Two Robotic Arms Playing Chess

Yaosheng (Jonathan) Xu
Harvard University
Cambridge, MA
yaoshengxu@g.harvard.edu

***Abstract*—In this paper, I implement a full-stack system that control two robotic arms playing chess against each other. Chess playing is a special case of bin picking task that requires (1) reasonable trajectory design that doesn't knock down chess pieces, (2) valid grasp pose that is able to stably pick or place chess pieces from any distance on the chess board. The robotic system contains two 9-degree-of-freedom robotic arms, and self-built chess board as well as chess pieces. Implementation details, including trajectory designs, dynamic controller, grasp selection, and simulators, are discussed further in the paper. In results, this robotic system is able to simulate chess playing given the chess moves; I present a simple video that demonstrate the last two moves of a famous end game (Anatoly Karpov v.s. Garry Kasparov, Mar-21-2017) to show the availability.** [1]

I. Introduction

Robots nowadays are capable of doing tasks in a wide range of environments [1], [2]. Among all kinds of tasks, bin picking is considered to be one of the most basic tasks that robotic arms can do. Nevertheless, a large number of papers have been published to research on the bin picking task, including customizable grasping conditions, irregular-shaped objects, multi-gripper switching, etc. [3]–[5]. Moreover, bin picking task can be used as a basic platform, and be derived to other interesting tasks, such as chess playing.

In my set up, I consider the standard rule of chess, which means the chess board is a 8x8 grid board, and there are totally 32 chess pieces. There is a robotic arm on each side, controlling the chess piece movement given the command. Two robotic arms are Kuka LBR iiwa welded with Schunk WSG 50, which gives 9-degree-of-freedom (DoF) for both robotic arms. To make the system work, I have to consider two major problems: (1) how to design a smooth trajectory that doesn't make collisions to other chess pieces, as well as how to gently place the chess piece at the center of the grid. (2) how to select the grasp pose that stably pick or place chess pieces, either far or close to the arm. In section III, I discuss the method I used in details, including the simulators, trajectory design, dynamic controller, and grasp selection. Then the evaluation of the whole system as well as related discussions are presented in section IV.

In a word, the goal of this project is to show that robots can do some interesting tasks by only using some simple techniques. Chess playing is a wonderful example that combine some simple techniques together as a full-stack system, and has some potential challenges which are different from vanilla bin picking task.

[1]https://youtu.be/4low-9Fh71c

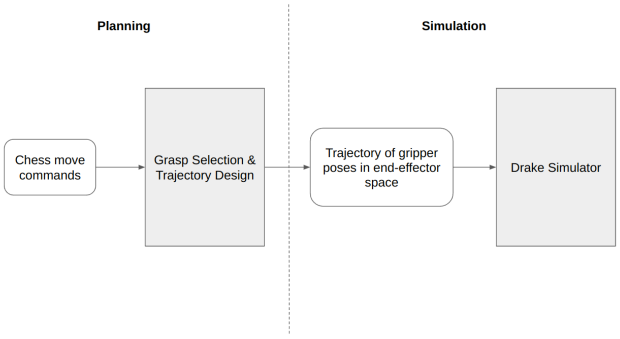


Fig. 1: Overview of the whole system. Two major parts are planning and simulation

II. Related Works

A wide range of manipulator controllers have been proposed to control the robot given higher-level commands [6]–[8]. Many works have discussed trade-offs among all kinds of manipulator controllers [9]. For example, Inverse kinematics (IK) controller solves a more global problem, but is not guaranteed to succeed, etc. [10]. In my chess playing settings, I choose to use differential inverse kinematics controller, because I am able to design smooth commands in end-effector space, and thus differential IK can simply tracking them in joint space. Differential IK will be further discussed in section III-C.

