2014 14th International Conference on Control, Automation and Systems (ICCAS 2014)
Oct. 22–25, 2014 in KINTEX, Gyeonggi-do, Korea

# Playing Alkagi with a Humanoid Robot

Joonsig Gong and Songhwai Oh

Department of Electrical and Computer Engineering and ASRI,
Seoul National University, Seoul 151-744, Korea
E-mail: {joonsig.gong, songhwai.oh}@cpslab.snu.ac.kr

**Abstract:** This paper presents a humanoid robot playing a game called Alkagi. Alkagi is a game in which one shoots a Go stone to hit an opponent's stone with the goal of making the opponent's stone fall out of the Go board. When the position of a Go stone is obtained using a vision system, the robot places its hand at the desired position and orientation in order to shoot the stone to a desired direction. The proposed method is demonstrated using an Aldebaran NAO humanoid robot.

**Keywords:** Alkagi, Motion Planning, Aldebaran NAO.

## 1. INTRODUCTION

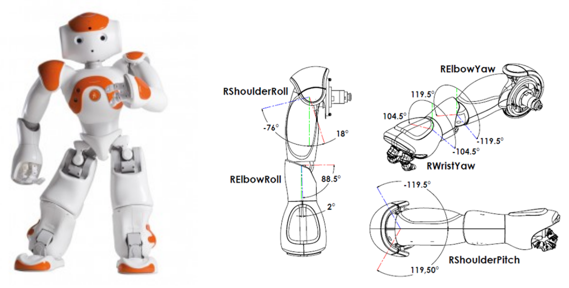
Making a companion robot in our daily life has become a popular research topic in robotics and artificial intelligence. There are a number of works which attempt to make a robot do the activities just like humans, such as playing soccer [1], table tennis [2], and chess [3]. It has been demonstrated that playing a game with a human is an extremely challenging task [1–3]. For example, when a robot plays the game of chess, it needs to figure out where the chess pieces are using its sensors, plan a motion to pick a piece and move it to a desired position, and learn a strategy to win a game.

In this paper, we present a humanoid robot playing a game called "Alkagi", which is a traditional Korean game using Go stones and board. An Aldebaran NAO humanoid robot [4] shown in Figure 1(a) is used in this paper. Alkagi is a Korean word meaning "hitting an egg" and the term egg refers to a Go stone. The game starts with two players having the same number of Go stones of different colors (black or white). Then each player takes a turn and shoots one of his or her stone with a finger to hit an opponent's stone in order to make the opponent's stone fall out of the Go board. The fallen stones cannot be used any longer. A player wins a game if all opponent's stones are cleared from the board. Figure 2 shows Alkagi being played on a Go board.

In order for a robot to play Alkagi, it has to first detect Go stones on the board using its vision system. Then the robot is required to determine which stone to hit and the hitting direction. The robot needs to place its hand based on the position of the chosen stone and the hitting direction using a motion planning algorithm.

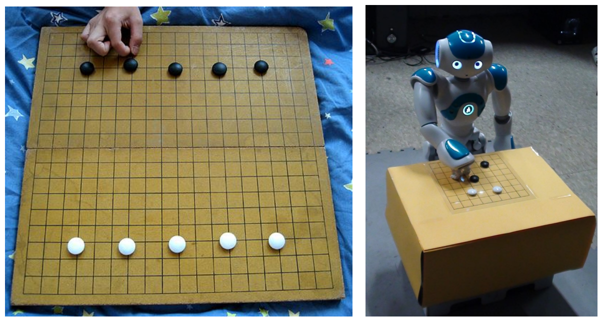
Since the game demands a high degree of accuracy even for a human, an accurate and elaborate planning method is necessary. To find a proper trajectory of a robot arm, we have applied a sampling-based motion planning algorithm called the rapidly-exploring random tree (RRT) [5], which is probabilistically complete. RRT has been extended and applied to a number of different path and motion planning problems by many researchers [6, 7].

