Robotic Scrabble

Bob Sumner
CS 7323 Autonomous Robotics
11 March 1997

Introduction

My goal in this project is to develop a control system for a robotic arm that will allow me to make it pick up pieces from a tray of letters and spell out a word on a Scrabble board. I will tell the robot which letters in its tray form the word I want it to play and the location on the Scrabble board of the start of the word. The arm will pick up the pieces and place them on the Scrabble board in their proper locations without disturbing the other pieces on the board.

The system should perform its task with a standard Scrabble board. That is, the robot should be able to perform its task using an unmodified Scrabble board and pieces just as they come new from the box. Any changes to the physical system must be made to the robot itself, not the Scrabble board.

I implement this project using a XR-4 robot arm manufactured by Rhino Robotics, Inc. The Rhino arm is a five axis robot arm with a motor driven gripper. I modified the gripper to facilitate the manipulation of Scrabble pieces.

Methodology

Care must be taken when choosing the methodology underlying a solution to a problem. In robotics, a spectrum of approaches exists that ranges from completely reactive systems with no world knowledge to non-reactive systems that rely heavily of world knowledge. There is no absolute correct or incorrect approach on this spectrum. Instead, the problem domain must be carefully examined and an approach selected that is appropriate to the specifics of the domain.

In the case of this project, the environment of a Scrabble-playing robotic arm suggest a non-reactive methodology. A Scrabble game takes place in a rigid, highly structured environment. During the coarse of a game, the Scrabble board and tray of letters need not move. The robotic arm, Scrabble board, and tray can all be attached to a workbench. (Note that I do not consider this a modification to the Scrabble game; it simply fixing the board's location.) Attaching the arm, tray, and board to the workbench essentially gives them an absolute, unchanging position in the world. The letters themselves are the only non-fixed items in the domain of this problem. I have chosen a completely non-reactive approach to developing the control system for the Scrabble playing robot.

Assumptions

The assumptions I make in developing the Scrabble-playing system in many ways define the methodology and approach. There is an interplay between assumptions that lead to a simpler, better solution and assumptions that are unfounded and therefore give poor results. I have both. First I'll state the good assumptions that help me to develop a solution of appropriate simplicity. Then I'll explain how other assumptions place limitations on performance.

As stated above, I make an underlying assumption about the environment. That is, I assume that the robot operates in a rigid world. The Scrabble board, tray of letters, and the robot itself are fixed in space. The table is sturdy and unmoving. The table top is flat and horizontal; pieces do not slide off due to gravity. Furthermore, the area above the Scrabble board is clear.

There is nothing suspended in the air that might interfere with the robot's motion. No person is going to try to push the arm out of place or take the piece out of the gripper while the robot is moving. These assumptions are justified. The robot does, in fact, operate in a rigid environment. I take advantage of these assumptions by simplifying the solution. For example, the robot need not worry about the board sliding while it is moving a letter from the tray to the board. Thus, a high degree of reactivity isn't needed. Instead, I can give the control program an accurate model of the environment and let the robot act in a non-reactive fashion.

However, though the assumptions I've made above are justified, they lead to some unfortunate results. When I conclude that a non-reactive system is sufficient, I am making other implicit assumptions that are not justified. Specifically, I'm assuming perfection. That is, I'm assuming that the robot's operation is perfect. If I tell it to go "here" a thousand times in a row, it will go exactly "here" every time. Implicit in this is that I have a perfect world model. When the robot puts a piece on the board, I assume that it was put exactly centered on the correct position on the board. The robot's world knowledge is updated based on this assumption. My non-reactive system has no feedback. If operation really was perfect, this would be acceptable. However, due to unpredictable errors such as play in the motors and inverse kinematics inaccuracies, the perfect operation assumption is not valid. The limitations of the system, in almost all cases, spring from this unjustified assumption.

