Artificial Intelligence for Physical Rehabilitation

Jacob Harer
Sang-Joon Lee
Srivathsa Rajagopal
Fulya Koc

04.28.2016

CS640: Artificial Intelligence - Graduate Project Report
INTRODUCTION

For this project, we implemented the Pacman game, which is controlled using a robotic arm. The robotic arm is designed for neuroscience research. It is made to help physical rehabilitation using video games with the usage of haptic feedback. The game uses an AI for the ghosts that would follow Pacman. Our design has three level difficulties, each with a different layout, and an increase of ghosts.

A system overview is shown above. The user interfaces with the robot arm which communicates with a c++ program running on its computer. The c++ program is responsible for measuring the position of the robotic arm and providing haptic feedback to the user. A python implementation of pacman is also running. The game has a socket connection to the c++ robotic arm controller and receives from it the current position of the ball of the arm. It converts the arms coordinate system to the grid system used by pacman and uses it to control pacman. The game is shown on a monitor to the user.

A video of our implementation in action is at https://youtu.be/in7PNrijI_o

GAME IMPLEMENTATION

Pacman has a simple design of a game layout that includes Pacman (the protagonist)
and ghosts (the antagonists). The layout features a grid of dots, and there are walls scattered on the layout to increase difficulty. The goal of Pacman is the consume all the dots on the game layout without getting eaten/caught by the ghosts. There are larger dots that give you the ability to eat the ghosts which give you time to continue consuming the rest of the dots before they respawn. The haptic force included through the robotic arm forms the walls of the game. This makes it so that the player cannot move out of the game layout and will have a better sensation of being in the game. The ball of the robotic arm where the player rests their hand is the point where pacman is shown in the game. The ball is moved in order to get Pacman to move on board. The coordinates of the ball is integrated through the C++ code of the robotic arm to the Python code which possesses the user interface of the game.

**PACMAN USER INTERFACE**

**Intermediate Level (Level 2)**
Equivalent Layout file:
```
% % % % % % % % % % % % %
% o . . . . . . . . . . o %
% . % % % . % % % % % % . %
% . . . G . . G . % % % . %
% . % % % % % % . % % % . %
% . % % % . o . . . . . . %
% . % % % . % % . % % % . %
% . . . . . P . . % . . . %
% % % % % % % % % % % % %
```

(.) represents a pac-dot or pellet.

(o) represents a power pellet

(P) represents the starting position of the Pacman

(G) represents the starting (and respawning) position of a ghost

Moderately complex layout with fewer escape routes

Relative Speed of Pacman to Ghost = 0.25

Number of Ghosts = 2

**Beginner Level (Level 1)**
Simple layout with a lot of escape routes
Relative Speed of Pacman to Ghost = 0.25
Number of Ghosts = 1

**Expert Level (Level 3)**

![Pacman Game Image]

Complex layout with less spacing between the routes (more number of thin walls)
Relative Speed of Pacman to Ghost = 0.25
Number of Ghosts = 3

**Haptic Feedback Design**

The Proficio robotic arm is capable of creating haptic force that creates sense of touch by applying force, vibrations or motion. The proficio’s haptic feedback act as friction space using PID controller of the robot arm system. The Kp, proportional gain and Kd, derivative parameter of the PID controller can be changed using these parameters. The strength of the haptic feedback can be controlled using the proportional gain, Kp.
The haptic forces used to create sensation of “walls” in the pacman game. The requirement for the pacman game was to have walls that cannot be penetrated or go through by the robotic arm. Therefore, we used a high proportional gain value with Kp value of Kp. The user is able to feel the wall while navigating through out the layout as well as vibration on the corners of the wall. This is due to simulation of two x and y forces that are pushing the robotic arm at the location, resulting in vibration.