Some previous works have been done on robotic chess playing. Matuszek et al. presented a robust chess-playing robotic system on Gambit, a 6-DoF robot manipulator system [11]. Angelkov et al. describe a system for automated chess playing with a robotic manipulator, which has 6-DoF arm with a parallel jaw gripper and six servos for movement of the joints [12]. All these papers, however, didn't post their implementation details. In this work, I choose a different robot manipulator, which has 9-DoF, to perform the chess playing task. My implementation is mainly in PyDrake [13], and I refer to the github for source code and implementation details.[2]

III. Method

A. System overview

The system consist of two major parts: planning and simulation. For planning, it takes chess move commands as input, and design corresponding gripper pose trajectory; Then, the Drake

[2]https://github.com/APM150/Two_Robotic_Arms_Playing_Chess

(a) (b)

Fig. 2: (a) Kuka LBR iiwa. (b) Schunk WSG 50

Fig. 3: Standard chess board

simulator will simulate the chess game given the trajectory. Figure 1 gives an overview of the described system.

B. Simulator Development

I will start introducing from the last block of Figure 1, which contains two parts: differential inverse kinematics controller and multibody plant. I will present multibody plant, which is simulator settings in this subsection; this includes robotic arms and chess models set up. For robotic arms, I choose Kuka LBR iiwa as the arm, which has 7-DoF; and Schunk WSG 50 as the gripper, which has 2-DoF. Figure 2 shows the arm and the gripper. Welding them together gives me 9-DoF, and the system contains two robotic arms, which is totally 18-DoF.

Following standard chess rules, I created the chess board as a 8x8 grid board, and 32 chess pieces, including 4 bishops, 4 knights, 4 rooks, 2 queens, 2 kings, and 16 pawns. The chess board grid is numbered in standard way: from left to right a to h, bottom to top 1 to 8 (see figure 3). This implies that the coordinate of each grid center in world frame is known, because the chess board is welded on the ground. To create these models that have valid collision geometry, I used convex decomposition from 3D mesh to Signed Distance Function

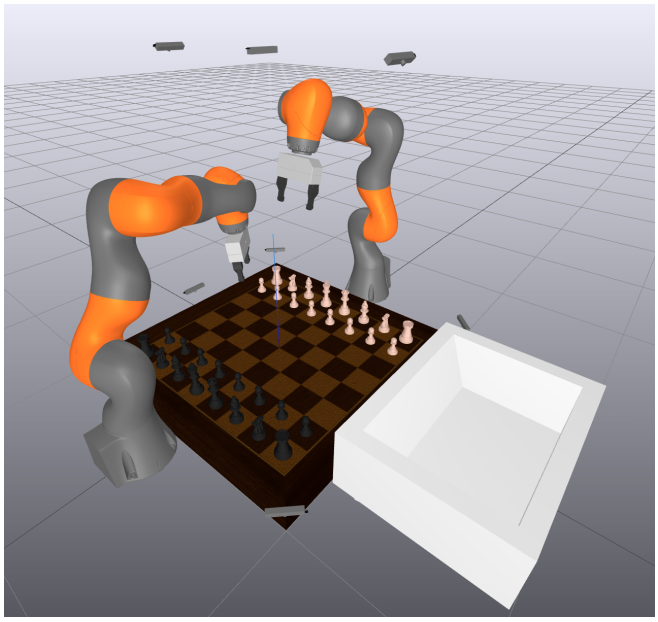


Fig. 4: Simulator set up overview.

(SDF) [3]. After loading these models, I added a bin for removed chess pieces, and 8 cameras surrounding the whole chess board for perception. Figure 4 captures the initialization of a full game.

C. Differential Inverse Kinematics Controller

Differential inverse kinematics controller is extremely powerful if a smooth trajectory in end-effector space can be designed. Previous works have shown its effectiveness with simple theory [14].

