A Model-Based Approach to Optimizing *Ms. Pac-Man* Game Strategies in Real Time

Greg Foderaro, *Member, IEEE*, Ashleigh Swingler, *Member, IEEE*, and Silvia Ferrari, *Senior Member, IEEE*

***Abstract*—This paper presents a model-based approach for computing real-time optimal decision strategies in the pursuit-evasion game of *Ms. Pac-Man*. The game of *Ms. Pac-Man* is an excellent benchmark problem of pursuit-evasion game with multiple, active adversaries that adapt their pursuit policies based on *Ms. Pac-Man*'s state and decisions. In addition to evading the adversaries, the agent must pursue multiple fixed and moving targets in an obstacle-populated environment. This paper presents a novel approach by which a decision-tree representation of all possible strategies is derived from the maze geometry and the dynamic equations of the adversaries or ghosts. The proposed models of ghost dynamics and decisions are validated through extensive numerical simulations. During the game, the decision tree is updated and used to determine optimal strategies in real time based on state estimates and game predictions obtained iteratively over time. The results show that the artificial player obtained by this approach is able to achieve high game scores, and to handle high game levels in which the characters speeds and maze complexity become challenging even for human players.**

***Index Terms*—Cell decomposition, computer games, decision theory, decision trees, *Ms. Pac-Man*, optimal control, path planning, pursuit-evasion games.**

I. Introduction

THE video game *Ms. Pac-Man* is a challenging example of pursuit-evasion games in which an agent (Ms. Pac-Man) must evade multiple dynamic and active adversaries (ghosts), as well as pursue multiple fixed and moving targets (pills, fruits, and ghosts), all the while navigating an obstacle-populated environment. As such, it provides an excellent benchmark problem for a number applications including recognizance and surveillance [1], search-and-rescue [2], [3], and mobile robotics [4], [5]. In *Ms. Pac-Man*, each ghost implements a different decision policy with random seeds and multiple modalities that are a function of Ms. Pac-Man's decisions. Consequently, the game requires decisions to be made in real time, based on observations of a stochastic and dynamic environment that is challenging to both human and artificial players [6]. This is evidenced by the fact that, despite the recent series of artificial intelligence competitions inviting researchers to develop artificial players to achieve the highest possible score, existing artificial players have yet to achieve the performance level of expert human players [7]. For instance, existing artificial players typically achieve average scores between 9000 and 18 000 and maximum scores between 20 000 and 35 000 [8]–[13]. In particular, the highest score achieved at the last *Ms. Pac-Man* screen capture controller competition was 36 280, while expert human players routinely achieve scores over 65 000 and in some cases as high as 920 000 [14].

Recent studies in the neuroscience literature indicate that biological brains generate exploratory actions by comparing the meaning encoded in new sensory inputs with internal representations obtained from the sensory experience accumulated during a lifetime or preexisting functional maps [15]–[19]. For example, internal representations of the environment and of the subject's body (body schema), also referred to as internal models, appear to be used by the somatosensory cortex (SI) for predictions that are compared to the reafferent sensory input to inform the brain of sensory discrepancies evoked by environmental changes, and generate motor actions [20], [21]. Computational intelligence algorithms that exploit models built from prior experience or first principles have also been shown to be significantly more effective, in many cases, than those that rely solely on learning [22]–[24]. One reason is that many reinforcement learning algorithms improve upon the latest approximation of the policy and value function. Therefore, a model can be used to establish a better performance baseline. Another reason is that model-free learning algorithms need to explore the entire state and action spaces, thus requiring significantly more data and, in some cases, not scaling up to complex problems [25]–[27].

Artificial players for *Ms. Pac-Man* to date have been developed using model-free methods, primarily because of the lack of a mathematical model for the game components. One approach has been to design rule-based systems that implement conditional statements derived using expert knowledge [8]–[12], [28], [29]. While it has the advantage of being stable and computationally cheap, this approach lacks extensibility and cannot handle complex or unforeseen situations, such as, high game levels, or random ghosts behaviors. An influence map model was proposed in [30], in which the game characters and objects exert an influence on their surroundings. It was also shown in [31] that, in the *Ms. Pac-Man* game, Q-learning and fuzzy-state aggregation can be used to learn in nondeterministic environments. Genetic algorithms and Monte Carlo searches have also been successfully implemented in [32]–[35]

Manuscript received October 18, 2014; revised September 02, 2015; accepted January 23, 2016. This work was supported by the National Science Foundation under Grant ECS 0925407.

G. Foderaro was with the Mechanical Engineering Department, Duke University, Durham, NC 27708 USA. He is now with Applied Research Associates, Inc. (e-mail: greg.foderaro@duke.edu).

A. Swingler is with the Mechanical Engineering and Materials Science Department, Duke University, Durham, NC 27708 USA (e-mail: ashleigh.swingler@duke.edu).

S. Ferrari is with the Mechanical and Aerospace Engineering Department, Cornell University, Ithaca, NY 14853 USA (e-mail: ferrari@cornell.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCIAIG.2016.2523508

to develop high-scoring agents in the artificial intelligence competitions. Due to the complexity of the environment and adversary behaviors, however, model-free approaches have had difficulty handling the diverse range of situations encountered by the player throughout the game [36].

The model-based approach presented in this paper overcomes the limitations of existing methods [14], [37]–[39] by using a mathematical model of the game environment and adversary behaviors to predict future game states and ghost decisions. Exact cell decomposition is used to obtain a graphical representation of the obstacle-free configuration space for *Ms. Pac-Man* in the form of a connectivity graph that captures the adjacency relationships between obstacle-free convex cells. Using the approach first developed in [40] and [41], the connectivity graph can be used to generate a decision tree that includes action and utility nodes, where the utility function represents a tradeoff between the risk of losing the game (capture by a ghost) and the reward of increasing the game score. The utility nodes are estimated by modeling the ghosts' dynamics and decisions using ordinary differential equations (ODEs). The ODE models presented in this paper account for each ghost's personality and multiple modes of motion. Furthermore, as shown in this paper, the ghosts are active adversaries that implement adaptive policies, and plan their paths based on Ms. Pac-Man's actions.

Extensive numerical simulations demonstrate that the ghost models presented in this paper are able to predict the paths of the ghosts with an average accuracy of 94.6%. Furthermore, these models can be updated such that when a random behavior or error occurs, the dynamic model and corresponding decision tree can both be learned in real time. The game strategies obtained by this approach achieve better performance than beginner and intermediate human players, and are able to handle high game levels, in which the character speed and maze complexity become challenging even for human players. Because it can be generalized to more complex environments and dynamics, the model-based approach presented in this paper can be extended to real-world pursuit-evasion problems in which the agents and adversaries may consist of robots or autonomous vehicles, and motion models can be constructed from exteroceptive sensor data using, for example, graphical models, Markov decision processes, or Bayesian nonparametric models [2], [42]–[46].

The paper is organized as follows. Section II reviews the game of *Ms. Pac-Man*. The problem formulation and assumptions are described in Section III. The dynamic models of *Ms. Pac-Man* and the ghosts are presented in Sections IV and V, respectively. Section VI presents the model-based approach to developing an artificial *Ms. Pac-Man* player based on decision trees and utility theory. The game model and artificial player are demonstrated through extensive numerical simulations in Section VII.

II. The *Ms. Pac-Man* Game

Released in 1982 by Midway Games, *Ms. Pac-Man* is a popular video game that can be considered as a challenging benchmark problem for dynamic pursuit and evasion games. In the *Ms. Pac-Man* game, the player navigates a character named

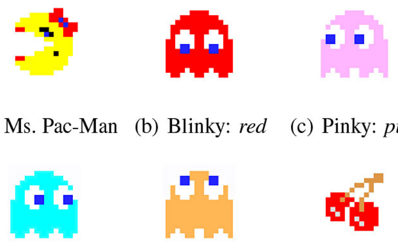
Fig. 1. Screen-capture of the *Ms. Pac-Man* game emulated on a computer.

Ms. Pac-Man through a maze with the goal of eating (traveling over) a set of fixed dots, called pills, as well as one or more moving objects (bonus items), referred to as fruits. The game image has the dimensions 224 × 288 pixels, which can be divided into a square grid of 8 × 8 pixel tiles, where each maze corridor consists of a row or a column of tiles. Each pill is located at the center of a tile and is eaten when Ms. Pac-Man is located within that tile [47].

Four ghosts, each with unique colors and behaviors, act as adversaries and pursue Ms. Pac-Man. If the player and a ghost move into the same tile, the ghost is said to capture Ms. Pac-Man, and the player loses one of three lives. The game ends when no lives remain. The ghosts begin the game inside a rectangular room in the center of the maze, referred to as the ghost pen, and are released into the maze at various times. If the player eats all of the pills in the maze, the level is cleared, and the player starts the process over, in a new maze, with incrementally faster adversaries.

Each maze contains a set of tunnels that allow Ms. Pac-Man to quickly travel to opposite sides of the maze. The ghosts can also move through the tunnels, but they do so at a reduced speed. The player is given a small advantage over ghosts when turning corners as well, where if a player controls Ms. Pac-Man to turn slightly before an upcoming corner, the distance Ms. Pac-Man must travel to turn the corner is reduced by up to approximately 2 pixels [47]. A player can also briefly reverse the characters' pursuit-evasion roles by eating one of four special large dots per maze referred to as power pills, which, for a short period of time, cause the ghosts to flee and give Ms. Pac-Man the ability to eat them [48]. Additional points are awarded when Ms. Pac-Man eats a bonus item. Bonus items enter the maze through a tunnel twice per level, and move slowly through the corridors of the maze. If they remain uneaten, the items exit the maze. A screenshot of the game is shown in Fig. 1, and the game characters are displayed in Fig. 2.

In addition to simply surviving and advancing through mazes, the objective of the player is to maximize the number of points earned, or score. During the game, points are awarded



(a) Ms. Pac-Man (b) Blinky: *red* (c) Pinky: *pink*

(d) Inky: *blue* (e) Sue: *orange* (f) Fruit: *cherry*

Fig. 2. Game characters and objects. (a) Ms. Pac-Man. (b) Blinky: *red*. (c) Pinky: *pink*. (d) Inky: *blue*. (e) Sue: *orange*. (f) Fruit: *cherry*.

when an object is eaten by Ms. Pac-Man. Pills are worth ten points each, a power pill gives 50 points, and the values of bonus items vary per level from 100 to 5000 points. When a power pill is active, the score obtained for capturing a ghost increases exponentially with the number of ghosts eaten in succession, where the total value is $\sum_{i=1}^{n} 100(2^n)$ and n is the number of ghosts eaten thus far. Therefore, a player can score 3000 points by eating all four ghosts during the duration of one power pill's effect. For most players, the game score is highly dependent on the points obtained for capturing ghosts. When Ms. Pac-Man reaches a score of 10 000, an extra life is awarded. In this paper, it is assumed that the player's objective is to maximize its game score and, thus, decision strategies are obtained by optimizing the score components, subject to a model of the game and ghost behaviors.

III. Problem Formulation and Assumptions

The *Ms. Pac-Man* player is viewed as a decision maker that seeks to maximize the final game score by a sequence of decisions based on the observed game state and predictions obtained from a game model. At any instant k, the player has access to all of the information displayed on the screen, because the state of the game $\mathbf{s}(k) \in \mathcal{X} \subset \mathbb{R}^n$ is fully observable and can be extracted without error from the screen capture. The time interval $(t_0, t_F]$ represents the entire duration of the game and, because the player is implemented using a digital computer, time is discretized and indexed by $k = 0, 1, \ldots, F$, where F is a finite end-time index that is unknown. Then, at any time $t_k \in (t_0, t_F]$, the player must make a decision $\mathbf{u}_M(k) \in \mathcal{U}(k)$ on the motion of Ms. Pac-Man, where $\mathcal{U}(k)$ is the space of admissible decisions at time t_k. Decisions are made according to a game strategy as follows.

Definition 3.1: A strategy is a class of admissible policies that consists of a sequence of functions

$$\sigma = \{\mathbf{c}_0, \mathbf{c}_1, \ldots\} \quad (1)$$

where $\mathbf{c}_k$ maps the state variables into an admissible decision

$$\mathbf{u}_M(k) = \mathbf{c}_k[\mathbf{s}(k)] \quad (2)$$

such that $\mathbf{c}_k[\cdot] \in \mathcal{U}(k)$, for all $\mathbf{s}(k) \in \mathcal{X}$.

In order to optimize the game score, the strategy σ is based on the expected profit of all possible future outcomes, which is estimated from a model of the game. In this paper, it is assumed that at several moments in time, indexed by t_i, the game can be modeled by a decision tree T_i that represents all possible decision outcomes over a time interval $[t_i, t_f] \subset (t_0, t_F]$, where $\Delta t = (t_f - t_i)$ is a constant chosen by the user. If the error between the predictions obtained by game model and the state observations exceed a specified tolerance, a new tree is generated, and the previous one is discarded. Then, at any time $t_k \in [t_i, t_f]$, the instantaneous profit can be modeled as a weighted sum of the reward V and the risk R and is a function of the present state and decision

$$\mathscr{L}[\mathbf{s}(k), \mathbf{u}_M(k)] = w_V V[\mathbf{x}(k), \mathbf{u}_M(k)] + w_R R[\mathbf{x}(k), \mathbf{u}_M(k)] \quad (3)$$

where w_V and w_R are weighting coefficients chosen by the user.

The decision-making problem considered in this paper is to determine a strategy $\sigma_i^* = \{\mathbf{c}_i^*, \ldots, \mathbf{c}_f^*\}$ that maximizes the cumulative profit over the time interval $[t_i, t_f]$

$$J_{i,f}\left[\mathbf{x}(i), \sigma_i\right] = \sum_{k=i}^{f} \mathscr{L}[\mathbf{x}(k), \mathbf{u}_M(k)] \quad (4)$$

such that, given T_i, the optimal total profit is

$$J_{i,f}^*\left[\mathbf{x}(i), \sigma_i^*\right] = \max_{\sigma_i} \left\{J_{i,f}\left[\mathbf{x}(i), \sigma_i\right]\right\}. \quad (5)$$

Because the random effects in the game are significant, any time the observed state $\mathbf{s}(k)$ significantly differs from the model prediction, the tree T_i is updated, and a new strategy σ_i^* is computed, as explained in Section IV-C. A methodology is presented in Sections III–VI to model the *Ms. Pac-Man* game and profit function based on guidelines and resources describing the behaviors of the characters, such as [49].

IV. Model of Ms. Pac-Man Behavior

In this paper, the game of *Ms. Pac-Man* is viewed as a pursuit-evasion game in which the goal is to determine the path or trajectory of an agent (Ms. Pac-Man) that must pursue fixed and moving targets in an obstacle-populated workspace, while avoiding capture by a team of mobile adversaries. The maze is considered to be a 2-D Euclidean workspace, denoted by $\mathcal{W} \subset \mathbb{R}^2$, that is populated by a set of obstacles (maze walls), $\mathcal{B}_1, \mathcal{B}_2, \ldots$, with geometries and positions that are constant and known *a priori*. The workspace $\mathcal{W}$ can be considered closed and bounded (compact) by viewing the tunnels, denoted by $\mathcal{T}$, as two horizontal corridors, each connected to both sides of the maze. Then, the obstacle-free space $\mathcal{W}_{\text{free}} = \mathcal{W} \backslash \{\mathcal{B}_1, \mathcal{B}_2, \ldots\}$ consists of all the corridors in the maze. Let $\mathcal{F}_\mathcal{W}$ denote an inertial reference frame embedded in $\mathcal{W}$ with origin at the lower left corner of the maze. In continuous time t, the state of Ms. Pac-Man is represented by a time-varying vector

$$\mathbf{x}_M(t) = [x_M(t)\ y_M(t)]^T \quad (6)$$

where x_M and y_M are the x, y-coordinates of the centroid of the Ms. Pac-Man character with respect to $\mathcal{F}_\mathcal{W}$, measured in units of pixels.

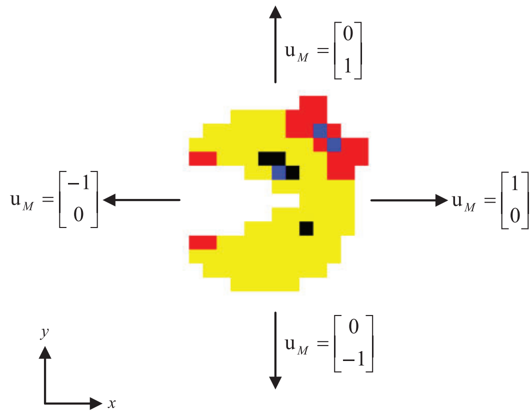


Fig. 3. Control vector sign conventions.

The control input for *Ms. Pac-Man* is a joystick, or keyboard, command from the player that defines a direction of motion for Ms. Pac-Man. As a result of the geometries of the game characters and the design of the mazes, the player is only able to select one of four basic control decisions (move up, move left, move down, or move right), and characters are restricted to two movement directions within a straight-walled corridor. The control input for Ms. Pac-Man is denoted by the vector

$$\mathbf{u}_M(t) = [u_M(t) v_M(t)]^T \tag{7}$$

where $u_M \in \{-1, 0, 1\}$ represents joystick commands in the x-direction and $v_M \in \{-1, 0, 1\}$ defines motion in the y-direction, as shown in Fig. 3. The control or action space, denoted by $\mathcal{U}$, for all agents is a discrete set

$$\mathcal{U} = [a_1, a_2, a_3, a_4] = \left\{ \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ -1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right\}. \tag{8}$$

Given the above definitions of state and control, it can be shown that Ms. Pac-Man's dynamics can be described by a linear, ordinary differential equation (ODE)

$$\dot{\mathbf{x}}_M(t) = \mathbf{A}(t)\mathbf{x}_M(t) + \mathbf{B}(t)\mathbf{u}_M(t) \tag{9}$$

where $\mathbf{A}$ and $\mathbf{B}$ are state–space matrices of appropriate dimensions [50].

In order to estimate Ms. Pac-Man's state, the ODE in (9) can be discretized, by integrating it with respect to time, using an integration step $\delta t << \Delta t = (t_f - t_i)$. The time index t_i represents all moments in time when a new decision tree is generated, i.e., the start of the game, the start of a new level, the start of game following the loss of one life, or the time when one of the actual ghosts' trajectories is found to deviate from the model prediction. Then, the dynamic equation for Ms. Pac-Man in discrete time can be written as

$$\mathbf{x}_M(k) = \mathbf{x}_M(k-1) + \alpha_M(k-1)\mathbf{u}_M(k-1)\delta t \tag{10}$$

where $\alpha_M(k)$ is the speed of Ms. Pac-Man at time k, which is subject to change based on the game conditions. The control input for the *Ms. Pac-Man* player developed in this paper is determined by a discrete-time state-feedback control law

$$\mathbf{u}_M(k) = c_k [\mathbf{x}_M(k)] \tag{11}$$

that is obtained using the methodology in Section VI, and may change over time.

The ghosts' dynamic equations are derived in Section V, in terms of state and control vectors

$$\mathbf{x}_G(k) = [x_G(k)\ y_G(k)]^T \tag{12}$$

$$\mathbf{u}_G(k) = [u_G(k)\ v_G(k)]^T \tag{13}$$

that are based on the same conventions used for *Ms. Pac-Man*, and are observed in real time from the game screen. The label G belongs to a set of unique identifiers $I_G = \{G | G \in \{R, B, P, O\}\}$, where R denotes the red ghost (Blinky), B denotes the blue ghost (Inky), P denotes the pink ghost (Pinky), and O denotes the orange ghost (Sue). Although an agent's representation occupies several pixels on the screen, its actual position is defined by a small 8 (pixel) × 8 (pixel) game tile, and capture occurs when these positions overlap. Letting $\tau[\mathbf{x}]$ represent the tile containing the pixel at position $\mathbf{x} = (x, y)$, capture occurs when

$$\tau[\mathbf{x}_M(k)] = \tau[\mathbf{x}_G(k)], \qquad \exists G \in I_G. \tag{14}$$

Because ghosts' behaviors include a pseudorandom component, the optimal control law for Ms. Pac-Man cannot be determined *a priori*, but must be updated based on real-time observations of the game [51]. Like any human player, the *Ms. Pac-Man* player developed by this paper is assumed to have full visibility of the information displayed on the game screen. Thus, a character state vector containing the positions of all game characters and of the bonus item $\mathbf{x}_F(k)$ at time k is defined as

$$\mathbf{x}(k) \triangleq \left[\mathbf{x}_M^T(k)\ \mathbf{x}_R^T(k)\ \mathbf{x}_B^T(k)\ \mathbf{x}_P^T(k)\ \mathbf{x}_O^T(k)\ \mathbf{x}_F^T(k)\right]^T \tag{15}$$

and can be assumed to be fully observable. Future game states can be altered by the player via the game control vector $\mathbf{u}_M(k)$. While the player can decide the direction of motion (Fig. 3), the speed of Ms. Pac-Man, $\alpha_M(k)$, is determined by the game based on the current game level, on the modes of the ghosts, and on whether Ms. Pac-Man is collecting pills. Furthermore, the speed is always bounded by a known constant ν, i.e., $\alpha_M(k) \le \nu$.

