Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

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

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

***Abstract* — The advent of computers and the World Wide Web diversified the way in which the game of Go is played. While traditional human-to-human play still remains an important form of game play, amateur players, along with some professional players, have shifted the play domain from "off-line" club houses to "on-line" Go servers. Computer Go is an important field of study to develop a software to play Go or a Go engine. In addition to human-to-human play, a 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, which is more than 4,000 years. We summarize this unavoidable change 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 named Cheonsoo-I 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, human-robot interface

I. INTRODUCTION

The entertainment industry is an established market, and the entertainment robot is at its dawn and growing quickly. In the future, the day will come when robots play an inevitably important role in the entertainment industry. However, 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 computers and the World Wide Web (WWW) revolutionizes the conventional human-to-human player pattern.

The game of Go [1] is a two-player game especially popular in East Asia and is gaining more popularity in other regions. The game is called Baduk in South Korea and Weiqi in China; 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 similar to when IBM's computer Chess program Deep Blue defeated human Chess master Garry Kasparov in 1997.

An extensive literature search reveals no paper publications about Go robotics. Therefore, the literature survey should be stated in the context of entertainment robotics. Dautenhahn et. al. in [4] provide a solution to an intellectual question of what a robot companion is. Their survey results show that a majority of people are favorable to the idea of a robot companion. Our work was performed based on the hypothesis of a favored robot companion. Goodrich and Schultz in [5] review Human-Robot Interaction and discuss it as a growing field of research and application which requires interdisciplinary understanding. They explore the topics of Human-Computer Interaction, Artificial Intelligence and Cybernetics as important fields of study. Our experience on this work indeed supports their view on the multi-disciplinary nature of Human-Robot Interaction. While unique challenges exist in Go robotics, one of the technical challenges for the authors is knowledge and experience across multiple fields of study. That is, a background on computer Go, socket programming, image processing, and robotics was prerequisite. Very few papers are found in the literature about board game robots. Chess-playing robots are found [6, 7, 8], probably because of the game's popularity.

The authors make several contributions through this paper. The first and most important is that we are the first, to our knowledge, to contribute work on Go robots and to articulate the human-computer interface in the existing literature. Another contribution is the proposed architecture interfacing GTP (Go Text Protocol) [9]. GTP is the *de facto* standard for computer Go engines to interface a Go GUI program that displays a Go board and stones.

This paper is organized as follows. Section II overviews the game of Go and computer Go. Section III presents the high level view of the Human-Computer-Network interface for the game of Go, and then our architecture is proposed in Section IV. Section V summarizes our implementation of a Go robot based on the proposed architecture. Section VI concludes the paper, and the following sections present acknowledgments and references.

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. One player represented by black stones and the other

978-1-4244-2794-9/09/$25.00 ©2009 IEEE

Figure 1. Final at the 27th KBS Baduk-wang Jeon (Battle of the Baduk King), Changho Lee vs. Sedol Lee, Mar. 16, 2009, South Korea.

player represented by white stones alternately place a stone on the board to obtain a higher score, essentially more territories, than the opponent. A territory is an empty intersection at the end of the game. Fig. 1 shows two 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 Fig. 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.

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 Fig. 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 not diagonally. The black stones on the bottom right should be removed as captured stones because all the liberties are taken by the opponent, or white stones. Another basic rule is the Ko rule to avoid an infinite loop of recurring board status. One cannot place a stone that causes the board status from the previous play to be repeated.

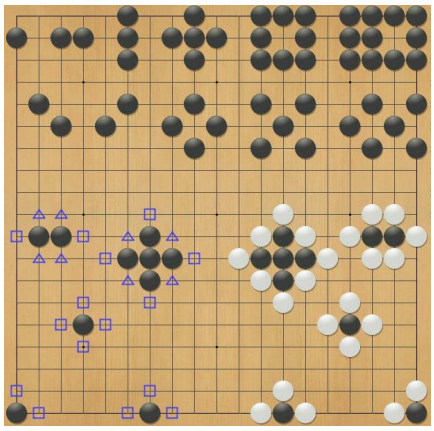

Figure 2. Artificially placed stones to explain adjacency (top) and liberty (bottom) concepts.

