

Foundations of an Interactive Game-Playing Robot

Darren James Earl

January 7, 2009

Abstract

Interactive Game playing requires the integration of several complex components. The robot must remain safe and leave the environment undisturbed, requiring obstacle avoidance. Obstacle avoidance requires fast sensors such as sonars to be integrated. Objects need to be identified necessitating some form of vision. Since vision is used and is comparatively slow, an internal world module must be created and maintained. In order to facilitate realtime performance, tasks need to be performed concurrently.

1 Introduction

The field of Social Robotics, robots must interact meaningfully with their human users. To this end, the robots have often assumed roles as guides, advisers, or coaches [3, 4]. In these roles, the robots have not physically interacted with their users but instead have simply communicated with them and utilized their embodiment to establish a stronger presence. Even in implementations where the robot participates in a physical game such as tag, the robot is reduced simply to providing meaningful commentary but no actions. On the other end of the spectrum, of which there are significantly fewer examples, a robot can take on the role of a toy [5]. Particularly in the case of Roball, the robot responded and adapted to the physical world, but only meaningfully controlled and manipulated its own movement. Additionally, these implementations are often of both hardware and software in nature. As such, the specific solution to the goal in these implementations tends to be highly specialized, relying on unique physical capa-

bilities and characteristics of the robot, and of little generalized value. This paper presents a foundation which a robot of standard configuration (with no actuators save those needed for locomotion) with modest sensor and computational capabilities can interact with the environment and participate in simple physical games. The paper focuses on simplest incarnation of these games in which a ball is simply passed back to a user.

2 Problem Description

Only three objects are explicitly important: the target, the ball, and the robot. The robot should robustly maneuver itself such that it hits the ball toward the target. While the environment should be clear enough not to prohibit the robot from lining up with the ball and target, the robot should also remain safely away from the target and avoid any other miscellaneous obstacles or boundaries. The sensors are restricted to odometry, a ring of range sensors, and one camera. Computation is limited to a consumer level laptop.

3 Setup

In this implementation, the robot used is the Pioneer 2. The robot is built with a ring 16 sonar detectors with a range of 5 meters. The camera is an off-the-shelf entry level color webcam capable of 10 frames per second at a resolution of 320 by 240. The camera has a severely restricted viewing angle of $.2\text{PI}$ radians or equivalently 36 degrees. The laptop has a single core 1.8 Hz processor. The robot is controlled via

the robot control software Player. This provides an interface for issuing movement commands in the form of forward and angular velocity.

4 Approach

The robot adheres to a behavior-based paradigm in the form of schemas, a methodology first developed by Arkin [1]. As such, independent behaviors evaluate a subset of filtered sensory data and output a desired vector of movement. These vectors are then combined via vector summation to produce a final vector of movement. This final vector determines the velocity and angular velocity of the robot until the next iteration of the control loop. A finite state machine selects which behaviors are active at a given iteration of the control loop and places restrictions on the final values of the velocity and angular velocity. The sonar array data refreshes at a rate faster than the control loop, always reflects the desired information (occupancy in the neighborhood surrounding the robot), and therefore does not need explicit modeling. The positions of the target and the ball are only intermittently observable by the robot and the robot is required to navigate to a position relative to their positions, therefore the positions require modeling. Furthermore, the image processing runs significantly slower than the control loop, and in order to achieve real-time reactivity in the robot, must be run concurrently. Consequently, the control loop does not have direct access to camera's measurements and must rely on the modeled locations and cues as to when they were last updated.

4.1 Representation

Range data is intrinsically encoded in polar coordinates—radius and angle. In the case of sonar sensors, the radius is the measurement retrieved and the angle is the sensor's physical orientation on the robot. In the case of the camera, the horizontal position of an object in the field of view in addition to the physical orientation of the camera determine the angle measurement. The radius can be determined by the size of the object with respect to some previously

held measure. By using polar coordinates instead of Cartesian coordinates, errors can be evaluated specifically as variance in either the angle or the distance as opposed to some non-linear combination of the two. However, special care must be exercised concerning the angles as they are not part of the continuous number line but rather are described by a mathematical ring of the interval $[-\pi, \pi)$. Thus the value should not ever be greater than π nor less than $-\pi$ and the perceived boundary of π and $-\pi$ is an artifact of representation. In actuality the function from near $-\pi$ to near π and vice versa is continuously defined. Additionally changes in angle should not be greater in magnitude than π . If for instance the counter-clockwise magnitude of the change in angle is larger than π , the equivalent clockwise rotation would be of magnitude less than π . The robot only models what can be directly observed, which in this case is the relative location of the objects.