The ghosts are found to obey one of three modes that are represented by a discrete variable $\delta_G(k)$, namely pursuit mode $[\delta_G(k) = 0]$, evasion mode $[\delta_G(k) = 1]$, and scatter mode $[\delta_G(k) = -1]$. The modes of all four ghosts are grouped into a vector $\mathbf{m}(k) \triangleq [\delta_R(k)\ \delta_B(k)\ \delta_P(k)\ \delta_O(k)]^T$ that is used to determine, among other things, the speed of Ms. Pac-Man.

The distribution of pills (fixed targets) in the maze is represented by a 28×36 matrix $\mathbf{D}(k)$ defined over an 8 (pixel) × 8 (pixel) grid used to discretize the game screen into tiles. Then, the element in the ith row and $j$the column at time k, denoted by $\mathbf{D}_{(i,j)}(k)$, represents the presence of a pill (+1), power pill (−1), or an empty tile (0). Then, a function $n : \mathbb{R}^{28\times 36} \to \mathbb{R}$, defined as the sum of the absolute values of all elements of $\mathbf{D}(k)$, can be used to obtain the number of pills (including power pills) that are present in the maze at time k. For example, when Ms. Pac-Man is eating pills $n[\mathbf{D}(k)] < n[\mathbf{D}(k-1)]$, and when it is traveling in an empty corridor,

TABLE I
SPEED PARAMETERS FOR MS. PAC-MAN

Game Level	β_1	β_2	β_3	β_4
1	0.71	0.80	0.79	0.90
2 - 4	0.79	0.90	0.83	0.95
5 - 20	0.87	1.00	0.87	1.00
21+	0.79	0.90	-	-

$n[\mathbf{D}(k)] = n[\mathbf{D}(k-1)]$. Using this function, the speed of Ms. Pac-Man can be modeled as follows:

$$\alpha_M(k) = \begin{cases} \beta_1\nu, \text{ if} \quad \mathbf{m}(k) \not\ni 1 \text{ and } n\left[\mathbf{D}(k)\right] < n\left[\mathbf{D}(k-1)\right] \\ \beta_2\nu, \text{ if} \quad \mathbf{m}(k) \not\ni 1 \text{ and } n\left[\mathbf{D}(k)\right] = n\left[\mathbf{D}(k-1)\right] \\ \beta_3\nu, \text{ if} \quad \mathbf{m}(k) \ni 1 \text{ and } n\left[\mathbf{D}(k)\right] < n\left[\mathbf{D}(k-1)\right] \\ \beta_4\nu, \text{ if} \quad \mathbf{m}(k) \ni 1 \text{ and } n\left[\mathbf{D}(k)\right] = n\left[\mathbf{D}(k-1)\right] \end{cases} \tag{16}$$

where β_1, β_2, β_3, and β_4 are known parameters that vary with the game level, as shown in Table I.

All elements of the matrix $\mathbf{D}(k)$ and vector $\mathbf{m}(k)$ are rearranged into a vector $\mathbf{z}(k)$ that represents the game conditions, and is obtained in real time from the screen (Section VII). As a result, the state of the game $\mathbf{s}(k) = [\mathbf{x}^T(k) \quad \mathbf{z}^T(k)]^T$ is fully observable. Furthermore, $\mathbf{s}(k)$ determines the behaviors of the ghosts as explained in Section V.

V. MODELS OF ADVERSARY BEHAVIOR

The Ms. Pac-Man character is faced by a team of antagonistic adversaries, four ghosts, that try to capture Ms. Pac-Man and cause it to lose a life when successful. Because the game terminates after Ms. Pac-Man loses all lives, being captured by the ghosts prevents the player from increasing its game score. Evading the ghosts is, therefore, a key objective in the game of *Ms. Pac-Man*. The dynamics of each ghost, ascertained through experimentation and online resources [47], are modeled by a linear differential equation in the form:

$$\mathbf{x}_G(k) = \mathbf{x}_G(k-1) + \alpha_G(k-1)\mathbf{u}_G(k-1)\delta t \tag{17}$$

where the ghost speed α_G and control input $\mathbf{u}_G$ depend on the ghost personality (G) and mode, as explained in the following Q3 subsections. The pursuit mode is the most common and represents the behavior of the ghosts while actively attempting to capture Ms. Pac-Man. When in pursuit mode, each ghost uses Q4 a different control law as shown in the following subsections. When Ms. Pac-Man eats a power pill, the ghosts enter evasion mode and move slowly and randomly about the maze. The scatter mode only occurs during the first seven seconds of each level and at the start of gameplay following the death of Ms. Pac-Man. In scatter mode, the ghosts exhibit the same random motion as in evasion mode, but move at "normal" speeds.

A. Ghost Speed

The speeds of the ghosts depend on their personality, mode, and position. In particular, the speed of Inky, Pinky, and Sue

TABLE II
SPEED PARAMETERS FOR BLUE, PINK, AND ORANGE GHOSTS

Game Level	η_1 (evasion)	η_2 (pursuit)	η_3 (tunnel)
1	0.50	0.75	0.40
2 - 4	0.55	0.85	0.45
5 - 20	0.60	0.95	0.50
21+	-	0.95	0.50

TABLE III
SPEED PARAMETERS FOR RED GHOST

Level	d_1	η_4	d_2	η_5
1	-	-	-	-
2	30	0.90	15	0.90
3 - 4	40	0.90	20	0.95
5	40	1.00	20	1.05
6 - 8	50	1.00	25	1.05
9 - 11	60	1.00	30	1.05
12 - 14	80	1.00	40	1.05
15 - 18	100	1.00	50	1.05
19 - 21+	120	1.00	60	1.05

can be modeled in terms of the maximum speed of Ms. Pac-Man (ν), and in terms of the ghost mode and speed parameters (Table II) as follows:

$$\alpha_G(k) = \begin{cases} \eta_1\nu, \text{ if } \delta_G(k) = 1 \\ \eta_2\nu, \text{ if } \delta_G(k) \neq 1 \text{ and } \tau[\mathbf{x}_G(k)] \notin \mathcal{T} \\ \eta_3\nu, \text{ if } \delta_G(k) \neq 1 \text{ and } \tau[\mathbf{x}_G(k)] \in \mathcal{T} \end{cases} \tag{18}$$

where $G = B, P, O$. The parameter η_1 (Table II) scales the speed of a ghost in evasion mode. When ghosts are in scatter or pursuit mode, their speed is scaled by parameter η_2 or η_3, depending on whether they are outside or inside a tunnel $\mathcal{T}$, respectively. The ghost speeds decrease significantly when they are located in $\mathcal{T}$, accordingly, $\eta_2 > \eta_3$, as shown in Table II.

Unlike the other three ghosts, Blinky has a speed that depends on the number of pills in the maze $n[\mathbf{D}(k)]$. When the value of $n(\cdot)$ is below a threshold d_1, the speed of the red ghost increases according to parameter η_4, as shown in Table III. When the number of pills decreases further, below $n[\mathbf{D}(k)] < d_2$, Blinky's speed is scaled by a parameter $\eta_5 \geq \eta_4$ (Table III). The relationship between the game level, the speed scaling constants, and the number of pills in the maze is provided in lookup table form in Table III. Thus, Blinky's speed can be modeled as

$$\alpha_G(k) = \begin{cases} \eta_4\nu, & \text{if } n[\mathbf{D}(k)]| \leq d_1 \\ \eta_5\nu, & \text{if } n[\mathbf{D}(k)] \leq d_2 \end{cases}, \quad \text{for} \quad G = R \tag{19}$$

and Blinky is often referred to as the aggressive ghost.

B. Ghost Policy in Pursuit Mode

Each ghost utilizes a different strategy for chasing Ms. Pac-Man, based on its own definition of a target position denoted by $\mathbf{y}_G(k) \in \mathcal{W}$. In particular, the ghost control law greedily selects the control input that minimizes the Manhattan distance between the ghost and its target from a set of admissible control inputs, or action space, denoted by $\mathcal{U}_G(k)$. The ghost action space depends on the position of the ghost at time k, as well as the geometries of the maze walls, and is defined similarly to the action space of Ms. Pac-Man in (8). Thus, based on the distance between the ghost position $\mathbf{x}_G(k)$ and the target position $\mathbf{y}_G(k)$, every ghost implements the following control law to reach $\mathbf{y}_G(k)$:

$$\mathbf{u}_G(k) = \begin{cases} \mathbf{c} & \text{if } \mathbf{c} \in \mathcal{U}_G(k) \\ \mathbf{d} & \text{if } \mathbf{c} \notin \mathcal{U}_G(k), \mathbf{d} \in \mathcal{U}_G(k) \\ [0\ 1]^T & \text{if } \mathbf{c} \notin \mathcal{U}_G(k), \mathbf{d} \notin \mathcal{U}_G(k) \end{cases} \tag{20}$$

where

$$\mathbf{c} \triangleq H(\mathbf{C}) \circ \operatorname{sgn}[\boldsymbol{\xi}_G(k)] \tag{21}$$

$$\mathbf{d} \triangleq H(\mathbf{D}) \circ \operatorname{sgn}[\boldsymbol{\xi}_G(k)] \tag{22}$$

$$\mathbf{C} \triangleq \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} |\boldsymbol{\xi}_G(k)| \tag{23}$$

$$\mathbf{D} \triangleq \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix} |\boldsymbol{\xi}_G(k)| \tag{24}$$

$$\boldsymbol{\xi}_G(k) \triangleq [\mathbf{x}_G(k) - \mathbf{y}_G(k)]\,. \tag{25}$$

Symbol $^\circ$ denotes the Schur product, $H(\cdot)$ is the elementwise Heaviside step function defined such that $H(0) = 1$, $\operatorname{sgn}(\cdot)$ is the elementwise signum or sign function, and $|\cdot|$ is the elementwise absolute value.

In pursuit mode, the target position for Blinky, the red ghost (R), is the position of Ms. Pac-Man [47]

$$\mathbf{y}_R(k) = \mathbf{x}_M(k) \tag{26}$$

as shown in Fig. 4. As a result, the red ghost is most often seen following the path of Ms. Pac-Man. The orange ghost (O), Sue, is commonly referred to as the shy ghost, because it typically tries to maintain a moderate distance from Ms. Pac-Man. As shown in Fig. 5, when Ms. Pac-Man is within a threshold distance c_O of Sue, the ghost moves toward the lower left corner of the maze, with coordinates $(x, y) = (0, 0)$. However, if Ms. Pac-Man is farther than c_O from Sue, Sue's target becomes the position of Ms. Pac-Man, i.e., [47]

$$\mathbf{y}_O(k) = \begin{cases} [0\ 0]^T, & \text{if } \|\mathbf{x}_O(k) - \mathbf{x}_M(k)\|_2 \le c_O \\ \mathbf{x}_M(k), & \text{if } \|\mathbf{x}_O(k) - \mathbf{x}_M(k)\|_2 > c_O \end{cases} \tag{27}$$

where $c_O = 64$ pixels, and $\|\cdot\|_2$ denotes the L_2-norm.

Unlike Blinky and Sue, the pink ghost (P), Pinky, selects its target $\mathbf{y}_P$ based on both the position and the direction of motion of Ms. Pac-Man. In most instances, Pinky targets a position in $\mathcal{W}$ that is at a distance c_P from Ms. Pac-Man, and in the direction of Ms. Pac-Man's motion, as indicated by the value of the control input $\mathbf{u}_M$ (Fig. 6). However, when Ms. Pac-Man is moving in the positive y-direction (i.e., $\mathbf{u}_M(k) = a_1$), Pinky's target is c_P pixels above and to the left of *Ms. Pac-Man*. Therefore, Pinky's target can be modeled as follows [47]:

$$\mathbf{y}_P(k) = \mathbf{x}_M(k) + \mathbf{G}[\mathbf{u}_M(k)]\mathbf{c}_P \tag{28}$$

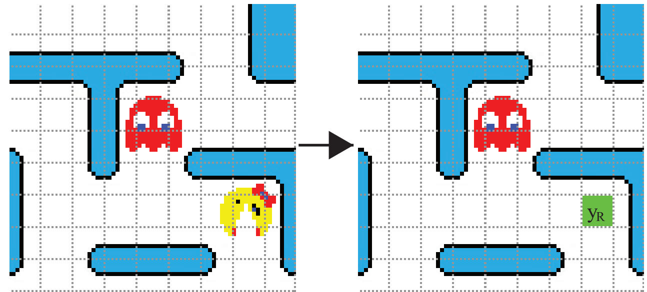


Fig. 4. Example of Blinky's target, $\mathbf{y}_R$. F4:1

where $\mathbf{c}_P = [32\ 32]^T$ pixels, and $\mathbf{G}(\cdot)$ is a matrix function of the control, defined as

$$\mathbf{G}(a_1) = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \quad \mathbf{G}(a_2) = \begin{bmatrix} -1 & 0 \\ 0 & 0 \end{bmatrix}$$
$$\mathbf{G}(a_3) = \begin{bmatrix} 0 & 0 \\ 0 & -1 \end{bmatrix} \quad \mathbf{G}(a_4) = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}. \tag{29}$$

The blue ghost (B), Inky, selects its target $\mathbf{y}_B$ based not only on the position and direction of motion of Ms. Pac-Man, but also on the position of the red ghost $\mathbf{x}_R$. As illustrated in Fig. 7, Inky's target is found by projecting the position of the red ghost in the direction of motion of Ms. Pac-Man ($\mathbf{u}_M$), about a point 16 pixels from $\mathbf{x}_M$, and in the direction $\mathbf{u}_M$. When Ms. Pac-Man is moving in the positive y-direction ($\mathbf{u}_M(k) = a_1$), however, the point for the projection is above and to the left of Ms. Pac-Man at a distance of 6 pixels. The reflection point can be defined as

$$\mathbf{y}_M^R(k) = \mathbf{x}_M(k) + \mathbf{G}[\mathbf{u}_M(k)]\mathbf{c}_B \tag{30}$$

where $\mathbf{c}_B = [16\ 16]^T$, and the matrix function $\mathbf{G}(\cdot)$ is defined as in (29). The position of the red ghost is then projected about the reflection point $\mathbf{y}_M^R$ in order to determine the target for the blue ghost [47]

$$\mathbf{y}_B(k) = 2 \cdot \mathbf{y}_M^R(k) - \mathbf{x}_R(k) \tag{31}$$

as shown by the examples in Fig. 7.

C. Ghost Policy in Evasion and Scatter Modes

At the beginning of each level and following the death of Ms. Pac-Man, the ghosts are in scatter mode for seven seconds. In this mode, the ghosts do not pursue the player but, rather, move about the maze randomly. When a ghost reaches an intersection, it is modeled to select one of its admissible control inputs $\mathcal{U}_G(k)$ with uniform probability (excluding the possibility of reversing direction).

If Ms. Pac-Man eats a power pill, the ghosts immediately reverse direction and enter the evasion mode for a period of time that decreases with the game level. In evasion mode, the ghosts move randomly about the maze as in scatter mode but with a lower speed. When a ghost in evasion mode is captured by Ms. Pac-Man, it returns to the ghost pen and enters pursuit mode on exit. Ghosts that are not captured return to pursuit mode when the power pill becomes inactive.

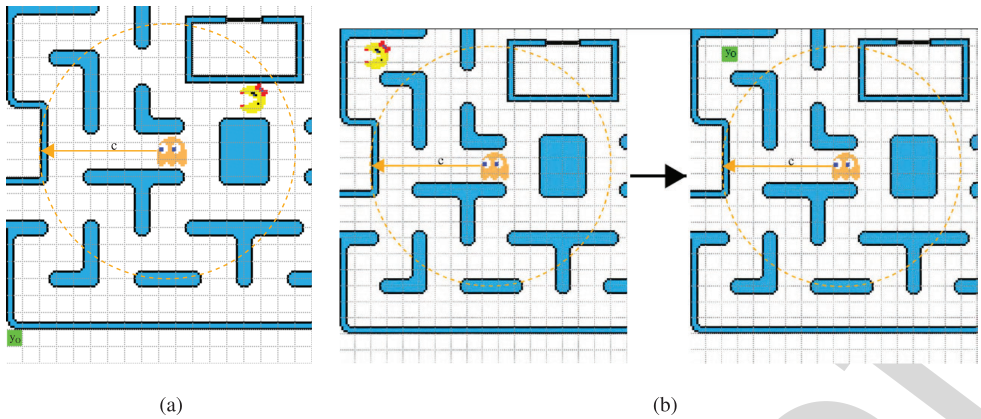


Fig. 5. Examples of Sue's target, $\mathbf{y}_O$. (a) $\|\mathbf{x}_O(k) - \mathbf{x}_M(k)\|_2 \leq c_O$. (b) $\|\mathbf{x}_O(k) - \mathbf{x}_M(k)\|_2 > c_O$.

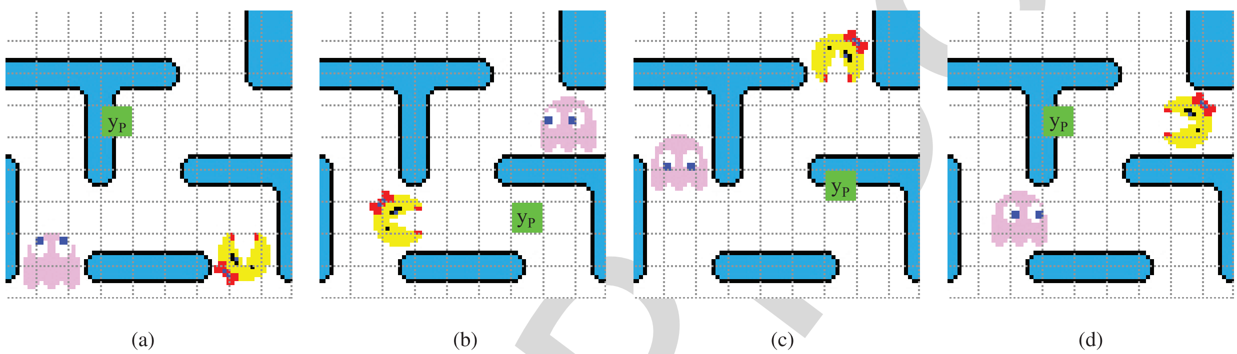


Fig. 6. Examples of Pinky's target, $\mathbf{y}_P$. (a) If $\mathbf{u}_M(k) = a_1$. (b) If $\mathbf{u}_M(k) = a_2$. (c) If $\mathbf{u}_M(k) = a_3$. (d) If $\mathbf{u}_M(k) = a_4$.

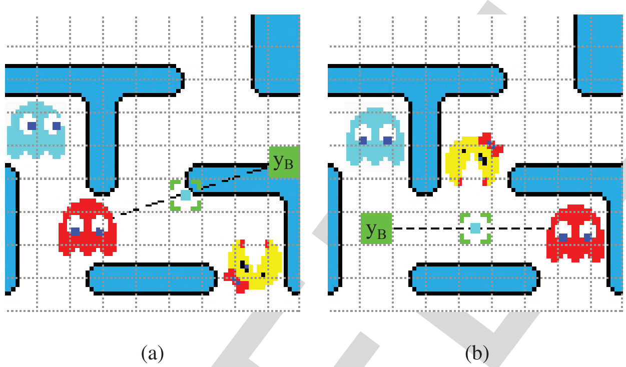


Fig. 7. Examples of Inky's target, $\mathbf{y}_B$. (a) If $\mathbf{u}_M(k) = a_1$. (b) If $\mathbf{u}_M(k) = a_3$.

VI. Methodology

This paper presents a methodology for optimizing the decision strategy of a computer player, referred to as the artificial *Ms. Pac-Man* player. A decision-tree representation of the game is obtained by using a computational geometry approach known as cell decomposition to decompose the obstacle-free workspace $\mathcal{W}_{\text{free}}$ into convex subsets, or cells, within which a path for Ms. Pac-Man can be easily generated [40]. As explained in Section VI-A, the cell decomposition is used to create a connectivity tree representing causal relationships between Ms. Pac-Man's position, and possible future paths [52]. The connectivity tree can then be transformed into a decision tree with utility nodes obtained from the utility function defined in Section VI-B. The optimal strategy for the artificial player is then computed and updated using the decision tree, as explained in Section VI-C.

A. Cell Decomposition and the Connectivity Tree

As a preliminary step, the corridors of the maze are decomposed into nonoverlapping rectangular cells by means of a line sweeping algorithm [53]. A cell, denoted κ_i, is defined as a closed and bounded subset of the obstacle-free space. The cell decomposition is such that a maze tunnel constitutes a single cell, as shown in Fig. 8. In the decomposition, two cells κ_i and κ_j are considered to be adjacent if and only if they share a mutual edge. The adjacency relationships of all cells in the workspace can be represented by a connectivity graph. A connectivity graph $\mathcal{G}$ is a nondirected graph, in which every node represents a cell in the decomposition of $\mathcal{W}_{\text{free}}$, and two nodes κ_i and κ_j are connected by an arc (κ_i, κ_j) if and only if the corresponding cells are adjacent.