In order to create a haptic wall, two input formation is required. We need to provide the location coordinate (x, y, z) of the center of the object (a box), and size of the box with length, width and height. This must be assigned for each of the outside wall (boundaries) of pacman as well as inner wall that represent the different configuration of wall for each level. The Proficio library provides a “summer” that allows to add these haptic forces. All haptic force walls must be added using these summer, finally as one summation of haptic forces that can be represented for the session.

For this project, we created three different .cpp files that creates haptic configuration for each of the 3 levels of the pacman game. Please refer to README_PACMAN for more information on each of the .cpp files.

An example of haptic wall is shown below. This as a 3-D visualization of the haptic wall that was created for the pacman game level one. As you can see that there are rectangular boxes that represent each of walls in the game.
POSITION MAPPING

Control of pacman via the robotic arm was done by mapping the ball of the robotic arm into pacman space and having pacman move toward the ball. As is shown in the above figure, the ball moves in x,y,z space. The x coordinate was thrown away and the x,y coordinates were used to make a linear mapping into the pacman game. The four corners in pacman were mapped to four specific points in the xy plane of the robot. These points are hardcoded in the current implementation but are easy to change.

AI ALGORITHM

For the game, we design an Artificial Intelligence for the ghosts so they will follow Pacman and provide a challenge to the player. This was done using a combination of the Minimax algorithm with alpha-beta pruning and the A* algorithm. Minimax performs well for cases when the ghosts are close to pacman, but it has a practical limit on the depth of the search tree before it begins to slow down computation and affect game playability. Because of this we approximate non winning-losing states with a heuristic. The heuristic is a summation. The distance metric is used to reward ghosts for taking moves which bring them closer to pacman. While the score metric is used to reward ghosts for denying pacman access to pellets or other sources of points.

The performance of this heuristic approximation is limited because the euclidean distance is not always a good estimator of actual path distance given the presence of obstacles. This is particularly relevant in pacman because of the large number of walls in the game. Because of this our minimax algorithm performs well when the distance between pacman and the ghosts is less than the depth of the search tree. When this is
not the case it must rely on the heuristics which are unreliable. In order to improve this performance we opted to switch to the A* algorithm to find the optimal path between the ghosts and pacman when the euclidean distance between them is greater than the maximum depth of minimax.

To perform A* efficiently we create a graph out of the pacman board. We define hallways in the pacman board to be cells in which movement is only allowed in two opposite directions. Cells which are not hallways and not walls are considered intersections. Practically this mean intersections are cells where ghosts will have to make a choice of which direction to turn. Each node in the graph represents an intersection., Nodes in the graph are connected if they are connected by a set of hallways.

In addition to the graph above to compute A* we also need to know pacman and the ghosts relationship to the graph. To do this we compute the cost to move from the ghost/pacmans current position to nearby intersections. This is trivial if they are currently on an intersection because we need only consider that intersection with a cost of 0. If they are on a hallway though we follow the hallway in both directions until we get its ending intersections and include both with a cost of the euclidean distance from our target to those intersections. The ghosts are used to define a set of start nodes and the cost to reach those start nodes. Pacman is used to define a set of goal nodes and the cost of those goal nodes. The A* algorithm is thus modified to take a set of start nodes each with a cost associated with it. This is equivalent to adding an additional node at the ghosts current position. Likewise each goal node has an extra cost associated with it which is added to the cost of any path which finishes at that position. This is equivalent to adding a single goal node at pacmans position.

As one final addition to the algorithm we wanted to reward ghosts for taking different paths to pacman in order to cause them to flank pacman. Without this A* will often have ghosts following each other in a line. We added an additional cost of ten to the paths in the A* algorithm if another ghost is already on that path. This causes ghosts to take separate paths unless there is no other reasonable path for them to take.

**EXPERIMENT OVERVIEW**

For experiments we designed three different pacman layout with varying degrees of difficulty. We also increased the number of ghosts from one to three as the difficulty of the level was increased. We tested patients on all three levels with both their strong
and weak arm. We recorded the score achieved on the trial, the outcome of the trial, and the path that was taken during the trial. We tested this for 6 different patients. The results are shown below.