Limitations

The primary limitation of the Scrabble-playing system is accuracy. The robot is able to pick a piece up from the tray of letters and place it on the board with an accuracy of +/- one square. That is the robot may place a piece halfway outside of the square in which the piece was

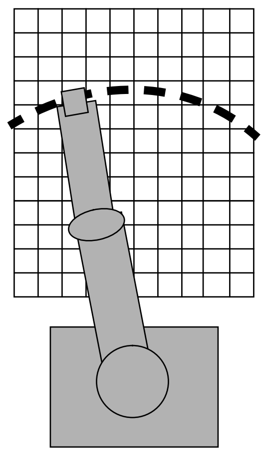

Figure 1. The position of the Scrabble board. Even when the arm is fully extended, some squares on the board cannot be reached.

intended to go. When the robot has to push a piece into a spot, the error is compounded. This is largely a result of play in the motors and inverse kinematics inaccuracies. There is perhaps a three degree play in the base motor. Since a Scrabble board requires very accurate placement, this play itself makes the system too inaccurate to play Scrabble. The problem is amplified as the arm is extended. I suspect there are some inverse kinematics errors because I have trouble making the arm move in a straight line. When I ask it to pick up a piece from the tray, it grabs the piece and should move straight up. However, the arm moves up and sideways, which sometimes knocks the letters out of the tray.

A severe limitation is the lack of feedback. The robot is unaware of any error it makes. So, if a piece is put on the board askew, the robot may latter bump into it, assuming that it was in a different location. A feedback loop would allow the arm to correct for errors introduced by play in the motors. One could argue that the system I have implemented is a glorified wind up toy. You tell it to go, and it goes. It preforms a sequence of operations from start to completion regardless of errors or unexpected problems. Sensor feedback could greatly improve performance.

A more basic limitation is the reach of the robot arm. I was unable to position the board in such a way that the robot can reach all parts of it. If the board is too close, the arm cannot reach down to it. If the board is too far, the arm's length isn't great enough to reach it. Placing the board at 45 degrees from the front of the robot afforded the best access. However, in this position the board's coordinate system was rotated from the robot's. I wanted to avoid unnecessary transformations which might add error to the positioning of the gripper. So, I chose a position that was aligned with the robot's coordinate system (Figure 1). The arm can reach about 60% of the board. The arm realizes when a square is out of reach. In this case it asks the user to place the piece for it. This is an example of cognizant failure.

Implementation

I take a divide-and-conquer approach to solving the problem. The high level task of spelling a word on the Scrabble board is recursively broken down into smaller and smaller problems. The small problems are then solved and combined in a bottom-up fashion until the entire solution is achieved. Solving the problem develops from bottom to top as follows:

- Establish a fixed environment including robot, board, and tray
- Add low-level "macros" such as picking and placing pieces
- Add knowledge of the world in the form of a board map
- Develop strategies to best place a piece given a certain board configuration
- Extend strategies to work with entire words

The underlying methodology behind my approach relies on having a fixed environment. So the first step in solving the problem is establishing a fixed environment and determining where objects are in this environment. When I started this project, the arm was already bolted to the

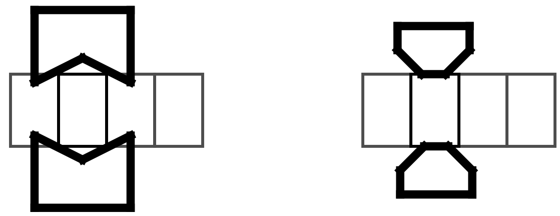

Figure 2: Original (left) and modified (right) grippers.

workbench. I must, then, secure the Scrabble board and tray to the same bench. The Rhino arm hardware performs inverse kinematics, allowing position control of the arm in x,y,z space. When selecting a place to bolt down the board and tray, I considered two factors. First, I wanted to align the board with the Rhino's coordinate system to avoid any unnecessary transformations. Second, I wanted to place the board and tray in easy reach of the arm. I was able to meet my first desire. But, no placement of the board allowed the robot to reach all squares. I bolted the board to the table in a position that allows the arm to reach most of the squares. All parts of the tray are in each reach of the arm.

