Intelligent Robot Design for Five-in-a-Row Game

Shu-Yin Chiang, Chien-Wu Tsai, and Hao-Ge Jiang

***Abstract*—In this research, an intelligent robot arm is designed to play a five-in-a-row automated game with a user. Hardware and software are developed for a system that integrates artificial intelligence technology with computer vision; a six-axis robot arm is used to achieve the goal of automated game played between human and robot. Results are demonstrated for the system using maximum weighting value and layer tree search and reinforcement learning algorithms. The system also provides protection of user safety during the game when a hand is in the vision range. The results show the system can perform successfully in the game.**

***Index Terms*—Robot arm, five-in-a-row game, vision system, maximum weighting value, layer search tree, reinforcement learning**

I. Introduction

In the era of Industry 4.0 and automation, robotic arms are widely used, not only on production lines, but to replace people in dangerous jobs, reduce costs, and provide fun and entertainment. The rise of artificial intelligence-related technologies has brought about many innovative research results. In particular, issues related to computer vision integration of robotics and artificial intelligence are the trend of future research and industrial development. In 2016, the algorithm AlphaGo, developed by the DeepMind team, defeated Sedol Lee, a world-famous Korean Go player. This computer algorithm is designed to learn and train the system based on artificial intelligence. AlphaGo uses deep learning under artificial intelligence to train Go programs. Monte Carlo search trees [1] are used to predict opponents' steps; Monte Carlo tree search includes selection, expansion, simulation, and verification process to find the best move for the game. Silver et al. [2-3] proposed a new reinforcement learning algorithm for AlphaGo. The algorithm includes a look-ahead search within the training loop to achieve rapid improvement, with accurate and stable learning.

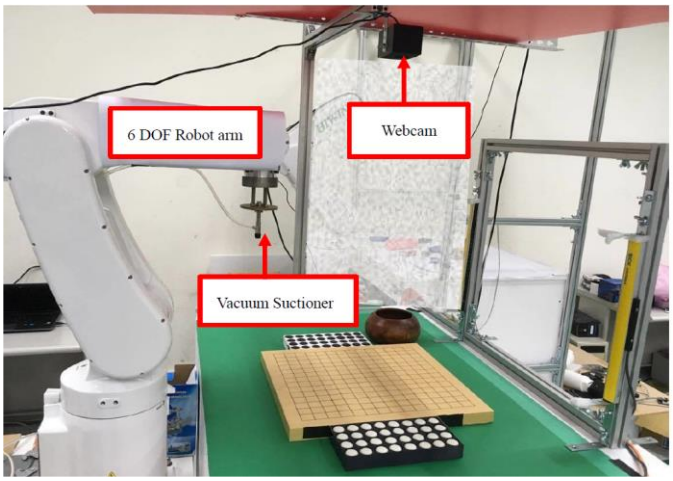
However, most of the research about Go game is designed in a computer system where either the user keys in the position of a stone or a human places the stones for the computer. Therefore, the most significant contribution of this current work is that both the user and robot place the stones all by themselves. In order to ensure the safety of the human-machine (robot arm) game, a skin-color detection system is used to stop robot arm movement when it recognizes that a person is playing a stone. Considering differences in race, ambient light brightness, and camera factors in skin color detection, YCbCr, HSV, and RGB color spaces are used to distinguish color blocks, and a probability calculation is performed through the PCM (Possibilistic C-Means) clustering method [4, 5] to detect the skin area.

To execute the game, the system needs the forward and inverse kinematics of a 6-axis robot arm to either control the angle of the robot or obtain the angle of the robot relative to the desired position to place the stone by itself. In addition, for the application of robotic arm and vision integration [6], environmental information is acquired from the camera image, and a transformation between image coordinates and world coordinates is calculated. To reach the desired position for the robot, the training system and the image system must be integrated to perform precise movements of the robot arm.

