Gridmaster AI: An Autonomous Connect 4 Robot

Anirudh Valiveru
6.4212: Robotic Manipulation
Massachusetts Institute of Technology
Cambridge, MA
anirudhv@mit.edu

***Abstract*—In this paper, I describe my work on Gridmaster, a full-stack robotic system meant to serve as an autonomous Connect 4 playing robot. Included are important details about every step of the project's stack, ranging from simulation, perception, and control using differential inverse kinematics. While most Connect 4 robotics projects online have custom-built a specialized rig that sits on top of the Connect 4 board, Gridmaster works out-of-the box on the generic 7-DoF Kuka Iiwa robot and a perception system using two RGB-D Cameras. As a result, Gridmaster is able to autonomously control and play a game of Connect 4, simulating a real-life game-playing robot. Limitations of my approach and potential future extensions are also discussed.**

I. Introduction

Designing a robotic system to play a recreational board game like Connect 4 can often be an insightful task when testing the capabilities of a manipulation system. Connect 4, in particular, involves the perception and manipulation of small and hard-to-manipulate coins that must be picked up and placed in the correct column to continue playing the game, providing an interesting challenge on the simulation, perception, and control fronts. Gridmaster solves the problem of manipulating chips by using a vaccum-gripper setup, simulated in Drake through weld constraints. Two RGB-D cameras (one on each side) are used to perceive the real-world poses of each coin that hasn't been placed in the grid. This paper also discusses necessary future extensions on the robot's game engine that will allow it to autonomously win against any human opponent.

II. Related Work

While Connect 4 is a board game generally only played casually, the skills required for a robot to play Connect 4 are quite transferable to more important daily tasks. These directions include cooking, washing dishes, doing laundry, and other household tasks that involve generalized perception and precise manipulation to accomplish a task. More specifically, however, several attempts to build a Connect4 robot have been made in the past, using the game as a canvas to research board-game AI strategy [1] and human-robot interaction [2]. Gridmaster is unique since it utilizes hardware designed for more general manipulation tasks, as opposed to a rig specially designed to play the game. This allows it to be deployed more readily to 7DoF manipulators that work similarly to the Kuka Iiwa arm.

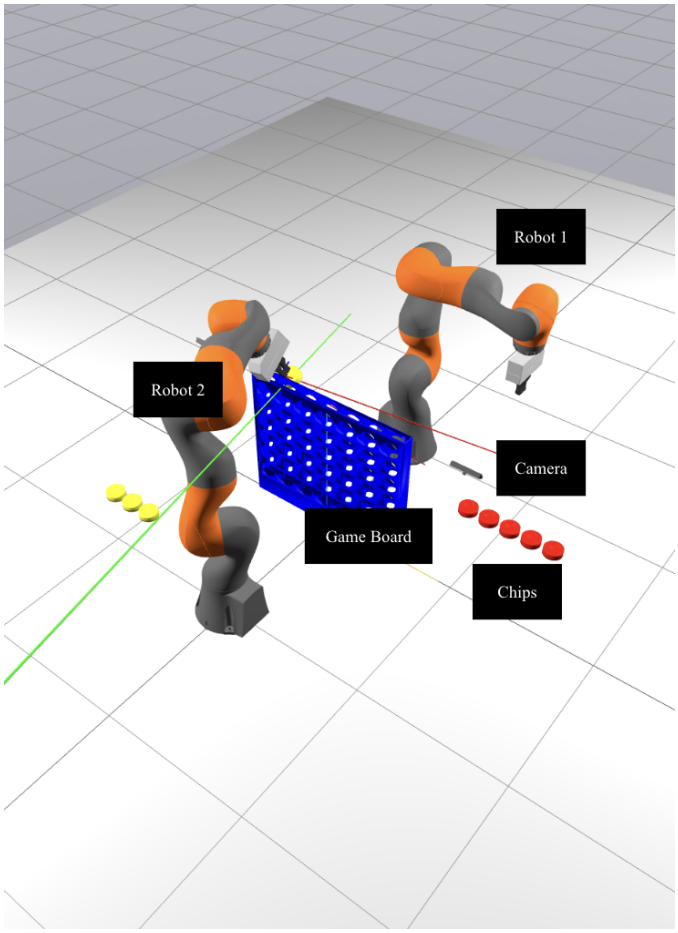

Fig. 1. **Simulation Setup**: This figure shows the scene used to simulate Gridmaster in Drake. The most important parts of the system include the game board, each of the two Kuka Iiwa robots, the two sets of coins (red and yellow), and the two RGB-D cameras used to perceive each coin's pose. The project's source code can be found at the following repository: https://github.com/anirudhv27/Connect4Robot

