A Model-based Cell Decomposition Approach to On-line Pursuit-Evasion Path Planning and the Video Game Ms. Pac-Man

Greg Foderaro, Ashleigh Swingler and Silvia Ferrari

***Abstract*—This paper presents an on-line approach for optimizing paths for a pursuit-evasion problem, in which an agent must visit several target positions within an environment while simultaneously avoiding one or more actively-pursuing adversaries. This problem is found in a variety of fields, such as robotic path planning, mobile-sensor applications, and path exposure. The methodology developed utilizes cell decomposition to construct a modified decision tree, which balances the reward associated with visiting target locations and the risk of capture by the adversaries. By computing paths on-line, the algorithm can quickly adapt to unexpected adversary behaviors and dynamic environments. The methodology developed in this paper is implemented as a controller for an artificial player in the Ms. Pac-Man arcade games and is entered into the IEEE CIG 2012 screen capture Ms. Pac-Man competition. The approach presented achieved a high score of 44,630 points.**

I. Introduction

The problem of determining an optimal strategy for an agent in a pursuit-evasion scenarios is prevalent in a variety of fields, including homeland security, battlefield logistics, and surveillance systems. These applications, in the most general sense, consist of two distinct groups, pursuers and evaders. Games provide ideal testing environments for intelligence theories and algorithms, as they contain sets of well defined rules and objectives in challenging dynamic workspaces, [1]. In addition to the aforementioned benefits, games are well suited as testbeds for determining the strengths and weaknesses of different algorithms. As a result, many competitions have been developed that use the level playing fields found in games to compare artificial intelligence methodologies.

The Ms. Pac-Man arcade game constitutes an excellent environment for testing algorithms addressing pursuit-evasion scenarios, as it is analogous to a multitude of current research interests, such as robotic path planning [2], [3], mobile-sensor networks [4], and path exposure [5], [6]. In this single-player video game, the user takes on the role of Ms. Pac-Man. The Ms. Pac-Man agent must navigate a two-dimensional maze while avoiding a team of pursuing adversaries, referred to as ghosts. In order to progress to following levels, the user must collect ("eat") a predetermined arrangement of targets ("dots") distributed throughout the workspace, an example of which can be seen in Fig. 1. When no dots remain in a given maze, the player advances to the next level of the game. Subsequent levels contain more complicated mazes, as well as faster adversaries. In the game, the user is awarded points for eating various objects throughout the maze. Therefore, one metric for determining the success of an algorithm as a controller for the Ms. Pac-Man agent is the final score obtained by the methodology. The screen-capture version of the Ms. Pac-Man competition [7] challenges researchers to develop high scoring algorithms, where the highest scoring agent (over several games) is the champion.

This paper presents an approach for determining the optimal trajectory for an agent that must balance the tasks of evading adversaries and fulfilling its target collection objective. The algorithm developed is tested using the Ms. Pac-Man game within the limits of the artificial intelligence competition rules and regulations. A variety of methodologies have been successfully applied as controllers for the Ms. Pac-Man agent. Wirth and Gallagher [8] created an artificial player based on an influence map model of the game mazes. The highest documented score by the aforementioned method was 19,490 points, however, the method averaged a score of only 6,848 points. Robles and Lucas [9] presented a simple tree search strategy for determining the paths for the Ms. Pac-Man agent, which averaged 9,630 points per game. The *Pambush* family of controllers, developed by Thawonmas *et al.*, [10], [11], has been very successful in recent Ms. Pac-Man screen capture competitions. These rule-based controllers dominated the competition circuit with trials reaching 30,010 points, and averaging scores around 15,000. Arguing that the limits of rule-based controllers had been reached, Ikehata and Ito [12] began work on an entirely different approach, implementing a Monte-Carlo tree search algorithm to avoid pincer moves by the ghosts. This methodology won the 2011 IEEE CIG Ms. Pac-Man screen capture competition with a score of 36,280 points.

The algorithm described in this paper utilizes *cell decomposition* [2] in order to transform the free configuration space, i.e. areas in the environment in which the agent can travel, into a set of convex cells. The set of cells obtained from the decomposition is used to create a decision tree for the agent. This tree is then used to select an optimal path within the environment, such that the agent minimizes the risk of capture by the adversaries and maximizes the reward associated with visiting the target locations.

The remainder of the paper is organized as follows: The formulation of the pursuit evasion problem, in terms of the Ms. Pac-Man game, is defined in Section II. Tracking strategies for the adversaries are modeled and verified through simulation in Section III. The results obtained from the methodology defined in Section IV are found in Section V.