The remainder of this paper is structured as follows. In Section 2, we provide an overview of our system for playing Alkagi and introduce our robot platform NAO. Section 3 discusses the motion planning method for finding the trajectory of a robot arm. Experimental results are presented in Section 4.



(a) (b)

Fig. 1 (a) An Aldebaran NAO humanoid robot. (b) Joints of the right arm of NAO and joint angle constraints.



(a) (b)

Fig. 2 (a) A photo of Alkagi being played on a Go board. (b) A photo of NAO playing Alkagi.

## 2. SYSTEM OVERVIEW

Figure 3 shows an overview of the proposed system for playing Alkagi. A robot can detect Go stones on the board using its vision system, classify them into black or white stones, and localize them. Then the robot decides which stone to hit and to which direction. The robot computes the position and orientation of its hand for the desired hitting motion. Then it finds a trajectory of its hand

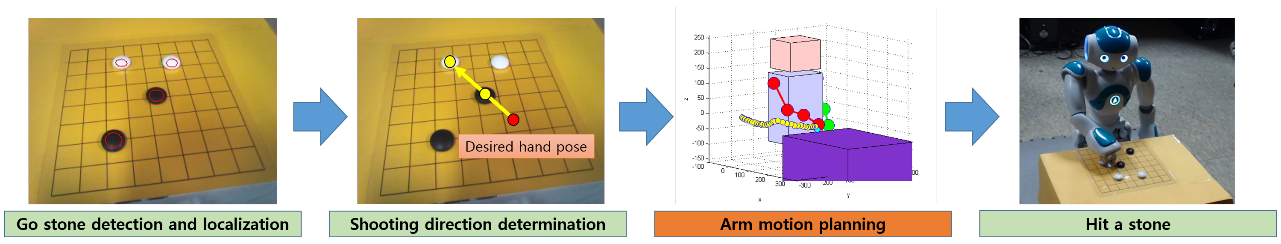

Desired hand pose
Go stone detection and localization
Shooting direction determination
Arm motion planning
Hit a stone

Fig. 3 An overview of the proposed system for playing Alkagi.

from its current pose to the desired pose (position and orientation). The robot executes the arm motion based on the computed trajectory and hits the target stone with its finger. This paper focuses on motion planning of a robot arm, which is the most critical step in the system.

A humanoid robot NAO from Aldebaran is used in this paper. A NAO robot is $58\,cm$ tall and weighs $4.8\,kg$ [4]. It has 25 degrees of freedom: two in the head, six in each arm, five in each leg, and one in the pelvis. Each joint can be controlled using a joint control API which is provided by Aldebaran. The API has a function that interpolates one or multiple joints to a target angle along a timed trajectory [8]. NAO is also equipped with various sensors: two VGA cameras on the head, ultrasonic range sensors on the chest, force sensitive resistors (FSR) on each foot, and an inertial measurement unit (IMU) in the torso.

## 3. MOTION PLANNING OF A NAO ARM

### 3.1 Rapidly-Exploring Random Tree

The rapidly-exploring random tree (RRT) [5] is a commonly used motion planning algorithm, which explores a high dimensional configuration space using random sampling and finds the shortest path to the goal region which is collision-free. RRT is a probabilistically complete method, i.e., the probability of finding a solution increases to one as the number of samples increases. Moreover, the implementation of RRT is simple compared to other methods.

For planning the motion of an arm, let $\mathcal{X} \subset \mathbb{R}^5$ be the configuration space of a robot arm, where $\mathcal{X} = \mathcal{X}_{free} \cup \mathcal{X}_{obs}$. $\mathcal{X}_{obs}$ represents the obstacle space which includes obstacles such as the robot torso, the Go board, and an infeasible region. $\mathcal{X}_{free} = \mathcal{X} \backslash \mathcal{X}_{obs}$ is a free space. Each configuration $x \in \mathcal{X}$ consists of five joint angles of the right arm. Each joint angle has a limit as shown in Figure 1(b) [8].

