Development of a Cheap Chess Robot: Planning and Perception

Billy Okal and Oliver Dunkley

Jacobs University, Automation Group
Campus Ring 1, 28759 Bremen
{b.okal, o.dunkley}@jacobs-university.de

Type of Work: Bachelor Thesis
Supervisor: Prof. Dr. Andreas Nüchter

1 Introduction

Interest in chess playing robots started as early as 1770 when Wolfgang von Kempelen unveiled what appeared to be an automatic chess player, that astonished and fascinated crowds, even after its fake nature was discovered in the 1820s [Sch99]. Chess playing robots caught more attention when the Deep Blue computer beat Gary Kasparov in a chess game [NN97], making a major breakthrough in artificial intelligence. Despite the fact that development of robots capable of playing chess has been historically known to be a complex task, major attempts have been made over the years. Classroom efforts such as Groen's lab course in sensor integration [GdBvIS92] are an early example. More recently, the Association for Advancement of Artificial Intelligence (AAAI) organized a small scale manipulation challenge with playing of chess as the competition task[1]. Today, many chess playing robots exist in various forms ranging from humanoids to industrial grade manipulators. However, the majority of these rely on either modified chess boards in terms using present colors and sizes to simplify the scene interpretation problem and/or modified chess pieces to aid the robot sensors, or use fixed vision sensors e.g. overhead mounted cameras with a fixed boards all of which serve to simplify the problem at a cost of flexibility, generality and reproducibility of designed methodology. Furthermore, most of the existing chess playing robots do not use established motion planning algorithms mainly because only a few degrees of freedom are involved and hence solving Inverse Kinematics equations to locate end effector poses is not considerably difficult although it could get easily expensive. We develop a system based solely on a single off the shelf webcam as a camera sensor, a cheap 4-DOF robotic arm assembled from standard robotics kits as an actuator and standard chess components. The robot arm was powered by Dynamixel AX-12 servos. We also develop a software solution that is both modular and generic allowing for

[1] http://aaai-robotics.ning.com/forum/topics/icra2010-and-aaai2010

easy interchange of major components such as the perception system, chess engine and motion planner.

2 Methods

To have a robot play a game of chess there are two main problems that need to be solved namely, perceiving the environment in which the chessboard and chess pieces exist and interpreting the state of the game from such perception sensor data and computing a feasible collision free path for moving a given chess piece on the chessboard. We do not delve into strategies and game logic for deciding which piece to move and use a standard chess engine to generate moves based on the given state of the game. For building a modular system, we use of the Robot Operating System (ROS) [QCG+09] for coordination and communication. We developed a system that is depicted in Figure 1. We solve the perception and planning problems using the procedures described in the sections that follow.

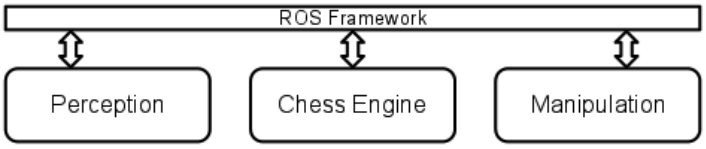


Figure 1: System Overview

2.1 Perception

We use of various first principles like approaches to solve the perception problem. In particular, we search for the chessboard in the scene at every game instance. We take an image of the scene at every game step and compare successive images to determine what has changed i.e. which moves have been played. We use image processing techniques such as line detection, Canny edge detection [SHB07] and image differences to determine the location of the four corners of the chessboard as well as the location of changes on the board and transform the resulting chessboard into a known frame by perspective projection. Once we get the positions of the changes on the chessboard using its size information and scaling, we determine which move to make by querying a standard chess engine. For implementation of the techniques, we use of the OpenCV[2] library.