III. Method

A. Overview

Figure 1 details the pipeline used to enable Gridmaster to play Connect 4 autonomously. As the game is played, Gridmaster keeps track of the game's board state internally, using this information to check for game-ending conditions at every turn.

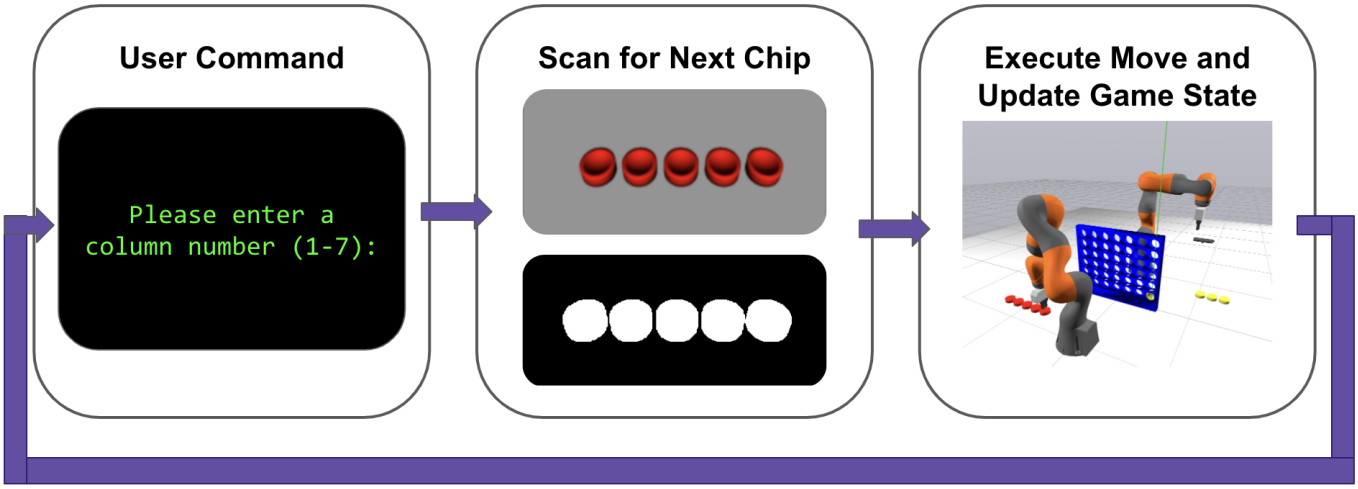

Fig. 2. **System Overview**: Gridmaster involves three major steps to play a turn of Connect 4. First, it takes a user command as input, specifying the column representing a turn. Next, it scans the area for a chip and converts pixel coordinates into real-world coordinates. Finally, it uses differential inverse kinematics to execute the move and update the game's state.

On initialization, Gridmaster sets up the simulation and scans both coin-loading areas to compute the position of each coin in the world frame. To work around simulation efficiency constraints, Gridmaster starts by loading five coins into the simulation to begin with, allowing the simulation to run without excess latency.

Once the simulation is set up, Gridmaster takes console input from the player, determining the column where the robot will play its turn. Each column corresponds with a world-frame pose that is passed into the differential inverse kinematics module [3] for the robot's control. To execute a turn, Gridmaster first moves to the appropriate coin's pose, as perceived at initialization. Then, the robot activates its suction gripper to pick up the coin in question, and moves to the world pose corresponding with the specified column. Finally, the robot deactivates suction to release the coin, resetting the gripper's pose to its starting position afterwards to allow the next player's turn!