Silvia Ferrari, Greg Foderaro, and Ashleigh Swingler are with the Department of Mechanical Engineering and Materials Science, Duke University, Durham, NC 27708 USA (email: sferrari@duke.edu, greg.foderaro@duke.edu, ashleigh.swingler@duke.edu)

978-1-4673-1194-6/12/$31.00 ©2012 IEEE

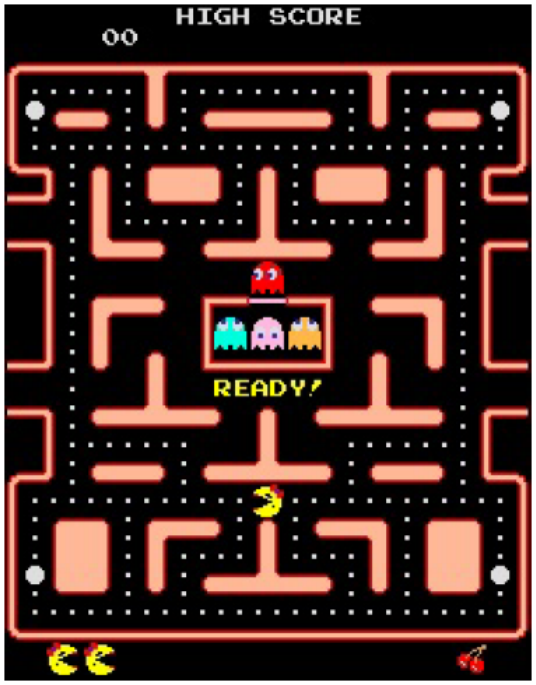


Fig. 1. Screenshot of Level 1 game maze.

II. Problem Formulation and Assumptions

The objective of this research is to develop an intelligent algorithm for determining the cost-optimal trajectory for a single mobile agent in a two-dimensional Euclidean workspace, $\mathcal{W} \subset \mathbb{R}^2$. Distributed throughout the workspace is a set of targets that the agent must collect, while avoiding collision with (similarly, capture by) a team of pursuing adversaries. An optimal path can, therefore, be defined as a continuous path in the workspace that simultaneously maximizes the number of targets acquired and minimizes the risk of capture by the adversaries. The geometry of the workspace and distribution of the targets are assumed known *a priori*. Adversary positions are determined in real-time, and the control laws of the adversaries are considered to be known.

The paradigm presented above can be found in the video game Ms. Pac-Man. In this game, the agent of interest, Ms. Pac-Man, has state and control vectors defined, respectively, as

$$\mathbf{x}_P = \begin{bmatrix} x_P & y_P \end{bmatrix}^T \quad (1)$$

$$\mathbf{u}_P = \begin{bmatrix} u_P & v_P \end{bmatrix}^T \quad (2)$$

The state vector, $\mathbf{x}_P$, represents the x and y coordinates, in pixels, of Ms. Pac-Man as compared to the reference frame of the workspace, $\mathcal{F}_\mathcal{W}$. This reference frame is such that all possible agent configurations are in the positive orthant of the Euclidean workspace. The control vector in (2) corresponds to the attempted direction of movement of the agent. The state and control vectors of the adversaries, referred to as ghosts, are defined similarly as $\mathbf{x}_G^I$ and $\mathbf{u}_G^I$, where I is a unique identifier for each ghost. The identifiers belong to the set $I_G = \{I | I \in \{r, p, b, o\}\}$, which correspond to the colors of the ghosts (red, pink, blue and orange).

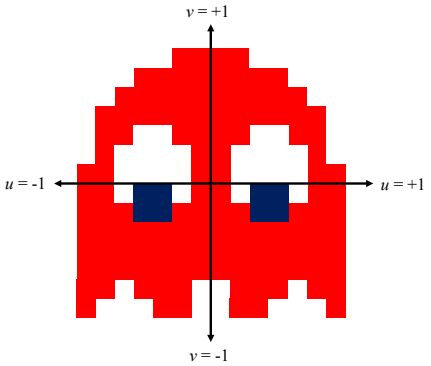


Fig. 2. Control Vector Conventions

The methodology developed requires the use of discretized time. Let k define a discrete instance in time such that $k\Delta t \in [t_i, t_f]$, where t_i indicates the time of the start of the game level, and t_f is the final time, i.e. completion of the level or death by collision with a ghost. Collision (or capture) occurs when Ms. Pac-Man and a ghost are located in the same 8 $\times$ 8 pixel tile,

$$\mathbf{x}_P(k) = \mathbf{x}_G^I(k) \quad \forall I \in I_G \quad (3)$$

The geometries of the agents, in conjunction with the design of the mazes, allow only for bidirectional motion. As a result, each point within the maze is associated with a set of admissible actions, $U[\mathbf{x}(k)] \in \mathcal{U}$, where $\mathcal{U}$ is the space of all possible control values for all game agents, such that

$$\mathcal{U} = [a_1, a_2, a_3, a_4] \equiv \left\{ \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ -1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right\} \quad (4)$$

Consequently, the control vectors of the ghosts and Ms. Pac-Man have the conventions found in Fig. 2.