Ms. Pac-Man can only move between adjacent cells, therefore, a causal relationship can be established from the adjacency relationships in the connectivity graph, and represented by a connectivity tree, as was first proposed in [52]. Let $\kappa[\mathbf{x}]$ denote the cell containing a point $\mathbf{x} = [xy]^T \in \mathcal{W}_{\text{free}}$. Given an initial position $\mathbf{x}_0$, and a corresponding cell $\kappa[\mathbf{x}_0]$, the connectivity tree associated with $\mathcal{G}$, and denoted by $\mathcal{C}$, is defined as an acyclic tree graph with root $\kappa[\mathbf{x}_0]$, in which every pair of nodes κ_i and κ_j connected by an arc are also connected by an arc

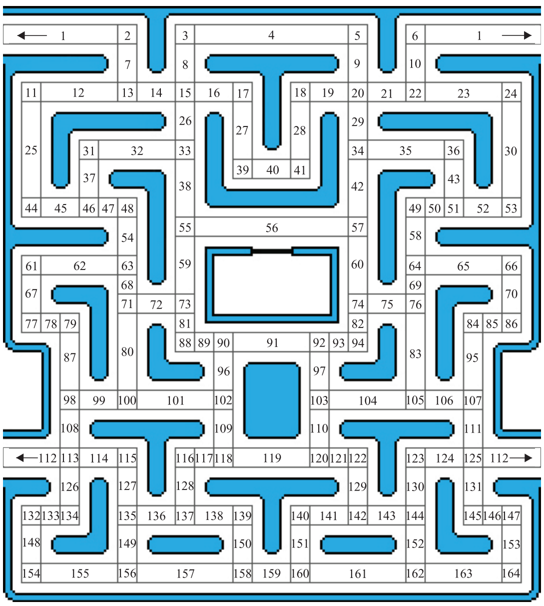


Fig. 8. Cell decomposition of *Ms. Pac-Man* second maze.

in $\mathcal{G}$. As in the connectivity graph, the nodes of a connectivity tree represent void cells in the decomposition. Given the position of Ms. Pac-Man at any time k, a connectivity tree with root $\kappa[\mathbf{x}_M(k)]$ can be readily determined from $\mathcal{G}$, using the methodology in [52]. Each branch of the tree then represents a unique sequence of cells that may be visited by *Ms. Pac-Man*, starting from $\mathbf{x}_M(k)$.

B. Ms. Pac-Man's Profit Function

Based on the game objectives described in Section II, the instantaneous profit of a decision $\mathbf{u}_M(k)$ is defined as a weighted sum of the risk of being captured by the ghosts, denoted by R, and the reward gained by reaching one of targets, denoted by V. Let $d(\cdot)$, $p(\cdot)$, $f(\cdot)$, and $b(\cdot)$ denote the rewards associated with reaching the pills, power pills, ghosts, and bonus items, respectively. The corresponding weights, ω_d, ω_p, ω_f, and ω_b denote known constants that are chosen heuristically by the user, or computed via a learning algorithm, such as temporal difference [39]. Then, the total reward can be defined as the sum of the rewards from each target type

$$\begin{aligned} V[\mathbf{s}(k), \mathbf{u}_M(k)] = \omega_d d[\mathbf{s}(k), \mathbf{u}_M(k)] + \omega_p p[\mathbf{s}(k), \mathbf{u}_M(k)] \\ + \omega_f f[\mathbf{s}(k), \mathbf{u}_M(k)] + \omega_b b[\mathbf{s}(k), \mathbf{u}_M(k)] \end{aligned} \tag{32}$$

and can be computed using the models presented in Section V, as follows.

The pill reward function $d(\cdot)$ is a binary function that represents a positive reward of 1 unit if Ms. Pac-Man is expected to eat a pill as result of the chosen control input $\mathbf{u}_M$, and is otherwise zero, i.e.,

$$d[\mathbf{x}(k), \mathbf{u}_M(k), \mathbf{z}(k)] = \begin{cases} 0, \text{ if } & \mathbf{D}[\mathbf{x}_M(k)] \neq 1 \\ 1, \text{ if } & \mathbf{D}[\mathbf{x}_M(k)] = 1. \end{cases} \tag{33}$$

A common strategy implemented by both human and artificial players is to use power pills to ambush the ghosts. When utilizing this strategy, a player waits near a power pill until the ghosts are near, it then eats the pill and pursues the ghosts which have entered evasion mode. The reward associated with each power pill can be modeled as a function of the minimum distance between Ms. Pac-Man and each ghost G

$$\rho_G[\mathbf{x}_M(k)] \triangleq \min |\mathbf{x}_M(k) - \mathbf{x}_G(k)| \tag{34}$$

where $|\cdot|$ denotes the L_1-norm. In order to take into account the presence of the obstacles (walls), the minimum distance in (34) is computed from the connectivity tree $\mathcal{C}$ obtained in Section VI-A, using the A * algorithm [53]. Then, letting ρ_D denote the maximum distance at which Ms. Pac-Man should eat a power pill, the power-pill reward can be written as

$$p[\mathbf{x}(k), \mathbf{u}_M(k), \mathbf{z}(k)] = \begin{cases} 0, & \text{if } \mathbf{D}[\mathbf{x}_M(k)] \neq -1 \\ \sum\limits_{G \in I_G} g[\mathbf{x}(k)], & \text{if } \mathbf{D}[\mathbf{x}_M(k)] = -1 \end{cases} \tag{35}$$

where

$$\begin{aligned} g[\mathbf{x}_M(k), \mathbf{x}_G(k)] = \vartheta_- \times H\{\rho_G[\mathbf{x}_M(k)] - \rho_D\} \\ + \vartheta_+ \times H\{\rho_D - \rho_G[\mathbf{x}_M(k)]\}. \end{aligned} \tag{36}$$

The parameters ϑ_- and ϑ_+ are the weights that represent the desired tradeoff between the penalty and reward associated with the power pill.

Because the set of admissible decisions for a ghost is a function of its position in the maze, the probability that a ghost in evasion mode will transition to a state $\mathbf{x}_G(k)$ from a state $\mathbf{x}_G(k-1)$, denoted by $P[\mathbf{x}_G(k)|\mathbf{x}_G(k-1)]$, can be computed from the cell decomposition (Fig. 8). Then, the instantaneous reward for reaching (eating) a ghost G in evasion mode is

$$\begin{aligned} &f[\mathbf{x}(k), \mathbf{u}_M(k), \mathbf{z}(k)] \\ &= \begin{cases} 0, & \text{if } \mathbf{x}_G(k) \neq \mathbf{x}_M(k) H[\delta_G(k) - 1] \\ P[\mathbf{x}_G(k)|\mathbf{x}_G(k-1)]\zeta(k), & \text{if } \mathbf{x}_G(k) = \mathbf{x}_M(k) \end{cases} \end{aligned} \tag{37}$$

where $\delta_G(k)$ represents the mode of motion for ghost G (Section IV), and the function

$$\zeta(k) = \left\{ 5 - \sum_{G \in I_G} H[\delta_G(k) - 1] \right\}^2 \tag{38}$$

is used to increase the reward quadratically with the number of ghosts reached.

Like the ghosts, the bonus items are moving targets that, when eaten, increase the game score. Unlike the ghosts, however, they never pursue Ms. Pac-Man, and, if uneaten after a given period of time they simply leave the maze. Therefore, at any time during the game, an attractive potential function

$$U_b(\mathbf{x}) = \begin{cases} \rho_F^2(\mathbf{x}), & \text{if } \rho_F(\mathbf{x}) \leq \rho_b \\ 0, & \text{if } \rho_F(\mathbf{x}) > \rho_b \end{cases}, \quad \mathbf{x} \in \mathcal{W}_{\text{free}} \tag{39}$$

can be used to pull Ms. Pac-Man toward the bonus item with a virtual force

$$F_b(\mathbf{x}) = -\nabla U_b(\mathbf{x}) \tag{40}$$

that decreases with ρ_F. The distance ρ_F is defined by substituting G with F in (34), ρ_b is a positive constant that represents the influence distance of the bonus item [53], and ∇ is the gradient operator. The instantaneous reward function for the bonus item is then defined such that the player is rewarded for moving toward the bonus item, i.e.,

$$b\left[\mathbf{x}(k), \mathbf{u}_M(k), \mathbf{z}(k)\right] = \operatorname{sgn}\left\{F_b\left[\mathbf{x}_M(k)\right]\right\} \circ \mathbf{u}_M(k). \quad (41)$$

The weight ω_b in (32) is then chosen based on the type and value of the bonus item for the given game level.

The instantaneous risk function is defined as the sum of the immediate risk posed by each of the four ghosts

$$R\left[\mathbf{x}(k), \mathbf{u}_M(k), \mathbf{z}(k)\right] = \sum_{G \in I_G} R_G\left[\mathbf{x}(k), \mathbf{u}_M(k), \mathbf{z}(k)\right] \quad (42)$$

where the risk of each ghost R_G depends on its mode of motion. In evasion mode ($\delta_G = 1$), a ghost G poses no risk to Ms. Pac-Man, because it cannot capture her. In scatter mode ($\delta_G = 0$), the risk associated with a ghost G is modeled using a repulsive potential function

$$U_G(\mathbf{x}) = \begin{cases} \left(\frac{1}{\rho_G(\mathbf{x})} - \frac{1}{\rho_0}\right)^2, & \text{if} \quad \rho_G(\mathbf{x}) \leq \rho_0 \\ 0, & \text{if} \quad \rho_G(\mathbf{x}) > \rho_0 \end{cases}, \quad \mathbf{x} \in \mathcal{W}_{\text{free}} \quad (43)$$

that repels Ms. Pac-Man with a force

$$F_G(\mathbf{x}) = -\nabla U_G(\mathbf{x}) \quad (44)$$

ρ_0 is the influence distance of Ms. Pac-Man, such that when Ms. Pac-Man is farther than ρ_0 from a ghost, the ghost poses zero risk. When a ghost is in the ghost pen or otherwise inactive, its distance to Ms. Pac-Man is treated as infinite.

The risk of a ghost in scatter mode is modeled such that Ms. Pac-Man is penalized for moving toward the ghost, i.e.,

$$R_G\left[\mathbf{x}(k), \mathbf{u}_M(k), \mathbf{z}(k)\right] = \operatorname{sgn}\left\{F_G[\mathbf{x}_M(k)]\right\} \circ \mathbf{u}_M(k) \quad (45)$$

for $\delta_G(k) = -1$. In pursuit mode [$\delta_G(k) = 0$], the ghosts are more aggressive and, thus, the instantaneous risk is modeled as the repulsive potential

$$R_G\left[\mathbf{x}(k), \mathbf{u}_M(k), \mathbf{z}(k)\right] = U_G(\mathbf{x}). \quad (46)$$

Finally, the risk of being captured by a ghost is equal to a large positive constant χ defined by the user

$$R_G\left[\mathbf{x}(k), \mathbf{u}_M(k), \mathbf{z}(k)\right] = \chi, \quad \text{for} \quad \tau[\mathbf{x}_M(k)] = \tau[\mathbf{x}_G(k)]. \quad (47)$$

This emphasizes the risk of losing a life, which would cause the game to end sooner and the score to be significantly lower.

Then the instantaneous profit function is a sum of the reward V and risk R

$$J[\mathbf{u}_M(k)] = V[\mathbf{s}(k), \mathbf{u}_M(k)] + R[\mathbf{x}(k), \mathbf{u}_M(k), \mathbf{z}(k)] \quad (48)$$

which is evaluated at each node in a decision tree constructed using the cell decomposition method described above.

C. *Decision Tree and Optimal Strategy*

As was first shown in [52], the connectivity tree $\mathcal{G}$ obtained via cell decomposition in Section VI-A can be transformed into a decision tree T_i that also includes action and utility nodes. A decision tree is a directed acyclic graph with a tree-like structure in which the root is the initial state, decision nodes represent all possible decisions, and state (or chance) nodes represent the state values resulting from each possible decision [54]–[56]. Each branch in the tree represents the outcomes of a possible strategy σ_i and terminates in leaf (or utility) node that contains the value of the strategy's cumulative profit $J_{i,f}$.

Let the tuple $T_i = \{C, D, J, A\}$ represent a decision tree comprising a set of chance nodes C, a set of decision nodes D, the utility function J, and a set of directed arcs A. At any time $t_i \in (t_0, t_F]$, a decision tree T_i for Ms. Pac-Man can be obtained from $\mathcal{G}$ using the following assignments.

1) The root is the cell $\kappa_i \in \mathcal{G}$ occupied by Ms. Pac-Man at time t_i.
2) Every chance node $\kappa_j \in C$ represents a cell in $\mathcal{G}$.
3) For every cell $\kappa_j \in C$, a directed arc $(\kappa_j, \kappa_l) \in A$ is added iff $\exists(\kappa_j, \kappa_l) \in \mathcal{G}$, $j \neq l$. Then, (κ_j, κ_l) represents the action decision to move from κ_j to κ_l.
4) The utility node at the end of each branch represents the cumulative profit $J_{i,f}$ of the corresponding strategy, σ_i, defined in (4).

Using the above assignments, the instantaneous profit can be computed for each node as the branches of the tree are grown using Ms. Pac-Man's profit function, presented in Section VI-B. When the slice corresponding to t_f is reached, the cumulative profit $J_{i,f}$ of each branch is found and assigned to its utility node. Because the state of the game can change suddenly as result of random ghost behavior, an exponential discount factor is used to discount future profits in $J_{i,f}$, and favor the profit that may be earned in the near future. From T_i, the optimal strategy σ_i^* is determined by choosing the action corresponding to the branch with the highest value of $J_{i,f}$. As explained in Section III, a new decision tree is generated when t_f is reached, or when the state observations differ from the model prediction, whichever occurs first.

VII. Simulation Results

The simulation results presented in this paper are obtained from the Microsoft's *Revenge of the Arcade* software, which is identical to the original arcade version of *Ms. Pac-Man*. The results in Section VII-A validate the ghost models presented in Section V, and the simulations in Section VII-B demonstrate the effectiveness of the model-based artificial player presented in Section VI. Every game simulated in this section is played from beginning to end. The artificial player is coded in C#, and runs in real time on a laptop with a Core-2 Duo 2.13-GHz CPU, and 8-GB RAM. At every instant, indexed by k, the state of the game $\mathbf{s}(k)$ is extracted from screen-capture images of the game using the algorithm presented in [41]. Based on the observed state value $\mathbf{s}(k)$, the control input to Ms. Pac-Man $\mathbf{u}_M$ is computed from the decision tree T_i, and implemented using simulated keystrokes. Based on $\mathbf{s}(k)$, the tree T_i is updated at

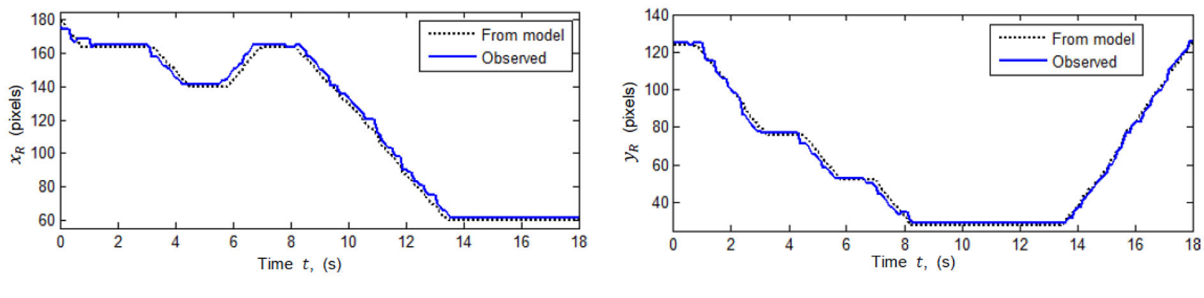


Fig. 9. Example of simulated and observed trajectories for the red ghost in pursuit mode.

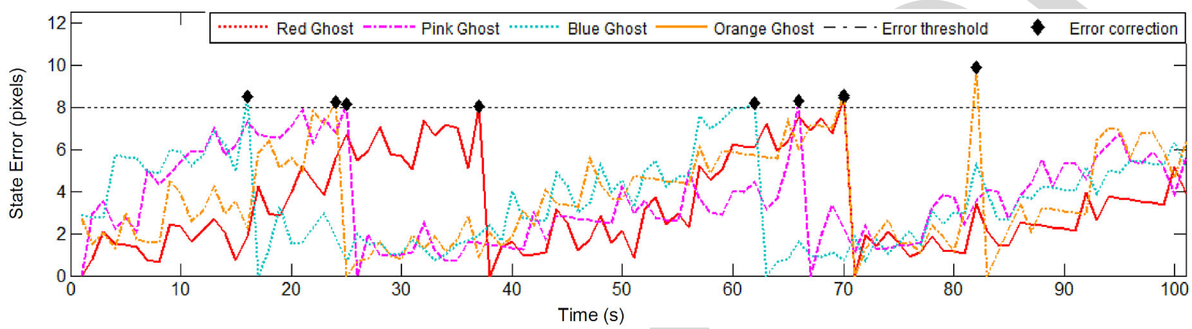


Fig. 10. Example of ghost-state error histories, and model updates (diamonds).

selected instants $t_i \in (t_0, t_f]$, as explained in Section VI-C. The highest recorded time to compute a decision was 0.09 s, and the mean times for the two most expensive steps of extracting the game state and computing the decision tree are on the order of 0.015 and 0.05 s, respectively.

A. Adversary Model Validation

The models of the ghosts in pursuit mode, presented in Section V-B, are validated by comparing the trajectories of the ghosts extracted from the screen capture code to those generated by integrating the models numerically using the same game conditions. When the ghosts are in other modes, their random decisions are assumed to be uniformly distributed [47]. The ghosts' state histories are extracted from screen-capture images while the game is being played by a human player. Subsequently, the ghost models are integrated using the trajectory of *Ms. Pac-Man* extracted during the same time interval. Fig. 9 shows an illustrative example of actual (solid line) and simulated (dashed line) trajectories for the red ghost, in which the model generated a path identical to that observed from the game. The small error between the two trajectories, in this case, is due entirely to the screen-capture algorithm.

The ghosts' models are validated by computing the percentage of ghost states that are predicted correctly during simulated games. Because the ghosts only make decisions at maze intersections, the error in a ghost's state is computed every time the ghost is at a distance of 10 pixels from an intersection. Then, the state is considered to be predicted correctly if the error between the observed and predicted values of the state is less than 8 pixels. If the error is larger than 8 pixels, the prediction is considered to be incorrect. When an incorrect prediction occurs, the simulated ghost state $\mathbf{x}_G$ is updated online using the observed state value as an initial condition in the ghost dynamic equation (17). Fig. 10 shows the error between simulated and observed state histories for all four ghosts during a sample time interval.

TABLE IV
GHOST MODEL VALIDATION RESULTS

Ghost	No. Correct Decisions	No. Incorrect Decisions	Model Accuracy
Red	21072	630	97.1%
Pink	19500	718	96.45%
Blue	19499	759	96.25%
Orange	19634	842	95.89%

The errors in ghost model predictions were computed by conducting game simulations until approximately 20 000 decisions were obtained for each ghost. The results obtained from these simulations are summarized in Table IV. In total, 79 705 ghost decisions were obtained, for an average model accuracy (the ratio of successes to total trials) of 96.4%. As shown in Table IV, the red ghost model is the least prone to errors, followed by the pink ghost model, the blue ghost model, and, last, the orange ghost model, which has the highest error rate. The model errors are due to imprecisions when decoding the game state from the observed game image, computation delay, missing state information (e.g., when ghost images overlap on the screen), and imperfect timing by the player when making turns, which has a small effect on Ms. Pac-Man's speed, as explained in Section II.

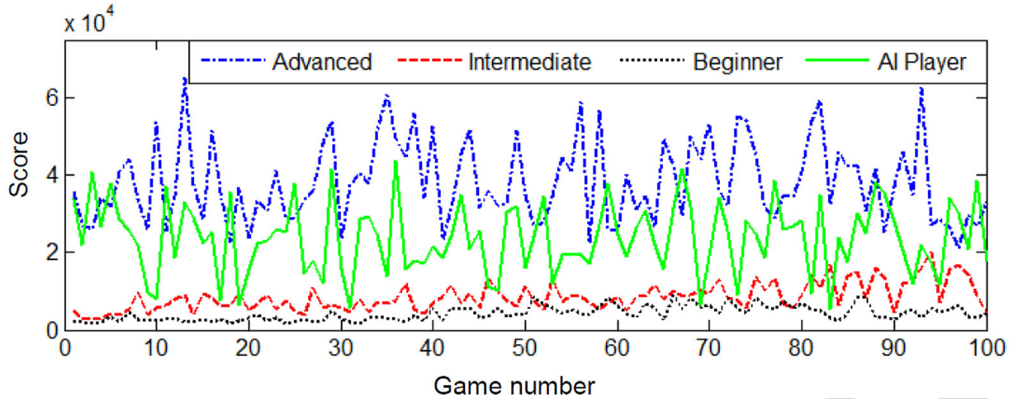


Fig. 11. Time histories of game scores obtained by human and AI players.

