AN IMPLEMENTATION OF PACMAN GAME USING ROBOTS

Madhav. Rao

India
madhavrrao@gmail.com
http://unix.eng.ua.edu/~mrrao

Abstract

As the field of robotics are advancing, robotics education needs to consider technological advances and societal level of interest. Realizing computer games in robotic platforms is one such technological advance for educating students in robotics science. Realizing computer games in robotics environment is still a challenge due to high investment factor in developing robot models. However the effort can lead to the enhanced interest in robotics education and further involvement in science and technology careers. Young gaming enthusiasts are aware of different playing strategies used in the computer games. If the course curriculum uses integrated approach by building a game in physical robotic environment, students' strategy developing skills are tapped and students can get jumpstarted in learning course with interest. Algorithms developed by students to realize the game, could find similar real world applications. One such popular game, Pacman is implemented using two iRobot Roomba robots. One robot is considered as Pacman, which escapes from other robot. The other robot, referred as Ghost, tries to attack Pacman. Network camera was used to find robot localization. Interprocess communication was included to share the data among different processes. The programs for Pacman and Ghost robots were built using Player/Stage, an open source package, compatible with iRobot platform. The algorithms were prior tested in Player/Stage simulation platform before implementing with real robots. The partial success of Pacman game in robots is discussed in the paper. This attempt should encourage in realizing more computer games in robotics education curriculum and generate immense interest in robotics education using low cost ready to use robots.

***Keywords*:** pose; blob; socket; video games, pacman; player/stage;

1. Introduction

Pacman, is a historically popular and entertaining [Lund (1999)] game in the world. Realizing computer animated games such as Pacman, using autonomous robots is often challenging. However the attempt will allow exploring solutions to some of the complex problems, by young and new programmers. Young video gaming enthusiasts are more involved in developing strategies to play the game effectively. The strategy if translated to a software program consisting of different hardware's: sensors, vision systems and robots, will provide an engaging experience in learning software, hardware's and other engineering skills. In a view to encourage robotics education, an attempt to realize video game using physical robots is stated in this paper.

To develop Pacman game, multiple robots need to be used in a certain mapped area. The player controls the Pacman by moving through a maze, eating pac-dots. When all dots are eaten, Pacman is taken to the next stage. Four enemies try to catch the Pacman and thereby reduce the life of the Pacman. This article modifies the real game and introduces one enemy instead of four to simplify the developmental cost. In addition pac-dots are not included in the maze and the game is restricted to one level.

Physically realizing a game using robots leads to high investment and hence a generic platform, referred as Player/Stage, described in [Anderson *et al.* (2007)], was used to simulate robot dynamics. Prior simulation provides a way to understand the essentials of the game, before implementing with physical models. Player/stage provides the necessary simulator for controller testing, prior to robot implementation. The Player/Stage demonstrates flexibility, transparency and speed that make it useful for robotics development. Hence programs were developed using Player/Stage platform.

The realization of Pacman in robotics involves several key algorithms and concepts such as attack and escape path finding, blob detection for robot localization, and interprocess communication for robot synchronization. The article shows implementation of Pacman game using simple and ready to use robots. Three different

experiments were conducted to verify the expected movements from the gaming robots. The approach takes a step towards realizing other similar games using robots. The attempt will encourage students to develop robotic algorithms with interest. The algorithms currently used in the prototype shows a simple approach. Higher degree of complexity in the algorithms can be easily included and realized in the future.

2. Background

Recently toy-robots are used to develop game based learning system [Chou *et al.* (2011)]. Off the shelf educational toy robot kits from Lego Mindstorms NXT are used immensely in the learning system [Lego mindstorms (Online)]. It is learnt that students learning skills are improved by the use of robots in education [Anderson *et al.* (2007)]. Robotics which is seen as a multidisciplinary field but, the software development is entirely a computer science area. Hence use of robots in realizing the popular games will tap majority of the students who are interested in the games to choose computer science as their major. Realization effort via robotic simulation techniques were also developed in the past [Josh *et al.* (2006)]. However realizing a game in a real robot platform is often neglected due to high cost investment in physical hardware's. The paper discusses realizing an engaging and popular Pacman game using multiple low cost robots, which will encourage the development of more gaming based robots.

iRobot's roomba is a low cost robot platform, which are extensively used in research and education. Various sensors such as wheel encoders, bump sensors, IR Wall sensors, cliff sensors, and IR receiver are available on off the shelf roomba robots. The platform also includes driving wheels, and LEDs [irobot (Online)]. The various sensors and actuators fulfill our requirement of motion based robots to realize the Pacman game.

Software development in robotics is hardware dependent. In addition robotics programming is complicated and time consuming. The Player/Stage Project, an open source tool simplifies controller development and sensor network systems [Gerkey *et al.* (2003)]. The Player platform provides a clean and simple interface to the robot's sensors and actuators over network. Client control programs talk to Player over Transmission Control Protocol (TCP), reading data from sensors, writing commands to actuators and configuring devices on the fly. In addition Player supports a wide variety of robot hardware and provides useful implementations of sophisticated sensing and control algorithms. Stage provides a population of simulated robots and sensors operating in a two-dimensional bitmapped environment. The devices are accessed through Player, as if they were real hardware. Hence the use of Player/Stage allows the programmers to independently develop the software and apply to one among many supported hardware's.