4.2 Vision

The robot utilizes a greedy graph-based vision algorithm developed by Pedro F. Felzenszwalb [2]. The algorithm is able to segment an image into semantically meaningful regions on the basis of local criteria and internal variability in regions. The algorithm runs in $n * \log(n)$ complexity, n being the number of pixels. A 320 by 240 image can be segmented in under a tenth of a second. The algorithm does significantly better in segmenting an image than simple thresholding and is robust in respect to gradations which can often be caused by shadows.

4.2.1 Ball Detection and Measurement

In a segmented image a component can be determined to be a ball using simple heuristics. The Average color of the region is an effective determinate. For a greenish ball, a good heuristic is simply the sum of the differences between the green and red values and the green and blue values. For a blue ball, the analogous heuristic is the difference between the blue and green values and the blue and red values. To further identify the ball a heuristic describing the number of pixels in the region divided by the circular area

defined by radius of the region can be used. These two heuristics are multiplied together and a simple thresholding on the value of the product determines if there is a round shape of the desired color. To rule out some troublesome false positives created by specularization on the ball, only regions with more than 1000 pixels are accepted. The distance to the robot can be easily calculated as a ratio between a previously observed value of the radius of the ball at one meter and the current radius. The angle at which the ball resides can be calculated as the product of the normalized offset from the center of the image to and widest viewing angle.

4.3 Concurrency

The image segmentation and object detection while fast (approximately 10Hz) is not on the same time scale as used or needed by the control loop (approximately 50 Hz). By using threads, image analysis can be executed concurrently, enabling the robot to progress and avoid obstacles without waiting for the vision sensory input. However, in addition to the well known problems of parallel programming (which largely can be mitigated through the use of mutexes), the sensory information gleaned from the visual processing could be outdated. The sensory information reflects the relative position of the ball and target at the time the frame was captured. By the time the frame is processed, the robot could have moved significantly. Consequently, the measurements need to be adjusted by the change in pose of the robot from when the frame was captured to when the frame has finished processing. Player provides the position of the robot in Cartesian coordinates and rotation relative to the starting position. While odometry tends to diverge from the true location after time, for short intervals it can be fairly accurate. By recording the pose of the robot before the frame is captured and comparing that to the pose after the frame has been processed, the sensory information can be updated with respect to the movement of the robot.

Given a sensory measurement of distance d and angle $theta$, the pose of the robot before capturing the frame (x_k, y_k, yaw_k) , and the pose of the robot after the frame is processed $(x_{k+1}, y_{k+1}, yaw_{k+1})$, the

corrected measurements d_{final} and $theta_{final}$ can be computed as follows:

$$\begin{aligned}
 dx &= (x_{k+1} - x_k) * \cos(-yaw_k) - (y_{k+1} - y_k) * \sin(-yaw_k) \\
 dy &= (x_{k+1} - x_k) * \sin(-yaw_k) + (y_{k+1} - y_k) * \cos(-yaw_k) \\
 x_{new} &= d * \cos(theta) - dx \\
 y_{new} &= d * \sin(theta) - dy \\
 theta_{final} &= \arctan(y_{new}, x_{new}) - (yaw_{k+1} - yaw_k), \text{ where } -PI < theta_{final} < PI \\
 d_{final} &= \sqrt{x_{new}^2 + y_{new}^2}
 \end{aligned}$$

4.4 Application of Kalman Filters

The odometry, sonar detectors, and measurements from the camera all suffer from significant amounts of noise. Additionally the target and the ball are dynamic objects, and thus the robot needs to be able to adapt to new sensory information. By utilizing kalman filters, the variances in measurements can be mitigated. Each sonar sensor is evaluated independently, and the measurements are modeled by a single dimension filter. In addition to the sonar sensors, only four pieces of datum need to be modeled—the relative polar coordinates of the ball and the target. For each object, the radius and angle data are modeled independently with single dimension kalman filters.

4.5 Movement

Since the robot does not model the world, save for tracking the relative positions of the ball and target, the expected values of the sonar sensor filters cannot be intelligently updated after the robot moves. Instead the covariance of the filter is increased to reflect the uncertainty in the estimated value. This allows newer information to be more quickly integrated into the filter but still provides some protection from false readings. Conversely, the relative positions of the ball and target can intelligently be updated with regard to the robot's movement.