$$v = [J^G(q)]^+ V^{G_d} \tag{1}$$

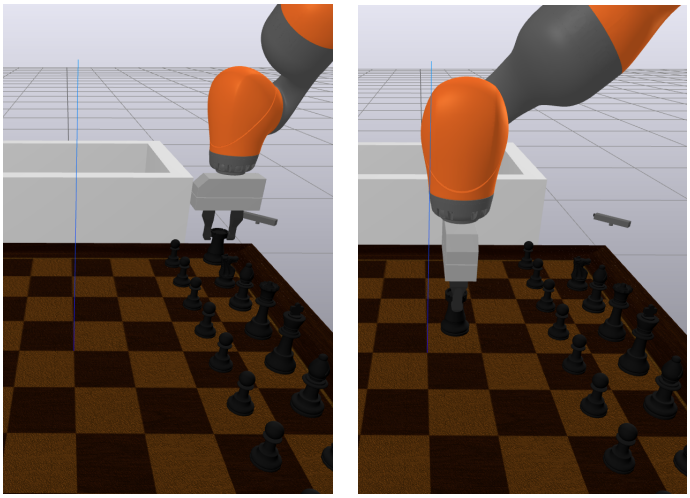
where v is the joint velocity, J^G is Jacobians in gripper frame, $+$ is pseudo-inverse, and V^{G_d} is desired gripper spatial velocity. By solving the pseudo-inverse, joint velocity can be outputted given spacial velocity, which is differentiated from pose trajectory. As a result, controlling the robot can be done by simply input the gripper pose trajectory.

D. Trajectory Design

Figuring out the input of the differential inverse kinematics controller lead to the first part in figure 1, which is planning. In a word, a desired trajectory is the one such that it successfully move a chess piece to the center of the desired grid, and doesn't knock down any other chess pieces while moving.

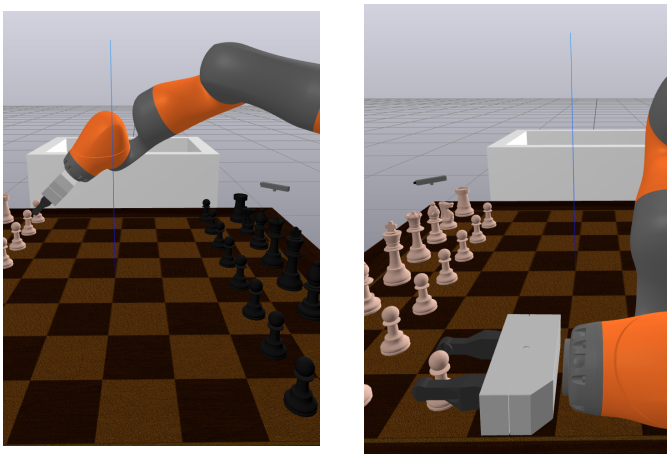
Let's first consider the case that the moving chess piece is located close to the robotic arm, and the goal grid is also close to the robotic arm. In this case, the gripper can vertically pick the chess piece, move it high enough, and again vertically place the chess piece in order to avoid any collisions while picking/placing/moving as possible. For example, if I want to move the black rook on a8 to d5 using the robotic arm on the

[3]https://github.com/gizatt/convex_decomp_to_sdf



(a) (b)

Fig. 5: (a) Picking the black rook on a8 using the robotic arm on the black side. (b) Placing the black rook on d5.



(a) (b)

Fig. 6: (a) Picking the white pawn on a2 using the robotic arm on the black side. (b) Placing the white pawn on h3.

black side, I can simply apply this scheme. Figure 5 is the visualization for this.

Not all cases, however, can be handled by this scheme. Consider the situation that either the moving chess piece is located far from the robotic arm, or the goal grid is far from the robotic arm (maybe even both), there is no way that the gripper can vertically pick or place. In this case, the gripper has to be horizontal or at least form an angle that is not vertical to the ground in order to reach the goal. For example, moving the white pawn from a2 to h3 using the robotic arm on the black side is considered "far pick" and "far place". I have to adjust the gripper to reach the chess piece and the goal. Figure 6 visualize this example in simulator.

From my experience, if the chess piece is 6 grids further or equal to the robotic arm, the gripper has to be adjusted from vertical to reach the chess piece. There is a trade-off between how horizontal the gripper should be and the possibility to knock down surrounding chess pieces. This could be determined by a heuristic that check how many other chess pieces are surrounded. I didn't implement this heuristic because of the time limit. But it's worth trying in future works. In both above cases (far and close), the chess piece is gently placed at the center of the goal grid. This can be done by assuming the flatness of the chess board, which means I can assume the z axis at the picking position is the same as the z axis at the placing position. In this way, the chess piece is gently placed instead of falling down, and this ensures that the chess piece would be at the center of the goal grid.