III. Modeling of Adversary Behavior

The Ms. Pac-Man game contains four pursuing adversaries, the movements of which are governed by unique policies for tracking Ms. Pac-Man. These policies are comprised of a set of rules that select a target location, $\mathbf{x}_T^I$, that is used in calculating the control vector, $\mathbf{u}_T^I$, for the respective ghost. Each policy is a function of the position (i.e. state, (1)) of Ms. Pac-Man in the maze. Although the pursuit behaviors of the ghosts differ in how target locations are selected, the ghosts share the same algorithm for moving towards their targets.

The target position of the red ghost, $I = r$, is set to the current location of Ms. Pac-Man,

$$\mathbf{x}_T^r(k) = \mathbf{x}_P(k) \quad (5)$$

This tracking strategy results in the red ghost most often pursuing the player from behind.

Using a slightly different policy, the pink ghost, $I = p$, targets a position in front of Ms. Pac-Man (in the direction that the player is moving),

$$\mathbf{x}_T^p(k) = \mathbf{x}_P(k) + \mathbf{A}_i d \quad \text{for } \mathbf{u}_P(k) = a_i \quad (6)$$

where $d = [32\ 32]^T$ pixels. By index, $\mathbf{u}_P(k) = a_i$ corresponds to the respective matrix below,

$$\mathbf{A}_1 = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \qquad \mathbf{A}_2 = \begin{bmatrix} -1 & 0 \\ 0 & 0 \end{bmatrix} \tag{7}$$
$$\mathbf{A}_3 = \begin{bmatrix} 0 & 0 \\ 0 & -1 \end{bmatrix} \qquad \mathbf{A}_4 = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$$

The strategy of the pink ghost is such that the player is often presented with an adversary pursuing from the front.

The blue ghost, $I = b$, is governed by the most complicated targeting policy. The target position is the location found by reflecting the position of the red ghost about a point, $\mathbf{x}_R(k)$, 16 pixels from the position of Ms. Pac-Man in the direction that the user is moving,

$$\mathbf{x}_T^b(k) = 2 \cdot \mathbf{x}_R(k) - \mathbf{x}_G^r(k) \tag{8}$$

The reflection point is defined as

$$\mathbf{x}_R(k) = \mathbf{x}_T^b(k) + \mathbf{A}_i e \quad \text{for } \mathbf{u}_P(k) = a_i \tag{9}$$

where $e = [16\ 16]^T$ pixels.

If Ms. Pac-Man is within a radius, c, of 64 pixels of the orange ghost, $I = o$, the target for this ghost is set to the the bottom left corner of the maze, ℓ. When Ms. Pac-Man is outside of this radius, the target is set as the position of Ms. Pac-Man.

$$\mathbf{x}_T^o(k) = \begin{cases} \ell & \text{if } ||\mathbf{x}_G^o(k) - \mathbf{x}_P(k)|| \leq c \\ \mathbf{x}_P(k) & \text{if } ||\mathbf{x}_G^o(k) - \mathbf{x}_P(k)|| > c \end{cases} \tag{10}$$

As previously stated, once the target positions have been obtained the ghosts use the same algorithm to move towards their targets. This manifests itself in the definition of the control vectors of the ghosts,

$$\mathbf{u}_G^I(k) = \begin{cases} a_i = H\{\mathbf{B}\} \circ sgn\{\mathbf{D}\} & \text{if} \quad a_i \in U[\mathbf{x}_G^I(k)] \\ a_j = H\{\mathbf{C}\} \circ sgn\{\mathbf{D}\} & \text{if} \quad a_i \notin U[\mathbf{x}_G^I(k)] \\ & \quad\ \ a_j \in U[\mathbf{x}_G^I(k)] \\ a_k = U[\mathbf{x}_G^I(k)]\{1\} & \text{if} \quad a_i \notin U[\mathbf{x}_G^I(k)] \\ & \quad\ \ a_j \notin U[\mathbf{x}_G^I(k)] \end{cases} \tag{11}$$