Autonomous robots require a continuous stream of data, to make instantaneous decisions. In this game, a way to detect the robots and transfer global position data to other robots was intended. Various sensors: odometer, inertial based systems and beacons are used in the past to determine the localization of robots. Odometer refers to integration of incremental motion information over time. However the sensor is highly susceptible to produce orientation errors and over time, will generate large lateral positional errors. Accelerometers, gyroscopes and other inertial navigation systems have an advantage that they are self-contained and they do not need external references. However inertial sensor data drift with time because of the need to integrate rate data. This makes it unsuitable for accurate positioning over an extended period of time. Active beacons are the most common navigation aids on ships and airplanes, due to high speed in detection of objects and minimal processing. However this incurs high cost in installation and maintenance. Global position system (GPS) is the best localization method available for outdoor navigation, but does not apply for indoor research activities [Borenstein *et al.* (1997)]. Visual detection by camera is one approach which is relatively inexpensive and provides global positioning data with relatively less erroneous results. The Visual detection is restricted to field of view and used in indoor research laboratories [Borenstein *et al.* (1997)]. The visual detection method used in a multirobot system, requires each individual robots to be identified in the captured image. Hence the robot is covered by color coded construction paper which acted as blobs in the whole region of image captured. Blob is a region that is either brighter or darker than the surroundings in the image or video. Object, mark and humans in videos and images were modeled as blobs in the past [Wang and Ju (2008)], [Kefalea (1998)]. Colors marked in the robots were used to localize the robots used in the platform.

3. Implementation

Two iRobot Create [irobot (Online)] robotics platform were used to implement the Pacman game. A command module included Gumstix architecture, a 400 MHz Intel PxA255 driven platform along with 802.11b serial extension board. The small size of the Gumstix (4 inches by 2 inches) along with low power consumption (approx. 0.6A) and a preinstalled version of linux provides a powerful robot control platform, that is compatible with Player/Stage robot simulation and control platform. The Bluetooth adapter module (BAM) is attached to the serial port of the Gumstix architecture. The BAM enables client facility to receive commands from a remote controlled computer. The implementation of Pacman using robots required a modified and simpler version of game. One robot was considered as Pacman. The Pacman robot goal was to escape from the Ghost. The other robot was referred as Ghost. The Ghost attempts to chase down and 'eat' (it accomplishes this by lightly colliding) Pacman. For the demonstration purpose, only one Ghost was used, rather than the traditional set of four Ghosts. This is to avoid complexity in communication between multiple Ghost robots. Three major tasks were identified for the robots:

- The Pacman estimates an escape route based on the location of the Ghost.
- The Ghost estimates the shortest attack path based upon the location of the Pacman .
- The robots follow an occupancy-grid map to identify their positions, which were captured by a network camera.

The architecture uses autonomous operations. While in a particular grid cell, the robot uploaded its position information to the centralized server system. The global position information was captured by a network camera placed above the game area. Remote intelligence was provided to the Pacman and Ghost through processes running on the server. The Ghost followed an algorithm to attack the Pacman. The Pacman followed an escape algorithm [Cowley *et al.* (2009), Cowley *et al.* (2007), Cowley *et al.* (2006)] while following the map [Butler *et al.* (2001)]. Since the iCreate platforms were light weight and do not have high processing power, the decision making algorithms were processed on the centralized server. The server determined the path for each robots, allowing robots to move quickly.

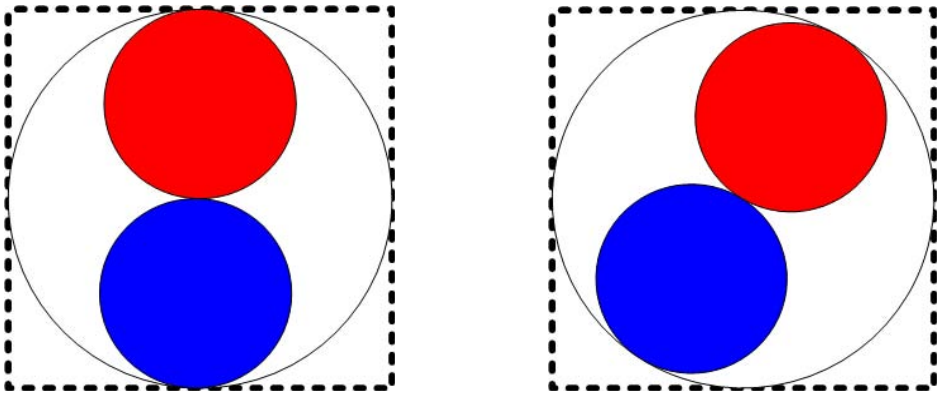
Network camera module was used to determine Pacman and Ghost locations. The blobs on the Roomba's were made of construction papers. The blobs identify the global position of the robots. The positions were then fed to the pacman and ghost processes. The usage of construction paper as blobs was influenced by [Anderson *et al.* (2007)]. The network camera identifies the blobs of different colors in the image. The idea was extended to detect two blobs for each robot for this game. Blob detection method described in [Wang and Ju (2008)] used an extensive algorithm; however a much simpler algorithm was used for this study. None of the filtering techniques were included in the blob detection algorithm. The camera captures the image in jpg format. The CImg library provides necessary functions to access the pixel values of the image [CImg, (Online)]. The following pseudo code was developed as the blob detection algorithm, to identify individual positions of two robots. The complete blob detection code is available in the source code package as supplemental file [Rao (Online)].

```
FUNCTION FindBlob( image, colorEvaluator )
        let blobList = empty list
        for each pixel in image
                if colorEvaluator( pixel ) returns true
                        addToBlobs( pixel, blobList )

        maxBlob = blobList[0]
        for each blob in blobList
                if area( blob ) > area( maxBlob )
                        maxBlob = blob
        return maxBlob

FUNCTION addToBlobs( pixel, blobList )
        for each blob in blobList
                if isPixelNearBlob( pixel, blob )
                        blob.addPixel( pixel )
```