AlphaGo is a computer program combining many algorithmic calculations and artificial intelligence. However, when it is playing with humans, it is still necessary for a human to place the stone to execute a human-machine game. The goal of this research is to design a system wherein artificial intelligence technology is integrated with computer vision, and a six-axis robot arm is used to achieve the goal of an automated game between human and robot. The game of Gomoku, also called Five-in-a-Row, is an abstract strategy board game traditionally played with Go pieces (black and white stones) on a Go board, usually the 15×15 board. The rest of this paper is structured as follows: The hardware structure and system design are introduced in Section 2, and the intelligent algorithm is proposed in Section 3. The experimental results and conclusion are presented in Sections 4 and 5, respectively.

II. Hardware and System Design

To design the game execution, not only is the software considered, but also the hardware design. As a key issue for success, the system hardware is shown in Figure 1. There is a laptop, a six-axis robot arm, a webcam, a vacuum suctioner controlled by an Arduino, and one set each of black and white stones arranged on both sides of a 15 x 15 board for the robot. A bowl with mixed black and white stones is placed for the user to play with the robot.



Shu-Yin Chiang and Chien-Wu Tsai are with Department of Information and Telecommunications Engineering, Ming Chuan University, Gui-Shan, Taoyuan 333, Taiwan (e-mail: sychiang@mail.mcu.edu.tw and cwtsai@mail.mcu.edu.tw, respectively.)

Hao-Ge Jiang is with School of Electrical and Electronic Engineering, Nanyang Technological University, 639798, Singapore. (e-mail: haoge001@ntu.edu.sg)

Fig. 1. System hardware.

A. *System Structure*

The flowchart of the system is shown in Figure 2. The system performs camera calibration to ensure that the board and the stones are in the right position to obtain the correct position from image to stone placement. The system is designed for the user to play the game first, and a dynamic object detection function informs the system that the user is playing a stone. When there is no object in the image, the robot calculates the position for placing the next stone using the designed algorithm; meanwhile, the system will continuously detect for skin color to avoid the robot arm hitting the user. This flow repeats until the game is over.

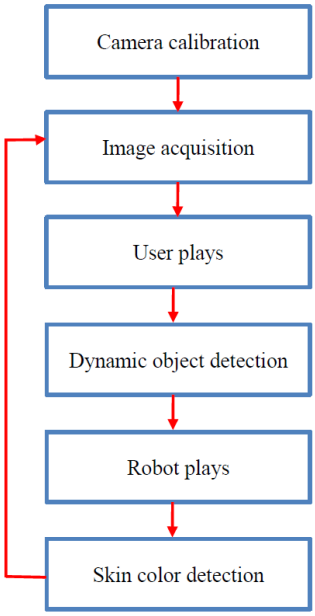


Fig. 2. System flowchart.

B. *Robot Arm*

The robot arm, RA605 manufactured by HIWIN, has 6 axes, as shown in Figure 3. The related parameters for the robot arm are summarized in Table I. Forward and inverse kinematics are used to control the robot motion to the desired position, as shown in Figure 4.

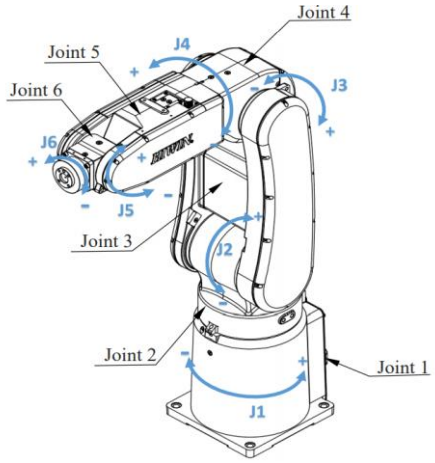


Fig. 3. 6-axis Robot arm.

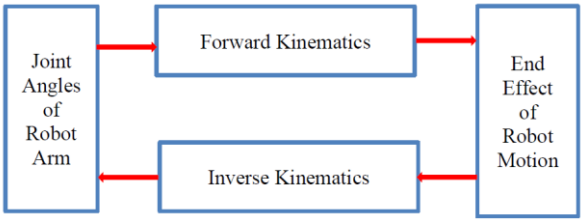


Fig. 4. Robot arm calculation.