where $\circ$ denotes the Schur product, $H\{\cdot\}$ represents the Heaviside function, and $|\cdot|$ is the absolute value. Let $\hat{\mathbf{x}}_G^I = [y_G^I\ x_G^I]^T$, then

$$\mathbf{B} = \begin{bmatrix} |\mathbf{x}_G^I(k) - \mathbf{x}_T^I(k)| & |\hat{\mathbf{x}}_G^I(k) - \hat{\mathbf{x}}_T^I(k)| \end{bmatrix} \tag{12}$$
$$\mathbf{C} = \begin{bmatrix} |\hat{\mathbf{x}}_G^I(k) - \hat{\mathbf{x}}_T^I(k)| & |\mathbf{x}_G^I(k) - \mathbf{x}_T^I(k)| \end{bmatrix} \tag{13}$$
$$\mathbf{D} = |\mathbf{x}_G^I(k) - \mathbf{x}_T^I(k)| \tag{14}$$

Due to the definition of the control vector in (11), the ghosts will not choose an action contrary to their current action, i.e. the ghosts will not reverse directions along their paths. Therefore, they can only effectively make pursuit decisions when located at a maze position where there are at least three possible directions for the ghosts to move.

It should also be noted that the models above do not apply at all times in the game. For the first 7 seconds of each play (either at the start of a new maze or when resuming play after the player has lost a life), the adversaries choose their decisions randomly when they reach an intersection, excluding the possibility of reversing direction. Similarly, when the player moves over a power pill, the ghosts immediately reverse direction and temporarily abandon their primary policies, exhibiting the same random behavior mentioned above.

A. Numerical Verification

The trajectories of the four ghosts were recorded during typical game play, i.e. $t > 7$ seconds, by a screen-capture program in an emulated Ms. Pac-Man game, [13]. Using the same path taken by Ms. Pac-Man, the ghost trajectories were simulated using the models developed above. Fig. 3 shows a comparison of the simulated paths to those extracted from the emulated game.

As stated above, the ghosts only effectively make decision at maze junctions with three or more possible direction for the ghosts to move. In order to verify the models, the number of correct decisions made by the simulated ghosts was counted. If an incorrect decision was made, the error was recorded, and the ghost trajectories were reset to those of the actual game. The emulated game was run to completion several times by a human player. In these games, the adversaries made 28,089 decisions. Simulations, using the method described above, were run for the targeting models developed. The simulated ghosts made 818 incorrect decisions, resulting in an accuracy of about 97.1 %.

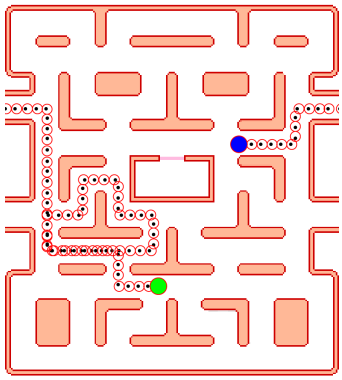
As seen in the results, the simulation of the trajectories closely approximates the actual paths. The errors that do occur are speculated to be a result of the screen-capture program. This hypothesis is strengthened by the fact that the models of the pink and blue ghosts were less accurate, as the tracking policies of the pink and blue ghosts depend on both the location of and direction that Ms. Pac-Man is moving. Because these policies have an additional input, screen-capture inaccuracies would have a greater affect on the simulations of the respective ghosts.

IV. Methodology

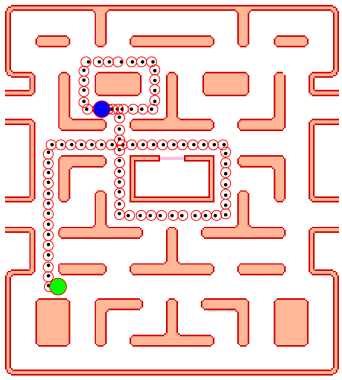
The methodology developed for the Ms. Pac-Man controller is as follows. The maze is decomposed into a finite set of cells, which is used to create a connectivity tree based on the adjacency relationships of the cells and the position of the Ms. Pac-Man agent. This tree is then used to calculate the instantaneous optimal path for the agent, based on the current game state extracted from the game image and the predicted future game state computed from the models described in Section III.