B. Simulation

Gravity is an essential component of what makes Connect 4 work as a game. As a result, simulating Connect 4 in particular requires the construction of assets with accurate collision geometries that interact with each other in realistic ways. Gridmaster is simulated using Drake's *HardwareStation* [4], which conveniently integrates the Kuka Iiwa arm [5] to enable perception and control directly from the scenario's YAML specification.

1) Connect 4 Board and coins: The Connect 4 board and coins are both represented in template *SDF* files, which stands for *Scene Description Format*. *SDF* represents both an object's visual and collision properties, both of which are important in the context of Gridmaster.

This work's simulation specifies three *SDF* files: one for the board, one to construct all red chips, and one to construct all yellow chips in the scene. Each *SDF*'s visual properties are represented by an *OBJ* file with accurate visual geometry. The chips' collision geometry are also specified using the same *OBJ* files.

However, the Connect 4 board's collision geometry was constructed separately using *SDF*'s geometry primitives. As a hollow asset with several holes, *Drake's* simulator defaults to utilizing the *OBJ* file's *convex hull* as its collision geometry if defined naively. This is a problem, because Connect 4 relies on the ability to place pieces *inside* each column! This hurdle was avoided by constructing the board's collision geometry face-by-face.

2) Suction Gripper: Connect 4's uniquely small pieces make Gridmaster an ideal candidate to use a *suction gripper* to accomplish its manipulation goals. This is in contrast to the traditional approach of using a two-finger gripper, like the *WSG Schunk*, which is the default end-effector implemented in *HardwareStation*. While two-fingered end-effectors are quite effective in general pick-and-place tasks, the thin margins of a Connect 4 gripper mean that any antipodal grasp computed is bound to be unstable. Connect 4 requires one to place these coins into small columns that are designed to fit them with very little leeway, making precision a critical goal. Therefore, suction grippers are a natural fit.

Unfortunately, *Drake* does not support suction out of the box in its simulation. Instead, all suction interactions are modeled using *AddWeldConstraint*, which allows one to activate and deactivate a constraint that welds two body frames together. All possible gripper-coin interactions are defined as *WeldConstraints* at initialization, and are disabled as a default. Activating a specific *WeldConstraint* serves as a coarse

approximation to grasping a particular coin using the suction gripper, for example.

Visually, the suction gripper is constructing by simply closing the two fingers of the *Schunk*. All weld constraints are defined relative to the Schunk's body.

C. Perception Module

Gridmaster's perception component is driven by two RGB-D sensors attached at fixed poses through the *HardwareStation* interface. These cameras are used to compute the poses of each individual coin for downstream inverse kinematics control. While perception is often solved using deep-learning based methods, the simulation's controlled nature allows us to detect voins through simple color-segmentation. A diagram describing Gridmaster's perception pipeline is included below:

Gridmaster's pipeline starts with raw color and depth images as input. Our simulation setup conveniently allows us to segment out the background using the uniform color of (148, 148, 148), which is the table's exact color in RGB.

After generating a binary segmentation mask for all chips on a given side, Gridmaster uses flood fill [6] to detect number of coins in the image, classifying the coin-number of each segmented pixel by computing all connected components in the mask. From there, the average pixel coordinate of each chip is computed, then unprojected from depth coordinates to the world frame using each camera's intrinsic and extrinsic parameters.

Algorithm 1 Flood Fill Algorithm

```
procedure FLOODFILL(mask, x, y, label)
    stack ← empty stack
    stack.push((x, y))
    while stack is not empty do
        (x, y) ← stack.pop()
        if x < 0 OR x ≥ width(mask) OR y < 0 OR y ≥ height(mask) then
            continue
        end if
        if mask[x][y] is occupied and unseen then
            mask[x][y] ← label
            stack.push((x - 1, y)) {left}
            stack.push((x + 1, y)) {right}
            stack.push((x, y - 1)) {up}
            stack.push((x, y + 1)) {down}
        end if
    end while
end procedure=0
```

Gridmaster's analytical approach to perception works due to Connect 4's inherent symmetries, along with the simulation's simplistic setup. A more complicated setup involving deep learning may be necessary to interpret noisy real-world data.