TABLE I
ROBOT ARM SPECIFICATION

Item		RA605-GC
Maximum reach radius		710 mm
Degree of Freedom		6
Nominal Load Capacity		5 kg
Motion Range	J1	± 165 °
	J2	+ 85° ~ −125°
	J3	+ 185° ~ −55°
	J4	± 190°
	J5	± 115°
	J6	± 360°
Position repeatability		± 0.02mm
Manipulator weight		40 kg

C. *Air Pump for suctioning the stone*

To enable the robot suction the stone to move it to the desired position, an air pump connected to an electromagnetic valve is used. The specification of the air pump is listed in Table II; shown in Figure 5 is the design of the electromagnetic valve and air pump connected to I/O pins of Arduino to control the holding and suctioning status.

TABLE II
AIR PUMP SPECIFICATIONS

Rated Voltage	DC 6 V ~ 12 V
Vacuum	400 mmHg
Pressure	0.7 kgf/cm^2

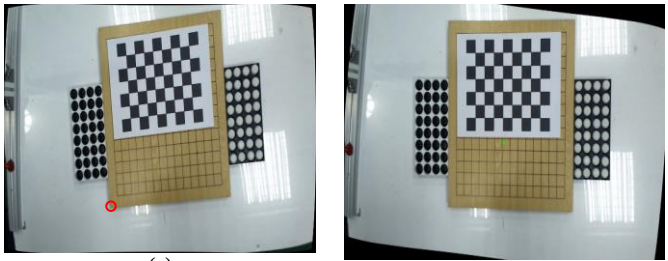
(a) (b)

Fig. 5 (a) Vacuum suctioner and (b) Air pump connected to electromagnetic valve.

D. *Calibration and Coordinate Transformation*

The webcam installed on the top, as shown in Figure 1, is used to obtain the image of the black and white stones and the Go board. To obtain the correct position of each stone and place a stone in the exact position, the system is first calibrated. The

first step of camera calibration is to find the edge points, and the second is to conduct image correction for the board to find the rotatable range within which to execute the game. The calibration is shown in Figure 6. In Figure 6(a), the left corner marked with a red circle is set as the reference point to rotate the chessboard, and then the calibration template is applied to the upper left corner of the chessboard grid line to correct the chessboard position. The calibrated result is shown in Figure 6 (b).



(a) (b)

Fig. 6. Camera calibration

The image coordinates are transformed to world coordinates, so the robot can use forward and inverse kinematics to access an object through the image system of the camera. The related equation is shown in (1), where the (u,v) is the image coordinate, (c_x, c_y) is the image center, (X,Y,Z) is the world coordinate, and f_x and f_y are focal length. A matrix with parameters r and t can be considered as the image matrix. Therefore, camera calibration can obtain the related parameters.

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \quad (1)$$

E. Stone Position

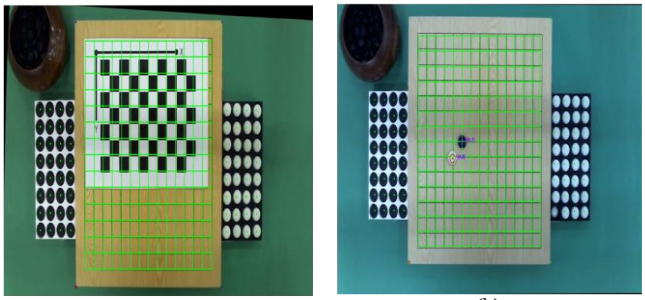
With the chessboard positioning image correction completed in the previous stage, through performing color space conversion, binarization processing, erosion and expansion processing, then Hough circle detection, the center position of each piece is located on the board, and the brightness value of each circle determines whether the stone is black or white, as shown in Figure 7. After obtaining the center position of all black and white pieces on the board, the system can play.