B. *Categories of Go Software*

The term *Go software* [10] is so polysemous that the accurate meaning should be clarified before further discussion. Firstly, 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 a Go engine, computer Go program, or computer Go intelligence. Some famous Go engines are The Many Faces of Go, MoGo, GNU Go, Silver Star, and Fuego.

Other categories of Go software of interest to us are the Go client program and the Go editing program. The 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 another from the pool as illustrated in Fig. 3. The Go editing program, more specifically the 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, Fig. 2 is a screen capture of an editing program GoGui. Typically, a well-designed Go client also provides the editing feature.

Here, we define two new terminologies: *Go Interface Software (GIS)* and *Go Interface Protocol (GIP)*. The former refers to a computer program that serves as a front end to a human player. For example, both the 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).

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 the Korea Baduk Association.

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. Fig. 4 summarizes the existing player configurations and the corresponding interface issues. Additionally, Fig. 5 illustrates the unified block diagram that explains how human, computer, and network are interconnected. Note that both figures are coherent; Fig. 5 unifies the existing structures in Fig. 4.

In general, a human player can choose to play against another human player or a computer player (Go Engine); the location of the opponent can be local or global. Global means the opponent is located across the network. Given that the opponent is chosen to be a local human player, the interface between them can be a traditional board and stones, Fig. 4 (a), or a computer, Fig. 4 (b). Typical input and output devices for the latter are a mouse and computer monitor.

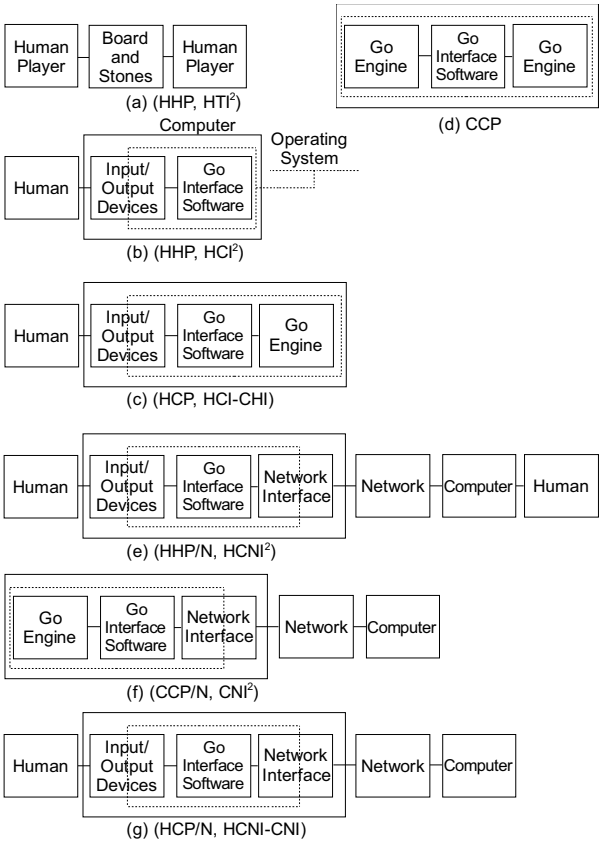

Figure 4. Player configurations and the corresponding interface issues. (a) HHP (Human-to-Human Player) with HTI (Human-Traditional equipment Interface), (b) HHP with HCI (Human-Computer Interface), (c) HCP (Human-to-Computer Player) with HCI-CHI (Computer-Human Interface), (d) CCP (Computer-to-Computer Player), (e) HHP/N (HHP over the Network) with HCNI (Human-Computer-Network Interface)-HCNI, abbreviated as $HCNI^2$, (f) CCP/N (CCP over the Network) with CNI (Computer-Network Interface)-CNI, abbreviated as CNI^2, (g) HCP/N (HCP over the Network) with HCNI-CNI. The notation is (Player configuration, Interface of a player (left) – Interface of a player (right)). If interfaces for both players are identical, a square abbreviates the notation, e.g. HCI^2 instead of HCI-HCI.