Last but not least, if one of the chess pieces is removed in the current step, the robotic arm should first pick the removed chess piece, and place it into the bin. When placing the chess piece into the bin, it follows the scheme of "far placing". What's more, there is no need to gently place the chess piece; this means the gripper only need to be moved above the bin, and let the chess piece fall down into the bin.

E. Grasp Selection

The final part is how to select a grasp that can stably pick a chess piece. It is necessary to point out that hard coding the position of the chess piece also works, because in the simulator setting I assume that the robot knows the coordinate of all the grids in world frame. Thus, I can code the coordinate of the grid center as x, y axis, and approximate the chess piece height as z axis; finally, combining the position with the rotation scheme mentioned in previous part gives a valid grasp pose.

Nevertheless, I can make the grasp more robust by applying antipodal grasps, so I don't need to approximate the chess piece height. Antipodal grasp is a heuristic for a two fingered gripper to find a grasp that has a large contact wrench cone; antipodal here means that the normal vectors of the contact (the z axis of the contact frames) are pointing in exactly opposite directions [10]. To sample valid antipodal grasps, I can design a cost function. Following the manipulation textbook chapter 5, I first form a term that reward all the points in the point cloud that aligned their normal well to the x-axis of the gripper:

$$cost = -\sum_i (n^i_{G_x})^2 \tag{2}$$

where $n^i_{G_x}$ is the x component of the ith point in the cropped point cloud expressed in the gripper frame. Besides, to make two fingers vertical to the chess board, I can penalize deviation of the gripper from vertical by adding an extra term:

$$cost+ = -\alpha[0 \quad 0 \quad -1]R^G \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \alpha R^G_{3,2} \tag{3}$$

where α is relative cost weight, and $R^G_{3,2}$ is the scalar element of the rotation matrix in the third row and second column. Finally, I just need to input the cropped point cloud of the

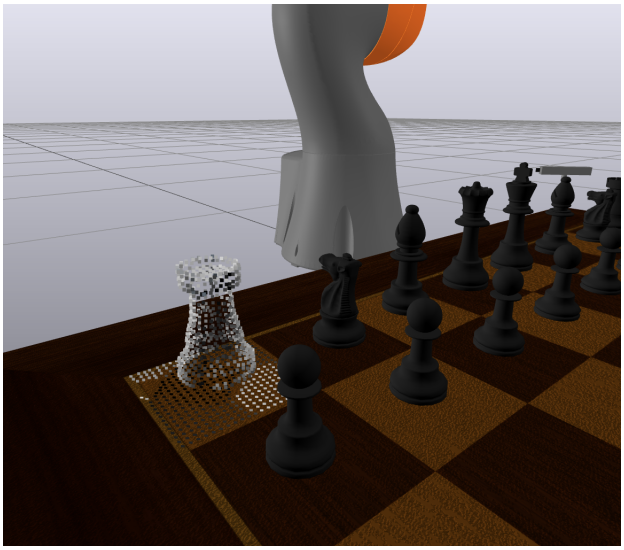


Fig. 7: Point cloud for the black rook at a8.