A temporal frame difference algorithm [7] is used to obtain the position of a stone placed by a user. First, the acquired image frame is numbered by I(t), t = 1, 2, …. Then the previous frame I(t-1) is subtracted from the current frame I(t) to create a change image as shown in (2).

$$T(t) = I(t) - I(t-1) \quad (2)$$

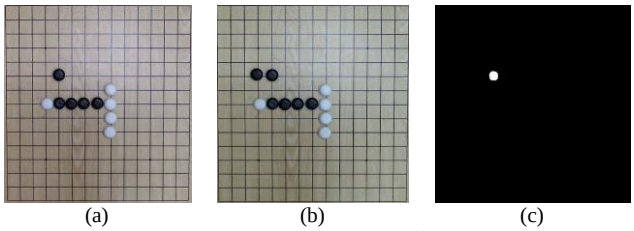
I(0) is the initial image that does not contain any stone.

Figure 8(a) is the original image, and the user has placed a stone in Figure 8(b). The result of the temporal frame difference shows in Figure 8(c).



(a) (b)

Fig. 7. (a) Calibration of the image to transform the intersections to x and y coordinates. (b) A black stone is at (5, 7) and a white stone is at (4, 6) position, respectively.



(a) (b) (c)

Fig. 8. (a) Image I(t-1) (b) Image I(t) (c) Frame difference method to obtain the position of the stone.

F. Game Rules

The user plays a black stone first, and the robot plays a white stone. Players alternate turns to place a stone of their color on an empty intersection. The winner is the first player to form an unbroken chain of five stones horizontally, vertically, or diagonally.

III. PROPOSED ALGORITHM

Since the Go board is 15 x 15, the system uses the array in (3) to record the status of the positions. Initially setting the p array to be zeros, as shown in (3).

$$p[i][j] = 0, i = 0,\ldots 14,\ j = 0,\cdots 14 \quad (3)$$

Next, if the position has a white stone placed then $p[i][j]=1$ and a black stone is $p[i][j]=-1$. After the user moves, the robot will judge all the open, horizontal, left, and right diagonal stones on the board before determining the next step.

A. Maximum Weighting Value

The decision-making process is based on the weighting value for each position. The weighting value for the next step is calculated according to the rules and situations defined in Table III.

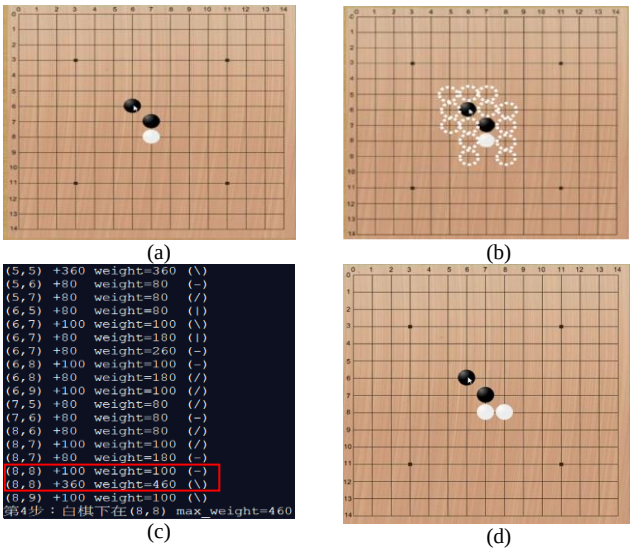
TABLE III
WEIGHTING VALUE FOR ROBOT UNDER DIFFERENT SITUATIONS

Situation	Robot Weighting	User Weighting
5 reju	20000	19000
open 4	5500	2500
double half open 4	4500	2500
half open 4	1500	25
open 3 and half open 4	1700	1400
open 3	500	360
open 2	100	80
half open 3	20	15

The robot will calculate all possible positions for all empty spaces where $p[i][j]=0$ to find out whether this position will created a situation found in Table III. If it fits a situation in Table III, then the weighting value is added to $w[i][j]$. The maximum $w[i][j]$ is the position where the robot will put the stone, ie, $p[i][j]=1$.

$$w[i][j]= w[i][j] + \textit{weighting value}$$
$$p[i][j]= 1, \text{ where } \textit{max } w[i][i] \quad (4)$$

For example, in Figure 9, the original play is shown in (a). The robot calculates all possible positions, as in Figure 9(b) to find out which position has a maximum weighting value. The weighting value for all possible positions is shown in Figure 9(c), and the maximum value is at (8, 8). The weighting value is 360, where the black stone forms open 3 and 100 where the white stone forms open 2. The maximum weighting for all possible positions is 460 (360+100), thus a white stone is placed at (8, 8) by the robot, as shown in Figure 9(d).