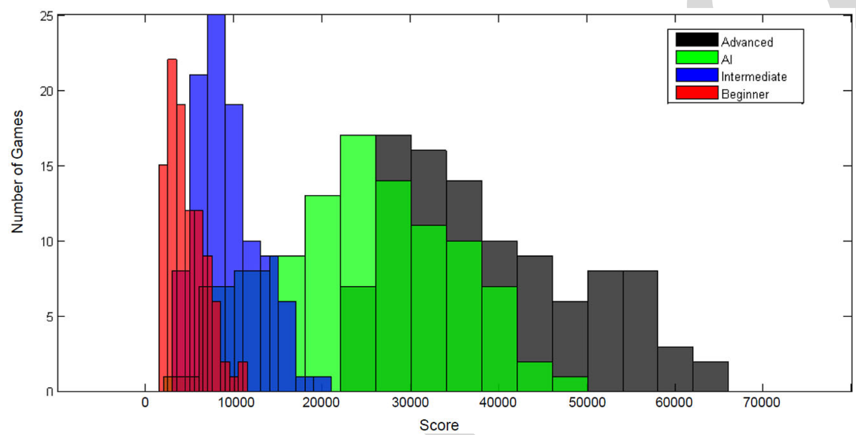


Fig. 12. Player score distribution for 100 games.

The difference in the accuracy of different ghost models arises from the fact that the differential equations in (26)–(28) and (31) include different state variables and game parameters. For example, the pink ghost model has a higher error rate than the red ghost model because its target position $\mathbf{y}_P$ is a function of Ms. Pac-Man state and control input, and these variables are both susceptible to observation errors, while the red ghost model only depends on Ms. Pac-Man state. Thus, the pink ghost model is subject not only to observation errors in $\mathbf{x}_M$, which cause errors in the red ghost model, but also to observation errors in $\mathbf{u}_M$.

B. Game Strategy Performance

The artificial player strategies are computed using the approach described in Section VI, where the weighting coefficients are $\omega_V = 1$, $\omega_R = 0.4$, $\omega_d = 8$, $\omega_p = 3$, $\omega_f = 15$, $\omega_b = 0.5$, $\chi = 20\ 000$, $\vartheta_- = -2.2$, and $\vartheta_+ = 1$, and are chosen by the user based on the desired tradeoff between the multiple conflicting objectives of Ms. Pac-Man [50]. The distance parameters are $\rho_0 = 150$ pixels and $\rho_b = 129$ pixels, and are chosen by the user based on the desired distance of influence for ghost avoidance and bonus item, respectively [53]. The time histories of the scores during 100 games are plotted in Fig. 11, and the score distributions are shown in Fig. 12. The minimum, average, and maximum scores are summarized in Table V.

TABLE V
PERFORMANCE RESULT SUMMARY OF AI AND HUMAN PLAYERS

Player	Minimum Score	Average Score	Maximum Score
AI	5210	23767	43720
Beginner	1610	4137	9210
Intermediate	2760	8542	20180
Advanced	21400	38172	65200

From these results, it can be seen that the model-based artificial (AI) player presented in this paper outperforms most of the computer players presented in the literature [8]–[14], which display average scores between 9000 and 18 000 and maximum scores between 20 000 and 36 280, where the highest score of 36 280 was achieved by the winner of the last *Ms. Pac-Man* screen competition at the 2011 Conference on Computational Intelligence and Games [14].

Because expert human players routinely outperform computer players and easily achieve scores over 65 000, the AI player presented in this paper is also compared to human players of varying skill levels. The beginner player is someone who has never played the game before, the intermediate player has basic knowledge of the game and some prior experience, and the advanced player has detailed knowledge of the game mechanics, and has previously played many games. All players completed the 100 games over the course of a few weeks, during multiple sittings, and over time displayed the performance plotted in Fig. 11. From Table V, it can be seen that the AI player presented in this paper performs significantly better than both the beginner and intermediate players on average, with its highest score being 43 720. However, the advanced player outperforms the AI player on average, and has a much higher maximum score of 65 200.

It can also be seen in Fig. 11 that the beginner and intermediate players improve their scores over time, while the advanced player does not improve significantly. In particular, when a simple least squares linear regression was performed on these game scores, the slope values were found to be 10.23 (advanced), 2.01 (AI), 74.32 (intermediate), and 36.67 (beginner). Furthermore, a linear regression t-test aimed at determining whether the slope of the regression line differs significantly from zero with 95% confidence was applied to the data in Fig. 11, showing that while the intermediate and beginner scores increase over time, the AI and advanced scores display a slope that is not statistically significantly different from zero (see [57] for a description of these methods). This suggests that beginner and intermediate players improve their performance more significantly by learning from the game, while the advanced player may have already reached its maximum performance level.

From detailed game data (not shown for brevity), it was found that human players are able to learn (or memorize) the first few levels of the game, and initially make fewer errors than the AI player. On the other hand, the AI player displays better performance than the human players later in the game, during high game levels when the game characters move faster, and the mazes become harder to navigate. These conditions force players to react and make decisions more quickly, and are found to be significantly more difficult by human players. Because the AI player can update its decision tree and strategy very frequently, the effects of game speed on the AI player's performance are much smaller than on human players. Finally, although the model-based approach presented in this paper does not include learning, methods such as temporal difference [39] will be introduced in future work to further improve the AI player's performance over time.

VIII. Conclusion

A model-based approach is presented for computing optimal decision strategies in the pursuit-evasion game *Ms. Pac-Man*. A model of the game and adversary dynamics are presented in the form of a decision tree that is updated over time. The decision tree is derived by decomposing the game maze using a cell decomposition approach, and by defining the profit of future decisions based on adversary state predictions, and real-time state observations. Then, the optimal strategy is computed from the decision tree over a finite time horizon, and implemented by an artificial (AI) player in real time, using a screen-capture interface. Extensive game simulations are used to validate the models of the ghosts presented in this paper, and to demonstrate the effectiveness of the optimal game strategies obtained from the decision trees. The AI player is shown to outperform beginner and intermediate human players, and to achieve the highest score of 43 720. It is also shown that although an advanced player outperforms the AI player, the AI player is better able to handle high game levels, in which the speed of the characters and spatial complexity of the mazes become more challenging.

Acknowledgment

The authors would like to thank R. Jackson at Stanford University, Stanford, CA, USA, for his contributions and suggestions.

References

[1] T. Muppirala, A. Bhattacharya, and S. Hutchin, "Surveillance strategies for a pursuer with finite sensor range," *Int. J. Robot. Res.*, vol. 26, no. 3, pp. 233–253, 2007.

[2] S. Ferrari, R. Fierro, B. Perteet, C. Cai, and K. Baumgartner, "A geometric optimization approach to detecting and intercepting dynamic targets using a mobile sensor network," *SIAM J. Control Optim.*, vol. 48, no. 1, pp. 292–320, 2009.

[3] V. Isler, S. Kannan, and S. Khanna, "Randomized pursuit-evasion with limited visibility," in *Proc. ACM-SIAM Symp. Discrete Algorithms*, 2004, pp. 1053–1063.

[4] V. Isler, D. Sun, and S. Sastry, "Roadmap based pursuit-evasion and collision avoidance," in *Proc. Robot. Syst. Sci.*, 2005.

[5] S. M. Lucas and G. Kendall, "Evolutionary computation and games," *IEEE Comput. Intell. Mag.*, vol. 1, no. 1, pp. 10–18, 2006.

[6] J. Schrum and R. Miikkulainen, "Discovering multimodal behavior in *Ms. Pac-Man* through evolution of modular neural networks," *IEEE Trans. Comput. Intell. AI Games*, vol. 8, no. 1, pp. 67–81, Mar. 2016.

[7] S. M. Lucas, "*Ms. Pac-Man* competition," *SIGEVOlution*, vol. 2, no. 4, pp. 37–38, Dec. 2007.

[8] N. Bell *et al.*, "Ghost direction detection and other innovations for *Ms. Pac-Man*," in *Proc. IEEE Symp. Comput. Intell. Games*, Aug. 2010, pp. 465–472.

[9] R. Thawonmas and H. Matsumoto, "Automatic controller of *Ms. Pac-Man* and its performance: Winner of the IEEE CEC 2009 software agent *Ms. Pac-Man* competition," in *Proc. Asia Simul. Conf.*, Oct. 2009.

[10] T. Ashida, T. Miyama, H. Matsumoto, and R. Thawonmas, "ICE Pambush 4," in *Proc. IEEE Symp. Comput. Intell. Games*, 2010.

[11] T. Miyama, A. Yamada, Y. Okunishi, T. Ashida, and R. Thawonmas, "ICE Pambush 5," in *Proc. IEEE Symp. Comput. Intell. Games*, 2011.

[12] R. Thawonmas and T. Ashida, "Evolution strategy for optimizing parameters in *Ms. Pac-Man* controller ICE Pambush 3," in *Proc. IEEE Symp. Comput. Intell. Games*, 2010, pp. 235–240.

[13] M. Emilio, M. Moises, R. Gustavo, and S. Yago, "Pac-mAnt: Optimization based on ant colonies applied to developing an agent for *Ms. Pac-Man*," in *Proc. IEEE Symp. Comput. Intell. Games*, 2010, pp. 458–464.

[14] N. Ikehata and T. Ito, "Monte-Carlo tree search in *Ms. Pac-Man*," in *Proc. IEEE Conf. Comput. Intell. Games*, Sep. 2011, pp. 39–46.

[15] A. A. Ghazanfar and M. A. Nicolelis, "Spatiotemporal properties of layer v neurons of the rat primary somatosensory cortex," *Cerebral Cortex*, vol. 9, no. 4, pp. 348–361, 1999.

[16] M. A. Nicolelis, L. A. Baccala, R. Lin, and J. K. Chapin, "Sensorimotor encoding by synchronous neural ensemble activity at multiple levels of the somatosensory system," *Science*, vol. 268, no. 5215, pp. 1353–1358, 1995.

[17] M. Kawato, "Internal models for motor control and trajectory planning," *Current Opinion Neurobiol.*, vol. 9, no. 6, pp. 718–727, 1999.

[18] D. M. Wolpert, R. C. Miall, and M. Kawato, "Internal models in the cerebellum," *Trends Cogn. Sci.*, vol. 2, no. 9, pp. 338–347, 1998.

[19] J. W. Krakauer, M.-F. Ghilardi, and C. Ghez, "Independent learning of internal models for kinematic and dynamic control of reaching," *Nature Neurosci.*, vol. 2, no. 11, pp. 1026–1031, 1999.

[20] M. A. Sommer and R. H. Wurtz, "Brain circuits for the internal monitoring of movements," *Annu. Rev. Neurosci.*, vol. 31, pp. 317, 2008.

[21] T. B. Crapse and M. A. Sommer, "The frontal eye field as a prediction map," *Progr. Brain Res.*, vol. 171, pp. 383–390, 2008.

[22] K. Doya, K. Samejima, K.-i. Katagiri, and M. Kawato, "Multiple model-based reinforcement learning," *Neural Comput.*, vol. 14, no. 6, pp. 1347–1369, 2002.

[23] A. J. Calise and R. T. Rysdyk, "Nonlinear adaptive flight control using neural networks," *IEEE Control Syst.*, vol. 18, no. 6, pp. 14–25, 1998.

[24] S. Ferrari and R. F. Stengel, "Online adaptive critic flight control," *J. Guid. Control Dyn.*, vol. 27, no. 5, pp. 777–786, 2004.

[25] C. G. Atkeson and J. C. Santamaria, "A comparison of direct and model-based reinforcement learning," in *Proc. Int. Conf. Robot. Autom.*, 1997.

[26] C. Guestrin, R. Patrascu, and D. Schuurmans, "Algorithm-directed exploration for model-based reinforcement learning in factored MDPs," in *Proc. Int. Conf. Mach. Learn.*, 2002, pp. 235–242.

[27] J. Si, *Handbook of Learning and Approximate Dynamic Programming*, New York, NY, USA: Wiley, 2004, vol. 2.

[28] A. Fitzgerald and C. B. Congdon, "A rule-based agent for *Ms. Pac-Man*," in *Proc. IEEE Congr. Evol. Comput.*, 2009, pp. 2646–2653.

[29] D. J. Gagne and C. B. Congdon, "Fright: A flexible rule-based intelligent ghost team for *Ms. Pac-Man*," in *Proc. IEEE Conf. Comput. Intell. Games*, 2012, pp. 273–280.

[30] N. Wirth and M. Gallagher, "An influence map model for playing *Ms. Pac-Man*," in *Proc. IEEE Symp. Comput. Intell. Games*, Dec. 2008, pp. 228–233.

[31] L. DeLooze and W. Viner, "Fuzzy Q-learning in a nondeterministic environment: Developing an intelligent *Ms. Pac-Man* agent," in *Proc. IEEE Symp. Comput. Intell. Games*, 2009, pp. 162–169.

[32] A. Alhejali and S. Lucas, "Evolving diverse *Ms. Pac-Man* playing agents using genetic programming," in *Proc. U.K. Workshop Comput. Intell.*, 2010.

[33] A. Alhejali and S. Lucas, "Using a training camp with genetic programming to evolve *Ms. Pac-Man* agents," in *Proc. IEEE Conf. Comput. Intell. Games*, 2011, pp. 118–125.

[34] T. Pepels, M. H. Winands, and M. Lanctot, "Real-time Monte Carlo tree search in Ms. Pac-Man," *IEEE Trans. Comput. Intell. AI Games*, vol. 6, no. 3, pp. 245–257, 2014.

[35] K. Q. Nguyen and R. Thawonmas, "Monte Carlo tree search for collaboration control of ghosts in *Ms. Pac-Man*," *IEEE Trans. Comput. Intell. AI Games*, vol. 5, no. 1, pp. 57–68, 2013.

[36] D. P. Bertsekas and S. Ioffe, "Temporal differences-based policy iteration and applications in neuro-dynamic programming," Lab. Inf. Decision Syst. Rep. 1996.

[37] B. Tong, C. M. Ma, and C. W. Sung, "A Monte-Carlo approach for the endgame of *Ms. Pac-Man*," in *Proc. IEEE Conf. Comput. Intell. Games*, Sep. 2011, pp. 9–15.

[38] S. Samothrakis, D. Robles, and S. Lucas, "Fast approximate max-n Monte Carlo tree search for *Ms. Pac-Man*," *IEEE Trans. Comput. Intell. AI Games*, vol. 3, no. 2, pp. 142–154, 2011.

[39] G. Foderaro, V. Raju, and S. Ferrari, "A model-based approximate λ iteration approach to online evasive path planning and the video game *Ms. Pac-Man*," *J. Control Theory Appl.*, vol. 9, no. 3, pp. 391–399, 2011.

[40] S. Ferrari and C. Cai, "Information-driven search strategies in the board game of CLUE," *IEEE Trans. Syst. Man Cybern. B, Cybern.*, vol. 39, no. 3, pp. 607–625, Jun. 2009.

[41] G. Foderaro, A. Swingler, and S. Ferrari, "A model-based cell decomposition approach to on-line pursuit-evasion path planning and the video game *Ms. Pac-Man*," in *Proc. IEEE Conf. Comput. Intell. Games*, 2012, pp. 281–287.

[42] M. Kaess, A. Ranganathan, and F. Dellaert, "iSAM: Incremental smoothing and mapping," *IEEE Trans. Robot.*, vol. 24, no. 6, pp. 1365–1378, Dec. 2008.

[43] M. Kaess *et al.*, "iSAM2: Incremental smoothing and mapping using the Bayes tree," *Int. J. Robot.*, vol. 31, no. 2, pp. 216–235, Feb. 2012.

[44] H. Wei and S. Ferrari, "A geometric transversals approach to analyzing the probability of track detection for maneuvering targets," *IEEE Trans. Comput.*, vol. 63, no. 11, pp. 2633–2646, 2014.

[45] H. Wei *et al.*, "Camera control for learning nonlinear target dynamics via Bayesian nonparametric Dirichlet-process Gaussian-process (DP-GP) models," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2014, pp. 95–102.

[46] W. Lu, G. Zhang, and S. Ferrari, "An information potential approach to integrated sensor path planning and control," *IEEE Trans. Robot.*, vol. 30, no. 4, pp. 919–934, 2014.

[47] J. Pittman, "The Pac-Man dossier," [Online]. Available: http://home.comcast.net/jpittman2/pacman/pacmandossier.html

[48] I. Szita and A. Lõrincz, "Learning to play using low-complexity rule-based policies: Illustrations through *Ms. Pac-Man*," *J. Artif. Intell. Res.*, pp. 659–684, 2007.

[49] M. Mateas, "Expressive AI: Games and artificial intelligence," in *Proc. DIGRA Conf.*, 2003.

[50] R. F. Stengel, *Optimal Control and Estimation*, New York, NY, USA: Dover, 1986.

[51] I. Szita and A. Lorincz, "Learning to play using low-complexity rule-based policies: Illustrations through *Ms. Pac-Man*," *J. Artif. Intell. Res.*, vol. 30, pp. 659–684, 2007.

[52] C. Cai and S. Ferrari, "Information-driven sensor path planning by approximate cell decomposition," *IEEE Trans. Syst. Man Cybern. B, Cybern.*, vol. 39, no. 3, pp. 672–689, Jun. 2009.

[53] J.-C. Latombe, *Robot Motion Planning*, Norwell, MA, USA: Kluwer, 1991.

[54] B. M. E. Moret, "Decision trees and diagrams," *ACM Comput. Surv.*, vol. 14, no. 4, pp. 593–623, 1982.

[55] F. V. Jensen and T. D. Nielsen, *Bayesian Networks and Decision Graphs*, New York, NY, USA: Springer-Verlag, 2001.

[56] M. Diehl and Y. Y. Haimes, "Influence diagrams with multiple objectives and tradeoff analysis," *IEEE Trans. Syst. Man Cyber. A*, vol. 34, no. 3, pp. 293–304, 2004.

[57] D. C. Montgomery, E. A. Peck, and G. G. Vining, *Introduction to Linear Regression Analysis*, New York, NY, USA: Wiley, 2012, vol. 821.

Greg Foderaro (M'XX) received the B.S. degree in mechanical engineering from Clemson University, Clemson, SC, USA, in 2009 and the Ph.D. degree in mechanical engineering and materials science from Duke University, Durham, NC, USA, in 2013. Q5

He is currently a Staff Engineer at Applied Research Associates, Inc. His research interests are in underwater sensor networks, robot path planning, multiscale dynamical systems, pursuit-evasion games, and spiking neural networks.

Ashleigh Swingler (M'XX) received the B.S. and M.S. degrees in mechanical engineering from Duke University, Durham, NC, USA, in 2010 and 2012, respectively. She is currently working toward the Ph.D. degree in the Department of Mechanical Engineering and Materials Science, Duke University. Q6

Her main research interests include disjunctive programming and approximate cell decomposition applied to robot and sensor path planning.

Silvia Ferrari (SM'XX) received the B.S. degree from Embry-Riddle Aeronautical University, Daytona Beach, FL, USA and the M.A. and Ph.D. degrees from Princeton University, Princeton, NJ, USA. Q7

She is a Professor of Mechanical and Aerospace Engineering at Cornell University, Ithaca, NY, USA, where she also directs the Laboratory for Intelligent Systems and Controls (LISC). Prior to joining the Cornell faculty, she was Professor of Engineering and Computer Science at Duke University, Durham, NC, USA, where she was also the Founder and Director of the NSF Integrative Graduate Education and Research Traineeship (IGERT) program on Wireless Intelligent Sensor Networks (WISeNet), and a Faculty Member of the Duke Institute for Brain Sciences (DIBS). Her principal research interests include robust adaptive control, learning and approximate dynamic programming, and information-driven planning and control for mobile and active sensor networks.

Prof. Ferrari is a member of the American Society of Mechanical Engineers (ASME), the International Society for Optics and Photonics (SPIE), and the American Institute of Aeronautics and Astronautics (AIAA). She is the recipient of the U.S. Office of Naval Research (ONR) Young Investigator Award (2004), the National Science Foundation (NSF) CAREER Award (2005), and the Presidential Early Career Award for Scientists and Engineers (PECASE) Award (2006).

QUERIES

Q1: Please provide city and postal code for Applied Research Associates, Inc.

Q2: Please note that names of games are italicized as per the IEEE style; there were many instances in your paper where Ms. Pac-Man was not used as the game title, but a character from the game. These were left in roman font. Please check for correctness.

Q3: Please specify which sections.

Q4: Please specify which sections.

Q5: Please provide membership year.

Q6: Please provide membership year.

Q7: Please provide membership year.