Let the $theta_k$ and d_k be the expected values for the angle and distance to an object before the robot moves. Let the pose of the robot before movement

to be (x_k, y_k, yaw_k) and the pose of the robot after movement be $(x_{k+1}, y_{k+1}, yaw_{k+1})$. The new expected values $theta_{k+1}$ and d_{k+1} can be calculated as follows:

$$\begin{aligned}
dx &= (x_{k+1} - x_k) * \cos(-yaw_k) - (y_{k+1} - y_k) * \sin(-yaw_k) \\
dy &= (x_{k+1} - x_k) * \sin(-yaw_k) + (y_{k+1} - y_k) * \cos(-yaw_k) \\
x_{new} &= d * \cos(theta) - dx \\
y_{new} &= d * \sin(theta) - dy \\
theta_{k+1} &= \arctan(y_{new}, x_{new}) - (yaw_{k+1} - yaw_k), \text{ where } -PI < theta_{k+1} < PI \\
d_{k+1} &= \sqrt{x_{new}^2 + y_{new}^2}
\end{aligned}$$

In addition to this correction, Gaussian noise with a variance equal to one tenth the change in the parameters is added.

4.6 Behaviors

Behaviors are encoded into schemas which describe desired movement given a subset of sensory or model data. Desired movement is simply a vector originating at the robot. Behaviors are combined by simple vector addition. To facilitate meaningful interactions among behaviors, all behavior vectors besides avoidance are limited to a magnitude no greater than one. Avoidance is designed provide minimal input until an object is detected within one meter, and then is able to override all other behaviors. This ensures that all behaviors can contribute to navigation in relatively open locations, but when object avoidance is needed it is able to assume command.

4.6.1 Avoidance

The avoidance behavior utilizes the sonar sensor filters such that the robot will be repelled from obstacles proportionally to their proximity. Each sonar sensor filter has an associated $theta$ value corresponding to its physical location on the robot. The repelling magnitude provided by a single value v is $-1/v$. To calculate the repelling magnitude of a single filter first by discretization the range $[0, 5]$ into n intervals i_1 to i_n . Then for each interval $i_k, 1 \leq k \leq n$ a repelling magnitude rm_k for a point in

the interval is calculated, and then the magnitude is multiplied by the probability of the interval p_k according to the associated Gaussian distribution. If any of the Gaussian distribution is located outside the range $[0, 5]$, the probability outside the range is simple added to the nearest defined interval. The repelling magnitude of the filter is the sum of all the products of $rm_k * p_k$. The repelling vector of a filter is simple the vector defined by the repelling magnitude of the filter and its $theta$ value.

4.6.2 Move Forward

To prevent the robot from stalling from lack of information, a small impetus for moving forward is provided. The output is a vector of magnitude .1 and orientation of 0.

4.6.3 Navigate

This behavior is used to position the robot in line with the ball and the target at a set distance away from the ball. In this implementation the desired point is one meter away from the ball and in line with the target. The magnitude is simply 1 but the angle $theta_{desired}$ is calculated as below.

Let the relative position of the target be $(r_t, theta_t)$ and the relative position of the ball be $(r_b, theta_b)$.

The respective Cartesian coordinates (x_t, y_t) are $(r_t * \cos(theta_t), r_t * \sin(theta_t))$ and (x_b, y_b) are $(r_b * \cos(theta_b), r_b * \sin(theta_b))$.

$$dx = x_b - x_t$$

$$dy = y_b - y_t$$

$$angle = \arctan(dy, dx)$$

$$r = \sqrt{dx^2 + dy^2}$$

$$x_{desired} = (r + 1) * \cos(angle) + x_t$$

$$y_{desired} = (r + 1) * \sin(angle) + y_t$$

$$theta_{desired} = \arctan(y_{desired}, x_{desired})$$

4.6.4 Ramming

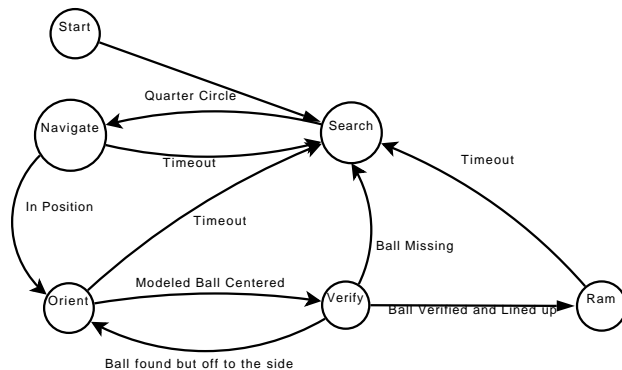
To effectively strike the ball, the robot needs to move at the center of the ball. The output vector is simply of magnitude 1 and points toward the ball.

4.6.5 Scanning

In order to gather information regarding the positions of the target and the ball, the robot sometimes needs to rotate in place. Scanning provides a vector with a modest magnitude and an angle to the left or right. This behavior is used in states where forward velocity is disallowed and thus only a controlled turn in place is enacted.