(a) (b) (c) (d)

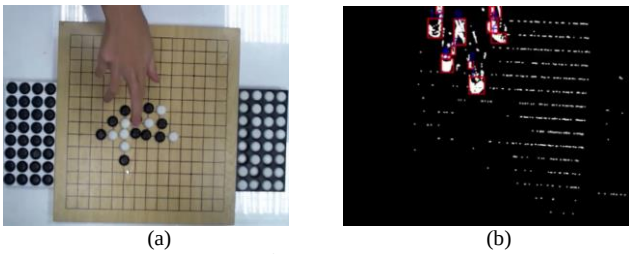
Fig. 9 (a) Original play. (b) Robot checks all positions to find the weight value for those possible white positions. (c) Weight value for each position is added, and the maximum is at (8, 8). (d) Robot places a white stone at (8, 8) position.

B. Layer Tree Search and Reinforcement Learning

To execute the game well, the robot must avoid the situation where a current maximum weighting is not the best move for the next step or may lead to losing in two or three steps. Hence, layer tree search and reinforcement learning (RL) algorithm is proposed to let the robot see the future moves in order to decide the current decision, not to make the decision based solely on the current situation. In the tree search, the simulation takes the board position s as an input and outputs a vector of move probabilities p for each action a, estimating the expected outcome z from position s. The simulation proceeds by selecting for each potential state: s a move, a with parameters of low visit count, high move probability, and high value according to the current state. The system trains the parameters with a reinforcement learning method. Finally, the layer tree search will stop the terminal position s from scoring according to the rules of the game to compute the game outcome z: -1 for a loss, 0 for a draw, and +1 for a win. Therefore, this rule uses the potential outcome to execute the game for the robot to win the game. The layer tree search algorithm does not search for all possible situations; it only finds the reasonable two or three steps the user may employ to form a half open 4 or open 3. Therefore, the robot places stones according to the layer tree search result and reinforcement learning process during the game.

C. Protection Scheme

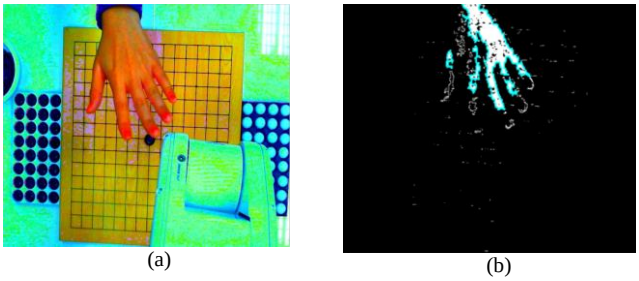
To avoid the robot touching the user during the game, a protection system is employed that includes dynamic object detection and skin color detection. The robot needs to recognize when a user is making a play, so a dynamic object detection system is designed, as shown in Figure 10. After the robot plays a stone, an image is detected by the camera; as long as any object is in the region; the system will not direct the robot to move. Figure 10 (a) is the real image, and Figure 10 (b) is the image obtained from the detection system. Any object besides stones will automatically be detected by the system; in this case, the fingers show that a user is placing a stone on the board. The robot will wait until no more image is detected to perform the next step.



(a) (b)

Fig. 10 (a) Real image (b) Image from dynamic detection system.