A Model-Based Approach to Optimizing *Ms. Pac-Man* Game Strategies in Real Time

Greg Foderaro, *Member, IEEE*, Ashleigh Swingler, *Member, IEEE*, and Silvia Ferrari, *Senior Member, IEEE*

***Abstract*—This paper presents a model-based approach for computing real-time optimal decision strategies in the pursuit-evasion game of *Ms. Pac-Man*. The game of *Ms. Pac-Man* is an excellent benchmark problem of pursuit-evasion game with multiple, active adversaries that adapt their pursuit policies based on *Ms. Pac-Man*'s state and decisions. In addition to evading the adversaries, the agent must pursue multiple fixed and moving targets in an obstacle-populated environment. This paper presents a novel approach by which a decision-tree representation of all possible strategies is derived from the maze geometry and the dynamic equations of the adversaries or ghosts. The proposed models of ghost dynamics and decisions are validated through extensive numerical simulations. During the game, the decision tree is updated and used to determine optimal strategies in real time based on state estimates and game predictions obtained iteratively over time. The results show that the artificial player obtained by this approach is able to achieve high game scores, and to handle high game levels in which the characters speeds and maze complexity become challenging even for human players.**

***Index Terms*—Cell decomposition, computer games, decision theory, decision trees, *Ms. Pac-Man*, optimal control, path planning, pursuit-evasion games.**

I. Introduction

The video game *Ms. Pac-Man* is a challenging example of pursuit-evasion games in which an agent (Ms. Pac-Man) must evade multiple dynamic and active adversaries (ghosts), as well as pursue multiple fixed and moving targets (pills, fruits, and ghosts), all the while navigating an obstacle-populated environment. As such, it provides an excellent benchmark problem for a number applications including recognizance and surveillance [1], search-and-rescue [2], [3], and mobile robotics [4], [5]. In *Ms. Pac-Man*, each ghost implements a different decision policy with random seeds and multiple modalities that are a function of Ms. Pac-Man's decisions. Consequently, the game requires decisions to be made in real time, based on observations of a stochastic and dynamic environment that is challenging to both human and artificial players [6]. This is evidenced by the fact that, despite the recent series of artificial intelligence competitions inviting researchers to develop artificial players to achieve the highest possible score, existing artificial players have yet to achieve the performance level of expert human players [7]. For instance, existing artificial players typically achieve average scores between 9000 and 18 000 and maximum scores between 20 000 and 35 000 [8]–[13]. In particular, the highest score achieved at the last *Ms. Pac-Man* screen capture controller competition was 36 280, while expert human players routinely achieve scores over 65 000 and in some cases as high as 920 000 [14].

Recent studies in the neuroscience literature indicate that biological brains generate exploratory actions by comparing the meaning encoded in new sensory inputs with internal representations obtained from the sensory experience accumulated during a lifetime or preexisting functional maps [15]–[19]. For example, internal representations of the environment and of the subject's body (body schema), also referred to as internal models, appear to be used by the somatosensory cortex (SI) for predictions that are compared to the reafferent sensory input to inform the brain of sensory discrepancies evoked by environmental changes, and generate motor actions [20], [21]. Computational intelligence algorithms that exploit models built from prior experience or first principles have also been shown to be significantly more effective, in many cases, than those that rely solely on learning [22]–[24]. One reason is that many reinforcement learning algorithms improve upon the latest approximation of the policy and value function. Therefore, a model can be used to establish a better performance baseline. Another reason is that model-free learning algorithms need to explore the entire state and action spaces, thus requiring significantly more data and, in some cases, not scaling up to complex problems [25]–[27].

Artificial players for *Ms. Pac-Man* to date have been developed using model-free methods, primarily because of the lack of a mathematical model for the game components. One approach has been to design rule-based systems that implement conditional statements derived using expert knowledge [8]–[12], [28], [29]. While it has the advantage of being stable and computationally cheap, this approach lacks extensibility and cannot handle complex or unforeseen situations, such as, high game levels, or random ghosts behaviors. An influence map model was proposed in [30], in which the game characters and objects exert an influence on their surroundings. It was also shown in [31] that, in the *Ms. Pac-Man* game, Q-learning and fuzzy-state aggregation can be used to learn in nondeterministic environments. Genetic algorithms and Monte Carlo searches have also been successfully implemented in [32]–[35]

Manuscript received October 18, 2014; revised September 02, 2015; accepted January 23, 2016. This work was supported by the National Science Foundation under Grant ECS 0925407.

G. Foderaro was with the Mechanical Engineering Department, Duke University, Durham, NC 27708 USA. He is now with Applied Research Associates, Inc. (e-mail: greg.foderaro@duke.edu).

A. Swingler is with the Mechanical Engineering and Materials Science Department, Duke University, Durham, NC 27708 USA (e-mail: ashleigh.swingler@duke.edu).

S. Ferrari is with the Mechanical and Aerospace Engineering Department, Cornell University, Ithaca, NY 14853 USA (e-mail: ferrari@cornell.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCIAIG.2016.2523508

to develop high-scoring agents in the artificial intelligence competitions. Due to the complexity of the environment and adversary behaviors, however, model-free approaches have had difficulty handling the diverse range of situations encountered by the player throughout the game [36].

The model-based approach presented in this paper overcomes the limitations of existing methods [14], [37]–[39] by using a mathematical model of the game environment and adversary behaviors to predict future game states and ghost decisions. Exact cell decomposition is used to obtain a graphical representation of the obstacle-free configuration space for *Ms. Pac-Man* in the form of a connectivity graph that captures the adjacency relationships between obstacle-free convex cells. Using the approach first developed in [40] and [41], the connectivity graph can be used to generate a decision tree that includes action and utility nodes, where the utility function represents a tradeoff between the risk of losing the game (capture by a ghost) and the reward of increasing the game score. The utility nodes are estimated by modeling the ghosts' dynamics and decisions using ordinary differential equations (ODEs). The ODE models presented in this paper account for each ghost's personality and multiple modes of motion. Furthermore, as shown in this paper, the ghosts are active adversaries that implement adaptive policies, and plan their paths based on Ms. Pac-Man's actions.

Extensive numerical simulations demonstrate that the ghost models presented in this paper are able to predict the paths of the ghosts with an average accuracy of 94.6%. Furthermore, these models can be updated such that when a random behavior or error occurs, the dynamic model and corresponding decision tree can both be learned in real time. The game strategies obtained by this approach achieve better performance than beginner and intermediate human players, and are able to handle high game levels, in which the character speed and maze complexity become challenging even for human players. Because it can be generalized to more complex environments and dynamics, the model-based approach presented in this paper can be extended to real-world pursuit-evasion problems in which the agents and adversaries may consist of robots or autonomous vehicles, and motion models can be constructed from exteroceptive sensor data using, for example, graphical models, Markov decision processes, or Bayesian nonparametric models [2], [42]–[46].

The paper is organized as follows. Section II reviews the game of *Ms. Pac-Man*. The problem formulation and assumptions are described in Section III. The dynamic models of *Ms. Pac-Man* and the ghosts are presented in Sections IV and V, respectively. Section VI presents the model-based approach to developing an artificial *Ms. Pac-Man* player based on decision trees and utility theory. The game model and artificial player are demonstrated through extensive numerical simulations in Section VII.

II. THE *MS. PAC-MAN* GAME

Released in 1982 by Midway Games, *Ms. Pac-Man* is a popular video game that can be considered as a challenging benchmark problem for dynamic pursuit and evasion games. In the *Ms. Pac-Man* game, the player navigates a character named Ms. Pac-Man through a maze with the goal of eating (traveling over) a set of fixed dots, called pills, as well as one or more moving objects (bonus items), referred to as fruits. The game image has the dimensions 224×288 pixels, which can be divided into a square grid of 8×8 pixel tiles, where each maze corridor consists of a row or a column of tiles. Each pill is located at the center of a tile and is eaten when Ms. Pac-Man is located within that tile [47].

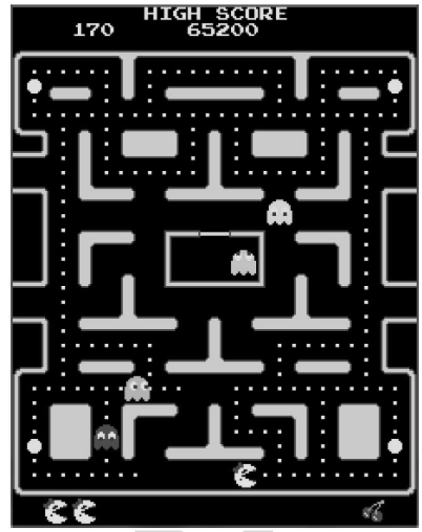
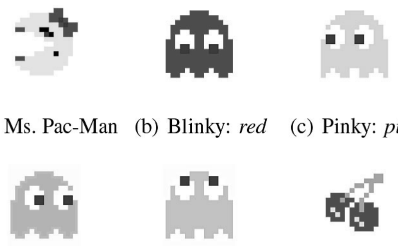


Fig. 1. Screen-capture of the *Ms. Pac-Man* game emulated on a computer.

Four ghosts, each with unique colors and behaviors, act as adversaries and pursue Ms. Pac-Man. If the player and a ghost move into the same tile, the ghost is said to capture Ms. Pac-Man, and the player loses one of three lives. The game ends when no lives remain. The ghosts begin the game inside a rectangular room in the center of the maze, referred to as the ghost pen, and are released into the maze at various times. If the player eats all of the pills in the maze, the level is cleared, and the player starts the process over, in a new maze, with incrementally faster adversaries.

Each maze contains a set of tunnels that allow Ms. Pac-Man to quickly travel to opposite sides of the maze. The ghosts can also move through the tunnels, but they do so at a reduced speed. The player is given a small advantage over ghosts when turning corners as well, where if a player controls Ms. Pac-Man to turn slightly before an upcoming corner, the distance Ms. Pac-Man must travel to turn the corner is reduced by up to approximately 2 pixels [47]. A player can also briefly reverse the characters' pursuit-evasion roles by eating one of four special large dots per maze referred to as power pills, which, for a short period of time, cause the ghosts to flee and give Ms. Pac-Man the ability to eat them [48]. Additional points are awarded when Ms. Pac-Man eats a bonus item. Bonus items enter the maze through a tunnel twice per level, and move slowly through the corridors of the maze. If they remain uneaten, the items exit the maze. A screenshot of the game is shown in Fig. 1, and the game characters are displayed in Fig. 2.

In addition to simply surviving and advancing through mazes, the objective of the player is to maximize the number of points earned, or score. During the game, points are awarded



(a) Ms. Pac-Man (b) Blinky: *red* (c) Pinky: *pink*

(d) Inky: *blue* (e) Sue: *orange* (f) Fruit: *cherry*

Fig. 2. Game characters and objects. (a) Ms. Pac-Man. (b) Blinky: *red*. (c) Pinky: *pink*. (d) Inky: *blue*. (e) Sue: *orange*. (f) Fruit: *cherry*.

when an object is eaten by Ms. Pac-Man. Pills are worth ten points each, a power pill gives 50 points, and the values of bonus items vary per level from 100 to 5000 points. When a power pill is active, the score obtained for capturing a ghost increases exponentially with the number of ghosts eaten in succession, where the total value is $\sum_{i=1}^{n} 100(2^n)$ and n is the number of ghosts eaten thus far. Therefore, a player can score 3000 points by eating all four ghosts during the duration of one power pill's effect. For most players, the game score is highly dependent on the points obtained for capturing ghosts. When Ms. Pac-Man reaches a score of 10 000, an extra life is awarded. In this paper, it is assumed that the player's objective is to maximize its game score and, thus, decision strategies are obtained by optimizing the score components, subject to a model of the game and ghost behaviors.

III. Problem Formulation and Assumptions

The *Ms. Pac-Man* player is viewed as a decision maker that seeks to maximize the final game score by a sequence of decisions based on the observed game state and predictions obtained from a game model. At any instant k, the player has access to all of the information displayed on the screen, because the state of the game $\mathbf{s}(k) \in \mathcal{X} \subset \mathbb{R}^n$ is fully observable and can be extracted without error from the screen capture. The time interval $(t_0, t_F]$ represents the entire duration of the game and, because the player is implemented using a digital computer, time is discretized and indexed by $k = 0, 1, \ldots, F$, where F is a finite end-time index that is unknown. Then, at any time $t_k \in (t_0, t_F]$, the player must make a decision $\mathbf{u}_M(k) \in \mathcal{U}(k)$ on the motion of Ms. Pac-Man, where $\mathcal{U}(k)$ is the space of admissible decisions at time t_k. Decisions are made according to a game strategy as follows.

Definition 3.1: A strategy is a class of admissible policies that consists of a sequence of functions

$$\sigma = \{\mathbf{c}_0, \mathbf{c}_1, \ldots\} \tag{1}$$

where $\mathbf{c}_k$ maps the state variables into an admissible decision

$$\mathbf{u}_M(k) = \mathbf{c}_k[\mathbf{s}(k)] \tag{2}$$

such that $\mathbf{c}_k[\cdot] \in \mathcal{U}(k)$, for all $\mathbf{s}(k) \in \mathcal{X}$.

In order to optimize the game score, the strategy σ is based on the expected profit of all possible future outcomes, which is estimated from a model of the game. In this paper, it is assumed that at several moments in time, indexed by t_i, the game can be modeled by a decision tree T_i that represents all possible decision outcomes over a time interval $[t_i, t_f] \subset (t_0, t_F]$, where $\Delta t = (t_f - t_i)$ is a constant chosen by the user. If the error between the predictions obtained by game model and the state observations exceed a specified tolerance, a new tree is generated, and the previous one is discarded. Then, at any time $t_k \in [t_i, t_f]$, the instantaneous profit can be modeled as a weighted sum of the reward V and the risk R and is a function of the present state and decision

$$\mathscr{L}[\mathbf{s}(k), \mathbf{u}_M(k)] = w_V V[\mathbf{x}(k), \mathbf{u}_M(k)] + w_R R[\mathbf{x}(k), \mathbf{u}_M(k)] \tag{3}$$

where w_V and w_R are weighting coefficients chosen by the user.

The decision-making problem considered in this paper is to determine a strategy $\sigma_i^* = \{\mathbf{c}_i^*, \ldots, \mathbf{c}_f^*\}$ that maximizes the cumulative profit over the time interval $[t_i, t_f]$

$$J_{i,f}\left[\mathbf{x}(i), \sigma_i\right] = \sum_{k=i}^{f} \mathscr{L}[\mathbf{x}(k), \mathbf{u}_M(k)] \tag{4}$$

such that, given T_i, the optimal total profit is

$$J_{i,f}^*\left[\mathbf{x}(i), \sigma_i^*\right] = \max_{\sigma_i} \left\{J_{i,f}\left[\mathbf{x}(i), \sigma_i\right]\right\}. \tag{5}$$

Because the random effects in the game are significant, any time the observed state $\mathbf{s}(k)$ significantly differs from the model prediction, the tree T_i is updated, and a new strategy σ_i^* is computed, as explained in Section IV-C. A methodology is presented in Sections III–VI to model the *Ms. Pac-Man* game and profit function based on guidelines and resources describing the behaviors of the characters, such as [49].

IV. Model of Ms. Pac-Man Behavior

In this paper, the game of *Ms. Pac-Man* is viewed as a pursuit-evasion game in which the goal is to determine the path or trajectory of an agent (Ms. Pac-Man) that must pursue fixed and moving targets in an obstacle-populated workspace, while avoiding capture by a team of mobile adversaries. The maze is considered to be a 2-D Euclidean workspace, denoted by $\mathcal{W} \subset \mathbb{R}^2$, that is populated by a set of obstacles (maze walls), $\mathcal{B}_1, \mathcal{B}_2, \ldots$, with geometries and positions that are constant and known *a priori*. The workspace $\mathcal{W}$ can be considered closed and bounded (compact) by viewing the tunnels, denoted by $\mathcal{T}$, as two horizontal corridors, each connected to both sides of the maze. Then, the obstacle-free space $\mathcal{W}_{\text{free}} = \mathcal{W} \backslash \{\mathcal{B}_1, \mathcal{B}_2, \ldots\}$ consists of all the corridors in the maze. Let $\mathcal{F}_\mathcal{W}$ denote an inertial reference frame embedded in $\mathcal{W}$ with origin at the lower left corner of the maze. In continuous time t, the state of Ms. Pac-Man is represented by a time-varying vector

$$\mathbf{x}_M(t) = [x_M(t)\ y_M(t)]^T \tag{6}$$

where x_M and y_M are the x, y-coordinates of the centroid of the Ms. Pac-Man character with respect to $\mathcal{F}_\mathcal{W}$, measured in units of pixels.

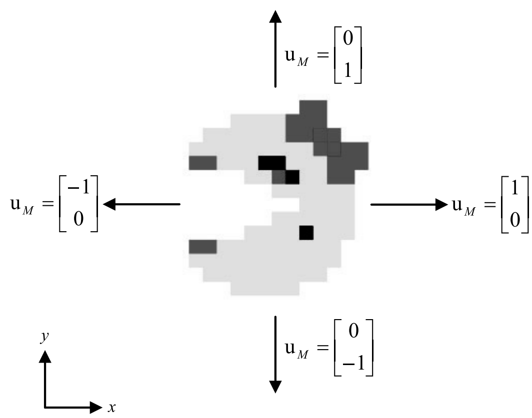


Fig. 3. Control vector sign conventions.

The control input for *Ms. Pac-Man* is a joystick, or keyboard, command from the player that defines a direction of motion for Ms. Pac-Man. As a result of the geometries of the game characters and the design of the mazes, the player is only able to select one of four basic control decisions (move up, move left, move down, or move right), and characters are restricted to two movement directions within a straight-walled corridor. The control input for Ms. Pac-Man is denoted by the vector

$$\mathbf{u}_M(t) = [u_M(t) v_M(t)]^T \tag{7}$$

where $u_M \in \{-1, 0, 1\}$ represents joystick commands in the x-direction and $v_M \in \{-1, 0, 1\}$ defines motion in the y-direction, as shown in Fig. 3. The control or action space, denoted by $\mathcal{U}$, for all agents is a discrete set

$$\mathcal{U} = [a_1, a_2, a_3, a_4] = \left\{ \begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}-1\\0\end{bmatrix}, \begin{bmatrix}0\\-1\end{bmatrix}, \begin{bmatrix}1\\0\end{bmatrix} \right\}. \tag{8}$$

Given the above definitions of state and control, it can be shown that Ms. Pac-Man's dynamics can be described by a linear, ordinary differential equation (ODE)

$$\dot{\mathbf{x}}_M(t) = \mathbf{A}(t)\mathbf{x}_M(t) + \mathbf{B}(t)\mathbf{u}_M(t) \tag{9}$$

where $\mathbf{A}$ and $\mathbf{B}$ are state–space matrices of appropriate dimensions [50].

In order to estimate Ms. Pac-Man's state, the ODE in (9) can be discretized, by integrating it with respect to time, using an integration step $\delta t << \Delta t = (t_f - t_i)$. The time index t_i represents all moments in time when a new decision tree is generated, i.e., the start of the game, the start of a new level, the start of game following the loss of one life, or the time when one of the actual ghosts' trajectories is found to deviate from the model prediction. Then, the dynamic equation for Ms. Pac-Man in discrete time can be written as

$$\mathbf{x}_M(k) = \mathbf{x}_M(k-1) + \alpha_M(k-1)\mathbf{u}_M(k-1)\delta t \tag{10}$$

where $\alpha_M(k)$ is the speed of Ms. Pac-Man at time k, which is subject to change based on the game conditions. The control input for the *Ms. Pac-Man* player developed in this paper is determined by a discrete-time state-feedback control law

$$\mathbf{u}_M(k) = c_k [\mathbf{x}_M(k)] \tag{11}$$

that is obtained using the methodology in Section VI, and may change over time.

The ghosts' dynamic equations are derived in Section V, in terms of state and control vectors

$$\mathbf{x}_G(k) = [x_G(k)\ y_G(k)]^T \tag{12}$$

$$\mathbf{u}_G(k) = [u_G(k)\ v_G(k)]^T \tag{13}$$

that are based on the same conventions used for *Ms. Pac-Man*, and are observed in real time from the game screen. The label G belongs to a set of unique identifiers $I_G = \{G | G \in \{R, B, P, O\}\}$, where R denotes the red ghost (Blinky), B denotes the blue ghost (Inky), P denotes the pink ghost (Pinky), and O denotes the orange ghost (Sue). Although an agent's representation occupies several pixels on the screen, its actual position is defined by a small 8 (pixel) $\times$ 8 (pixel) game tile, and capture occurs when these positions overlap. Letting $\tau[\mathbf{x}]$ represent the tile containing the pixel at position $\mathbf{x} = (x, y)$, capture occurs when

$$\tau[\mathbf{x}_M(k)] = \tau[\mathbf{x}_G(k)], \qquad \exists G \in I_G. \tag{14}$$

Because ghosts' behaviors include a pseudorandom component, the optimal control law for Ms. Pac-Man cannot be determined *a priori*, but must be updated based on real-time observations of the game [51]. Like any human player, the *Ms. Pac-Man* player developed by this paper is assumed to have full visibility of the information displayed on the game screen. Thus, a character state vector containing the positions of all game characters and of the bonus item $\mathbf{x}_F(k)$ at time k is defined as

$$\mathbf{x}(k) \triangleq \left[\mathbf{x}_M^T(k)\ \mathbf{x}_R^T(k)\ \mathbf{x}_B^T(k)\ \mathbf{x}_P^T(k)\ \mathbf{x}_O^T(k)\ \mathbf{x}_F^T(k)\right]^T \tag{15}$$

and can be assumed to be fully observable. Future game states can be altered by the player via the game control vector $\mathbf{u}_M(k)$. While the player can decide the direction of motion (Fig. 3), the speed of Ms. Pac-Man, $\alpha_M(k)$, is determined by the game based on the current game level, on the modes of the ghosts, and on whether Ms. Pac-Man is collecting pills. Furthermore, the speed is always bounded by a known constant ν, i.e., $\alpha_M(k) \leq \nu$.