4.7 States

In order to effectively pass the ball towards the target, the robot must complete a sequence of steps to achieve correct positioning and gather data. A finite state machine is used to determine what behaviors are needed to be active and to organize when and how the robot achieves a step.



(a) Finite State Machine

4.7.1 Orient

The robot is not allowed forward velocity but only angular velocity. The robot rotates itself using the ram behavior and when the robot is relatively oriented on the ball the state transitions to the Verify State. Orient can time out to Search. If it transitions to Search, the variance on the position of the ball is increased.

4.7.2 Navigation

In Navigation, the robot utilizes the Move Forward, Avoidance, and Navigate behaviors. If the robot desires to turn faster than $\text{PI}/4$ radians per second, forward motion is stopped. If the robot detects that it is positioned within .1 meters of the desired position denoted by the Navigate behavior it transitions into the Orient state. If a time out occurs the robot transitions to the Search State.

4.7.3 Ram

The robot solely utilizes the ram behavior in an effort to strike the ball. After three seconds the state transitions to Search and the variance in the location of the ball and target is increased.

4.7.4 Search

The robot solely utilizes the scanning behavior and allows only angular velocity. The robot roughly completes a slow circle. After a full rotation, the state transitions to Navigation.

4.7.5 Verify

The robot does not utilize any control behaviors but instead waits for the camera to complete an analysis. If the ball is detected within the field of vision but not within a certain threshold, the state transitions to Orient. If the ball is detected within the field of vision and is within a small viewing angle in front of the robot, the state transitions to Ram. If the ball is not detected within the field of vision the state transitions to Search and the covariance of the ball's position is increased.

4.8 Simulation

Using Stage, the simulator packaged with Player, a toy world was constructed consisting of static obstacles and representations of the ball and target. An artificial camera process was created to mirror reflect several of the characteristics in the real world. In particular, the objects could only be detected if the were

within the same viewing angle of the physical camera, roughly $.2\pi$ radians. Secondly, the measurement data was delayed a tenth of a second, comparable to the real implementation, and then integrated into the system model using the odometry adjusted values. The agent quickly navigated to the desired position and successfully rammed the ball regardless of its starting location. The agent remained at a safe distance to all obstacles.

4.9 Physical

The robot was placed in a pen of roughly 6 by 6 meters with the ball targets. A green ball represented the ball to be passed and a smaller blue ball represented a user. In addition to the established delay caused by processing the image, there is an unmodeled latency in retrieving the camera data. If the latency is low, the robot successfully navigated into the correct position several times and struck the green ball in the correct direction. In contrast to the simulation, the speed that the robot traveled at was significantly slower and the robot did need to search for the balls several times. However, as latency increases the data from the camera degrades quickly—leading to large modeling errors. The robot reasons on this inaccurate data, causing erratic behavior and excessive wandering and searching. However, the object avoidance still functions and the robot can at least avoid damage to itself or the environment. The problem is mitigated by low forward and angular velocities, but high latencies quickly degrade performance below a real time timescale.

4.10 Conclusion

A robot using modest sensory capabilities, modest computation, and standard configuration can interact meaningfully with the environment. The limitations on the equipment prevented a more resounding success. In particular the field of view on the camera largely detracted from the ability of the robot to effectively track the ball. Coupled with insufficient computation abilities, the robot could only manipulate nearly static objects. A more robust object recognition would allow actual users to be detected

instead of simplistic balls. As large unmodeled latencies are introduced into image capturing, the modeling of data derived from vision degrades to unusable levels. The robot, acting on this model, acts erratically but at least maintains a base level of safety provided by obstacle avoidance.

5 References

References

- [1] Ron Arkin, “Motor Schema Based Navigation for a Mobile Robot: An Approach to Programming by Behavior” *Proceedings of the IEEE Conference on Robotics and Automation*, Raleigh, NC, pages 264-271, March 31 - April 3, 1987.
- [2] Pedro F. Felzenszwalb and Daniel P. Huttenlocher “Efficient Graph-Based Image Segmentation” *International Journal of Computer Vision*, Volume 59, Number 2, September 2004
- [3] T. Fong, Illah Nourbakhsh and Kerstin Dautenhahn “A Survey of Socially Interactive Robots” *Robotics and Autonomous Systems*, 42(3-4), pages 143-166, 2003.
- [4] Cory Kidd and Cynthia Breazeal “Robots at Home: Understanding Long-Term Human-Robot Interaction” IROS-082008
- [5] Francois Michaud and Serge Caron “Roball, the rolling robot” *Autonomous Robots*, 12(2), pages 211-222, 2002.