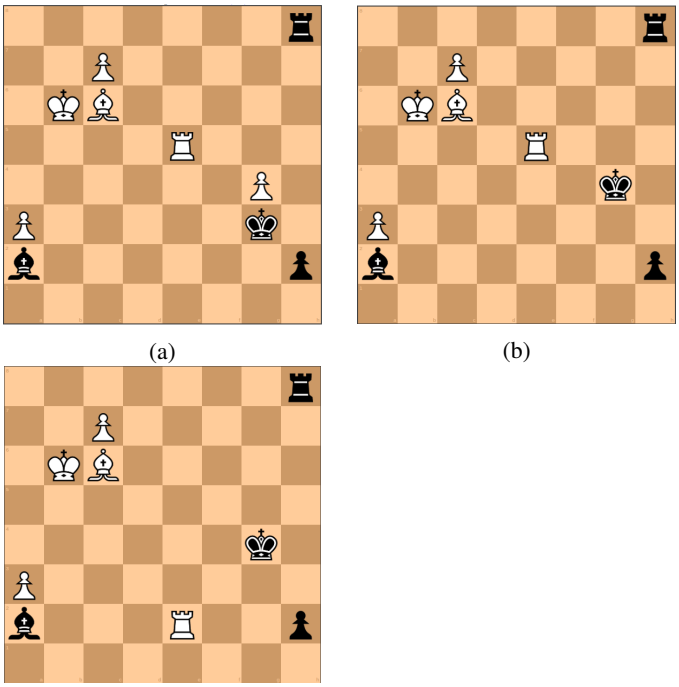
desired grid created by 8 depth cameras around, to calculate antipodal grasp candidates. An example visualization of the point cloud is shown in figure 7.

A drawback of the this antipodal grasp method is that there is no guarantee the selected grasp is exactly vertical to the chess board because it's sampling based. Even I can choose the grasp candidate to be only vertical to the chess board, I lose the ability to handle far picking in this case. So there is no theoretical guarantee that antipodal grasp would generate a perfect grasp that stably pick the chess piece.

IV. Evaluation and Discussion

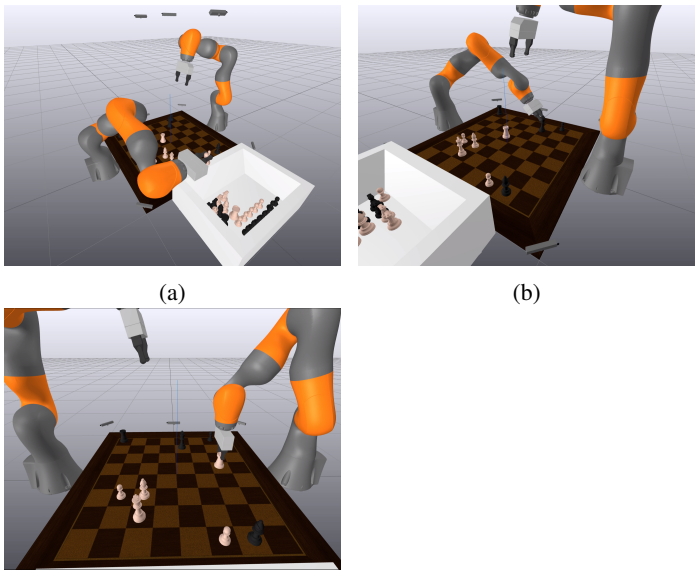
Put all things together, I can simulate any chess game given the commands. To show the significance of the work, but also due to the capacity I can show on the paper, I choose to simulate the last two steps of a famous end game (Anatoly Karpov v.s. Garry Kasparov [Mar-21-2017], see figure 8). I choose this example because (1) it contains a step that remove a chess piece into the bin. (2) it contains pick and place for both close distance and far distance, which is discussed in section III-D. The corresponding simulation is shown in figure 9. Two robotic arms successfully finish the last two step of this end game using the methods in section III.

For future works, one way to make the problem much harder is eliminating the assumption that the robotic arms know the coordinate of the chess board grid. In this case, the robot arm has to use perception to recognize the chess piece. Some previous papers has worked on this problem, and used a classifier hierarchy to recognize chess pieces [11]. This would include detecting the location, recognizing the color, as well as the type of the chess piece. Besides this, I can potentially add an Alpha-Beta Tree Search algorithm that can bring chess AI to real life (simulator). Playing chess using Alpha-Beta Tree Search has been solved years ago [16]. The potentiality of this project is big, and one can add all sorts of functionality above it.



(a) (b) (c)

Fig. 8: (a) Anatoly Karpov v.s. Garry Kasparov [Mar-21-2017] last 2 steps. (b) Step 1: black player remove white pawn on g4, using black king on g3. (c) Step 2: white player move white rook on e5 to e2.