A. Cell Decomposition and the Connectivity Tree

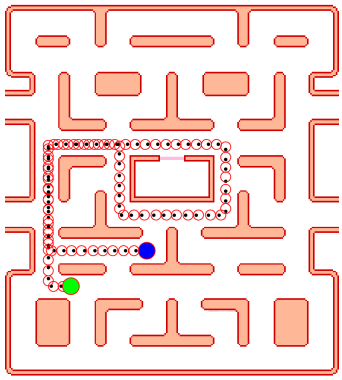
Cell decomposition is a commonly implemented approach in robotic path planning for obstacle avoidance, [14]–[16]. The obstacle free region is decomposed into a finite set of non-overlapping polygons, referred to as cells. The obstacles of classic cell decomposition correspond to the walls of the maze in the Ms. Pac-Man testing environment. Using a line-sweeping algorithm, the free space of the maze is decomposed



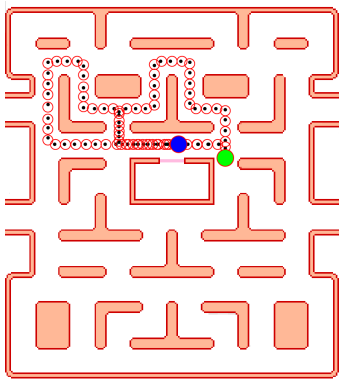
(a) Red Ghost



(b) Pink Ghost



(c) Blue Ghost



(d) Orange Ghost

Fig. 3. Comparisons of the ghost trajectories extracted from the game (circles) and the simulations from the derived models (dots). The solid green circle is the initial position, and the solid blue circle is the final position.

into a set of rectangular cells, an example of which is found in Fig. 4.

The decomposition can be used to create a connectivity graph corresponding to the adjacency relationships of the workspace cells.

Definition 5.1: A *connectivity graph* is a non-directed graph that is used to represent the cells in a decomposed workspace. Nodes correspond to cells in the decomposition, while arcs represent the adjacency of connected cells.

Due to the assumption of a continuous path, the user may only move from the cell containing its current location to an adjacent cell, thus creating a causal relationship between current location and potential future paths. This relationship is commonly represented by a connectivity tree.

Definition 5.2: A *connectivity tree* is a tree graph associated with a connectivity graph, $\mathcal{G}$, and the current cell of the agent, $\mathbf{x}_P(k) \in \kappa_P$. The tree has κ_P as the root and branches with length L.

The branches of the connectivity tree correspond to ordered sets of cells that contain possible paths for the agent beginning in κ_P. Each arc in the tree is assigned a weight, which changes based on the game scenario. Fig. 5 shows an example connectivity tree based on the decomposition found in Fig. 4.

As a result of the cell properties and character mobility, a unique action value can be assigned to each arc in the connectivity tree. Therefore, the set of admissible actions is a function of $\mathbf{x}_P(k)$ as well as $\mathbf{u}_P(k-1)$, because as the agent is not allowed to immediately reverse, the set of admissible actions is the complement of $\mathbf{u}_P(k-1)$.

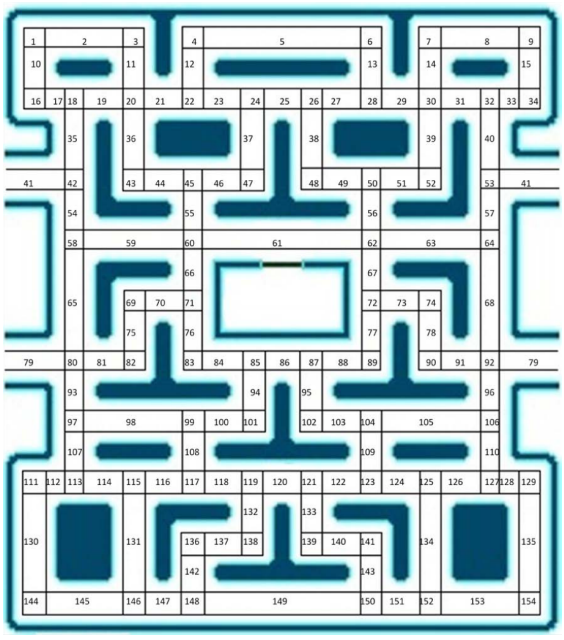


Fig. 4. Cell decomposition of the Level 1 maze (as seen in [17])

B. State Extraction and Optimal Agent Strategy

In order to decode the game state from the image obtained via screen-capture, a simple method is implemented that