For the skin color detection system to take into account differences in race, ambient light brightness, and camera factors, YCbCr, HSV, and RGB color spaces are used to distinguish color regions. Hence, the PCM (Possibilistic C-Means) grouping method is used to determine the skin color region. The minimized cost function of PCM for finding the skin color system is shown in (5), employed to find the region of the skin. In equation (5), x_i, $i=1,...,N$ represents the i^{th} data of the total N data in the set X, θ_j, $j=1,...,m$ represents the representatives of m groups (each represented by C_j), the whole set is expressed as Θ. U is a matrix, and u_{ij} is its element. The (i, j) term represents the degree of compatibility of the i^{th} data vector x_i with the j^{th} representative θ_j. γ_j as a positive parameter indicating that each parameter is associated with the cluster C_j. Hence, the results of skin color detection are shown in Figure 11 (a) and (b). Figure 11 (a) is the real image, and Figure 11 (b) is the image from the skin color detection system, indicating that the skin color of a user is detected. Then the robot will stop immediately to protect the user.



(a) (b)

Fig. 11 (a) Real image (b) Image from dynamic detection system

$$J_{PCM}(U,\Theta)=\sum_{i=1}^{N}\sum_{j=1}^{m}u_{ij}^{q}\left\|x_i-\theta_j\right\|^2+\sum_{j=1}^{m}\gamma_j\sum_{i=1}^{N}(1-u_{ij})^q \quad (5)$$

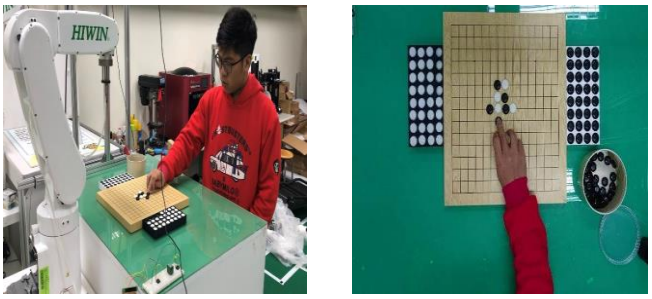
IV. Experimental Results

In this section, we demonstrate the results of a game played between the robot and user. Figure 12 shows the robot can suction a stone from the plate to the board successfully.

(a) (b)

Figure 12. (a) Robot suctioning a stone (b) Robot places the stone.

Next, the user plays in Figure 13 (a) and the position of the stone can be obtained from the camera above as shown in Figure 13 (b).



(a) (b)

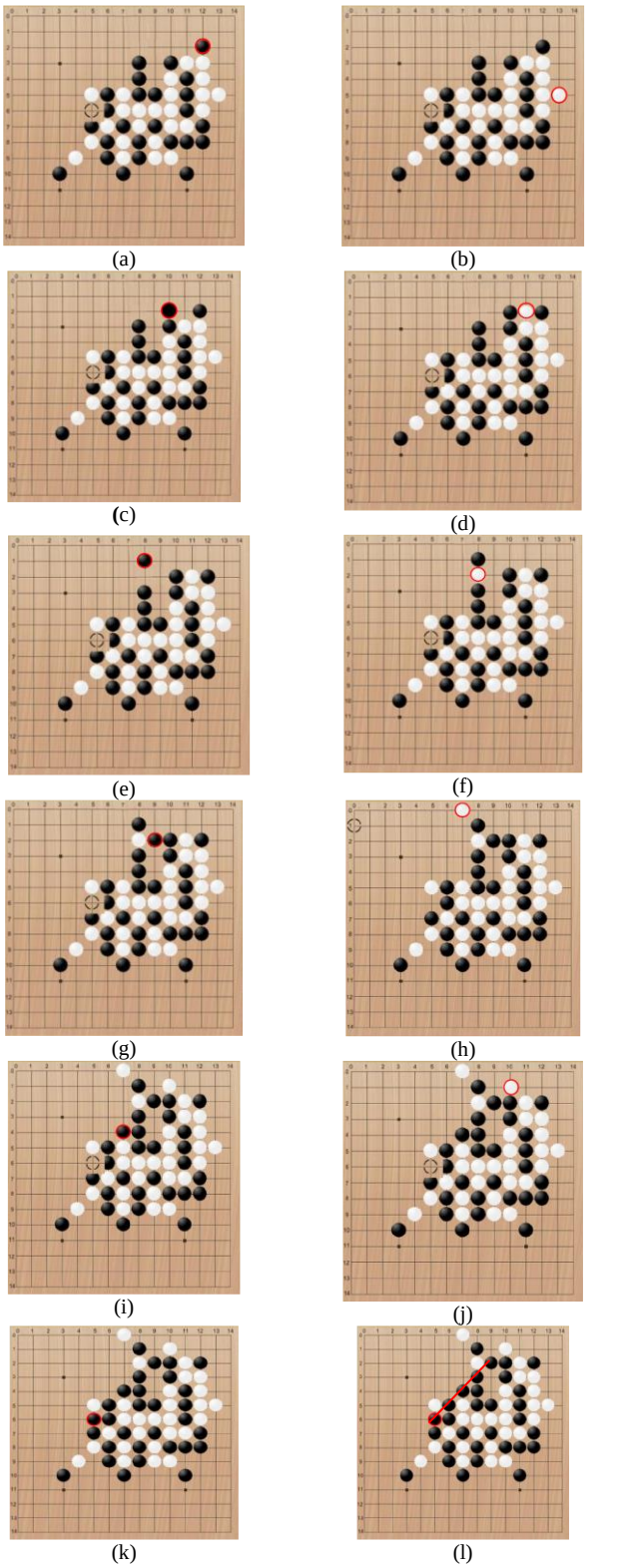
Figure 13. (a) User plays (b) Image for the robot