The minimum and maximum x and y pose values obtained from the blob detection algorithm were used to construct bounding box of the blob. The blobs were made of a single color and hence midpoint between the centers of the two blobs of a robot provides an approximately correct center position for individual robots as shown in Figure 1. The angular position (yaw) of the robots was determined by the relative coordinates of the center of two blobs. The pose information for both robots were then sent to each robot processes via sockets interprocess communication channel.



a) Zero angular positioned robot b) Angular positioned robot

Fig.1. Illustration of blobs used in identifying robots position in the Pacman game.

On detecting the red and green blobs, the center position of each blob was calculated and an imaginary rectangle was drawn along the dotted lines. The dotted lines observed represented the boundary of the blob in a single grid cell. The robot movement continued as long as it did not fall completely within a cell. The path-planning algorithm was then called when a robot completely fell on the cell. The network camera's frame-rate is adjusted such that the robot's movements were captured without passing any boundaries of the cell. Note that the map's dimensions were restricted to the network camera's field of view. The Ghost and Pacman robots blob colors were fixed throughout. Red and blue construction papers were mounted on Ghost and yellow and green colors were mounted on Pacman throughout the experiments.

4. Behavioral Model

A hybrid behavioral model is depicted in Figure 2. There were 3 processes running on the server: the camera, ghost, and pacman processes. The camera process exists to ultimately generate position and orientation: referred as poses in the Figure 2. The position data were used by the pacman and ghost processes. The pacman and ghost processes exist to execute the behaviors of the two robots. The Pacman and Ghost robot contain nearly mirror behavior, with the exception of the path planning algorithms (Attack for the Ghost and Escape for the Pacman). Their behavior models only contain a single state, except the end state which is triggered when the Ghost captures the Pacman. This state has been omitted for clarity and simplicity. The behavior model of each robot has three-layers. The high level path planning layer (Escape and Attack planning layers) plans a path for the robot. The path planning layer takes poses of robots as inputs. The layer outputs a path, which is then taken as input for the path execution layer. Execution layer determines how to execute the path given as input. The lowest layer, obstacle avoidance, exists for obvious reasons.

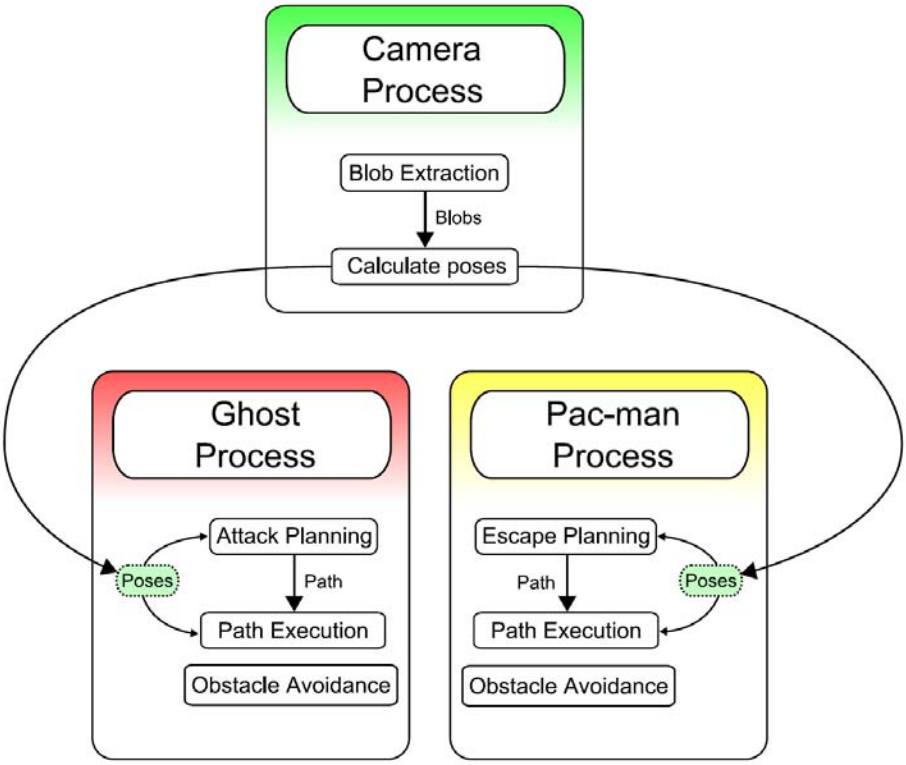
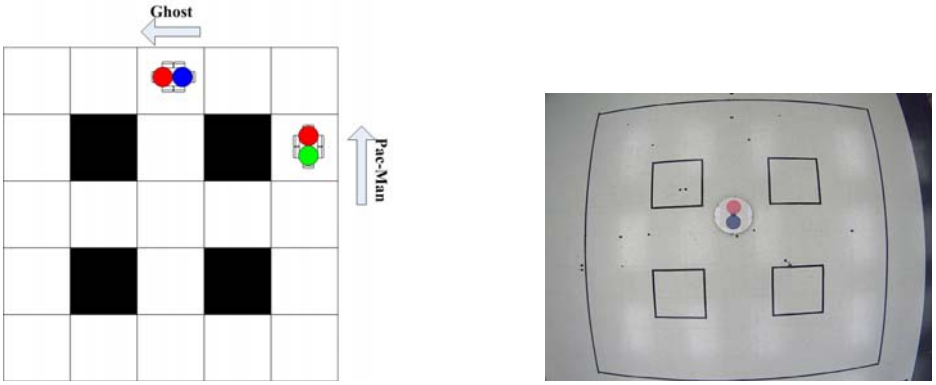