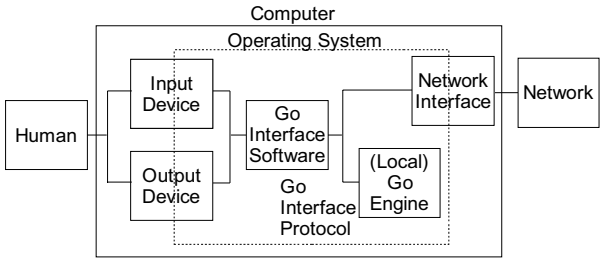

Figure 5. Block diagram of unified structure for human–computer-network interface. Go Interface Software selects the input among Go Engine, Network Interface, and Input/ Output Devices.

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, $HCNI^2$). Typically, the counterpart computer is a Go server, so GIS serves as a Go Client Program. In a Go program, an Internet Socket serves as the universal gateway to the Network Interface as well.

Regarding a computer player, the first documentation of a human playing a full game against a Go engine that appears in the literature is by Zobrist [11] published in 1970. The structure of (HCP, HCI-CHI) evolved from the monolithic structure in [11], in which GIS and Go Engine are not modularized. While some programs still maintain the monolithic structure, it is recommended that GIS and Go Engine are separately modularized because there is a *de facto* standard protocol to communicate between them. Another HCP configuration is depicted in Fig. 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 the development of a strong Go engine becomes an important research topic. Fig. 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.

TABLE 1. SUMMARY OF PLAY CONFIGURATION AND INTERFACES

	Human	Computer
Human	(HHP, HTI^2), (HHP, HCI^2) (HHP/N, $HCNI^2$)	(HCP, HCI-CHI) (HCP/N, HCNI-CNI)
Computer	Reciprocal to HCP	CCP (CCP/N, CNI^2)

TABLE 2. GIS'S COUNTERPART BLOCK FOR DIFFERENT OPPONENTS.

	Player	
	Human	Computer
Local	Input/Output Devices	Local Go Engine
Global	Network Interface	Network Interface

In summary, Fig. 4 and Fig. 5 are tabulated in Table 1 and Table 2, respectively. The former summarizes the existing interface issues upon different player configurations, and the latter provides a framework to unify Fig. 4. Note that all the player configurations and interfaces are nicely wrapped into Fig. 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 regarding how best to enjoy a Go game. One can choose HHP or HHP/N. The choice between human and computer is currently not a dilemma because the computer is much weaker than a human. An intermediate player can easily defeat, generally speaking, most 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 pleasant sound when a stone is placed on a board. Fig. 6 (a) shows the proper technique of holding a stone between index and middle fingers. The stone should be smashed down by the middle finger. The impact of the sound is considered to reflect the player's level and experience. Fig. 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 experience and skill in this game.

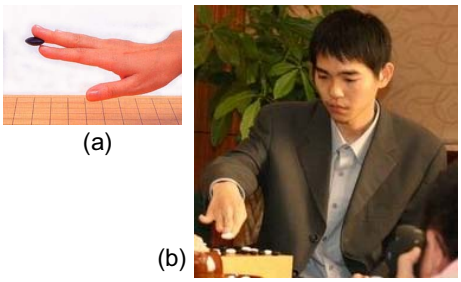

Figure 6. Stone placement (a) proper way to hold a stone, (b) Sedol Lee's stone placement exposing his experience in the game.