After fixing the locations of the board and tray, I needed to develop a transformation from Rhino space to Scrabble space. A Scrabble board is marked with a 15x15 grid. I chose to represent this grid with a zero indexed two dimensional array. To develop a transformation from (x,y,z) in Rhino space to (r,c) in Scrabble space, I took manual measurements necessary to calculate the size of a square on the board. I used this size with the measured origin to make a base/offset calculation that transforms (r,c) to (x,y,z).

At this point it became clear that I needed to modify the Robot's gripper. I chose a gripper that is well suited for placing pieces on a board without bumping into other pieces (Figure 2). I structured the robot in a way that is most appropriate for the task.

Next I developed low level commands or macros to perform simple tasks such as picking pieces up and placing pieces on the board. The (r,c) to (x,y,z) transformation is all that is needed to place a piece on the board after the robot is holding it. But, picking pieces up from the tray was a difficult task that merits some discussion. The pieces in the tray are flush with each other, so the robot can only grab them along their top edge. However, to place a piece on the board, the gripper must hold the piece flat-ways. My solution to this problem involved the following steps:

1. Pick the piece up and out of the tray by the top edge
2. Move to a clear work space and balance the piece on its bottom edge
3. Knock the piece over, letter side up, with the gripper
4. Pick it up flat, either oriented top/bottom or left/right

This introduces a key point. There are two ways the gripper can pick up a piece: holding it along the top and bottom edges, or holding it along the left and right edges. I added low level control over which orientation is used.

Two final low level functions are that of pushing and dropping. A low level push function simply tries to push a specific piece up, down, right, or left one square. The drop functions moves the gripper above a square on the board and tries to drop a piece in the square.

At this point in the implementation, I combined the low level routines into higher level ones to yield more sophisticated behavior. First, I gave the robot a world map. In this case, the robot's world is the Scrabble board. So, the map is simply a 15x15 binary mask. Zero means there is no piece in the square and one means there is a piece. When the robot places a piece on the board, it marks the appropriate bit in the mask to indicate that the spot is now occupied.

With this knowledge, the robot has the ability to decide how to place a piece on the board. I defined a simple set of heuristics that the robot uses to decide how to place a piece on the board. Possible placement methods (Figure 3) include placing it directly on the square by holding it top/bottom or left/right, placing it next to the square and pushing it in, or dropping it. The method selected depends on the surrounding pieces. This is the extent of my implementation for this project. I've completed the steps up to this point. However, a final step remains for future work.

The final step is to extend the placement routine from a single letter to an entire word. Notice that the order in which the letters are placed is important. One placement ordering might be easier than another. For example, one ordering might block pieces in such a way that some have to be pushed whereas another ordering may allow them all to be placed directly. I want to select the ordering that is most accurate. Pushing pieces is less accurate than placing them directly, and dropping is even less accurate than pushing. So, I will develop a cost function that takes as input a certain ordering and gives as output a cost based on the types of placements required to execute this ordering. In order to select the best ordering for a word, I will enumerate all permutations of a word's ordering and run each permutation through the cost function. I will simply select the function with the lowest cost.

Future Work

I chose a completely non-reactive solution to the Scrabble problem. The approach goes most of the way to placing pieces on the board. However, it fails in that it is not accurate enough to place pieces on the board without bumping others. The thrust of future work would be solving this problem, i.e. developing ways to make the placement of letters more accurate. This will inevitably involve adding some sort of feedback loop. Vision (using a camera mounted above the work space) could be used to detect the position of letters on the board. A LED attached to the gripper and two cameras could be used to extract the 3d location of the gripper. Both of these feedback methods could improve performance. However, it should be noted that using these methods would greatly complicate the problem. Instead of a simple 15x15 array, some more complex data structure must be used as a map of the board. Planning with this complex structure will be more difficult. Plus, implementing the vision algorithms is certainly not a trivial task.

This project demonstrates an almost universal truth about solving problems with computers. 85% of a problem can be solved easily and quickly. The last 15% is orders of magnitude more difficult than the first 85%.

