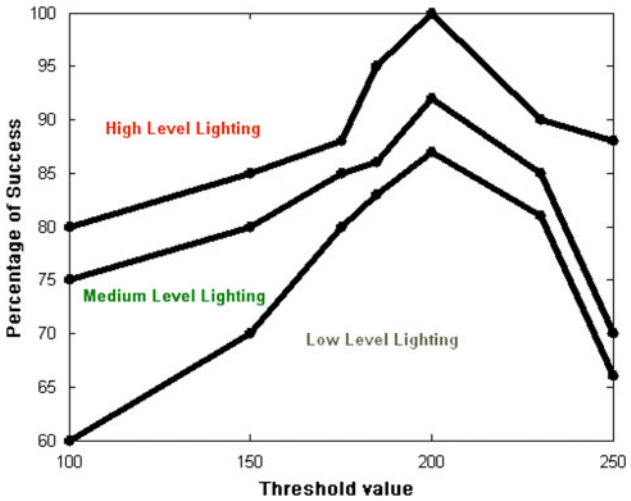

Fig. 8 Threshold values for different lighting levels

experiment, six different people coming from different age groups and having different skills were used. Each experiment was performed five times and results are classified. Table 1 illustrates the results of these experiments from which it can be indicated that the system is unbeatable, and the best success against the system can be a draw. This is because the characteristics of the Min–Max algorithm.

The second part of the experiments was done for the evaluation of the proposed vision system. The main intention of these experiments is to obtain the optimum threshold in different lighting levels. To achieve this, different threshold values are tested in different lighting conditions namely: low, medium and high. The test results, illustrated in Fig. 8, reveal that '200' is a good value for the desired threshold in all lighting conditions.

During the calibration step, any environmental change on the lighting levels does not influence the proposed vision system within the calculated threshold value. However, if any error is recognized by the system, automatic threshold arranging system is triggered and the threshold value is recalculated automatically to overcome the problem. Another important problem is the performance of digital servomotors used by the arm. Three HS-5745MG [14] digital servomotors are used by the system which are very powerful but requires to be rested to prevent deviation during the game. Experiments prove that the deviation rate increases steadily after 15 min running.

Overall, the hardware part of the proposed system consists of a low-cost Lynx-6 manipulator and a basic web camera for vision operations. Min–Max algorithm is adapted to the AI engine of the system and a basic but efficient image processing algorithm is also proposed to detect positions of the pieces on the game board. The proposed software architecture relies on a plugin-based structure which makes the developed application flexible and encourages researcher to develop their personal applications with minimum effort. The video of the system including the application and the introduction of the developed software can be reached in (http://uk.youtube.com/watch?v=HFuBMOLuAn0).

The developed software tool (IRSS) is designed and implemented as an open source tool which can be used in various application fields in science and industry. The tool mainly supports cost-effective education and training in robotics which helps researchers and students to improve their learning on robotic in simulation and real-time control. Accordingly, students and researchers can employ the tool to find solutions to several existing scientific and educational problems in the field of robotic. For instance, the developed tool can be adapted to overcome the fundamental visual serving and tracking tasks. Besides, the system allows researchers to develop different applications including board games, educational issues, and other corresponding issues in robotics. That is because the software architecture of the implemented tool includes a modular-based structure which supports plug-ins and helps researchers to design and integrate different applications to the tool easily. Feedbacks show that simulation-based learning especially in robotic is effective. The more effective learning occurs when simulation and real-time control takes place simultaneously which is one of the fundamental success of the proposed IRSS tool.

This free Software package will be on the official web page of the project in a short time. The corresponding address is (http://comp.eng.ankara.edu.tr/staff/msguzel_eng.html).

Acknowledgments This paper is an extension and modified version of our conference paper [15].

References

1. Ruben Vuittonet, Jeff Gray (2006) Tic-Tac-LEGO, an investigation into coordinated robotic control, In: Proc. of the 44th annual southeast regional conference, pp 796–781
2. Manoel Alvaro, Soares Souza (2006) Playing a tic-tac-toe game against a robot manipulator using a vision system. Revista Ciencias Exates 12:123–128
3. Sungur M, Halici U (1992) "Optimizing neural networks for playing tic tac toe", In: Proc. of International Joint Conference on Neural Networks, IEEE INSS IJCNN, Baltimore, USA
4. Sebastian Sigel (2001) Training an artificial neural network to play tic-tac-toe, ECE 539 Term Project
5. Chellapilla K, Fogel DB (1999) "Evolution, neural networks, games and intelligence", In: Proc. of the IEEE, 87(9): pp 1471–1496
6. Matuszek C, Mayton B, Aimi R, Deisenroth MP, Liefeng Bo, Chu R, Kung M, LeGrand L, Smith JR, Fox D (2011) "Gambit: an autonomous chess-playing robotic system, "Robotics and Automation (ICRA),2011 IEEE International Conference on May 2011, pp 4291–4297
7. Rajani NF, Dar G, Biswas R, Ramesha CK (2011) "Solution to the tic-tac-toe problem using hamming distance approach in a neural network, "Intelligent Systems, Modelling and Simulation (ISMS), 2011 Second International Conference, 3(6): pp 25–27
8. Koyuncu Baki, Güzel Mehmet (2007) Software development for the kinematic analysis of a lynx 6 robot arm. Intern J Appl Sci, Eng Technol 4(4):228–233
9. Lynxmotion Inc,2007 "http://www.lynxmotion.com"
10. Rafael C. Gonzales, Richard E. Woods (2002) Prentice Hall, digital image processing, pp 108–112
11. Russell S, Norving P (2001) Prentice Hall, Artificial intelligence a modern approach, pp 67–76
12. Paulo Pinto (2002) Introducing the min–max algorithm
13. Díez SG, Laforge J, Saerens M (2013) "minimax: an optimally randomized MINIMAX algorithm, "Cybernetics, IEEE Transactions on, vol 43
14. ServoCity (2008), http://www.servocity.com
15. Mehmet Güzel, Yasin Hınıslıoğlu (2009) Artificial low cost robot arm system playing tic-tac-toe, 9th Conference on Autonomous Robot Systems and Competitions, Robotica, Branco, Portugal, pp 7–8