The ghosts are found to obey one of three modes that are represented by a discrete variable $\delta_G(k)$, namely pursuit mode $[\delta_G(k) = 0]$, evasion mode $[\delta_G(k) = 1]$, and scatter mode $[\delta_G(k) = -1]$. The modes of all four ghosts are grouped into a vector $\mathbf{m}(k) \triangleq [\delta_R(k)\ \delta_B(k)\ \delta_P(k)\ \delta_O(k)]^T$ that is used to determine, among other things, the speed of Ms. Pac-Man.

The distribution of pills (fixed targets) in the maze is represented by a 28×36 matrix $\mathbf{D}(k)$ defined over an 8 (pixel) $\times$ 8 (pixel) grid used to discretize the game screen into tiles. Then, the element in the ith row and $j$the column at time k, denoted by $\mathbf{D}_{(i,j)}(k)$, represents the presence of a pill (+1), power pill (−1), or an empty tile (0). Then, a function $n : \mathbb{R}^{28\times 36} \rightarrow \mathbb{R}$, defined as the sum of the absolute values of all elements of $\mathbf{D}(k)$, can be used to obtain the number of pills (including power pills) that are present in the maze at time k. For example, when Ms. Pac-Man is eating pills $n[\mathbf{D}(k)] < n[\mathbf{D}(k-1)]$, and when it is traveling in an empty corridor,

TABLE I
SPEED PARAMETERS FOR MS. PAC-MAN

Game Level	β_1	β_2	β_3	β_4
1	0.71	0.80	0.79	0.90
2 - 4	0.79	0.90	0.83	0.95
5 - 20	0.87	1.00	0.87	1.00
21+	0.79	0.90	-	-

$n[\mathbf{D}(k)] = n[\mathbf{D}(k-1)]$. Using this function, the speed of Ms. Pac-Man can be modeled as follows:

$$\alpha_M(k) = \begin{cases} \beta_1\nu, \text{ if } & \mathbf{m}(k) \not\ni 1 \text{ and } n\left[\mathbf{D}(k)\right] < n\left[\mathbf{D}(k-1)\right] \\ \beta_2\nu, \text{ if } & \mathbf{m}(k) \not\ni 1 \text{ and } n\left[\mathbf{D}(k)\right] = n\left[\mathbf{D}(k-1)\right] \\ \beta_3\nu, \text{ if } & \mathbf{m}(k) \ni 1 \text{ and } n\left[\mathbf{D}(k)\right] < n\left[\mathbf{D}(k-1)\right] \\ \beta_4\nu, \text{ if } & \mathbf{m}(k) \ni 1 \text{ and } n\left[\mathbf{D}(k)\right] = n\left[\mathbf{D}(k-1)\right] \end{cases} \tag{16}$$

where β_1, β_2, β_3, and β_4 are known parameters that vary with the game level, as shown in Table I.

All elements of the matrix $\mathbf{D}(k)$ and vector $\mathbf{m}(k)$ are rearranged into a vector $\mathbf{z}(k)$ that represents the game conditions, and is obtained in real time from the screen (Section VII). As a result, the state of the game $\mathbf{s}(k) = [\mathbf{x}^T(k) \quad \mathbf{z}^T(k)]^T$ is fully observable. Furthermore, $\mathbf{s}(k)$ determines the behaviors of the ghosts as explained in Section V.

V. MODELS OF ADVERSARY BEHAVIOR

The Ms. Pac-Man character is faced by a team of antagonistic adversaries, four ghosts, that try to capture Ms. Pac-Man and cause it to lose a life when successful. Because the game terminates after Ms. Pac-Man loses all lives, being captured by the ghosts prevents the player from increasing its game score. Evading the ghosts is, therefore, a key objective in the game of *Ms. Pac-Man*. The dynamics of each ghost, ascertained through experimentation and online resources [47], are modeled by a linear differential equation in the form:

$$\mathbf{x}_G(k) = \mathbf{x}_G(k-1) + \alpha_G(k-1)\mathbf{u}_G(k-1)\delta t \tag{17}$$

where the ghost speed α_G and control input $\mathbf{u}_G$ depend on the ghost personality (G) and mode, as explained in the following Q3 subsections. The pursuit mode is the most common and represents the behavior of the ghosts while actively attempting to capture Ms. Pac-Man. When in pursuit mode, each ghost uses Q4 a different control law as shown in the following subsections. When Ms. Pac-Man eats a power pill, the ghosts enter evasion mode and move slowly and randomly about the maze. The scatter mode only occurs during the first seven seconds of each level and at the start of gameplay following the death of Ms. Pac-Man. In scatter mode, the ghosts exhibit the same random motion as in evasion mode, but move at "normal" speeds.

A. Ghost Speed

The speeds of the ghosts depend on their personality, mode, and position. In particular, the speed of Inky, Pinky, and Sue can be modeled in terms of the maximum speed of Ms. Pac-Man (ν), and in terms of the ghost mode and speed parameters (Table II) as follows:

TABLE II
SPEED PARAMETERS FOR BLUE, PINK, AND ORANGE GHOSTS

Game Level	η_1 (evasion)	η_2 (pursuit)	η_3 (tunnel)
1	0.50	0.75	0.40
2 - 4	0.55	0.85	0.45
5 - 20	0.60	0.95	0.50
21+	-	0.95	0.50

$$\alpha_G(k) = \begin{cases} \eta_1\nu, \text{ if } \delta_G(k) = 1 \\ \eta_2\nu, \text{ if } \delta_G(k) \neq 1 \text{ and } \tau[\mathbf{x}_G(k)] \notin \mathcal{T} \\ \eta_3\nu, \text{ if } \delta_G(k) \neq 1 \text{ and } \tau[\mathbf{x}_G(k)] \in \mathcal{T} \end{cases} \tag{18}$$

where $G = B, P, O$. The parameter η_1 (Table II) scales the speed of a ghost in evasion mode. When ghosts are in scatter or pursuit mode, their speed is scaled by parameter η_2 or η_3, depending on whether they are outside or inside a tunnel $\mathcal{T}$, respectively. The ghost speeds decrease significantly when they are located in $\mathcal{T}$, accordingly, $\eta_2 > \eta_3$, as shown in Table II.

Unlike the other three ghosts, Blinky has a speed that depends on the number of pills in the maze $n[\mathbf{D}(k)]$. When the value of $n(\cdot)$ is below a threshold d_1, the speed of the red ghost increases according to parameter η_4, as shown in Table III. When the number of pills decreases further, below $n[\mathbf{D}(k)] < d_2$, Blinky's speed is scaled by a parameter $\eta_5 \geq \eta_4$ (Table III). The relationship between the game level, the speed scaling constants, and the number of pills in the maze is provided in lookup table form in Table III. Thus, Blinky's speed can be modeled as

TABLE III
SPEED PARAMETERS FOR RED GHOST

Level	d_1	η_4	d_2	η_5
1	-	-	-	-
2	30	0.90	15	0.90
3 - 4	40	0.90	20	0.95
5	40	1.00	20	1.05
6 - 8	50	1.00	25	1.05
9 - 11	60	1.00	30	1.05
12 - 14	80	1.00	40	1.05
15 - 18	100	1.00	50	1.05
19 - 21+	120	1.00	60	1.05

$$\alpha_G(k) = \begin{cases} \eta_4\nu, & \text{if } n[\mathbf{D}(k)]| \leq d_1 \\ \eta_5\nu, & \text{if } n[\mathbf{D}(k)] \leq d_2 \end{cases}, \quad \text{for} \quad G = R \tag{19}$$

and Blinky is often referred to as the aggressive ghost.

B. Ghost Policy in Pursuit Mode

Each ghost utilizes a different strategy for chasing Ms. Pac-Man, based on its own definition of a target position denoted by $\mathbf{y}_G(k) \in \mathcal{W}$. In particular, the ghost control law greedily selects the control input that minimizes the Manhattan distance between the ghost and its target from a set of admissible control inputs, or action space, denoted by $\mathcal{U}_G(k)$. The ghost action space depends on the position of the ghost at time k, as well as the geometries of the maze walls, and is defined similarly to the action space of Ms. Pac-Man in (8). Thus, based on the distance between the ghost position $\mathbf{x}_G(k)$ and the target position $\mathbf{y}_G(k)$, every ghost implements the following control law to reach $\mathbf{y}_G(k)$:

$$\mathbf{u}_G(k) = \begin{cases} \mathbf{c} & \text{if } \mathbf{c} \in \mathcal{U}_G(k) \\ \mathbf{d} & \text{if } \mathbf{c} \notin \mathcal{U}_G(k), \mathbf{d} \in \mathcal{U}_G(k) \\ [0\ 1]^T & \text{if } \mathbf{c} \notin \mathcal{U}_G(k), \mathbf{d} \notin \mathcal{U}_G(k) \end{cases} \tag{20}$$

where

$$\mathbf{c} \triangleq H(\mathbf{C}) \circ \operatorname{sgn}[\boldsymbol{\xi}_G(k)] \tag{21}$$

$$\mathbf{d} \triangleq H(\mathbf{D}) \circ \operatorname{sgn}[\boldsymbol{\xi}_G(k)] \tag{22}$$

$$\mathbf{C} \triangleq \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} |\boldsymbol{\xi}_G(k)| \tag{23}$$

$$\mathbf{D} \triangleq \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix} |\boldsymbol{\xi}_G(k)| \tag{24}$$

$$\boldsymbol{\xi}_G(k) \triangleq [\mathbf{x}_G(k) - \mathbf{y}_G(k)]\,. \tag{25}$$

Symbol $^\circ$ denotes the Schur product, $H(\cdot)$ is the elementwise Heaviside step function defined such that $H(0) = 1$, $\operatorname{sgn}(\cdot)$ is the elementwise signum or sign function, and $|\cdot|$ is the elementwise absolute value.

In pursuit mode, the target position for Blinky, the red ghost (R), is the position of Ms. Pac-Man [47]

$$\mathbf{y}_R(k) = \mathbf{x}_M(k) \tag{26}$$

as shown in Fig. 4. As a result, the red ghost is most often seen following the path of Ms. Pac-Man. The orange ghost (O), Sue, is commonly referred to as the shy ghost, because it typically tries to maintain a moderate distance from Ms. Pac-Man. As shown in Fig. 5, when Ms. Pac-Man is within a threshold distance c_O of Sue, the ghost moves toward the lower left corner of the maze, with coordinates $(x, y) = (0, 0)$. However, if Ms. Pac-Man is farther than c_O from Sue, Sue's target becomes the position of Ms. Pac-Man, i.e., [47]

$$\mathbf{y}_O(k) = \begin{cases} [0\ 0]^T, & \text{if } \|\mathbf{x}_O(k) - \mathbf{x}_M(k)\|_2 \le c_O \\ \mathbf{x}_M(k), & \text{if } \|\mathbf{x}_O(k) - \mathbf{x}_M(k)\|_2 > c_O \end{cases} \tag{27}$$

where $c_O = 64$ pixels, and $\|\cdot\|_2$ denotes the L_2-norm.

Unlike Blinky and Sue, the pink ghost (P), Pinky, selects its target $\mathbf{y}_P$ based on both the position and the direction of motion of Ms. Pac-Man. In most instances, Pinky targets a position in $\mathcal{W}$ that is at a distance c_P from Ms. Pac-Man, and in the direction of Ms. Pac-Man's motion, as indicated by the value of the control input $\mathbf{u}_M$ (Fig. 6). However, when Ms. Pac-Man is moving in the positive y-direction (i.e., $\mathbf{u}_M(k) = a_1$), Pinky's target is c_P pixels above and to the left of *Ms. Pac-Man*. Therefore, Pinky's target can be modeled as follows [47]:

$$\mathbf{y}_P(k) = \mathbf{x}_M(k) + \mathbf{G}[\mathbf{u}_M(k)]\mathbf{c}_P \tag{28}$$

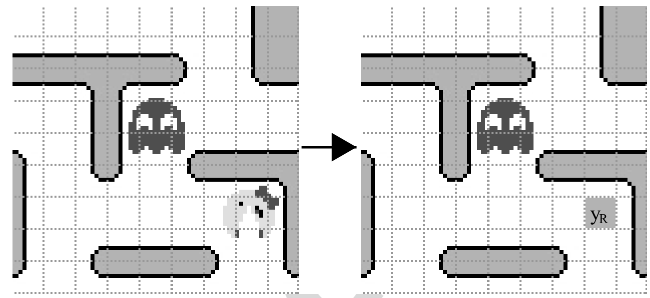


Fig. 4. Example of Blinky's target, $\mathbf{y}_R$. F4:1

where $\mathbf{c}_P = [32\ 32]^T$ pixels, and $\mathbf{G}(\cdot)$ is a matrix function of the control, defined as

$$\mathbf{G}(a_1) = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \quad \mathbf{G}(a_2) = \begin{bmatrix} -1 & 0 \\ 0 & 0 \end{bmatrix}$$
$$\mathbf{G}(a_3) = \begin{bmatrix} 0 & 0 \\ 0 & -1 \end{bmatrix} \quad \mathbf{G}(a_4) = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}. \tag{29}$$

The blue ghost (B), Inky, selects its target $\mathbf{y}_B$ based not only on the position and direction of motion of Ms. Pac-Man, but also on the position of the red ghost $\mathbf{x}_R$. As illustrated in Fig. 7, Inky's target is found by projecting the position of the red ghost in the direction of motion of Ms. Pac-Man ($\mathbf{u}_M$), about a point 16 pixels from $\mathbf{x}_M$, and in the direction $\mathbf{u}_M$. When Ms. Pac-Man is moving in the positive y-direction ($\mathbf{u}_M(k) = a_1$), however, the point for the projection is above and to the left of Ms. Pac-Man at a distance of 6 pixels. The reflection point can be defined as

$$\mathbf{y}_M^R(k) = \mathbf{x}_M(k) + \mathbf{G}[\mathbf{u}_M(k)]\mathbf{c}_B \tag{30}$$

where $\mathbf{c}_B = [16\ 16]^T$, and the matrix function $\mathbf{G}(\cdot)$ is defined as in (29). The position of the red ghost is then projected about the reflection point $\mathbf{y}_M^R$ in order to determine the target for the blue ghost [47]

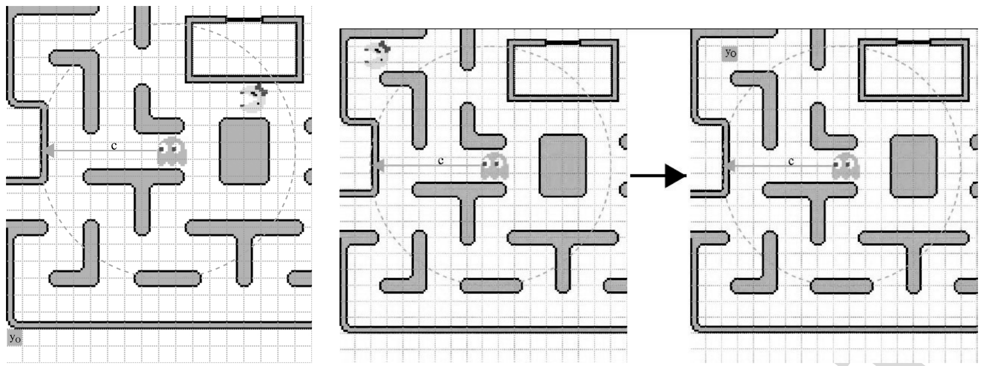
$$\mathbf{y}_B(k) = 2 \cdot \mathbf{y}_M^R(k) - \mathbf{x}_R(k) \tag{31}$$

as shown by the examples in Fig. 7.

C. Ghost Policy in Evasion and Scatter Modes

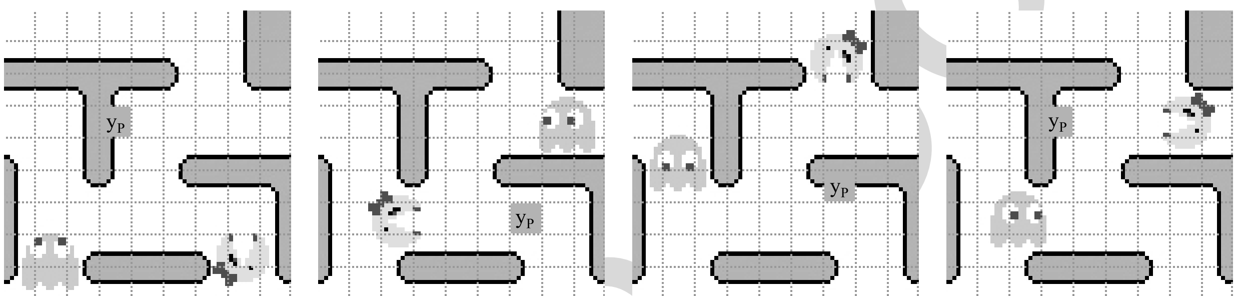
At the beginning of each level and following the death of Ms. Pac-Man, the ghosts are in scatter mode for seven seconds. In this mode, the ghosts do not pursue the player but, rather, move about the maze randomly. When a ghost reaches an intersection, it is modeled to select one of its admissible control inputs $\mathcal{U}_G(k)$ with uniform probability (excluding the possibility of reversing direction).

If Ms. Pac-Man eats a power pill, the ghosts immediately reverse direction and enter the evasion mode for a period of time that decreases with the game level. In evasion mode, the ghosts move randomly about the maze as in scatter mode but with a lower speed. When a ghost in evasion mode is captured by Ms. Pac-Man, it returns to the ghost pen and enters pursuit mode on exit. Ghosts that are not captured return to pursuit mode when the power pill becomes inactive.



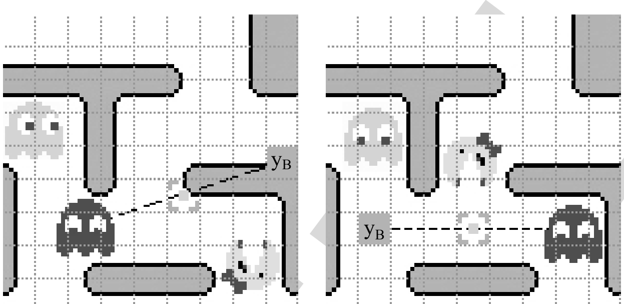
(a) (b)

Fig. 5. Examples of Sue's target, $\mathbf{y}_O$. (a) $\|\mathbf{x}_O(k) - \mathbf{x}_M(k)\|_2 \leq c_O$. (b) $\|\mathbf{x}_O(k) - \mathbf{x}_M(k)\|_2 > c_O$.



(a) (b) (c) (d)

Fig. 6. Examples of Pinky's target, $\mathbf{y}_P$. (a) If $\mathbf{u}_M(k) = a_1$. (b) If $\mathbf{u}_M(k) = a_2$. (c) If $\mathbf{u}_M(k) = a_3$. (d) If $\mathbf{u}_M(k) = a_4$.



(a) (b)

Fig. 7. Examples of Inky's target, $\mathbf{y}_B$. (a) If $\mathbf{u}_M(k) = a_1$. (b) If $\mathbf{u}_M(k) = a_3$.

VI. Methodology

This paper presents a methodology for optimizing the decision strategy of a computer player, referred to as the artificial *Ms. Pac-Man* player. A decision-tree representation of the game is obtained by using a computational geometry approach known as cell decomposition to decompose the obstacle-free workspace $\mathcal{W}_{\text{free}}$ into convex subsets, or cells, within which a path for Ms. Pac-Man can be easily generated [40]. As explained in Section VI-A, the cell decomposition is used to create a connectivity tree representing causal relationships between Ms. Pac-Man's position, and possible future paths [52]. The connectivity tree can then be transformed into a decision tree with utility nodes obtained from the utility function defined in Section VI-B. The optimal strategy for the artificial player is then computed and updated using the decision tree, as explained in Section VI-C.

A. Cell Decomposition and the Connectivity Tree

As a preliminary step, the corridors of the maze are decomposed into nonoverlapping rectangular cells by means of a line sweeping algorithm [53]. A cell, denoted κ_i, is defined as a closed and bounded subset of the obstacle-free space. The cell decomposition is such that a maze tunnel constitutes a single cell, as shown in Fig. 8. In the decomposition, two cells κ_i and κ_j are considered to be adjacent if and only if they share a mutual edge. The adjacency relationships of all cells in the workspace can be represented by a connectivity graph. A connectivity graph $\mathcal{G}$ is a nondirected graph, in which every node represents a cell in the decomposition of $\mathcal{W}_{\text{free}}$, and two nodes κ_i and κ_j are connected by an arc (κ_i, κ_j) if and only if the corresponding cells are adjacent.

