Robotic Go: Exploring a Different Perspective on Human-Computer Interaction with the Game of Go

Tae-Hyung Kim, Adam Nisbett, Donald C. Wunsch II
Department of Electrical and Computer Engineering
Missouri University of Science and Technology
Rolla, USA
tk424@mst.edu, jan976@mst.edu, dwunsch@mst.edu

Abstract—The advent of computers and the World Wide Web diversified the way game of Go is played. While the traditional human-to-human player still remains an important form of a game play, amateur players along with some professional players have been being shifted the play domain from “off-line” club house to “on-line” Go servers. On the other hand, computer Go is an important field of study to develop a software to play Go or Go engine. In addition to human-to-human play setting, Go engine or computer intelligence to play Go adds another axis to play configuration: human-to-computer play and computer-to-computer play. These revolutions in the game of Go happened in an extremely short period of time compared to the history of the game, more than 4,000 years. We summarize this unavoidable change of waves for the first time in the literature, to our knowledge, and propose a novel way to interact with the current technological advances. We present the new Human-Machine-Computer-Network Interface concept and our implementation of the machine interface with a robot arm. This Lynxmotion robotic arm successfully places stones on a board under the proposed architecture.

Keywords—Baduk robot, Go robot, Weiqi robot, entertainment robot, baduk, game of Go, weiqi, human-machine-computer interface

I. INTRODUCTION

It has been only about two decades since the traditional way of playing a board game face-to-face has been diversified. The advent of a computer and World Wide Web (WWW) revolutionizes the conventional human-to-human player pattern. On the other hand, the entertainment industry is an established market and the entertainment robot is at its dawn and fast-growing. In the future, the day will come when robots play an inevitably important role in the entertainment industry.

The game of Go [1] is a two-player game especially popular in East Asia and is gaining more popularity in regions other than this region. The game is called Baduk in South Korean and Weiqi in China and its history spans more than 4,000 years. The popularity of Go in the East exceeds or is comparable to that of Chess in the West. In the meantime, the Artificial/Computational Intelligence society has been paying increasingly more attention to Go. Now, computer Go [2, 3] is considered to be a new unconquered challenge in these fields as IBM's computer Chess program Deep Blue defeated a human Chess master Garry Kasparov in 1997.

No paper about Go robotics was found in the literature. Lack of papers on Go robotics is not because of the field's insignificance, but because of the following combined reasons: immature market, invisibility of the game to enterprising engineers and scientists, and its multidisciplinary nature. While there are unique challenges in Go robot, one of the technical challenges for the authors is knowledge and experience across multiple field of studies. That is, background on computer Go, socket programming, image processing, and robotics was prerequisite, so research cooperation is essential.

There are several contributions by the authors. The first and most important is we are the first to our knowledge in the literature representing works on Go robot. Additionally, we are the first to our knowledge to articulate the human-computer interface in the literature. Another contribution is the proposed architecture interfacing GTP (Go Text Protocol) [4]. GTP is the de facto standard for computer Go engines to interface a Go GUI program that displays Go board and stones.

This paper is organized as follows. Section II overviews game of Go and computer Go. Section III presents the high level view of Human-Machine-Computer interface for game of Go and then our architecture is proposed in section IV. Section V shows sampled results. Section VI is the conclusions and the following sections present reference, and acknowledgments.

II. GAME OF GO AND GO SOFTWARES

A. Game of Go: Basic Game Rules

Go is a territory game on a board with intersections. The full board size has 19x19 intersections and smaller board sizes (6x6, 9x9, and 13x13) are used for learning and research purposes. Black and white players alternately place a stone on the board to obtain higher score, essentially more territories, than the opponent. A territory is an empty intersection at the end of the game. Figure 1 shows two world top professional players discussing the territories on the board.

The basic game rule is simple. The placed stones form a group with adjacent stones and they remain on the board unless they are captured. Adjacency is defined only in
parallel or vertically along the lines, not diagonally. In Figure 2, the stones on the top row form groups. On the right corner, eleven stones construct a group. Conversely, none of the stones on the second row are grouped because all of them are located diagonally.

Figure 1. Final at the 27th KBS King of Baduk Title, Changho Lee vs. Sedol Lee, Mar. 16, 2009

Figure 2. Artificially placed stones to explain adjacency (top) and liberty (bottom) concepts.