A. *Maximum Weighting Value*

Figure 14 shows the steps of the robot playing by using the maximum weighting value. In Figure 14(a), the user places a stone at (12, 2) position. Then the robot calculates that the maximum weighting value is at (13, 5) in Figure 14(b). The user places a stone at (10, 2) in Figure 14(c), and the robot calculates that the maximum weighting value is at (11, 2) in Figure 14(d). The user places a stone at (8, 1) in Figure 14(e), and the robot places a stone at (8, 2) in Figure 14(f). The user and the robot subsequently place stones at (9, 2), (7, 0), (7, 4), (10, 1), as shown in Figure 14(g), (h), (i), and (j), respectively. Finally, the user places a stone at (5, 6) in Figure 14(k), which forms a diagonal line, as shown in Figure 14(l), so the robot loses the game. The maximum weighting value only considers the current state, and it cannot predict all of the user's plays.

B. *Layer Tree Search and Reinforcement Learning*

To show partial results of the layer tree search and reinforcement learning method for the same situation, the image of Figure 14(a) is also shown in Figure 15(a). In this case, the robot calculates the position based on the layer tree search and reinforcement learning algorithm and finds that placement



(a) (b) (c) (d) (e) (f) (g) (h) (i) (j) (k) (l)

Figure 14 Steps for the robot playing the game by maximum weighting value and losing.

at position (8, 1) will allow the user to win in several steps. Therefore, the robot places a stone at (8, 1) in Figure 15(b) to block the user from placing a stone there. The game will

continue for a long time if the robot calculates several steps ahead using this method.

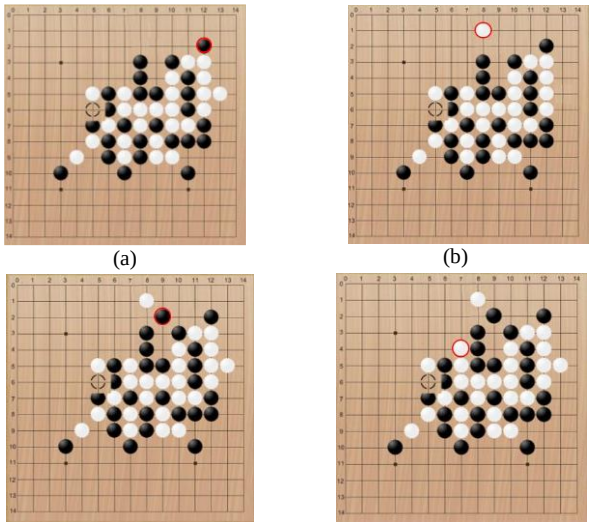


Figure 15 Steps for the robot playing the game using Layer Tree Search and Reinforcement Learning

The results of 10 test games executed between the robot and a user are summarized in Table IV. First, the robot uses maximum weighting value to determine its moves and the results show that the robot wins 8 games out of 10 for a winning percentage rate of 80%. Next, the robot changes its algorithm to Layer Tree Search and Reinforcement Learning to play games with the same user. The robot wins 10 games in 10 for a winning percentage of 100%.