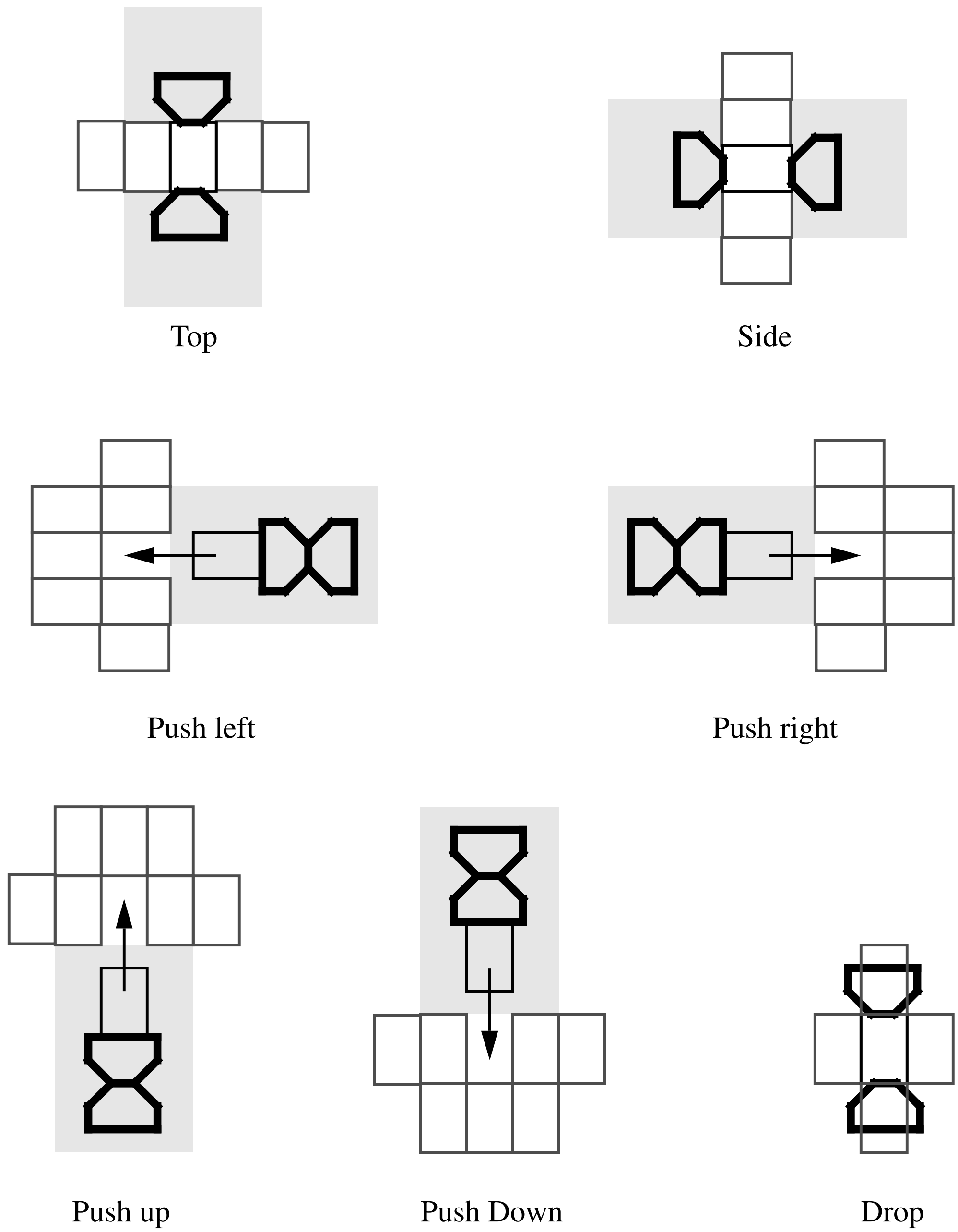

Figure 3: Seven possible placement types. The gray area must be free of pieces in order to use a certain placement method. Notice the Drop placement has no gray area.