On the other hand, a group of stones (including a group of one stone) is captured when the last liberty of a group is removed, i.e. surrounded fully by the opponent's stones. A liberty is defined as an empty adjacent intersection of an occupied stone position. On the bottom left of Figure 2, a liberty is marked with a square or a triangle. A triangle is used to emphasize that the corresponding liberty is from an occupied stone located either vertically or in parallel, but diagonally. The black stones on the bottom right should be removed because all the liberties are taken by the opponent or white stones. Another basic rule is Ko rule to avoid infinite loop of the recurring board status. One cannot place a stone at where the board status for the previous play is repeated.

B. Categories of Go Softwares

A term Go software [5] is so polysemous that the accurate meaning should be clarified to discuss further. First off, computer Go is a field of study to create a computer program that plays the game of Go. Its ultimate goal to date is to defeat the human champion. Such a program is referred to as Go engine, computer Go program, or computer Go intelligence. Some of famous Go engines are The Many Faces of Go, MoGo, GNU Go, Silver Star, and Fuego.

Other categories of Go software of our interest are Go client program and Go editing program. Go client program is a client program to connect a Go server, a pool of Go players. It follows the client-server model and allows one to play against the other from the pool as illustrated in Figure 3. Go editing program, more specifically Go GUI (Graphic User Interface) program, provides a GUI that displays the board status. One can also store and edit game records with it. For example, Figure 2 is a screen capture of an editing program GoGui. Typically, well-designed Go client also provides the editing feature.

Figure 3. Go client program: (a) the client-server concept to overcome spatial constraint to connect Go players around the world, (b) list of connected players at Cyberoro, a server recognized by Korea Baduk Association.

Here, we define two new terminologies: Go Interface Software (GIS) and Go Interface Protocol (GIP). The former refers a computer program serves as a front end to a human player. For example, both Go client program and editing program are GIS with different features. The latter is a protocol used by GIS to communicate with a Go Engine. GIP sets a group of standard commands/ procedures so that GIS can exchange information about game plays with a Go engine. The De facto standards are GMP (Go Modem Protocol) and the newer GTP (Go Text Protocol).

III. BIG PICTURE: PLAYER CONFIGURATIONS AND THE INTERFACE ISSUES

GIS plays a central role in interconnecting human and computer players over the network. The interface issues can be best-explained with block diagrams. Figure 4 summarizes the existing player configurations and the corresponding interface issues. Additionally, Figure 4 illustrates the unified block diagram that explains how human, computer, and network are interconnected. Note both figures are coherent and Figure 5 unifies the existing structures in Figure 4.

<table>
<thead>
<tr>
<th>Human</th>
<th>Computer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human</td>
<td>(HHP, HTI\textsuperscript{2}), (HHP, HCI\textsuperscript{2}), (HHP/N, HCNI\textsuperscript{2})</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Computer</td>
<td>Reciprocal to HCP</td>
</tr>
</tbody>
</table>

Table 1. Summary of play configuration and interfaces.

<table>
<thead>
<tr>
<th>Player</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Human</td>
<td>Computer</td>
</tr>
<tr>
<td>Local</td>
<td>Input/ Output Devices</td>
</tr>
<tr>
<td>Global</td>
<td>Network Interface</td>
</tr>
</tbody>
</table>

Table 2. GIS's counterpart block for different opponents.
In general, a human player can choose to play against another human player or a computer player (Go Engine) of which choice can be local or global via network. Say the opponent is chosen to be a local human player. The interface between them can be a traditional board and stones, Figure 4 (a), or a computer Figure 4 (b). Typical input and output devices for the latter are mouse and computer monitor. Note that playing on a board and stones denoted as (HHP, HTI²) is preferred not because the inertia of the traditional ways still exists, but because a uniquely interesting factor in Go is a pleasantly keen sound upon placement of a stone. This will be discussed further in the following section. Another popular configuration is to play against other human players over the network denoted as (HHP/N, HCNI²). Typically, the counterpart computer is a Go server, so GIS serves as Go Client Program. In a Go program, Internet Socket serves as the universal gateway to the Network Interface as well.

Regarding a computer player, the first human play against a Go engine playing a full game appears in the literature as by Zobrist [6]. The structure of (HCP, HCI-CHI) evolved from the monolithic structure in [6], which GIS and Go Engine are not modularized. Currently, GIS and Go Engine are separately modularized because GIP is a de facto standard protocol to communicate between them. Another HCP configuration is depicted in Figure 4 (g), (HCP/N, HCNI-CNI). This is a setting to play against a global computer player. On the other hand, CCP is becoming more important as development of a strong Go engine became an important research topic. Figure 4 (d) and (f) present the two existing settings. The former is a local test to set matches between two Go engines. A tool such as twogtp that interfaces GTPs of two Go engines corresponds to GIS. The latter is the structure for a global test to a computer Go server (CGOS) which automatically organizes matches between Go engines. CGOS mandates to use a custom client program and to implement GTP as the GIP.