**PLAYER EXPERIMENTATION RESULTS**

The following table shows the results of the experiment on six users. Each user attempted one trial for each level for both right and left arm. All users had right arm as dominant arm.

<table>
<thead>
<tr>
<th>Subject</th>
<th>Difficulty</th>
<th>Level</th>
<th>Ghost Speed</th>
<th>Num of Ghosts</th>
<th>Right Arm Score</th>
<th>Left Arm Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sri</td>
<td>Beginner</td>
<td>1</td>
<td>0.25</td>
<td>1</td>
<td>831</td>
<td>1023</td>
</tr>
<tr>
<td>Jacob</td>
<td>Beginner</td>
<td>1</td>
<td>0.25</td>
<td>1</td>
<td>817</td>
<td>798</td>
</tr>
<tr>
<td>SJ</td>
<td>Beginner</td>
<td>1</td>
<td>0.25</td>
<td>1</td>
<td>1088</td>
<td>1027</td>
</tr>
<tr>
<td>Fulya</td>
<td>Beginner</td>
<td>1</td>
<td>0.25</td>
<td>1</td>
<td>846</td>
<td>0</td>
</tr>
<tr>
<td>Ajjen</td>
<td>Beginner</td>
<td>1</td>
<td>0.25</td>
<td>1</td>
<td>-499</td>
<td>-416</td>
</tr>
<tr>
<td>Elham</td>
<td>Beginner</td>
<td>1</td>
<td>0.25</td>
<td>1</td>
<td>689</td>
<td>-476</td>
</tr>
<tr>
<td>Sri</td>
<td>Intermediate</td>
<td>2</td>
<td>0.25</td>
<td>2</td>
<td>-408</td>
<td>-405</td>
</tr>
<tr>
<td>Jacob</td>
<td>Intermediate</td>
<td>2</td>
<td>0.25</td>
<td>2</td>
<td>-451</td>
<td>757</td>
</tr>
<tr>
<td>SJ</td>
<td>Intermediate</td>
<td>2</td>
<td>0.25</td>
<td>2</td>
<td>-228</td>
<td>-175</td>
</tr>
<tr>
<td>Fulya</td>
<td>Intermediate</td>
<td>2</td>
<td>0.25</td>
<td>2</td>
<td>760</td>
<td>-371</td>
</tr>
<tr>
<td>Ajjen</td>
<td>Intermediate</td>
<td>2</td>
<td>0.25</td>
<td>2</td>
<td>-396</td>
<td>676</td>
</tr>
<tr>
<td>Elham</td>
<td>Intermediate</td>
<td>2</td>
<td>0.25</td>
<td>2</td>
<td>-615</td>
<td>-450</td>
</tr>
<tr>
<td>Sri</td>
<td>Expert</td>
<td>3</td>
<td>0.25</td>
<td>3</td>
<td>-179</td>
<td>98</td>
</tr>
<tr>
<td>Jacob</td>
<td>Expert</td>
<td>3</td>
<td>0.25</td>
<td>3</td>
<td>265</td>
<td>-465</td>
</tr>
<tr>
<td>SJ</td>
<td>Expert</td>
<td>3</td>
<td>0.25</td>
<td>3</td>
<td>72</td>
<td>-452</td>
</tr>
<tr>
<td>Fulya</td>
<td>Expert</td>
<td>3</td>
<td>0.25</td>
<td>3</td>
<td>59</td>
<td>-117</td>
</tr>
</tbody>
</table>
As you can see from above result, the scores for right arms were generally higher for each of the levels, and winning percentages were slightly higher with users dominant arm.

The figure below gives an illustration of right arm vs. left arm score for beginner level. (Blue - right arm, Red - left arm) It shows that for the beginner level the right arm (dominant arm) had yielded slightly higher final game score for most of the users.