The dilemma occurs due to the interface issues. One advantage of HHP is the stone placement sound and the physical feel of the stone that add pleasure to game plays. The disadvantage is the limited range of opponents caused by space-time constraints. On the other hand, players on the HHP/N setting typically use a mouse and 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 the removal of the pleasant touches and placement sounds. Note that players prefer to choose (HHP, HTI^2) over (HHP, HCI^2) due to the unstimulating nature of a mouse and computer monitor.

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

A club house is a part of Go culture, which serves as a pool of players. Its domain is being shifted on-line, which caused a decay in the amateur club house culture. A traditional club house is a system that partly compensates for the space-time constraints of HHP. It is referred to as kiwon in Korean, ki-in in Japanese, and qiyuan in Chinese, which may also mean a nationwide association. National associations such as Hanguk Kiwon (Korea Baduk Association), Nihon Ki-in (Japan Go Association), and Zhongguo Qiyuan (Chinese Weiqi Association) operate club houses for professional players and supporting organizations**.**

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

C. *Proposed architecture to solve a Go player's dilemma*

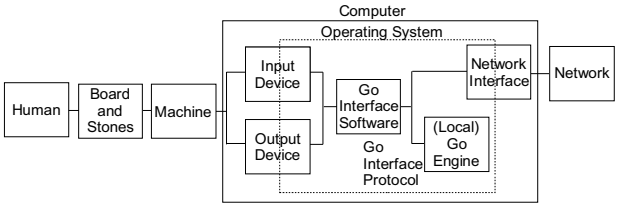

Figure 7. The proposed architecture of Human-Machine-Computer-Network Interface (HMCNI).

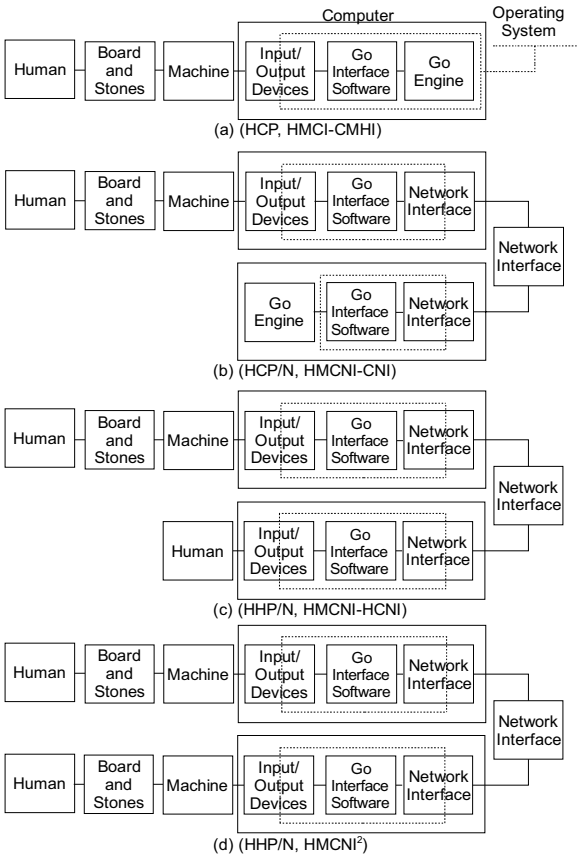

Figure 8. Player configurations and interface issues.

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

V. Implementation of A Go Robot: Cheonsoo-I

Our implementation of a Go robot successfully places stones on the custom-made 9x9 board. The stones have a diameter of 19mm and are 6.5mm thick. The computer's interaction with the physical board is separated into two independent functions: identifying where the pieces are located on the board and when the human player has made a move, and manipulating stones on the robot's turn.

The first task is passive and is accomplished by using a standard webcam, Logitech QuickCam for Notebooks, positioned above the board as shown in Fig. 9 (b). The images are processed with the Roborealm software, RoboRealm v.1.8.22.1 [12], in order to determine the position of the playing board in the image and which intersections contain stones.