In sum, Figure 4 and Figure 5 are tabulated in Table 1 and Table 2, respectively. The former is about the existing interface issues upon different player configuration and the latter provides a framework to unify Figure 4. Note all the player configuration and interfaces are nicely wrapped into Figure 5.

IV. PROPOSED ARCHITECTURE FOR HUMAN-MACHINE-COMPUTER INTERFACE

A. Dilemma of a contemporary player: traditional board and stones or online Go server?

Contemporary players suffer from a dilemma to enjoy a Go game. One can choose HHP or HHP/N. Choice between human and computer is not a dilemma because computer is much weaker than human. An intermediate player can easily defeat, moderately speaking, most of Go engines available on Go servers.

Uniquely to this game, the stone placement sound plays an important role in the enjoyment of Go. A beginner learns how to hold a stone properly in order to make a pleasantly keen sound when a stone is placed on a board. Figure 6 (a)
shows the proper way by holding a stone between index and middle fingers. The stone should be smashed down by the middle figure. Impact of the sound is considered to reflect the player's level and experience. Figure 6 (b) shows a world top-level professional, Sedol Lee (9 Dan), placing a stone with curvy fingers and wrist to magnify the stone placement sound. This small gesture reflects his gallantly keen mind with experience and skills in this game.

Figure 6. Stone placement (a) proper way to hold a stone, (b) Sedol Lee's stone placement exposing his experience in the game.

The dilemma occurs due to the interface issues. Advantages of HHP is the stone placement sound and touches that add pleasure to game plays. The disadvantage is the limited range of opponents caused by the space-time constraints. On the other hand, players on the HHP/N setting typically use a mouse and a computer monitor as the input/output device. Its obvious advantage is a large pool of opponents that overcome the space-time constraints of HHP. The drawback is a player should suffer from the removal of the pleasant touches and placement sounds. Note players prefer to choose (HHP, HTI²) over (HHP, HCI²) due to the dullness of a mouse and computer monitor.

B. Playing on a Go server is an unavoidable path to decay amateur club houses

A club house was a part of Go culture, which served as a pool of players. Its domain is being shifted on-line, which caused decay in amateur club house culture.

A traditional club house is a system that partly compensates the space-time constraints of HHP. It is referred to as kwon in Korean, ki-in in Japanese, and qi yuan in Chinese, which may also mean a nation-wide association. A national association such as Hanguk Kiwon (Korea Baduk Association), Nihon Ki-in (Japan Go Association), and Zhongguo Qiuyuan (Chinese Weiqi Association) is a club house for professional players and subsequent organizations.

In the arena of amateurs, the club house culture is declining. Popularity of Go servers directly impacts the industry. For example, club houses in South Korea once popular are disappearing. Playing Go on-line is now an unavoidable social phenomenon.

C. Proposed architecture to solve the dilemma

Figure 7 depicts the proposed architecture to provide a solution to a Go player's dilemma: choice between traditional equipments and a Go server. The key idea is to use a robot or machine to play a game for the opponent. Note two additional blocks are added in Figure 7 compared to Figure 5. The traditional equipments of a board and stones are placed between the human user on the left and the machine. Figure 8 details the unified structure in Figure 7. It illustrates possible player configurations and the corresponding interface issues and all of them are nicely coherent to the existing structures explained in Figure 4.
V. IMPLEMENTATION OF GO ROBOT

Our implementation of Go robot successfully places stones in the 9x9 board. The computer’s interaction with the physical board is separated into two independent functions: identifying where the pieces are located on the board as well as when the human player has made a move, and manipulating stones on the robot’s turn.

![Diagram of robotic arm](image)

Figure 9. (a) Ideal model of robotic arm, (b) Hardware implementation of the machine input/output interface.

The first task is passive and is accomplished by using a standard webcam positioned above the board as shown in Figure 9 (b). The images are processed with the Roborealm software [7] in order to determine the position of the playing board in the image, and which intersections contain stones. The second task is accomplished using the Lynxmotion robotic arm [8] with a modified gripper attachment to move stones.

A. Vision System and Image Processing

Roborealm is used to sample and process images from the webcam in order to determine the board state. Red markers located at the corners of the board are used to crop and transform the image so that the board intersections occur at known locations. Then the image is processed using basic threshold and blob filtering tools as shown in Figure 10 so that each stone is isolated.

![Image processing sequence](image)

Figure 10. An example of image processing sequence

After the image is filtered, each blob’s center of gravity is compared with the intersection locations and it is marked with the closest intersection. If the stone is not within the area of the playing board it is marked as being off the board and its coordinates are stored.