Fig 2. Schematic showing the behavioral model of robots in Pacman game.

The map forms the area, where the robots move to attack and escape from other robot. While the true Pacman game area is much more involved, the modified game area was used, to simplify developing the algorithm. The map is a grid consisting of black and white colored cells as shown in Figure 3. The black cells represented obstacle areas and white cells indicated open areas. The small, square areas enclosed by black tape in the inner part of the map represented the boundaries of the obstacles. The path following decision was made by the robot only when the robot was found in the open area. Each robot ensured that it does not cross the boundaries of the obstacle areas while moving. The path planning level and sub level of robot server processes ensures that the robot stays within the open area. Interprocess communication between camera and robot processes were designed using socket communication. Socket communication was chosen above standard input output pipes and file operations due to the communication speed and operational simplicity. Two TCP/IP sockets were used on the camera process to create channels for the two robot processes. However, both channels received the same information: the poses of two robots used in the realization.

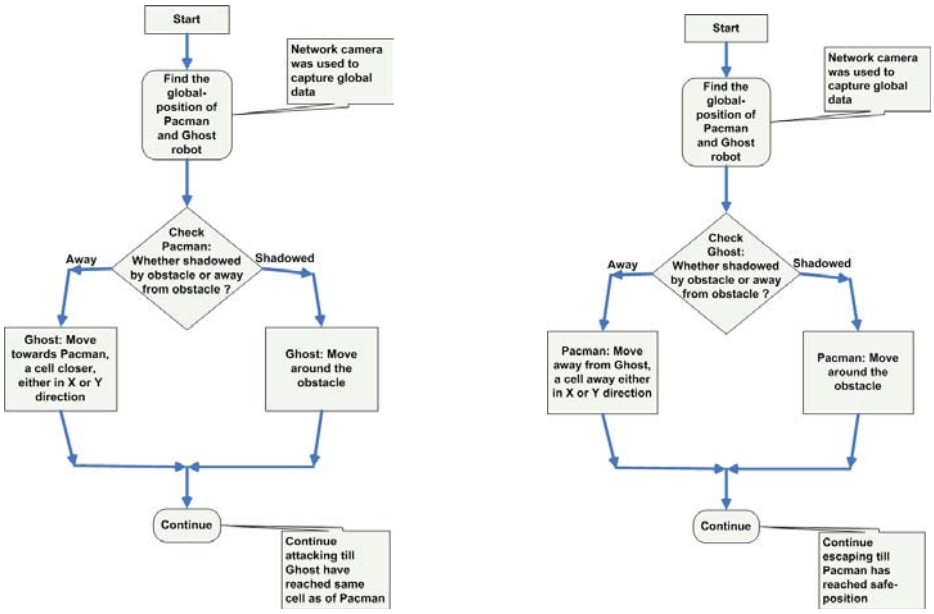


a) Map on paper b) real view of map

Fig.3. Map representation of Pacman game.

5. Attack and Escape Algorithms

Our Pacman game did not followed all of the classic rules as explained in [Katie and Zimmerman (2003)]. While the Pacman attempted to escape from the Ghost, the Pills and Dots were not integrated in this game. In our version, the game was completed when the Ghost touched the Pacman robot. Also, the point system was not included in the game, and time was never a constraint. However, the game followed a simple path-planning approach for the two robots employed in the game. A similar architecture was explained in [Butler *et al.* (2001)]. The attack and escape algorithms feed their respective robots with the data of next immediate cell position to reach. The game ends if the Ghost reached the Pacman's cell and touched Pacman. The algorithm also avoided the obstacle area while moving. The flowcharts for the attack and escape algorithms are shown in Figure 4.