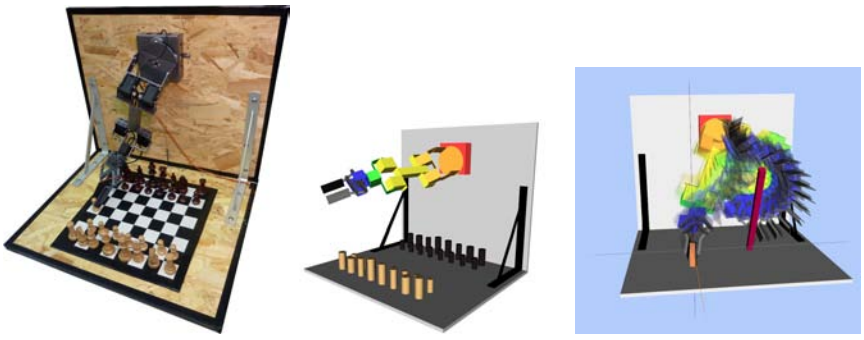
[2] `http://opencv.willowgarage.com`

2.2 Manipulation

With the new move determined, we again use chessboard size information to determine the precise locations to move a given piece, which provides the goal and start pose for the planning process. We use a seasoned motion planning algorithm called BiRRT (Bi-directional Rapidly Exploring Random Trees) [Dia10] a variant of RRT which provides a probabilistically complete algorithm [LaV98]. We compare this method to that of computing the Inverse Kinematics of the end effector (gripper) each time with the belief that the BiRRT motion planning algorithm would take advantage of the topological compactness of the search space as most of the goals are very close in this scenario. Either of the procedures gives us a trajectory to follow. We then execute the computed trajectories on the robot taking into account further joint considerations and reachability of the workspace. We added some constraints into the planning system so that the chess pieces are moved in an 'upright' manner throughout the manipulation process and also to filter the found solutions. For implementation, we make use of OpenRAVE [Dia10] motion planning suite which comes with a set of motion planning algorithms including BiRRT and open interfaces for using the planners.

3 Results

Experiments with the perception procedures showed reliable chess move detection with real-time performance comparable to previous cases with modified chessboards and/or modified chess pieces or fixed cameras. The trajectories generated using seasoned motion planning algorithms were also of better quality compared to those generated by simple Inverse Kinematics. We also noticed that the BiRRT algorithm took considerably shorter time for planning compared to solving Inverse Kinematics for locating the end effector pose. Figure 2 shows the setup of the experiments and a sample trajectory.



(a) Real Setup (b) Simulation Setup (c) Sample Trajectory

Figure 2: Real and simulated setup and a sample trajectory

Further media including video material depicting the results can be found on our groups

video archive at `http://www.youtube.com/user/AutomationAtJacobs` and further details and discussion of the work is provided in a corresponding thesis document.

4 Conclusion

We demonstrated that the constraint of using a modified chessboard and/or pieces can be relaxed without greatly compromising performance. We have also demonstrated that the constraint of having a fixed camera directly above the chessboard can be done away with while still enjoying reasonable real-time performance. Use of seasoned motion planning algorithms with various optimizations and constraints was also found to produce smoother paths for manipulation chess pieces. We showed all of these using very simple hardware and a limited amount of time. We also noticed that our imprecise hardware and limited DOF posed great difficulties even for seasoned planning algorithms making realization of a full successful game not possible due to overheating of motors. The limitation in DOF can however be easily averted at cheap cost by adding just two more DOF to the robot arm.

References

[Dia10] R. Diankov. *Automated Construction of Robotic Manipulation Programs*. PhD thesis, Carnegie Mellon University, Robotics Institute, August 2010.

[GdBvIS92] F. C. A. Groen, G. A. der Boer, A. van Inge, and R. Stam. A chess playing robot: lab course in robot sensor integration. In *Instrumentation and Measurement Technology Conference, 1992. IMTC '92., 9th IEEE*, pages 261 –264, May 1992.

[LaV98] S.M. LaValle. Rapidly-exploring random trees: A new tool for path planning. Technical Report TR-98-11, Iowa State University, Computer Science Department, 1998.

[NN97] M. Newborn and M. Newborn. *Kasparov Vs. Deep Blue: Computer Chess Comes of Age*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1997.

[QCG+09] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and Y. N. Andrew. ROS: an open-source Robot Operating System. In *ICRA Workshop on Open Source Software*, 2009.

[Sch99] S. Schaffer. Enlightened Automata. In *The Sciences in Enlightened Europe. (Eds. William Clark, Jan Golinski, and Simon Schaffer)*, pages 126– 165, 1999.

[SHB07] M. Sonka, V. Hlavac, and R. Boyle. *Image Processing, Analysis, and Machine Vision*. Thomson-Engineering, 2007.