If motion is detected in the images (by comparing several consecutive frames) the image processing sequence is suspended until the image stabilizes. The board state can then be compared to the state in memory to determine if the human player has made a move.

B. Stone Manipulation with Robotic Arm

The Lynxmotion robotic arm has six degrees of freedom and is controlled by the Roborealm software through the Lynxmotion SSC-32 servo controller board.

If the robotic arm is modeled as an ideal mechanism as represented in Figure 9 (a), the Cartesian coordinates of the ‘wrist’, along with the angle of the hand from a vertical position, can be transformed into the necessary angles for each servo using the following equations:

$$\theta_0 = \tan^{-1}\left(\frac{x}{y}\right)$$  

$$\theta_1 = \frac{\pi}{2} - \tan^{-1}\left(\frac{x}{\sqrt{x^2 + y^2}}\right) - \cos^{-1}\left(\frac{l_1^2 + x^2 + y^2 + z^2 - l^2}{2l_1\sqrt{x^2 + y^2 + z^2}}\right)$$  

$$\theta_2 = \cos^{-1}\left(\frac{x^2 + y^2 + z^2 - l_1^2}{2l_1}\right)$$  

$$\theta_3 = \pi - \theta_1 - \theta_2, \quad \theta_4 = \theta + \frac{\pi}{4}$$  

In order to compensate for the non-ideal characteristics of the actual arm, caused by backlash in the servo gears and sag due to gravity, the Cartesian coordinate input necessary for each intersection was found manually. Roborealm accepts input specifying the board positions that a stone should be picked up and placed, and the necessary coordinates are looked up and used to calculate the servo angles.

The stones used for playing Go are difficult for a two-fingered machine to manipulate, due to their rounded shape, smooth surface and low profile. The tips of the gripper mechanism were not able to reliably pick up the stones with the standard rubber tips. Through experimentation, a successful design was achieved through a wire attachment to the tips of the gripper mechanism. Figure 11 shows the configurations that were tested during the process of designing the modified tips.

The original configuration with rubber tips as shown in Figure 11 (a) was not able to pick up the stones due to their low profile. The rubber tips were removed as in Figure 11 (b), and then the stones could be picked up, but a very slight error in positioning of the tips would cause the stone to slide out from between the grippers. Figure 11 (c) shows a four-pronged wire attachment to the grippers that proved somewhat successful, but still required fairly precise positioning. The last design as shown in Figure 11 (d) proved to be successful in consistently picking up the stones, even when there was positioning error of up to one quarter of the distance between grid intersections.
VI. CONCLUSIONS

There are two major contributions of this paper. The first contribution is to introduce an unified architecture for the player configuration and interface issue that explains the existing player configurations and interface issues. The second and more important contribution is to propose an unified Human-Machine-Computer-Network Interface that is implemented with a Lynxmotion robot arm.

Regarding the first contribution, the existing player configurations and interface issues are summarized and an unified structure is introduced to explain the current settings. In this process, two new terminologies are defined: Go Interface Software and Go Interface Protocol. The former refers to any software that interfaces the existing components. For example, Go software such as Go client program, Go editing program, and twogtp fall nicely into the category of Go Interface Software. The latter, Go Interface Protocol, is defined as a protocol used by Go Interface Software to communicate with a Go Engine. It is a group of standard commands/ procedures so that Go Interface Software can exchange information about game plays with a Go engine and the current De facto standard is Go Text Protocol.

The second contribution can be interpreted as a solution to “A Go player’s dilemma.” This dilemma has been existing since a computer is used to play a Go game. In these days, a player should choose between pleasantness of traditional human interface, i.e. Go board and stones, and ease of access to a pool of other Go players or Go server by using the “dull” input/ output devices of a computer. The proposed Human-Machine-Computer-Network Interface concept along with a “Go playing robot” address this issue.

Our prototype robot arm successfully places stones on a 9x9 board. The implementation requires multi-disciplinary background on computer Go, socket programming, image processing, and robotics and extensive amount of time to experiment the proper shape of a robot arm tip to stably pick and place a stone of which shape may slight vary.

VII. ACKNOWLEDGMENT

Authors would like to thank Intelligent Systems Center and National Science Foundation for the financial support to perform this research. This paper is also submitted to 2009 IEEE International Conference on Systems, Man, and Cybernetics, San Antonio, Texas, USA.

VIII. REFERENCES

[5] “Sensei’s library - a collaborative web site about and around the game of Go (also called Igo, Weiqi, and Baduk),” available online at http://senseis.xmp.net/.