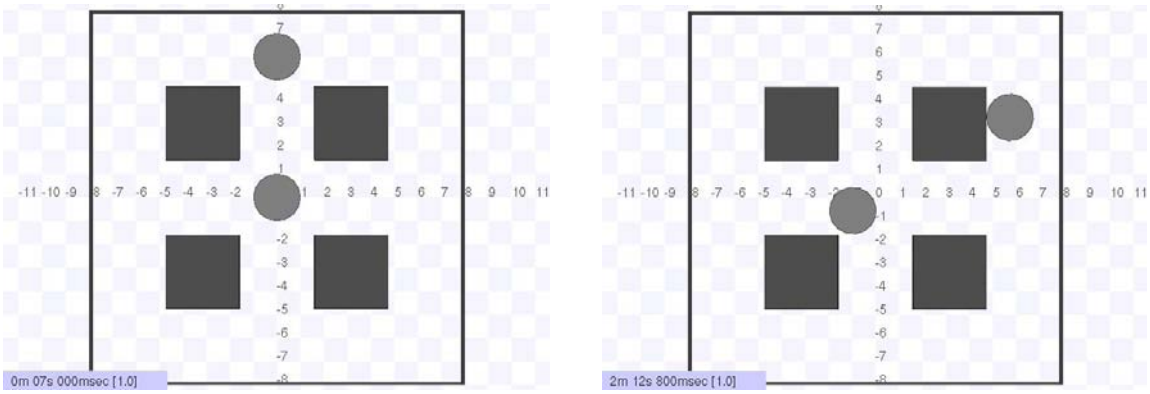


a) Attack algorithm b) Escape algorithm

Fig.4. Algorithms developed for realizing Pacman game using robots.

6. Player/Stage Simulation

For the most part, the roomba's maneuvering code were taken from the test roomba, originally developed by Professor M. Anderson and her research group in the University of Alabama. The Pacman and Ghost were deployed as two robots in the Player/stage environment. The code developed for the simulation uses file i/o as the communication medium to pass the position data to the two processes, since the player software has a glitch in the camera interface; hence integration of the camera module in simulation was not possible. The game area introduced in the simulation is similar to the occupancy grid. The simulation of robots, using game area is shown in Figure 5. The robots were shown with no blobs. The simulation provided testing of the software developed, for physical iRobot platforms.



a) Initial position b) Middle of play

Fig.5 Simulation of Pacman game in Player/Stage.

7. Experiments

7.1. *Setup*

The overall hardware setup for roomba robots to identify their global positions was as follows:

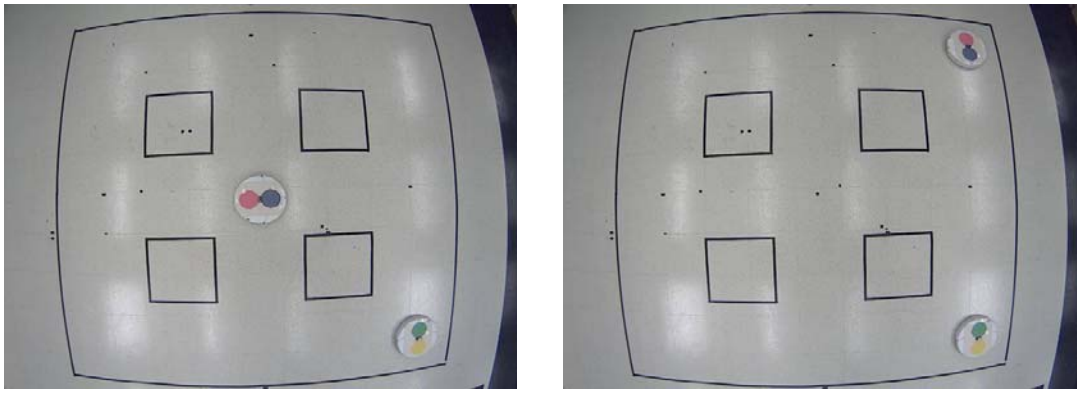
- Attach the blobs to two roomba robots
- Installed network camera Axis-203MW on the ceiling of the lab. The camera was installed to cover the game area with the least distortion.
- The network camera was connected to a local router. A desktop computer was then connected using the same router.
- The Roomba robots were reachable from the desktop through their BAM modules over a Bluetooth usb device.
- The Player/Stage software was installed in the computer. The Pacman and Ghost processes were developed within Player/stage. Additionally, the CImg library was installed for reading the images used in the blob detection algorithm.
- Process communication through standard input and output and file pipes produced delay and hence TCP/IP sockets were used for interprocess communication.

The experiments to realize the game were carried in steps. Stepwise testing included keeping either of the robot static and other robot moving to destination. The stationary positions of the robot were changed to find the number of moves required for other robot to attack or escape. Number of moves was referred to the number of cells traversed by robot, to reach its destination. The total number of cells in the map, seen by the camera is limited to 25. Each cell is represented by the square in the map region shown in Figure 3 (b). Each positioning of the robot is treated as different experiment and was repeated five times each, to ensure the validity of the processes developed. The last step included the motion of both robots and realizing the Pacman game. Only four experimental positions were discussed in each steps.