Once a starting state and a goal region are defined in the configuration space, the RRT algorithm iteratively samples a random state $x_{rand}$ from $\mathcal{X}_{free}$ and expands the search tree $\mathcal{T}$, which is initialized with the starting state. $x_{rand}$ is sampled from a uniform distribution over $\mathcal{X}_{free}$ but we can make the distribution biased towards the goal region in order to find a solution faster. The tree $\mathcal{T}$ is extended by connecting $x_{near}$, the nearest node of the tree to $x_{rand}$, and a new state $x_{new} \in \mathcal{X}_{free}$ in the direction of $x_{rand}$. $x_{new}$ is computed by applying an input for a predefined unit of time in the direction of $x_{rand}$. Once the tree $\mathcal{T}$ reaches the goal region, the trajectory for an arm can be extracted by searching the tree recursively from the goal region to the starting state.

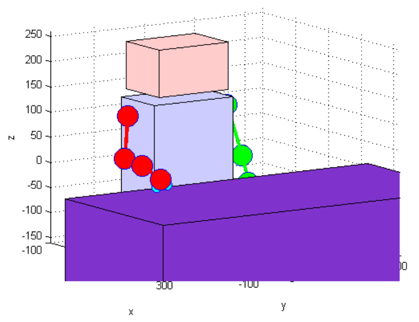


Fig. 4 A snapshot of testing a trajectory using a forward kinematics based simulator.

### 3.2 Implementation Details

In order to find the required joint angles to place a robot hand at the desired location, one has to solve the inverse kinematics problem. Kofinas et al. [1] presented an exact, analytical solution to both forward and inverse kinematics of NAO. However, the inverse kinematics problem for hands is not fully described in their work since the angles of wrist joints of NAO are not considered. Aldebaran Robotics provides a motion API which contains an inverse kinematics solver [8], but it is not possible to find a collision-free trajectory using this API. To avoid complexities and ambiguities given by solving inverse kinematics, we use the results obtained using forward kinematics. We have implemented a simulator solving forward kinematics for NAO that computes the position and orientation of a hand when the joint angles of the arm are given as an input. All the transformations between two frames adjacent to a joint are described with the Denavit-Hartenberg (DH) parameters [9]. Figure 4 shows our forward kinematics simulator for NAO.

Using forward kinematics, we compare current 6D position of a hand with the goal 6D position of a hand whenever we extend the search tree $\mathcal{T}$. We continue extending $\mathcal{T}$ until the distance between two positions is within a predefined distance threshold.

## 4. EXPERIMENTAL RESULTS

We have tested the proposed Alkagi system on a NAO robot. The region of a Go board, in which a NAO robot can shoot a stone in the forward direction, is a $60\,mm \times 30\,mm$ rectangular area. This region is determined based on the movement of the right arm of NAO

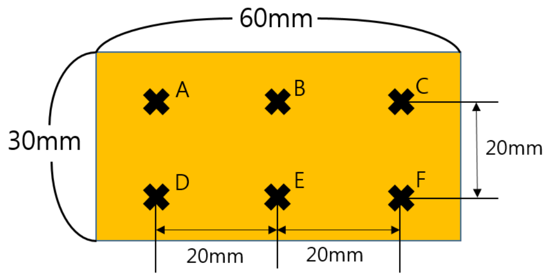


Fig. 5 Tested locations on a Go board.

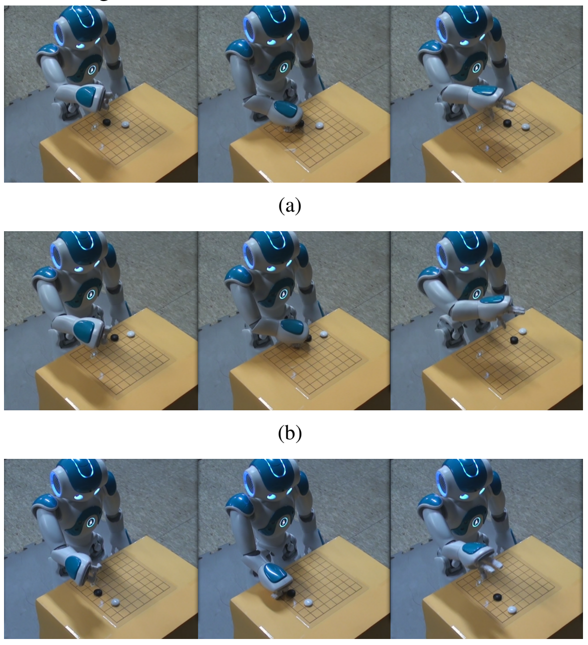


Fig. 6 Photos of a NAO robot playing Alkagi.