Ms. Pac-Man can only move between adjacent cells, therefore, a causal relationship can be established from the adjacency relationships in the connectivity graph, and represented by a connectivity tree, as was first proposed in [52]. Let $\kappa[\mathbf{x}]$ denote the cell containing a point $\mathbf{x} = [xy]^T \in \mathcal{W}_{\text{free}}$. Given an initial position $\mathbf{x}_0$, and a corresponding cell $\kappa[\mathbf{x}_0]$, the connectivity tree associated with $\mathcal{G}$, and denoted by $\mathcal{C}$, is defined as an acyclic tree graph with root $\kappa[\mathbf{x}_0]$, in which every pair of nodes κ_i and κ_j connected by an arc are also connected by an arc

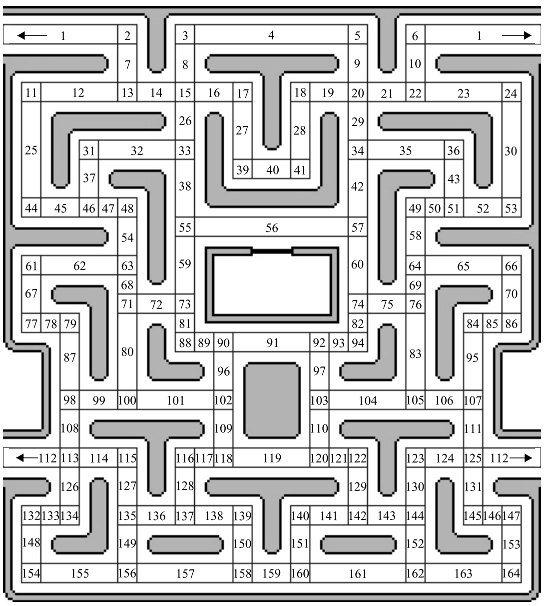


Fig. 8. Cell decomposition of *Ms. Pac-Man* second maze.

in $\mathcal{G}$. As in the connectivity graph, the nodes of a connectivity tree represent void cells in the decomposition. Given the position of Ms. Pac-Man at any time k, a connectivity tree with root $\kappa[\mathbf{x}_M(k)]$ can be readily determined from $\mathcal{G}$, using the methodology in [52]. Each branch of the tree then represents a unique sequence of cells that may be visited by *Ms. Pac-Man*, starting from $\mathbf{x}_M(k)$.

B. Ms. Pac-Man's Profit Function

Based on the game objectives described in Section II, the instantaneous profit of a decision $\mathbf{u}_M(k)$ is defined as a weighted sum of the risk of being captured by the ghosts, denoted by R, and the reward gained by reaching one of targets, denoted by V. Let $d(\cdot)$, $p(\cdot)$, $f(\cdot)$, and $b(\cdot)$ denote the rewards associated with reaching the pills, power pills, ghosts, and bonus items, respectively. The corresponding weights, ω_d, ω_p, ω_f, and ω_b denote known constants that are chosen heuristically by the user, or computed via a learning algorithm, such as temporal difference [39]. Then, the total reward can be defined as the sum of the rewards from each target type

$$V[\mathbf{s}(k), \mathbf{u}_M(k)] = \omega_d d[\mathbf{s}(k), \mathbf{u}_M(k)] + \omega_p p[\mathbf{s}(k), \mathbf{u}_M(k)] + \omega_f f[\mathbf{s}(k), \mathbf{u}_M(k)] + \omega_b b[\mathbf{s}(k), \mathbf{u}_M(k)] \quad (32)$$

and can be computed using the models presented in Section V, as follows.

The pill reward function $d(\cdot)$ is a binary function that represents a positive reward of 1 unit if Ms. Pac-Man is expected to eat a pill as result of the chosen control input $\mathbf{u}_M$, and is otherwise zero, i.e.,

$$d[\mathbf{x}(k), \mathbf{u}_M(k), \mathbf{z}(k)] = \begin{cases} 0, & \text{if } \mathbf{D}[\mathbf{x}_M(k)] \neq 1 \\ 1, & \text{if } \mathbf{D}[\mathbf{x}_M(k)] = 1. \end{cases} \quad (33)$$

A common strategy implemented by both human and artificial players is to use power pills to ambush the ghosts. When utilizing this strategy, a player waits near a power pill until the ghosts are near, it then eats the pill and pursues the ghosts which have entered evasion mode. The reward associated with each power pill can be modeled as a function of the minimum distance between Ms. Pac-Man and each ghost G

$$\rho_G[\mathbf{x}_M(k)] \triangleq \min |\mathbf{x}_M(k) - \mathbf{x}_G(k)| \quad (34)$$

where $|\cdot|$ denotes the L_1-norm. In order to take into account the presence of the obstacles (walls), the minimum distance in (34) is computed from the connectivity tree $\mathcal{C}$ obtained in Section VI-A, using the A * algorithm [53]. Then, letting ρ_D denote the maximum distance at which Ms. Pac-Man should eat a power pill, the power-pill reward can be written as

$$p[\mathbf{x}(k), \mathbf{u}_M(k), \mathbf{z}(k)] = \begin{cases} 0, & \text{if } \mathbf{D}[\mathbf{x}_M(k)] \neq -1 \\ \sum\limits_{G \in I_G} g[\mathbf{x}(k)], & \text{if } \mathbf{D}[\mathbf{x}_M(k)] = -1 \end{cases} \quad (35)$$

where

$$g[\mathbf{x}_M(k), \mathbf{x}_G(k)] = \vartheta_- \times H\{\rho_G[\mathbf{x}_M(k)] - \rho_D\} + \vartheta_+ \times H\{\rho_D - \rho_G[\mathbf{x}_M(k)]\}. \quad (36)$$

The parameters ϑ_- and ϑ_+ are the weights that represent the desired tradeoff between the penalty and reward associated with the power pill.

Because the set of admissible decisions for a ghost is a function of its position in the maze, the probability that a ghost in evasion mode will transition to a state $\mathbf{x}_G(k)$ from a state $\mathbf{x}_G(k-1)$, denoted by $P[\mathbf{x}_G(k)|\mathbf{x}_G(k-1)]$, can be computed from the cell decomposition (Fig. 8). Then, the instantaneous reward for reaching (eating) a ghost G in evasion mode is

$$f[\mathbf{x}(k), \mathbf{u}_M(k), \mathbf{z}(k)] = \begin{cases} 0, & \text{if } \mathbf{x}_G(k) \neq \mathbf{x}_M(k) H[\delta_G(k)-1] \\ P[\mathbf{x}_G(k)|\mathbf{x}_G(k-1)]\zeta(k), & \text{if } \mathbf{x}_G(k) = \mathbf{x}_M(k) \end{cases} \quad (37)$$

where $\delta_G(k)$ represents the mode of motion for ghost G (Section IV), and the function

$$\zeta(k) = \left\{ 5 - \sum_{G \in I_G} H[\delta_G(k) - 1] \right\}^2 \quad (38)$$

is used to increase the reward quadratically with the number of ghosts reached.

Like the ghosts, the bonus items are moving targets that, when eaten, increase the game score. Unlike the ghosts, however, they never pursue Ms. Pac-Man, and, if uneaten after a given period of time they simply leave the maze. Therefore, at any time during the game, an attractive potential function

$$U_b(\mathbf{x}) = \begin{cases} \rho_F^2(\mathbf{x}), & \text{if } \rho_F(\mathbf{x}) \leq \rho_b \\ 0, & \text{if } \rho_F(\mathbf{x}) > \rho_b \end{cases}, \quad \mathbf{x} \in \mathcal{W}_{\text{free}} \quad (39)$$

can be used to pull Ms. Pac-Man toward the bonus item with a virtual force

$$F_b(\mathbf{x}) = -\nabla U_b(\mathbf{x}) \quad (40)$$

that decreases with ρ_F. The distance ρ_F is defined by substituting G with F in (34), ρ_b is a positive constant that represents the influence distance of the bonus item [53], and ∇ is the gradient operator. The instantaneous reward function for the bonus item is then defined such that the player is rewarded for moving toward the bonus item, i.e.,

$$b\left[\mathbf{x}(k), \mathbf{u}_M(k), \mathbf{z}(k)\right] = \operatorname{sgn}\left\{F_b\left[\mathbf{x}_M(k)\right]\right\} \circ \mathbf{u}_M(k). \quad (41)$$

The weight ω_b in (32) is then chosen based on the type and value of the bonus item for the given game level.

The instantaneous risk function is defined as the sum of the immediate risk posed by each of the four ghosts

$$R\left[\mathbf{x}(k), \mathbf{u}_M(k), \mathbf{z}(k)\right] = \sum_{G \in I_G} R_G\left[\mathbf{x}(k), \mathbf{u}_M(k), \mathbf{z}(k)\right] \quad (42)$$

where the risk of each ghost R_G depends on its mode of motion. In evasion mode ($\delta_G = 1$), a ghost G poses no risk to Ms. Pac-Man, because it cannot capture her. In scatter mode ($\delta_G = 0$), the risk associated with a ghost G is modeled using a repulsive potential function

$$U_G(\mathbf{x}) = \begin{cases} \left(\frac{1}{\rho_G(\mathbf{x})} - \frac{1}{\rho_0}\right)^2, & \text{if} \quad \rho_G(\mathbf{x}) \le \rho_0 \\ 0, & \text{if} \quad \rho_G(\mathbf{x}) > \rho_0 \end{cases}, \quad \mathbf{x} \in \mathcal{W}_{\text{free}} \quad (43)$$

that repels Ms. Pac-Man with a force

$$F_G(\mathbf{x}) = -\nabla U_G(\mathbf{x}) \quad (44)$$

ρ_0 is the influence distance of Ms. Pac-Man, such that when Ms. Pac-Man is farther than ρ_0 from a ghost, the ghost poses zero risk. When a ghost is in the ghost pen or otherwise inactive, its distance to Ms. Pac-Man is treated as infinite.

The risk of a ghost in scatter mode is modeled such that Ms. Pac-Man is penalized for moving toward the ghost, i.e.,

$$R_G\left[\mathbf{x}(k), \mathbf{u}_M(k), \mathbf{z}(k)\right] = \operatorname{sgn}\left\{F_G[\mathbf{x}_M(k)]\right\} \circ \mathbf{u}_M(k) \quad (45)$$

for $\delta_G(k) = -1$. In pursuit mode [$\delta_G(k) = 0$], the ghosts are more aggressive and, thus, the instantaneous risk is modeled as the repulsive potential

$$R_G\left[\mathbf{x}(k), \mathbf{u}_M(k), \mathbf{z}(k)\right] = U_G(\mathbf{x}). \quad (46)$$

Finally, the risk of being captured by a ghost is equal to a large positive constant χ defined by the user

$$R_G\left[\mathbf{x}(k), \mathbf{u}_M(k), \mathbf{z}(k)\right] = \chi, \quad \text{for} \quad \tau[\mathbf{x}_M(k)] = \tau[\mathbf{x}_G(k)]. \quad (47)$$

This emphasizes the risk of losing a life, which would cause the game to end sooner and the score to be significantly lower.

Then the instantaneous profit function is a sum of the reward V and risk R

$$J[\mathbf{u}_M(k)] = V[\mathbf{s}(k), \mathbf{u}_M(k)] + R[\mathbf{x}(k), \mathbf{u}_M(k), \mathbf{z}(k)] \quad (48)$$

which is evaluated at each node in a decision tree constructed using the cell decomposition method described above.

C. Decision Tree and Optimal Strategy

As was first shown in [52], the connectivity tree $\mathcal{G}$ obtained via cell decomposition in Section VI-A can be transformed into a decision tree T_i that also includes action and utility nodes. A decision tree is a directed acyclic graph with a tree-like structure in which the root is the initial state, decision nodes represent all possible decisions, and state (or chance) nodes represent the state values resulting from each possible decision [54]–[56]. Each branch in the tree represents the outcomes of a possible strategy σ_i and terminates in leaf (or utility) node that contains the value of the strategy's cumulative profit $J_{i,f}$.

Let the tuple $T_i = \{C, D, J, A\}$ represent a decision tree comprising a set of chance nodes C, a set of decision nodes D, the utility function J, and a set of directed arcs A. At any time $t_i \in (t_0, t_F]$, a decision tree T_i for Ms. Pac-Man can be obtained from $\mathcal{G}$ using the following assignments.

1) The root is the cell $\kappa_i \in \mathcal{G}$ occupied by Ms. Pac-Man at time t_i.
2) Every chance node $\kappa_j \in C$ represents a cell in $\mathcal{G}$.
3) For every cell $\kappa_j \in C$, a directed arc $(\kappa_j, \kappa_l) \in A$ is added iff $\exists(\kappa_j, \kappa_l) \in \mathcal{G}$, $j \neq l$. Then, (κ_j, κ_l) represents the action decision to move from κ_j to κ_l.
4) The utility node at the end of each branch represents the cumulative profit $J_{i,f}$ of the corresponding strategy, σ_i, defined in (4).

Using the above assignments, the instantaneous profit can be computed for each node as the branches of the tree are grown using Ms. Pac-Man's profit function, presented in Section VI-B. When the slice corresponding to t_f is reached, the cumulative profit $J_{i,f}$ of each branch is found and assigned to its utility node. Because the state of the game can change suddenly as result of random ghost behavior, an exponential discount factor is used to discount future profits in $J_{i,f}$, and favor the profit that may be earned in the near future. From T_i, the optimal strategy σ_i^* is determined by choosing the action corresponding to the branch with the highest value of $J_{i,f}$. As explained in Section III, a new decision tree is generated when t_f is reached, or when the state observations differ from the model prediction, whichever occurs first.

VII. Simulation Results

The simulation results presented in this paper are obtained from the Microsoft's *Revenge of the Arcade* software, which is identical to the original arcade version of *Ms. Pac-Man*. The results in Section VII-A validate the ghost models presented in Section V, and the simulations in Section VII-B demonstrate the effectiveness of the model-based artificial player presented in Section VI. Every game simulated in this section is played from beginning to end. The artificial player is coded in C#, and runs in real time on a laptop with a Core-2 Duo 2.13-GHz CPU, and 8-GB RAM. At every instant, indexed by k, the state of the game $\mathbf{s}(k)$ is extracted from screen-capture images of the game using the algorithm presented in [41]. Based on the observed state value $\mathbf{s}(k)$, the control input to Ms. Pac-Man $\mathbf{u}_M$ is computed from the decision tree T_i, and implemented using simulated keystrokes. Based on $\mathbf{s}(k)$, the tree T_i is updated at

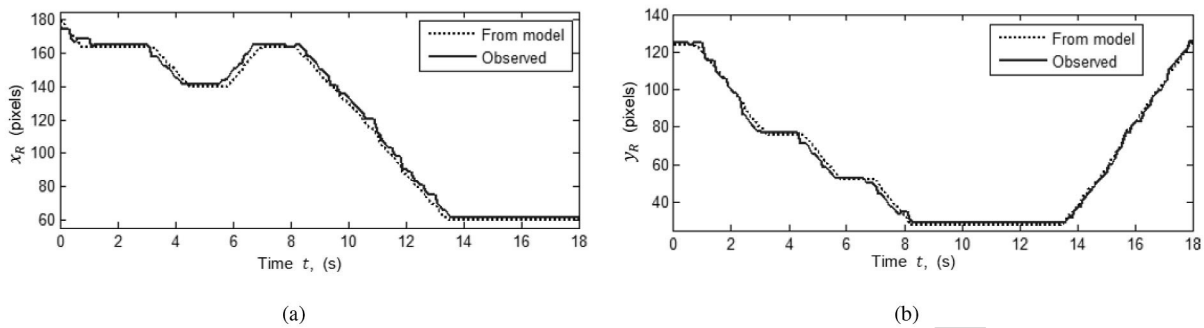


Fig. 9. Example of simulated and observed trajectories for the red ghost in pursuit mode.

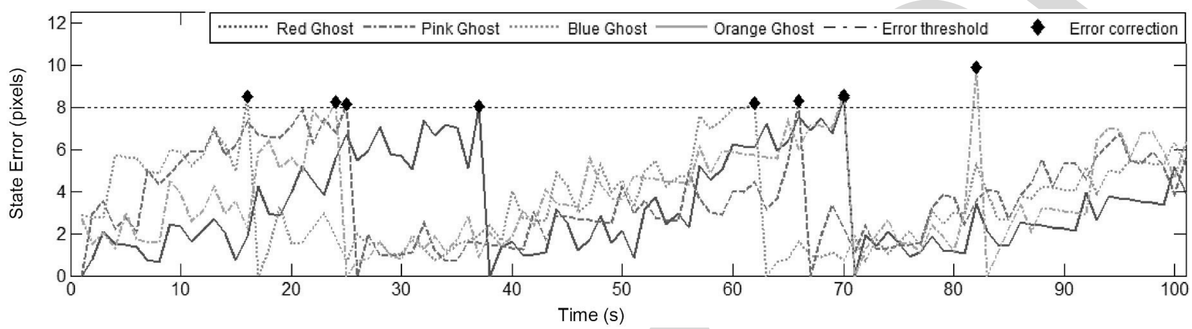


Fig. 10. Example of ghost-state error histories, and model updates (diamonds).

selected instants $t_i \in (t_0, t_f]$, as explained in Section VI-C. The highest recorded time to compute a decision was 0.09 s, and the mean times for the two most expensive steps of extracting the game state and computing the decision tree are on the order of 0.015 and 0.05 s, respectively.

A. *Adversary Model Validation*

The models of the ghosts in pursuit mode, presented in Section V-B, are validated by comparing the trajectories of the ghosts extracted from the screen capture code to those generated by integrating the models numerically using the same game conditions. When the ghosts are in other modes, their random decisions are assumed to be uniformly distributed [47]. The ghosts' state histories are extracted from screen-capture images while the game is being played by a human player. Subsequently, the ghost models are integrated using the trajectory of *Ms. Pac-Man* extracted during the same time interval. Fig. 9 shows an illustrative example of actual (solid line) and simulated (dashed line) trajectories for the red ghost, in which the model generated a path identical to that observed from the game. The small error between the two trajectories, in this case, is due entirely to the screen-capture algorithm.

The ghosts' models are validated by computing the percentage of ghost states that are predicted correctly during simulated games. Because the ghosts only make decisions at maze intersections, the error in a ghost's state is computed every time the ghost is at a distance of 10 pixels from an intersection. Then, the state is considered to be predicted correctly if the error between the observed and predicted values of the state is less than 8 pixels. If the error is larger than 8 pixels, the prediction is considered to be incorrect. When an incorrect prediction occurs, the simulated ghost state $\mathbf{x}_G$ is updated online using the observed state value as an initial condition in the ghost dynamic equation (17). Fig. 10 shows the error between simulated and observed state histories for all four ghosts during a sample time interval.

TABLE IV
GHOST MODEL VALIDATION RESULTS

Ghost	No. Correct Decisions	No. Incorrect Decisions	Model Accuracy
Red	21072	630	97.1%
Pink	19500	718	96.45%
Blue	19499	759	96.25%
Orange	19634	842	95.89%

The errors in ghost model predictions were computed by conducting game simulations until approximately 20 000 decisions were obtained for each ghost. The results obtained from these simulations are summarized in Table IV. In total, 79 705 ghost decisions were obtained, for an average model accuracy (the ratio of successes to total trials) of 96.4%. As shown in Table IV, the red ghost model is the least prone to errors, followed by the pink ghost model, the blue ghost model, and, last, the orange ghost model, which has the highest error rate. The model errors are due to imprecisions when decoding the game state from the observed game image, computation delay, missing state information (e.g., when ghost images overlap on the screen), and imperfect timing by the player when making turns, which has a small effect on Ms. Pac-Man's speed, as explained in Section II.

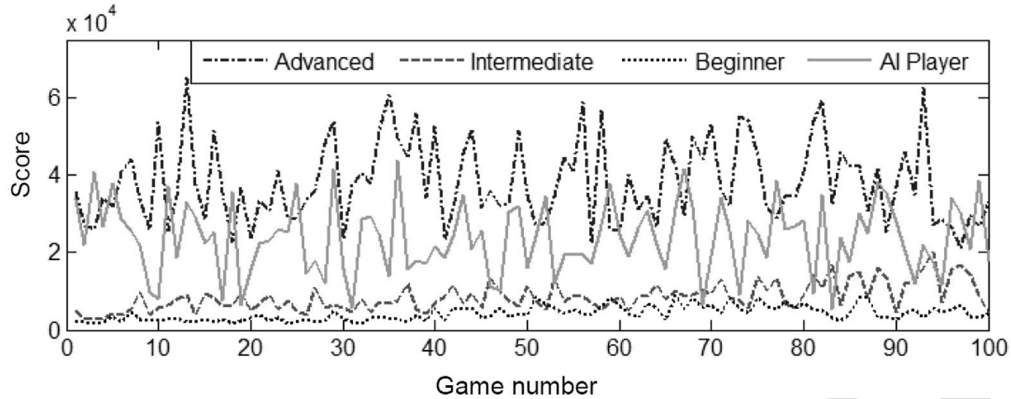


Fig. 11. Time histories of game scores obtained by human and AI players.