7.2. *Ghost was moving, Pacman was static*

In this experiment, Pacman is held static, while Ghost is given freedom to approach Pacman. The Pacman was held at different positions and Ghost was expected to reach Pacman as shown in Figure 6. Each different positions of Pacman is considered different experiment #. The Table 1 data represents the number of moves for the Ghost to reach Pacman starting from their respective positions. The experimental result coincides with the expected number of moves made by Ghost to reach Pacman. All different positional experiments were observed as successful attempts.

a) Experiment1 b) Experiment2



c) Experiment3 d) Experiment4

Fig.6 Camera view of different Ghost positions used in the experiments.

Table 1. Number of moves Ghost takes to reach Pacman.

Experiment #	# of moves	Comments
1	8	Successful
2	4	Successful
3	4	Successful
4	4	Successful

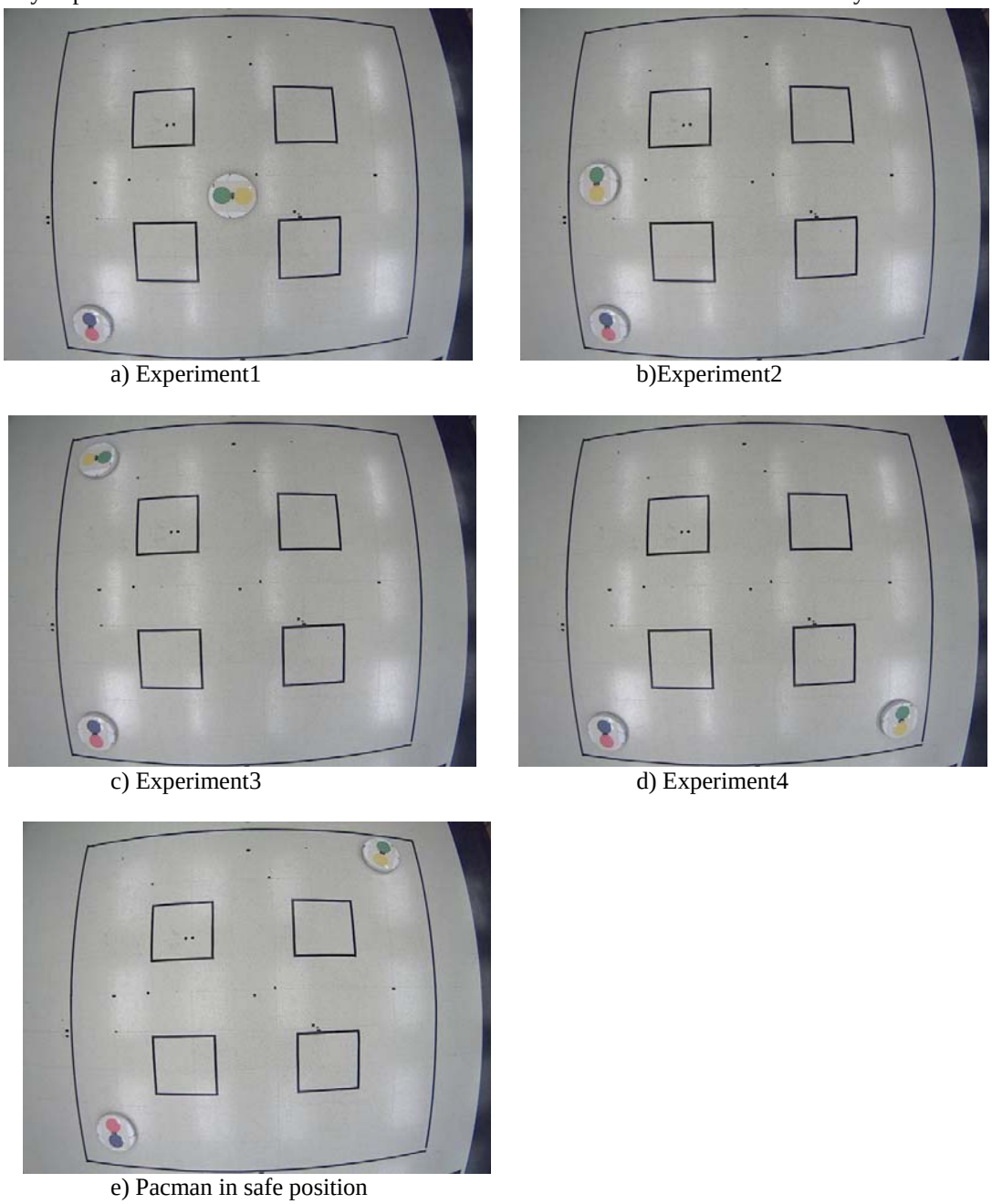
7.3. *Pacman was moving, Ghost was static*

In this experiment, Ghost is held static, while Pacman is given freedom to reach the safest position from Ghost. The Ghost was held at different positions and Pacman was expected to reach the safest place at every attempt as shown in Figure 7. Each different positions of Ghost is referred as experiment #. The Table 2 data represents the # of moves for the Pacman to reach to the safest position from Ghost as shown in Figure 7. The experimental results coincide with the expected number of moves made by Pacman to reach the safest place. All different positional experiments were observed as successful attempts.