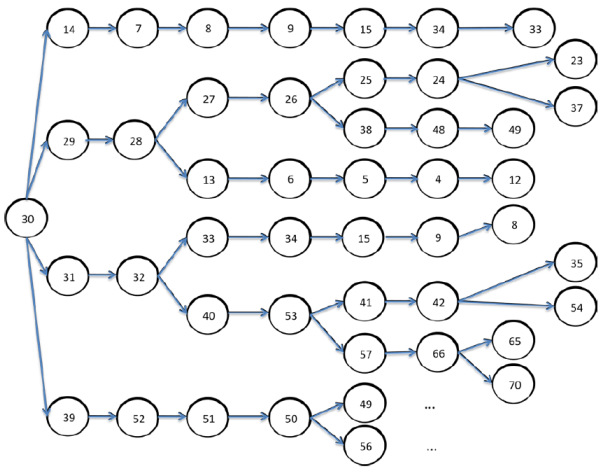


Fig. 5. Connectivity tree of the cell decomposition (as seen in [17])

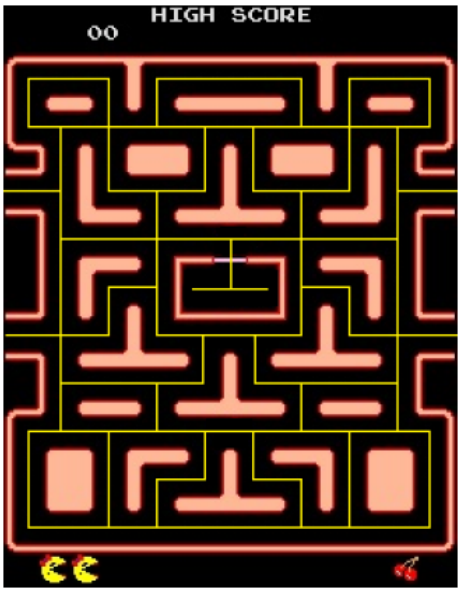


Fig. 6. Stored list of pixels in the level 1 maze to be searched for state extraction. The stored pixels are drawn in yellow.

searches for pixels of particular colors, which correspond to different game objects. When a pixel containing a color of interest is found, the adjacent space is searched for additional pixels of the same color using a flood fill algorithm. If enough pixels of a given color are found, the game object corresponding to this color is recorded at the particular location. As many of the game objects have similar colors, e.g. the red ghost and the strawberry, object specific properties are searched for. For example, the red pixels of the strawberry may have an adjacent green pixel, while the red pixels of the red ghost may have an adjacent blue pixel. In addition, the pixels representing the eyes of the ghosts are examined. The direction that the eyes point corresponds to the direction of motion of the particular ghost. Therefore, an analysis of the eyes results in additional information regarding the game state. To increase the efficiency of the state extraction, the method checks a stored set of pixels instead of the entire maze, as illustrated in Fig. 6. Similarly, the dot states are updated by checking a saved list of locations for pixels of the particular dot color.

The player's instantaneous reward for visiting a particular cell in the decomposition can be calculated by a reward function, $\mathcal{L}$. This function is defined as a trade-off between the predicted benefit and risk for the agent in visiting a particular cell. The reward function is defined as follows,

$$\begin{aligned}\mathcal{L}[\mathbf{x}_P(k),\mathbf{u}_P(k)] &= w_D D[\mathbf{x}_P(k),\mathbf{u}_P(k)] \\ &+ w_B B[\mathbf{x}_P(k),\mathbf{u}_P(k)] \\ &+ w_F F[\mathbf{x}_P(k),\mathbf{u}_P(k)] \\ &+ w_R R[\mathbf{x}_P(k),\mathbf{u}_P(k)]\end{aligned} \tag{15}$$

where D, B, F and R are event specific reward and risk functions. The function D corresponds to the number of dots that can be eaten in a particular cell. When Ms. Pac-Man passes over a power-pill, the player is able to eat the "frightened" ghosts. The function B is associated with the predicted score that can be achieved by eating frightened ghosts. The final reward component of (15), F, corresponds to the possibility of eating a fruit. The terms w_D, w_B, and w_F are weighting coefficients for the respective functions. These user defined values vary from level to level as they are related to factors such as character speed and maze complexity. Note that once a player has passed through a cell the associated reward changes. The function R and weighting coefficient w_R correspond to the risk of capture, and are governed by the distances between the agent and the ghosts,

$$R[\mathbf{x}_p(k),\mathbf{u}_p(k)] = \sum_{I\in I_G}[q(\mathbf{x}_p,\mathbf{x}_G^I)-\rho_0]^2+\gamma \tag{16}$$

The function $q(\mathbf{x}_p,\mathbf{x}_G^I)$ is the minimum distance between Ms. Pac-Man and the ghost, I, as measured along the corridors of the maze. The parameter ρ_0 is a user-defined parameter distance-of-influence weight that is selected such that ghosts at a distance greater than ρ_0 from the user are not considered when calculating the risk term, i.e. $q \rightarrow \rho_0$, $R \rightarrow 0$. To emphasize the cost of losing a life, γ is set as a very large positive number, e.g. $\gamma = 100,000$, if Ms. Pac-Man is predicted to collide with a ghost and zero otherwise.