D. Control Using Differential Inverse Kinematics

Gridmaster uses differential inverse kinematics to control desired poses. This is implemented using the *AddIiwaDifferentialIK* functionality of *HardwareStation*, which allows the robot to use Differential Inverse Kinematics to control the end-effector to a desired end position and orientation. The system's implementation of DiffIK is quite simple, only taking the desired pose as input. Gridmaster also doesn't implement any form of motion planning or trajectory optimization to compute ideal gripper trajectories. This decision was simply a result of time-constraints; a more robust system would add constraints to the poses and velocities to ensure smoother motion that can handle more diverse cases.

Inputs to the *DifferentialInverseKinematics* controller are wired using a custom *LeafSystem* called *PoseSource* that broadcasts a constant *RigidTransform* pose to its output port. Therefore, any control signal is given to the robot by setting the *PoseSource*'s pose output to the desired end-effector pose.

Any single turn always involves the following sequence of steps:

1) Move end-effector above the next chip.
2) Grab the chip using suction.
3) Move end-effector above the specified column.
4) Release the chip into the column.
5) Reset the robot's position.
6) Update the game's state.

Steps 1, 3, and 5 all involve actuating the robot to the relevant pose. To ensure that they reach all of these poses, Gridmaster only moves on to the next step when each joint velocity is reasonably close to zero. These joint velocities are accessed through the "plant" subsystem, which provides direct access to both Iiwa robots in the system.

E. Game Engine: MinMax

Combining all of the above steps allows the robot to play Connect 4 against each other. At this point, however, adding a game engine will allow Gridmaster to play (and win) against any human opponent using close-to-optimal strategy! At the time of writing this paper, Gridmaster simply serves as a framework to play Connect 4 using a robot, and takes human inputs to make gameplay decisions. However, this section discusses potential extensions to the robot's AI that will allow it to play autonomously.

1) MinMax: Any turn of Connect 4 can be seen as a decision, where a series of optimal decisions is one that allows one to win the game. MinMax works under the principle that the optimal move is one that allows you to maximize your score by the end of the game, assuming that your opponent is also optimizing the same objective for themselves.

The Minimax [7] decision-making principle is often used in game-theory to describe the process of making decisions to minimize one's worst-case outcomes (assuming that the opponent is playing optimally). More formally, in a two-player game:

$$v_i = \max_{a_i} \min_{a_j} v_i(a_i, a_j)$$

, where a_i, a_j represents agent $i's$ and agent "j strategies, respectively, and v_i represents i's utility.