7.4. *Pacman and Ghost were moving*

The Pacman and Ghost initial positions were set as shown in Figure 8 and the experimental data were shown in Table 3. This particular experimental case provided incorrect results, which was not expected. The Ghost robot tries to move away from the Pacman upon reaching any of the corners, which was not the expected outcome using our algorithm. One of the possibilities for the deviation in robot movements were attributed to the radial distortion of the camera capture process. Radial distortion near the corners, provides positional error in the robots. The cell near the corner if offseted, refers to the wrong cell and Ghost tries to reach the incorrect cell. In addition, the two robots try to update its path on reaching a cell, hence near corners, incorrect update of attack and escape path were developed autonomously.

Consistently expected robot moves were obtained when either of the robots remained stationary. The stationary



a) Experiment1 b)Experiment2

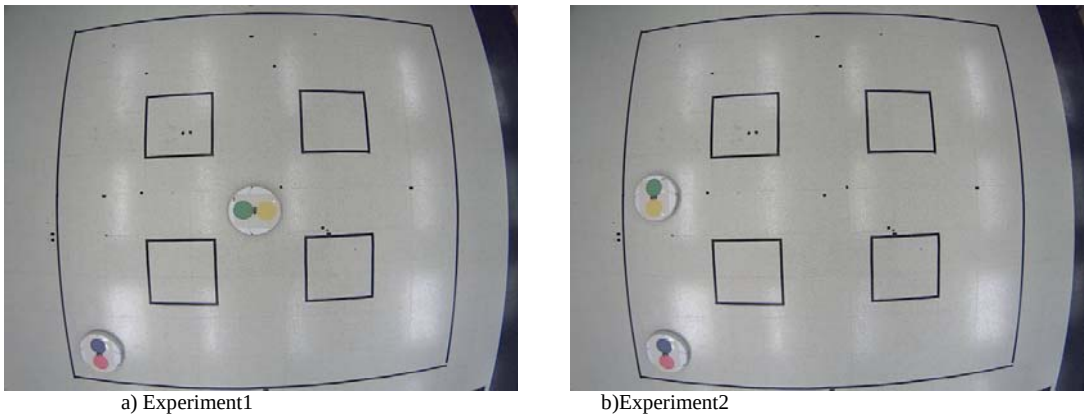
c) Experiment3 d) Experiment4

e) Pacman in safe position

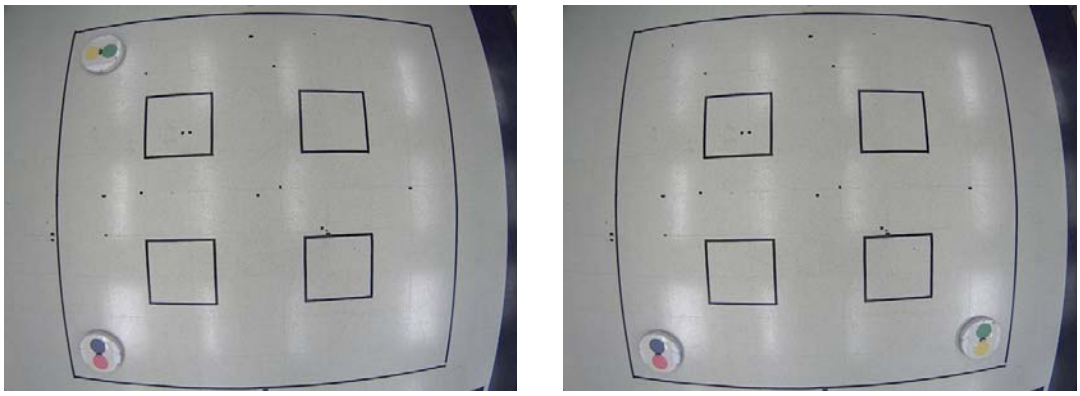
Fig.7 Camera view of different Pacman positions used in the experiments.

robots positioned at the corners provided successful results. The success of the stationary robot experiments were attributed to the initial identification and a fixed path planning of the moving robot. Readers are suggested to view the supplemental video file [Rao (Online)].

Radial distortion correction emphasized as Brown-Conrady model in [Zhang (2000)] is planned to be included to obtain correction in the camera view of the map as shown in Figure 9. The Figure shows the radial division of grids in the map. Using the Brown-Conrady model, near the corners of the map, the robot is expected to make radial displacement. However at the center of the map, robot follows a linear displacement. This should minimize the deviations obtained in the results, when both the robots are moving.



a) Experiment1 b)Experiment2



c) Experiment3 d) Experiment4

Fig.8 Camera view of different positions taken by Pacman and Ghost.

Table 2. Number of moves Pacman, takes to reach safe position.

Experiment #	# of moves	Comments
1	4	Successful
2	6	Successful
3	4	Successful
4	4	Successful

Table 3. Number of moves Ghost, takes to reach Pacman, while both are moving.

Experiment #	# of moves	Comments
1	Never ends	Incorrect
2	8	Successful
3	21	Incorrect
4	23	Incorrect

8. Results and Analysis

If either of the robots were static, the prototype model worked flawlessly. However, when both the robots were moving, the results seem to deviate from the expected moves. The following analysis was made in regard to this problem.