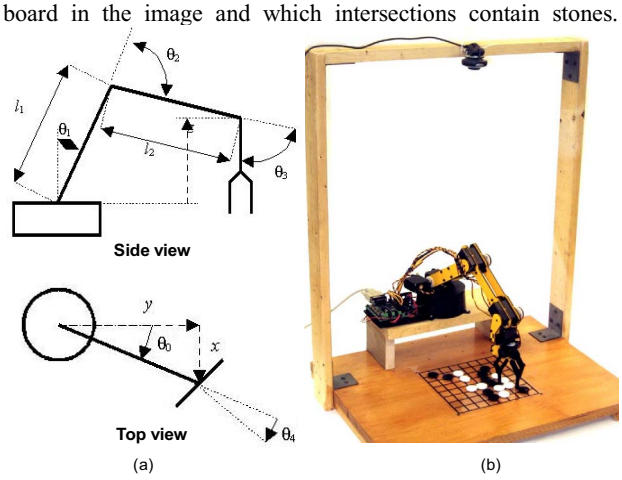

Figure 9. Cheonsoo-I (a) Ideal model of robotic arm, (b) Hardware implementation of the machine input/output interface.

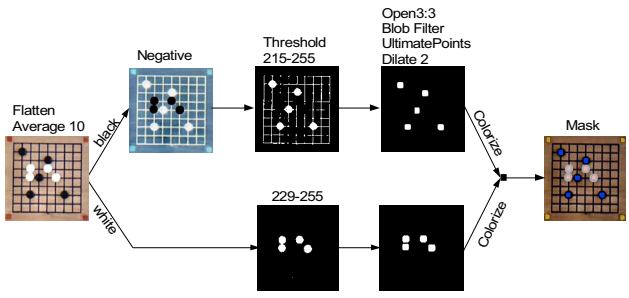

Figure 10. An example of image processing sequence.

The second task is accomplished using the Lynxmotion robotic arm, Lynx 6 Robotic Arm Combo Kit for PC [13], 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's state. Red markers located at the corners of the board are used to crop and transform the image so that the board's intersections occur at known locations. Then the image is processed using basic threshold and blob filtering tools as shown in Fig. 10 so that each stone is isolated.

After the image is filtered, each blob's center of gravity is compared with the intersection locations and 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's 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 Fig. 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:

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

$$\theta_1 = \frac{\pi}{2} - \tan^{-1}\left(\frac{z}{\sqrt{x^2+y^2}}\right) - \cos^{-1}\left(\frac{l_1^2 + x^2 + y^2 + z^2 - l_2^2}{2l_1\sqrt{x^2+y^2+z^2}}\right) \tag{2}$$

$$\theta_2 = \cos^{-1}\left(\frac{x^2+y^2+z^2-l_1^2-l_2^2}{2l_1l_2}\right) \tag{3}$$

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

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 standard rubber tips of the gripper mechanism were not able to reliably pick up the stones. Through experimentation, a successful design was achieved through a wire attachment to the tips of the gripper mechanism.

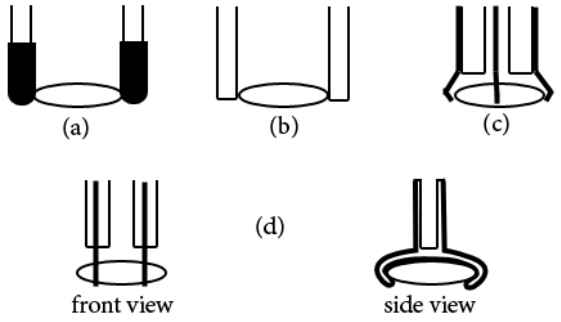

Figure 11. Gripper tips (a) unmodified with standard rubber tips (b) unmodified without rubber tips (c) four-pronged wire attachment (d) final design.