The objective of the methodology is to determine an optimal sequence of actions for the agent that both maximizes the point reward and minimizes the risk of capture. Let the optimal strategy, σ^*, be defined by a sequence of functions, c_k, that map the state, $\mathbf{x}_P$ to an admissible action. Mathematically, the optimal strategy is as follows,

$$\sigma^* = c_i, ..., c_F \tag{17}$$

where

$$\mathbf{u}_P(k) = c_k[\mathbf{x}_P(k)], \text{for } k = 1, ..., F \tag{18}$$

This in turn maximizes the cost-to-go function evaluated over

the interval $[t_i, t_F]$,

$$J_{i,F}[\mathbf{x}_P(i)] = \sum_{k=i}^{F} \alpha_k \mathcal{L}[\mathbf{x}_P(k), \mathbf{u}_P(k)] \tag{19}$$

where t_i is the current time and t_F is the final time. A discount factor, α_k, that is an exponential function of k, is applied such that the rewards at future times are discounted compared to immediate ones.

Using the above formulation, the connectivity tree can be used as a decision tree, where the instantaneous rewards are computed as the branches of the tree are grown. In addition, the cumulative cost $J_{i,k}[\mathbf{x}_p(i)]$ can be evaluated using the instantaneous reward, and then stored at each node iteratively over time.

$$J_{i,k}[\mathbf{x}_P(i)] = \sum_{j=i}^{k} \alpha_j \mathcal{L}[\mathbf{x}_P(j), \mathbf{u}_P(j)] \tag{20}$$

Therefore, the optimal path can be selected by choosing the branch corresponding to the maximum value of $J_{i,F}$. The optimal branch then governs the optimal strategy, σ^*, by chaining together the sequence of control values stored in the arcs of the branch.

During game-play, as stated in Section III, the movements of the adversaries are not always governed by the tracking policies. The ghosts exhibit random behavior during the first 7 seconds of play and when they are in a frightened state. To address the random ghost decisions at the start of play, the ghost models are ignored, and the risk term in (16) is replaced with an artificial repulsive potential term. Artificial potential fields are a well-known robot motion planning approach that treat agents as particles under the influence of potential fields [2]. A repulsive potential "pushes" the agent away from a ghosts if they are within a threshold distance. This potential is defined as,

$$P[\mathbf{x}_P(k), \mathbf{u}_P(k)] = \begin{cases} \left[\frac{1}{\rho(\mathbf{x}_P, \mathbf{x}_G)} - \frac{1}{\rho_0}\right]^2 & \text{if} \quad \rho(\mathbf{x}_P, \mathbf{x}_G) \leq \rho_0 \\ 0 & \text{if} \quad \rho(\mathbf{x}_P, \mathbf{x}_G) > \rho_0 \end{cases} \tag{21}$$

The instantaneous reward function, (15), can then be updated to include P,

$$\begin{aligned} \mathcal{L}[\mathbf{x}_P(k), \mathbf{u}_P(k)] &= w_D D[\mathbf{x}_P(k), \mathbf{u}_P(k)] && (22) \\ &+ w_B B[\mathbf{x}_P(k), \mathbf{u}_P(k)] \\ &+ w_F F[\mathbf{x}_P(k), \mathbf{u}_P(k)] && (23) \\ &+ w_P P[\mathbf{x}_P(k), \mathbf{u}_P(k)] \end{aligned}$$

where w_P is a weighting coefficient.

When Ms. Pac-Man has eaten a power pill, the future states of the ghosts cannot be predicted by the tracking models. Instead, when the ghosts are frightened they are modeled as probabilistic states, such that a ghost would have a state $\mathbf{x}_G^{Ij}$ with probability p_{Ij}. Essentially, each time a ghost reaches an intersection in the predictive simulation, new instances of that ghost are created for each potential action with a corresponding probability of occurrence. Then, the reward for eating a ghost with state $\mathbf{x}_G^{Ij}$ is weighted by the probability p_{Ij}, such that

$$B[\mathbf{x}_P(k), \mathbf{u}_P(k)] = p_{Ij} b[\mathbf{x}_P(k), \mathbf{u}_P(k)] \tag{24}$$

As this is more computationally intensive than using the deterministic ghost model, the branch length, L, is reduced when a power pill is active.