(a) (b) (c)

Fig. 9: (a) The robotic arm on black side move the white pawn on g4 into the bin. (b) The robotic arm on black side move the black king from g3 to g4. (c) The robotic arm on white side move the white rook from e5 to e2.

V. Conclusion

In this project, I implement a system that has two robotic arms playing chess. This is a full-stack system which utilize many techniques I learned in the robotic manipulation course 6.4212 at MIT. Before taking this course, I have zero experience in robotic manipulation. This project shows that the most basic concepts in robotic manipulation, such as differential inverse kinematics controller, antipodal grasp, etc., can already do a wide range of interesting tasks. Still, the project has some limitation regarding the trajectory design and grasp selection: the gripper is not robust enough to determine the most suitable gripper pose for picking or placing. Even with antipodal grasp, there is still no guarantee that it is a perfect grasp that stably pick/place with zero probability knocking down other chess pieces. The future works of this project is discussed in the previous section.

In summary, the goal of this project is to utilize the concepts I learned in the course to implement a full-stack system. At the end, I successfully show the demonstration of how to control two robotic arms (18-DoF in total) to play chess.

References

[1] Amazon, "Meet Astro, a home robot unlike any other," September 2021

[2] R. Sparrow, "Kicking a robot dog," 2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI), March 2016

[3] H. Tachikake, W. Watanabe, "A Learning-based Robotic Bin-picking with Flexibly Customizable Grasping Conditions," 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), October 2020

[4] J. Guo, L. Fu, M. Jia, K. Wang, S. Liu, "Fast and Robust Bin-picking System for Densely Piled Industrial Objects," 2020, https://arxiv.org/pdf/2012.00316.pdf

[5] M. Fujita, Y. Domae, R. Kawanishi, etc., "Bin-picking Robot using a Multi-gripper Switching Strategy based on Object Sparseness," 2019 IEEE 15th International Conference on Automation Science and Enginering (CASE), August 2019

[6] K. Salisbury, "Active stiffness control of a manipulator in Cartesian coordinates," Proc. of the 19th IEEE Conference on Decision and Control, 1980.

[7] N. Hogan, "Impedance Control: An Approach to Manipulation. Part I - Theory," Journal of Dynamic Systems, Measurement and Control, vol. 107, pp. 1–7, Mar 1985

[8] O. Khatib, "A Unified Approach for Motion and Force Control of Robot Manipulators: The Operational Space Formulation," IEEE Journal of Robotics and Automation, vol. 3, no. 1, pp. 43-53, February 1987

[9] M. Lahijanian, M. Svorenova, A. A. Morye, B. Yeomans, D. Rao, I. Posner, P. Newman, H. Kress-Gazit, M. Kwiatkowska, "Resource-Performance Trade-off Analysis for Mobile Robot Design," IEEE Robotics and Automation Letters (RA-L 2018), 2018

[10] R. Tedrake, "Robotic Manipulation - Chapter 8," 2020-2022

[11] M. Cynthia, M. Brian, A. Roberto, D. Marc Peter, B. Liefeng, C. Robert, K. Mike, L. Louis, S. Joshua, F. Dieter, "Gambit: A Robust Chess-Playing Robotic System," https://rse-lab.cs.washington.edu/postscripts/chess-icra-11.pdf

[12] A. Dimitrija, K. Natasa, K. Saso, "Automated Chess Playing with a Robot Manipulator," International Journal of Engineering Issues, vol. 2015, pp. 45–51

[13] R. Tedrake, "Drake: Model-based design in the age of robotics and machine learning," May, 2021

[14] D. A. Drexler, I. Harmati, "Joint Constrined Differential Inverse Kinematics Algorithm for Serial Manipulators," periodica polytechnica, January 2013

[15] R. Tedrake, "Robotic Manipulation - Chapter 5," 2020-2022

[16] W. Buana, P. dan, L. Heryawan, "Applying Alpha-Beta Algorithm in A Chess Engine," Jurnal TeknoSains, vol. 6, pp. 1–58, 2016