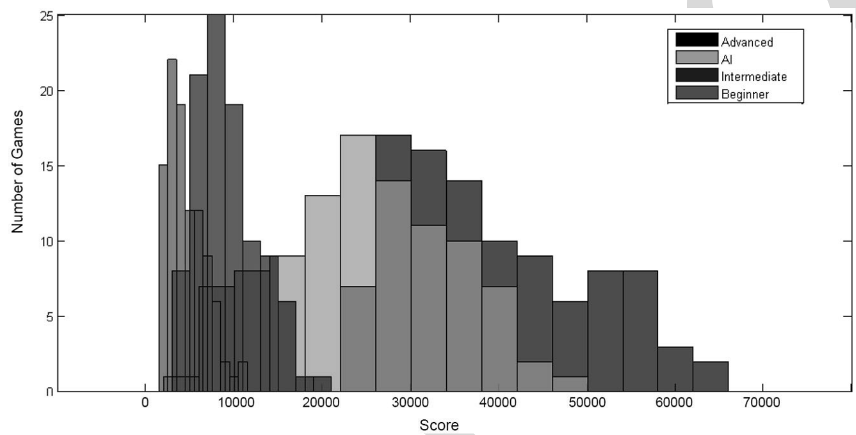


Fig. 12. Player score distribution for 100 games.

The difference in the accuracy of different ghost models arises from the fact that the differential equations in (26)–(28) and (31) include different state variables and game parameters. For example, the pink ghost model has a higher error rate than the red ghost model because its target position $\mathbf{y}_P$ is a function of Ms. Pac-Man state and control input, and these variables are both susceptible to observation errors, while the red ghost model only depends on Ms. Pac-Man state. Thus, the pink ghost model is subject not only to observation errors in $\mathbf{x}_M$, which cause errors in the red ghost model, but also to observation errors in $\mathbf{u}_M$.

B. Game Strategy Performance

The artificial player strategies are computed using the approach described in Section VI, where the weighting coefficients are $\omega_V = 1$, $\omega_R = 0.4$, $\omega_d = 8$, $\omega_p = 3$, $\omega_f = 15$, $\omega_b = 0.5$, $\chi = 20\ 000$, $\vartheta_- = -2.2$, and $\vartheta_+ = 1$, and are chosen by the user based on the desired tradeoff between the multiple conflicting objectives of Ms. Pac-Man [50]. The distance parameters are $\rho_0 = 150$ pixels and $\rho_b = 129$ pixels, and are chosen by the user based on the desired distance of influence for ghost avoidance and bonus item, respectively [53]. The time histories of the scores during 100 games are plotted in Fig. 11, and the score distributions are shown in Fig. 12. The minimum, average, and maximum scores are summarized in Table V.

TABLE V
PERFORMANCE RESULT SUMMARY OF AI AND HUMAN PLAYERS

Player	Minimum Score	Average Score	Maximum Score
AI	5210	23767	43720
Beginner	1610	4137	9210
Intermediate	2760	8542	20180
Advanced	21400	38172	65200

From these results, it can be seen that the model-based artificial (AI) player presented in this paper outperforms most of the computer players presented in the literature [8]–[14], which display average scores between 9000 and 18 000 and maximum scores between 20 000 and 36 280, where the highest score of 36 280 was achieved by the winner of the last *Ms. Pac-Man* screen competition at the 2011 Conference on Computational Intelligence and Games [14].

Because expert human players routinely outperform computer players and easily achieve scores over 65 000, the AI player presented in this paper is also compared to human players of varying skill levels. The beginner player is someone who has never played the game before, the intermediate player has basic knowledge of the game and some prior experience, and the advanced player has detailed knowledge of the game mechanics, and has previously played many games. All players completed the 100 games over the course of a few weeks, during multiple sittings, and over time displayed the performance plotted in Fig. 11. From Table V, it can be seen that the AI player presented in this paper performs significantly better than both the beginner and intermediate players on average, with its highest score being 43 720. However, the advanced player outperforms the AI player on average, and has a much higher maximum score of 65 200.

It can also be seen in Fig. 11 that the beginner and intermediate players improve their scores over time, while the advanced player does not improve significantly. In particular, when a simple least squares linear regression was performed on these game scores, the slope values were found to be 10.23 (advanced), 2.01 (AI), 74.32 (intermediate), and 36.67 (beginner). Furthermore, a linear regression t-test aimed at determining whether the slope of the regression line differs significantly from zero with 95% confidence was applied to the data in Fig. 11, showing that while the intermediate and beginner scores increase over time, the AI and advanced scores display a slope that is not statistically significantly different from zero (see [57] for a description of these methods). This suggests that beginner and intermediate players improve their performance more significantly by learning from the game, while the advanced player may have already reached its maximum performance level.

From detailed game data (not shown for brevity), it was found that human players are able to learn (or memorize) the first few levels of the game, and initially make fewer errors than the AI player. On the other hand, the AI player displays better performance than the human players later in the game, during high game levels when the game characters move faster, and the mazes become harder to navigate. These conditions force players to react and make decisions more quickly, and are found to be significantly more difficult by human players. Because the AI player can update its decision tree and strategy very frequently, the effects of game speed on the AI player's performance are much smaller than on human players. Finally, although the model-based approach presented in this paper does not include learning, methods such as temporal difference [39] will be introduced in future work to further improve the AI player's performance over time.

VIII. Conclusion

A model-based approach is presented for computing optimal decision strategies in the pursuit-evasion game *Ms. Pac-Man*. A model of the game and adversary dynamics are presented in the form of a decision tree that is updated over time. The decision tree is derived by decomposing the game maze using a cell decomposition approach, and by defining the profit of future decisions based on adversary state predictions, and real-time state observations. Then, the optimal strategy is computed from the decision tree over a finite time horizon, and implemented by an artificial (AI) player in real time, using a screen-capture interface. Extensive game simulations are used to validate the models of the ghosts presented in this paper, and to demonstrate the effectiveness of the optimal game strategies obtained from the decision trees. The AI player is shown to outperform beginner and intermediate human players, and to achieve the highest score of 43 720. It is also shown that although an advanced player outperforms the AI player, the AI player is better able to handle high game levels, in which the speed of the characters and spatial complexity of the mazes become more challenging.

Acknowledgment

The authors would like to thank R. Jackson at Stanford University, Stanford, CA, USA, for his contributions and suggestions.

References

[1] T. Muppirala, A. Bhattacharya, and S. Hutchin, "Surveillance strategies for a pursuer with finite sensor range," *Int. J. Robot. Res.*, vol. 26, no. 3, pp. 233–253, 2007.
[2] S. Ferrari, R. Fierro, B. Perteet, C. Cai, and K. Baumgartner, "A geometric optimization approach to detecting and intercepting dynamic targets using a mobile sensor network," *SIAM J. Control Optim.*, vol. 48, no. 1, pp. 292–320, 2009.
[3] V. Isler, S. Kannan, and S. Khanna, "Randomized pursuit-evasion with limited visibility," in *Proc. ACM-SIAM Symp. Discrete Algorithms*, 2004, pp. 1053–1063.
[4] V. Isler, D. Sun, and S. Sastry, "Roadmap based pursuit-evasion and collision avoidance," in *Proc. Robot. Syst. Sci.*, 2005.
[5] S. M. Lucas and G. Kendall, "Evolutionary computation and games," *IEEE Comput. Intell. Mag.*, vol. 1, no. 1, pp. 10–18, 2006.
[6] J. Schrum and R. Miikkulainen, "Discovering multimodal behavior in *Ms. Pac-Man* through evolution of modular neural networks," *IEEE Trans. Comput. Intell. AI Games*, vol. 8, no. 1, pp. 67–81, Mar. 2016.
[7] S. M. Lucas, "*Ms. Pac-Man* competition," *SIGEVOlution*, vol. 2, no. 4, pp. 37–38, Dec. 2007.
[8] N. Bell *et al.*, "Ghost direction detection and other innovations for *Ms. Pac-Man*," in *Proc. IEEE Symp. Comput. Intell. Games*, Aug. 2010, pp. 465–472.
[9] R. Thawonmas and H. Matsumoto, "Automatic controller of *Ms. Pac-Man* and its performance: Winner of the IEEE CEC 2009 software agent *Ms. Pac-Man* competition," in *Proc. Asia Simul. Conf.*, Oct. 2009.
[10] T. Ashida, T. Miyama, H. Matsumoto, and R. Thawonmas, "ICE Pambush 4," in *Proc. IEEE Symp. Comput. Intell. Games*, 2010.
[11] T. Miyama, A. Yamada, Y. Okunishi, T. Ashida, and R. Thawonmas, "ICE Pambush 5," in *Proc. IEEE Symp. Comput. Intell. Games*, 2011.
[12] R. Thawonmas and T. Ashida, "Evolution strategy for optimizing parameters in *Ms. Pac-Man* controller ICE Pambush 3," in *Proc. IEEE Symp. Comput. Intell. Games*, 2010, pp. 235–240.
[13] M. Emilio, M. Moises, R. Gustavo, and S. Yago, "Pac-mAnt: Optimization based on ant colonies applied to developing an agent for *Ms. Pac-Man*," in *Proc. IEEE Symp. Comput. Intell. Games*, 2010, pp. 458–464.
[14] N. Ikehata and T. Ito, "Monte-Carlo tree search in *Ms. Pac-Man*," in *Proc. IEEE Conf. Comput. Intell. Games*, Sep. 2011, pp. 39–46.
[15] A. A. Ghazanfar and M. A. Nicolelis, "Spatiotemporal properties of layer v neurons of the rat primary somatosensory cortex," *Cerebral Cortex*, vol. 9, no. 4, pp. 348–361, 1999.
[16] M. A. Nicolelis, L. A. Baccala, R. Lin, and J. K. Chapin, "Sensorimotor encoding by synchronous neural ensemble activity at multiple levels of the somatosensory system," *Science*, vol. 268, no. 5215, pp. 1353–1358, 1995.
[17] M. Kawato, "Internal models for motor control and trajectory planning," *Current Opinion Neurobiol.*, vol. 9, no. 6, pp. 718–727, 1999.
[18] D. M. Wolpert, R. C. Miall, and M. Kawato, "Internal models in the cerebellum," *Trends Cogn. Sci.*, vol. 2, no. 9, pp. 338–347, 1998.
[19] J. W. Krakauer, M.-F. Ghilardi, and C. Ghez, "Independent learning of internal models for kinematic and dynamic control of reaching," *Nature Neurosci.*, vol. 2, no. 11, pp. 1026–1031, 1999.
[20] M. A. Sommer and R. H. Wurtz, "Brain circuits for the internal monitoring of movements," *Annu. Rev. Neurosci.*, vol. 31, pp. 317, 2008.
[21] T. B. Crapse and M. A. Sommer, "The frontal eye field as a prediction map," *Progr. Brain Res.*, vol. 171, pp. 383–390, 2008.
[22] K. Doya, K. Samejima, K.-i. Katagiri, and M. Kawato, "Multiple model-based reinforcement learning," *Neural Comput.*, vol. 14, no. 6, pp. 1347–1369, 2002.

[23] A. J. Calise and R. T. Rysdyk, "Nonlinear adaptive flight control using neural networks," *IEEE Control Syst.*, vol. 18, no. 6, pp. 14–25, 1998.

[24] S. Ferrari and R. F. Stengel, "Online adaptive critic flight control," *J. Guid. Control Dyn.*, vol. 27, no. 5, pp. 777–786, 2004.

[25] C. G. Atkeson and J. C. Santamaria, "A comparison of direct and model-based reinforcement learning," in *Proc. Int. Conf. Robot. Autom.*, 1997.

[26] C. Guestrin, R. Patrascu, and D. Schuurmans, "Algorithm-directed exploration for model-based reinforcement learning in factored MDPs," in *Proc. Int. Conf. Mach. Learn.*, 2002, pp. 235–242.

[27] J. Si, *Handbook of Learning and Approximate Dynamic Programming*, New York, NY, USA: Wiley, 2004, vol. 2.

[28] A. Fitzgerald and C. B. Congdon, "A rule-based agent for *Ms. Pac-Man*," in *Proc. IEEE Congr. Evol. Comput.*, 2009, pp. 2646–2653.

[29] D. J. Gagne and C. B. Congdon, "Fright: A flexible rule-based intelligent ghost team for *Ms. Pac-Man*," in *Proc. IEEE Conf. Comput. Intell. Games*, 2012, pp. 273–280.

[30] N. Wirth and M. Gallagher, "An influence map model for playing *Ms. Pac-Man*," in *Proc. IEEE Symp. Comput. Intell. Games*, Dec. 2008, pp. 228–233.

[31] L. DeLooze and W. Viner, "Fuzzy Q-learning in a nondeterministic environment: Developing an intelligent *Ms. Pac-Man* agent," in *Proc. IEEE Symp. Comput. Intell. Games*, 2009, pp. 162–169.

[32] A. Alhejali and S. Lucas, "Evolving diverse *Ms. Pac-Man* playing agents using genetic programming," in *Proc. U.K. Workshop Comput. Intell.*, 2010.

[33] A. Alhejali and S. Lucas, "Using a training camp with genetic programming to evolve *Ms. Pac-Man* agents," in *Proc. IEEE Conf. Comput. Intell. Games*, 2011, pp. 118–125.

[34] T. Pepels, M. H. Winands, and M. Lanctot, "Real-time Monte Carlo tree search in Ms. Pac-Man," *IEEE Trans. Comput. Intell. AI Games*, vol. 6, no. 3, pp. 245–257, 2014.

[35] K. Q. Nguyen and R. Thawonmas, "Monte Carlo tree search for collaboration control of ghosts in *Ms. Pac-Man*," *IEEE Trans. Comput. Intell. AI Games*, vol. 5, no. 1, pp. 57–68, 2013.

[36] D. P. Bertsekas and S. Ioffe, "Temporal differences-based policy iteration and applications in neuro-dynamic programming," Lab. Inf. Decision Syst. Rep. 1996.

[37] B. Tong, C. M. Ma, and C. W. Sung, "A Monte-Carlo approach for the endgame of *Ms. Pac-Man*," in *Proc. IEEE Conf. Comput. Intell. Games*, Sep. 2011, pp. 9–15.

[38] S. Samothrakis, D. Robles, and S. Lucas, "Fast approximate max-n Monte Carlo tree search for *Ms. Pac-Man*," *IEEE Trans. Comput. Intell. AI Games*, vol. 3, no. 2, pp. 142–154, 2011.

[39] G. Foderaro, V. Raju, and S. Ferrari, "A model-based approximate λ iteration approach to online evasive path planning and the video game *Ms. Pac-Man*," *J. Control Theory Appl.*, vol. 9, no. 3, pp. 391–399, 2011.

[40] S. Ferrari and C. Cai, "Information-driven search strategies in the board game of CLUE," *IEEE Trans. Syst. Man Cybern. B, Cybern.*, vol. 39, no. 3, pp. 607–625, Jun. 2009.

[41] G. Foderaro, A. Swingler, and S. Ferrari, "A model-based cell decomposition approach to on-line pursuit-evasion path planning and the video game *Ms. Pac-Man*," in *Proc. IEEE Conf. Comput. Intell. Games*, 2012, pp. 281–287.

[42] M. Kaess, A. Ranganathan, and F. Dellaert, "iSAM: Incremental smoothing and mapping," *IEEE Trans. Robot.*, vol. 24, no. 6, pp. 1365–1378, Dec. 2008.

[43] M. Kaess *et al.*, "iSAM2: Incremental smoothing and mapping using the Bayes tree," *Int. J. Robot.*, vol. 31, no. 2, pp. 216–235, Feb. 2012.

[44] H. Wei and S. Ferrari, "A geometric transversals approach to analyzing the probability of track detection for maneuvering targets," *IEEE Trans. Comput.*, vol. 63, no. 11, pp. 2633–2646, 2014.

[45] H. Wei *et al.*, "Camera control for learning nonlinear target dynamics via Bayesian nonparametric Dirichlet-process Gaussian-process (DP-GP) models," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2014, pp. 95–102.

[46] W. Lu, G. Zhang, and S. Ferrari, "An information potential approach to integrated sensor path planning and control," *IEEE Trans. Robot.*, vol. 30, no. 4, pp. 919–934, 2014.

[47] J. Pittman, "The Pac-Man dossier," [Online]. Available: http://home.comcast.net/jpittman2/pacman/pacmandossier.html

[48] I. Szita and A. Lõrincz, "Learning to play using low-complexity rule-based policies: Illustrations through *Ms. Pac-Man*," *J. Artif. Intell. Res.*, pp. 659–684, 2007.

[49] M. Mateas, "Expressive AI: Games and artificial intelligence," in *Proc. DIGRA Conf.*, 2003.

[50] R. F. Stengel, *Optimal Control and Estimation*, New York, NY, USA: Dover, 1986.

[51] I. Szita and A. Lorincz, "Learning to play using low-complexity rule-based policies: Illustrations through *Ms. Pac-Man*," *J. Artif. Intell. Res.*, vol. 30, pp. 659–684, 2007.

[52] C. Cai and S. Ferrari, "Information-driven sensor path planning by approximate cell decomposition," *IEEE Trans. Syst. Man Cybern. B, Cybern.*, vol. 39, no. 3, pp. 672–689, Jun. 2009.

[53] J.-C. Latombe, *Robot Motion Planning*, Norwell, MA, USA: Kluwer, 1991.

[54] B. M. E. Moret, "Decision trees and diagrams," *ACM Comput. Surv.*, vol. 14, no. 4, pp. 593–623, 1982.

[55] F. V. Jensen and T. D. Nielsen, *Bayesian Networks and Decision Graphs*, New York, NY, USA: Springer-Verlag, 2001.

[56] M. Diehl and Y. Y. Haimes, "Influence diagrams with multiple objectives and tradeoff analysis," *IEEE Trans. Syst. Man Cyber. A*, vol. 34, no. 3, pp. 293–304, 2004.

[57] D. C. Montgomery, E. A. Peck, and G. G. Vining, *Introduction to Linear Regression Analysis*, New York, NY, USA: Wiley, 2012, vol. 821.

Greg Foderaro (M'XX) received the B.S. degree in mechanical engineering from Clemson University, Clemson, SC, USA, in 2009 and the Ph.D. degree in mechanical engineering and materials science from Duke University, Durham, NC, USA, in 2013. Q5

He is currently a Staff Engineer at Applied Research Associates, Inc. His research interests are in underwater sensor networks, robot path planning, multiscale dynamical systems, pursuit-evasion games, and spiking neural networks.

Ashleigh Swingler (M'XX) received the B.S. and M.S. degrees in mechanical engineering from Duke University, Durham, NC, USA, in 2010 and 2012, respectively. She is currently working toward the Ph.D. degree in the Department of Mechanical Engineering and Materials Science, Duke University. Q6

Her main research interests include disjunctive programming and approximate cell decomposition applied to robot and sensor path planning.

Silvia Ferrari (SM'XX) received the B.S. degree from Embry-Riddle Aeronautical University, Daytona Beach, FL, USA and the M.A. and Ph.D. degrees from Princeton University, Princeton, NJ, USA. Q7

She is a Professor of Mechanical and Aerospace Engineering at Cornell University, Ithaca, NY, USA, where she also directs the Laboratory for Intelligent Systems and Controls (LISC). Prior to joining the Cornell faculty, she was Professor of Engineering and Computer Science at Duke University, Durham, NC, USA, where she was also the Founder and Director of the NSF Integrative Graduate Education and Research Traineeship (IGERT) program on Wireless Intelligent Sensor Networks (WISeNet), and a Faculty Member of the Duke Institute for Brain Sciences (DIBS). Her principal research interests include robust adaptive control, learning and approximate dynamic programming, and information-driven planning and control for mobile and active sensor networks.

Prof. Ferrari is a member of the American Society of Mechanical Engineers (ASME), the International Society for Optics and Photonics (SPIE), and the American Institute of Aeronautics and Astronautics (AIAA). She is the recipient of the U.S. Office of Naval Research (ONR) Young Investigator Award (2004), the National Science Foundation (NSF) CAREER Award (2005), and the Presidential Early Career Award for Scientists and Engineers (PECASE) Award (2006).

QUERIES

Q1: Please provide city and postal code for Applied Research Associates, Inc.

Q2: Please note that names of games are italicized as per the IEEE style; there were many instances in your paper where Ms. Pac-Man was not used as the game title, but a character from the game. These were left in roman font. Please check for correctness.

Q3: Please specify which sections.

Q4: Please specify which sections.

Q5: Please provide membership year.

Q6: Please provide membership year.

Q7: Please provide membership year.