Fig. 11 shows the configurations that were tested during the process of designing the modified tips. The original configuration with rubber tips as shown in Fig. 11 (a) was not able to pick up the stones due to their low profile. The rubber tips were removed as in Fig. 11 (b), and then the stones could be picked up; however, a very slight error in positioning of the tips would cause the stone to slide out from between the grippers, which could catastrophically wreck the arrangement of the stones while playing an actual game. Fig. 11 (c) shows a four-pronged wire attachment to the grippers that proved somewhat successful, but it still required fairly precise positioning. The last design as shown in Fig. 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 a unified architecture that explains the existing player configurations and interface issues. The second and more important contribution is to propose a 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 a 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 with the existing components. For example, Go software, such as the 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 reinterpreted as a solution to "a Go player's dilemma." This dilemma has existed since a computer was first used to play a Go game. These days, a player must choose between the pleasantness of traditional human interface, i.e. Go board and stones, and ease of access to a pool of other Go players or a Go server by using the "unstimulating" input/output devices of a computer. The proposed Human-Machine-Computer-Network interface concept, along with a "Go playing robot," addresses this issue.

Our prototype robot arm Cheonsoo-I successfully places stones on a 9x9 board. The implementation requires a multi-disciplinary background on computer Go, socket programming, image processing, and robotics, as well as an extensive amount of time to experiment with the proper shape of a robot arm tip to stably pick up and place stones of slightly varied shapes.

VII. Acknowledgment

The authors would like to thank the Intelligent Systems Center, Mary K. Finley endowment, and the National Science Foundation, contract number ECCS-0725382, for the financial support to perform this research.

VIII. References

[1] Soo-Hyun Jeong, "Understanding the Contemporary Baduk," Nanampublisher, Seoul, South Korea, 2004, [in Korean].

[2] Martin Mueller, "Computer Go Survey," Artificial Intelligence, vol. 134, pp. 145-179, 2001.

[3] Bruno Bouzy and Tristan Cazenave, "Computer Go: An AI Oriented Survey," Artificial Intelligence, vol. 132, pp. 39-103, 2001.

[4] Kerstin Dautenhahn, Sarah Woods, Christina Kaouri, Michael L. Walters, Kheng Lee Koay, Iain Werry, "What is a Robot Companion–Friend, Assistant or Butler?," IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1192-1197, Aug.2-6, 2005.

[5] Michael A. Goodrich and Alan C. Schultz, "Human-Robot interaction: A Survey," Now Publishers, pp. 203-275, 2007.

[6] F.C.A. Groen, G.A. der Boer, A. van Inge, R. Stam, "A chess playing robot: Lab course in robot sensor integration," 9th IEEE Instrumentation and Measurement Technology Conference, Metropolitan, USA, pp. 261 - 264, 1992.

[7] Shuying Zhao, Chao Chen, Chunjiang Liu, Meng Liu, "Algorithm of location of chess-robot system based on computer vision," Control and Decision Conference, Yantai, China, pp. 5215-5218, 2008.

[8] Emir Sokic and Melita Ahic-Djokic, "Simple Computer Vision System for Chess Playing Robot Manipulator as a Project-based Learning Example," IEEE International Symposium on Signal Processing and Information Technology, Sarajevo, pp.75-79, 2008.

[9] "GTP - Go Text Protocol," available online at http://www.lysator.liu.se/~gunnar/gtp/.

[10] "Sensei's library (a collaborative website about he game of Baduk, Go, or Weiqi)," available online at http://senseis.xmp.net/.

[11] Albert Lindsey Zobrist, "Feature Extraction and Representation for Pattern Recognition and the Game of Go," Ph.D. dissertation, University of Wisconsin, 1970.

[12] "Roborealm – Robotic Machine Vision Software," available online at http://www.roborealm.com/.

[13] "Lynxmotion Robot Kit," available online at http://www.lynxmotion.com/