The figure below shows the right arm vs. left arm at intermediate level. Surprisingly left arm performed better for some users. Our hypothesis is that this result is due to “learning” factor of human brain where we quickly adapt to the new experience from the beginner level and apply & do better while second time around. We also notices this behaviour with order of trial when performing right arm first before left arm. Left arm performed much better when trials were performed after right arm.
The following figure illustrates performance of each user with right arm vs. left arm for expert level.
ANALYSES

The graphs below show the plotting of the X-Y coordinates recorded for the game play. The movement of the robotic arm is shown for both the dominant and the weak arm. Since all the members of the team have a dominant right arm, we are just considering the Right arm to be the dominant one, unless otherwise stated.

The movement recorded from the right arm shows, as expected, a relatively smoother path trace. The zig-zag movements observed in the graph are caused by the haptic feedback walls that are used to model the Pacman board layout.

**Right Arm vs. Left Arm: Beginner Level**

![Figure: Beginner Right Arm XY Position](image)
Figure: Beginner Left Arm XY Position

**Right Arm vs. Left Arm: Intermediate Level**

Figure: Intermediate Right Arm XY Position
**CONCLUSION**

It is difficult to play Pacman with robotic arm, especially with Ghost adversaries that is implemented with an AI algorithm.

There is a slight learning curve that is associated with playing the Pacman game with the Robotic arm. This is additionally influenced by the adaptation to the plane of robotic arm motion.

The experiment results show that the dominant arm had better control of robotic arm movement and was able to achieve higher score. But it is also important to note that the difference between the dominant and the weak arms isn’t too high which suggests that the movements required to play the game are very simple and easy to learn.

**FUTURE WORK**

**3D version of Pacman**

We could have a 3-D version of Pacman game instead of the conventional 2-D version. This would involve work on the design of the game itself. Also, the game has to be exciting and easy to play with a very small learning curve.
Add additional orientation options (Y-Z plane, X-Y plane)

The current layout of the robotic arm motion is in the X-Y plane. With the monitor or the screen displayed to the users is in the X-Z plane, it hinders the natural movements in gameplay. We could add a new orientation to the project to make it more intuitive.

Design levels specific to rehabilitation prescriptions

We could have different versions of Pacman boards created right out of the layout text file. This would automate a lot of haptic wall creation work and the project could potentially end up as an application platform for non-technical people to create boards (text file) on the fly and have the game running for it within minutes. In a way, this can be a type of varying the complexity of game.

Use haptic feedback to make it more challenging

We could incorporate haptic feedback all along the path in addition to the already existing haptics for the wall. This haptic feedback could be of much less intensity and may aid as a frictional force to require additional strength to play the game.

Using different AI algorithms for individual ghost

The original version of Pacman game has different ghosts using different algorithms like chase, ambush, predefined motion and random. We could use this idea.

Having the robotic arm assist the user - Automated Training of Movements

There is a way to train the arm to learn from the movements by recording them and playing back as a repeat. This could help the physiotherapists to train their patients accurately. The physiotherapist could trace a certain path on the Pacman board to follow and the patient could just place their arm into robotic arm’s slot while it retraces the path that it was taught.

Design the game more accessible to left handed people

Current version supports only the right handed people. The Proficio arm is capable of
physical flipping to transform into a left robotic arm. We would need to re-map the coordinates and potentially make it much more user-friendly for left-handed people.

CITATIONS

   a. [http://ai.berkeley.edu/project_overview.html](http://ai.berkeley.edu/project_overview.html)
   b. [http://ai.berkeley.edu/multiagent.html](http://ai.berkeley.edu/multiagent.html)
   [http://www.redblobgames.com/pathfinding/a-star/implementation.html](http://www.redblobgames.com/pathfinding/a-star/implementation.html)
3. [https://github.com/BarrettTechnology/libbarrett](https://github.com/BarrettTechnology/libbarrett)
5. [https://youtu.be/9Hd4SGyF3TE?t=2m40s](https://youtu.be/9Hd4SGyF3TE?t=2m40s)