| Go Location | $e_{pos}$ | $e_{ori}$ | $\sigma_{pos}$ | $\sigma_{ori}$ |
|---|---|---|---|---|
| A | 0.24 | 3.87 | 0.07 | 0.24 |
| B | 0.62 | 2.10 | 0.15 | 0.54 |
| C | 0.30 | 8.52 | 0.11 | 0.41 |
| D | 0.14 | 9.24 | 0.05 | 0.22 |
| E | 0.17 | 4.13 | 0.05 | 0.14 |
| F | 0.46 | 2.53 | 0.14 | 0.53 |

Table 1 Results from the arm motion planning experiment. The NAO arm is moved toward different Go locations shown in Figure 5. For each location, the experiment is repeated ten times. $e_{pos}$ and $e_{ori}$ are the average position (in $mm$) and orientation (in degrees) errors, respectively. $\sigma_{pos}$ and $\sigma_{ori}$ are the standard deviations of position and orientation errors, respectively.

when the torso is fixed. We have tested the performance of NAO for hitting a Go stone. Figure 5 shows locations on the board we have tested the system. For each location, motion planning of the right arm and the actual movement of the NAO arm were performed ten times. Then we compared the actual position and orientation of a hand with the desired goal position and orientation of a hand.

Table 1 shows the result from the experiment. We can see that the position error is very small (less than 1 $mm$). But, the orientation error shows higher values for positions, such as C and D. Despite the relatively high orientation errors, however, NAO was able to shoot a Go stone and hit the opponent's stone successfully as shown in Figure 6.

## 5. CONCLUSION

In this paper, we have demonstrated that NAO, a humanoid robot, can play the game of Alkagi. The system implements motion planning of a robot arm using the RRT path planning algorithm. In experiments, we have shown that a NAO robot can reliably reach the Go stone at the desired position and orientation and shoot the Go stone. In our future work, we plan to use the whole body of a robot to broaden the area in which a robot hand can reach. It requires an efficient path planning method considering the fact that a humanoid robot has over 25 degrees of freedom.

## 6. ACKNOWLEDGEMENT

This work was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2013R1A1A2009348).

## REFERENCES

[1] N. Kofinas, E. Orfanoudakis, and M. Lagoudakis, "Complete analytical inverse kinematics for NAO," in *Proc. of the IEEE 13th International Conference on Autonomous Robot Systems (Robotica)*, 2013.

[2] L. Acosta, J. Rodrigo, J. A. Mendez, G. Marichal, and M. Sigut, "Ping-pong player prototype," *IEEE Robotics & Automation Magazine*, vol. 10, no. 4, pp. 44–52, 2003.

[3] N. Dantam, P.Kolhe, and M.Stilman, "The motion grammar for physical human-robot games," in *Proc. of the IEEE International Conference on Robotics and Automation*, May 2011.

[4] "Aldebaran Robotics." [Online]. Available: http://www.aldebaran.com/

[5] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *International Journal of Robotics Research*, vol. 20, no. 3, pp. 378–400, May 2001.

[6] J. Kuffner and S. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Proc. of the IEEE International Conference on Robotics and Automation*, April 2000.

[7] J. Suh and S. Oh, "A cost-aware path planning algorithm for mobile robots," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robotics and Systems*, Oct. 2012.

[8] "Nao Software Documentation 1.14." [Online]. Available: https://community.aldebaran-robotics.com/resources/documentation/

[9] J. Craig, Ed., *Introduction to Robotics: Mechanics and Control*. Pearson Prentice Hall, 2005.