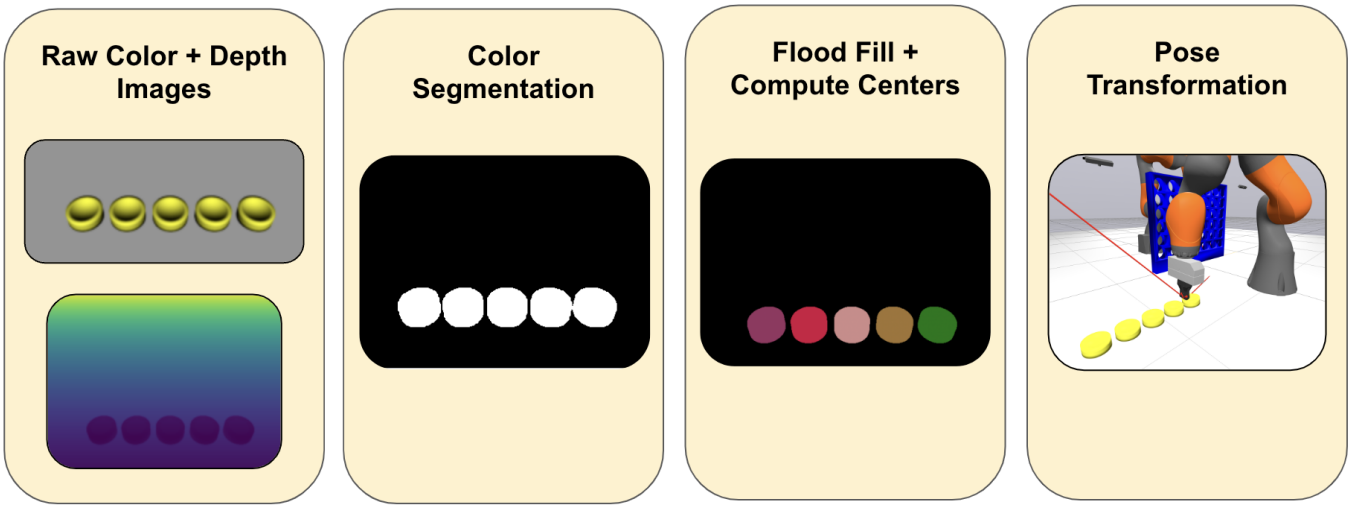

Fig. 3. **Perception Pipeline**: To perceive the initial poses of each individual coin, Gridmaster takes raw RGB-D images as input and runs simple color segmentation to generate a binary mask isolating each coin. Then, a flood-fill algorithm is used to label each pixel as belonging to a particular coin, and the average of each "cluster" is used as the coin's pose in pixel coordinates. Finally, the depth image position is un-projected to world coordinates using the RGB-D camera's intrinsic and extrinsic parameters, allowing the robot to grasp the coin with its end-effector.

Therefore, the naive MinMax algorithm involves scoring a state-space tree of all possible strategies recursively. This is done by trying all possible moves, computing the score of opponent's optimal response to each move by recursively calling MinMax for them, and finally selecting the move that has the least-bad opponent response. While simple, the MinMax tree-search algorithm serves as the basis of many game AIs, including the notorious Stockfish chess AI.

There are two major challenges to implementing tree-search algorithms that can play two-player games like Connect 4 or Chess. The first is tractability. With 21 red and 21 yellow pieces that can be placed anywhere in a 7-wide, 6-tall game board, the space of possible game states expands quite quickly, and requires certain optimizations like *alpha-beta pruning* and *transmutation tables* [8] to avoid recomputing states and heuristically prioritize exploring states that are more likely to win. Designing an optimal game-playing AI is a fascinating problem with several directions to explore and optimize; however, more details about this are out of scope for this project.

To summarize, while the final version of Gridmaster does not implement any sophisticated tree-search algorithm due to time constraints, game engine design is an important next-step that would allow Gridmaster to truly master the game of Connect 4.

IV. Results

In its present state, Gridmaster fully functions as a station that allows two players to play Connect 4 using a 7-DoF robot, such as the Kuka Iiwa. Gridmaster combines simulation techniques to represent the Connect 4 system, perception involving two RGB + depth cameras, and a control system leveraging differential inverse kinematics to actuate to desired poses. One caveat to this implementation is general simulation latency of our approach, which is unavoidable due to the many degrees of freedom required to simulate the whole scene. Simulating the full scene with 21 red coins and 21 yellow coins is impossible in the current setup, which also means that Gridmaster unfortunately doesn't adhere strictly to a real-life scenario. The controlled scene visuals and hard-coded suction gripper simulation are also limitations in a similar vein. Regardless, Gridmaster serves as a useful proof-of-concept for a real-life Connect 4 playing robot.

V. Discussion and Future Work

While Gridmaster's full-stack robotic system is fully capable of playing a game of Connect 4 against a human opponent, there are several limitations to the current approach.

For one, this work's perception system is limited by the predetermined RGB-D sensor rig, which is often impractical to set up and calibrate in a real-world scenario, and can't handle scenarios where there are chips lying outside of the camera's field-of-view. A more robust system might, for example, have a single arm-mounted camera that can move to scan an environment for chips. Gridmaster also doesn't use perception to determine the board's state. Instead, we opt to keep track of the desired game state in parallel, based on the desired action at each turn. This is a strong limitation that doesn't account for instances where a chip might fall into the wrong column, or not fall into a column at all. A more robust system might be able to detect this and try again.