- Distorted position values near the corner of game-area: As stated previously, Brown-Conrady distortion model can minimize this error. Another modification includes change in the robot's drive algorithm, which can be tailored made to the game area. When reaching the corners of the game area, a fixed calibrated turn should move the robot towards the next cell. However, this requires thorough calibration and study of floor traction used in the game area.
- Socket communication: Shared memory instead of sockets as interprocess communication can be used in the future. Share memory approach would probably make the interprocess communication more efficient, and the overhead involved in creating the sockets would be avoided.

9. Conclusions and Future work

Despite having a few deviations in the results, a step towards realizing a game in robots was successfully implemented. This should encourage visualizing more games in robotic physical environment. Even though, simplistic set of rules were implemented, a modified version of the game can be easily integrated via algorithms. In the future, radial distortion needs to be covered and re evaluate the robotic movements. More complex rules needs to be added to the game in the physical robotic environment. Complex rules such as each cell position affecting the robots in some meaningful way is planned to be included in the future. Multiple Ghost robots is planned to be integrated in the environment, thereby an enhanced version of escape planning algorithm needs to be implemented. The realization of game in physical and simulation (Player/Stage) environment can be used to solve other real world problems such as detecting intrusions using game theory [Liang and Xiao (2010)]. The attempt described in the paper, forms a logical platform for others to realize more games using robots. Robot based games involves the development of various processes which will encourage students to learn the development of both software and hardware.

Acknowledgments

I would like to thank Mr. Ryan. Horton for his ideas on including blob detection as localization method. I would like to appreciate Dr. Monica. Anderson at the University of Alabama for allowing space and roomba robots to complete robotics experiments.

References

[1] Lund, H. H. (1999). Ai in children's play with lego robots, Proceedings of AAAI 1999 Spring Symposium Series, AAAI Press, Menlo.
[2] Anderson, M.; Thaete, L.; Wieg, N. (2007). Player/stage: A unifying paradigm to improve robotics education delivery.
[3] Chou, L-D.; Liu, T-C.; Li, D.; Chen, Y-S.; Leong, M. T.; Lee, P-H.; Lin, Y-C. (2011). Development of a game-based learning system using toy robots, Advanced Learning Technologies (ICALT), 11th IEEE International Conference, pp. 202–204.
[4] Lego mindstorms, http://mindstorms.lego.com, [Online].
[5] Josh, F.; Cheryl, S.; William, S. (2006). A video game-based mobile robot simulation environment, Intelligent Robots and Systems, IEEE/RSJ International Conference, October, pp. 3749 –3754.
[6] irobot, http://www.irobot.com, [Online].
[7] Gerkey, B. P.; Vaughan, R. T.; Howard, A. (2003). The player/stage project: Tools for multi-robot and distributed sensor systems, Proceedings of the 11th International Conference on Advanced Robotics, pp. 317–323.
[8] Borenstein, J.; Everett, H. R.; Feng, L.; Wehe, D. (1997). Mobile robot positioning sensors and techniques, Journal of Robotic Systems, vol. 14, no. 4, pp. 231–249.
[9] Wang, L.; Ju, H. (2008). A robust blob detection and delineation method. Education Technology and Training, International Workshop on Geoscience and Remote Sensing. ETT and GRS, vol. 1, December, pp. 827–830.
[10] Kefalea, E. (1998). Object localization and recognition for a grasping robot, Industrial Electronics Society, Proceedings of the 24th Annual Conference of the IEEE, vol. 4, September, pp. 2057–2062.
[11] Cowley, B.; Charles, D.; Black, M.; Hickey, R. (2009). Analyzing player behavior in pacman using feature-driven decision theoretic predictive modeling, Computational Intelligence and Games, IEEE Symposium, September, pp. 170 –177.
[12] Cowley, B.; Charles, D.; Black, M.; Hickey, R. (2007). Using decision theory for player analysis in pacman, Proceedings of SAB workshop on Adaptive Approaches to Optimizing player satisfaction, Italy, vol. 1, pp. 41–50.
[13] Cowley, B.; Charles, D.; Black, M.; Hickey, R. (2006). Data driven decision theory for player analysis in pacman, Proceedings of the optimizing player satisfaction Workshop, Stanford University, Stanford, CA: AAAI Press, vol. 1, pp. 25–30.
[14] Butler, Z.; Byrnes, S.; Rus, D.; (2001). Distributed motion planning for modular robots with unit-compressible modules, Intelligent Robots and Systems, Proceedings. IEEE/RSJ International Conference, vol. 2, pp. 790 –796.
[15] Wang, L.; Ju, H.; (2008). A robust blob detection and delineation method, Education Technology and Training, International Workshop on Geoscience and Remote Sensing. ETT and GRS. International Workshop, vol. 1, December, pp. 827 –830.
[16] Cimg, http://cimg.sourceforge.net, [Online].
[17] Katie, S.; Zimmerman, E. (2003). Rules of play: Game design fundamentals, The MIT Press., vol. 1, no. 1.
[18] Zhang, Z.; (2000). A flexible new technique for camera calibration, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, no. 11, pp. 1330 – 4.
[19] Liang, X.; Xiao, Y.; (2010). Studying bio-inspired coalition formation of robots for detecting intrusions using game theory, Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions, vol. 40, no. 3, pp. 683 –693.
[20] Rao, M.; http://unix.eng.ua.edu/~mrrao/pacman, [Online].