TABLE IV
SUCCESS RATE FOR THE ROBOT EMPLOYING DIFFERENT METHODS

Method	Success/Games	Percentage won
Maximum weighting	8/10	80%
Layer tree search and RL	10/10	100%

V. CONCLUSIONS

In this paper, an intelligent robot is designed to play the five-in-row game with a human. In the system design, a 6-axis robot arm with a vision system is used to recognize the positions where stones are placed and the proposed algorithm calculations are used to win the game. The maximum weighting value algorithm can only win in some cases because it solely calculates the current situation without considering the following steps. The layer tree search and reinforcement learning algorithm helps the robot simulate several steps ahead until the outcome is a win, so the robot will not place a stone in a position which is merely the current best choice, but which not the best choice after two or three steps. The system is also equipped with a protection system so that a human arm will not be accidentally hurt by the robot arm. The results show that the robot can automatically play this game with a human, and even win games consistently by employing an artificial intelligence system.

REFERENCES

[1] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A Survey of Monte Carlo Tree Search Methods," *IEEE Transactions on Computational Intelligence and AI in Games*, Volume 4, Issue 1, March 2012, pp. 1-43.

[2] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis, "Mastering the Game of Go without Human Knowledge," *Nature*, Volume 550, pp. 354–359, October 2017.

[3] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis, "Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm," *arXiv*:1712.01815, 2017.

[4] B. K. Chakraborty and M. K. Bhuyan, "Skin segmentation using Possibilistic Fuzzy C-means clustering in presence of skin-colored background," 2015 *IEEE Recent Advances in Intelligent Computational Systems*.

[5] Y. Lei, W. Yuan, H. Wang, Y Wenhu, W. Bo, "A Skin Segmentation Algorithm Based on Stacked Autoencoders," *IEEE Transactions on Multimedia*, Volume 19, No. 4, pp. 740-749, 2017.

[6] M. S. Sultan, X. Chen, G. Ma, J. Xue, W. Ni, T. Zhang, and W. Zhang "Hand-eye 3D Pose Estimation for a Drawing Robot," *IEEE International Conference on Mechatronics and Automation* , pp.1325–1331, 2013.

[7] C. C. Chang, T. L. Chia, and C. K. Yang, "A modified temporal difference method for change detection," *Optical Engineering*, Volume 44, No. 2, Feb. 2005, pp. 027001.

Shu-Yin Chiang received M.S. and Ph.D. degrees from the University of Michigan, Ann Arbor, in 1994 and 1999, respectively, both in electrical engineering. She is currently Chair and Professor in the Department of Information and Telecommunications Engineering, Ming Chuan University, Taiwan, R.O.C. Her research interests include intelligent control, robot applications and wireless sensor networks.

Chien-Wu Tsai received a B.S. degree from Tatung Institute of Technology, Taipei, Taiwan, in 1987, a M.S. from University of Southern California, Los Angeles, in 1992, and Ph.D. degree from National Taiwan University in 2001, all in computer science and information engineering. From 1995 to 2004, he held several teaching positions in the Electronic Engineering Department, Lunghwa University of Science and Technology, Taoyuan, Taiwan. Currently, he is an Associate Professor with the Department of Information and Telecommunications Engineering, Ming Chuan University, Taoyuan, Taiwan. His current research interests are in areas of computer vision, robot control and video signal processing.

Hao-Ge Jiang received B.S. degree from Ming Chuan University, Taoyuan, Taiwan, in 2019 in information and telecommunications engineering. He is currently a Ph.D. student of School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore. His research interests include intelligent control, and robot applications.