Finally, Gridmaster also has several control limitations. More specifically, the naive implementation of differential inverse kinematics often leads to unwanted collisions with the board. The incorporation of a motion-planning algorithm like *Rapidly Exploring Random Trees (RRT)* [9] or one of its

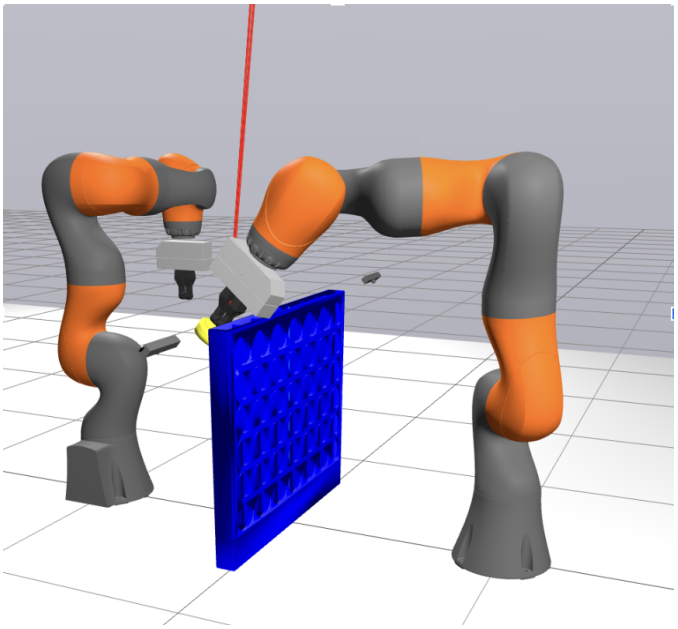

Fig. 4. **Potential Failure Mode**: This image is an example of a failure-mode caused by the naive control scheme employed by Gridmaster. Without motion planning and trajectory optimization, Gridmaster often makes rapid and erratic movements, and collides with the board as seen in this figure. This can be avoided through more sophisticated control.

variants, along with trajectory optimization with smoothness constraints and smarter interpolation, may enable smoother motion. The joint-velocity condition to switch between moves is also often too restrictive, leading to long periods of time where we are forced to wait as the arm comes to a halt, even if the end-effector is close to the correct pose.

In summary, Gridmaster demonstrates the utility of integrating the full robotics stack to perform precise manipulation tasks that may allow one to play a board game like Connect 4. Although limited, Gridmaster represents a greater goal of enabling robots to effectively perform anthropomorphic tasks. Enabling a robot to work within the confines of the human-designed world can unlock a world of impressive possibilities, as robots integrate their naturally-quick computation power with real-world interaction to solve problems that may be quite difficult for humans.

References

[1] X. Song and L. Hultros, "Connect four robot: Implementation of ai-strategies in a connectfour robot," 2018.

[2] A. Ayub and A. R. Wagner, "Teach me what you want to play: learning variants of connect four through human-robot interaction," in *Social Robotics: 12th International Conference, ICSR 2020, Golden, CO, USA, November 14–18, 2020, Proceedings 12*. Springer, 2020, pp. 502–515.

[3] "Ch. 3 - Basic Pick and Place — manipulation.csail.mit.edu," https://manipulation.csail.mit.edu/pick.htmldiff_ik_w_constraints, [Accessed 12-12-2023].

[4] "Drake: Model-Based Design and Verification for Robotics — drake.mit.edu," https://drake.mit.edu/, [Accessed 12-12-2023].

[5] "industrial intelligence 4.0_beyond automation — KUKA AG — kuka.com," https://www.kuka.com/en-us, [Accessed 12-12-2023].

[6] "Flood fill - Wikipedia — en.wikipedia.org," https://en.wikipedia.org/wiki/Flood_fill, [Accessed 12-12-2023].

[7] "Minimax — Brilliant Math & Science Wiki — brilliant.org," https://brilliant.org/wiki/minimax/, [Accessed 12-12-2023].

[8] "Solving Connect 4: how to build a perfect AI — blog.gamesolver.org," http://blog.gamesolver.org/, [Accessed 12-12-2023].

[9] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime motion planning using the rrt," in *2011 IEEE international conference on robotics and automation*. IEEE, 2011, pp. 1478–1483.