In order to increase the likelihood of Ms. Pac-Man capturing frightened ghosts, when a decision is made to traverse a cell containing a power pill, the planned set of actions is interrupted when the distance between Ms. Pac-Man and the power pill is small. The controller lets the player either rest or moves the player back and forth at the same position until a ghost is close. Then, the agent moves over the power pill and resumes the methodology described above.

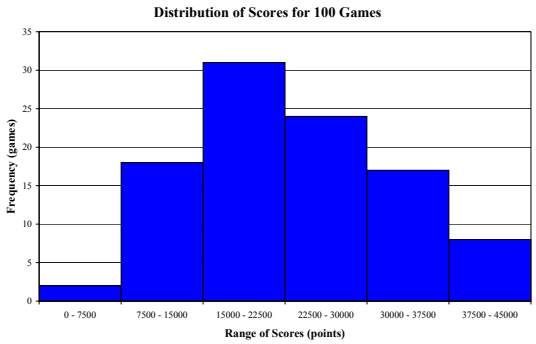


Fig. 7. Trial Score Distribution

V. Results

The controller developed by this research was implemented as an artificial player in the web version of the Ms. Pac-Man game (freely available at [13]). Using screen capture and the methodology defined above, the classic arcade game was run to completion (i.e. 'game over') 100 times using a connectivity tree with a branch length of $L = 16$ cells for regular game play and $L = 11$ when a power pill is active and the ghosts exhibit random behavior. Table I documents the major statistics of the results obtained, and Fig. 7 shows the score distribution. The highest score obtained over the 100 trials was $44,630$ points, which is higher than the score of the 2011 screen-capture Ms. Pac-Man competition [12] winner, which achieved a score of 36,280 points.

The majority of player deaths can be attributed to inaccuracies in the extracted game state due to computational delays, as well as incomplete adversary models. Although the models presented in Section III adequately represent the tracking policies, the details of additional behaviors, including duration of the "frightened" ghost states and the timing of sudden ghost direction reversals were not sufficiently defined in the methodology. Errors accumulated due to the sensitivity of the adversary dynamics to inaccuracies in the extracted game state severely affect path planning for the agent. The approach presented above attempts to compensate for these

errors by frequently reevaluating player decisions. However in high risk scenarios, there is sometimes not enough time to correct poor decisions before capture.

Although the algorithm developed has weaknesses attributed to inaccuracies in the extracted game state, it is able to overcome many of the shortcomings of other methods. Previous approaches have either ignored the tracking policies of the ghosts or have considered incorrect or highly approximate models. As a result, these algorithms are unable to effectively plan ahead, and consequently, often move their player into dangerous situations. Many previous methodologies use metrics dominated by ghost proximities and directions of movement, which often result in the mislabeling of optimal paths as dangerous.

The methodology described in this paper is an effective and competitive controller for the Ms. Pac-Man screen capture competition. However, as explained above, the methodology has some weaknesses. Future work will focus on improving the performance of the cell decomposition approach by including additional features in the game model. Implementing a more detailed model of the game state is expected to reduce the number of scenarios that result in incorrect predictions of future game states, and, in turn, decrease the number of suboptimal controller decisions. Furthermore, the accuracy of the game state extraction and the size of the connectivity tree are limited by computational delays. Optimizing the code of the algorithms and using a faster computer will help to increase the success of the methodology.

TABLE I
PERFORMANCE OF ARTIFICIAL MS. PAC-MAN PLAYER OVER 100 RUNS

	Score	Mazes Cleared
Highest	44630	5
Lowest	6200	0
Average	23296	2.57

VI. CONCLUSIONS

A novel approach for an artificial player of the game Ms. Pac-Man is presented. The game state is decoded from the game image by use of a basic screen-scraping technique, and the future game state is predicted through a verified ghost model. The method uses cell decomposition in combination with a connectivity tree to compute optimal decisions for the player and returns a control value corresponding to the resulting player action. During instances in the game when the objects are known to behave randomly, such as the first few seconds of the game or when Ms. Pac-Man eats a power pill, the algorithm uses an alternate probabilistic ghost model and employs a potential field method in combination with the connectivity tree. Results are presented that show a high score of 44,630 points over 100 runs.

ACKNOWLEDGMENTS

This work is partially supported by the Office of Naval Research Code 321, the National Science Foundation under Grant ECCS 1028506, and the National Science Foundation under Grant DGE 1068871.

REFERENCES

[1] S. M. Lucas and G. Kendall, "Evolutionary computation and games," *IEEE Computer Intelligence Magazine*, vol. 1, no. 1, 2006.
[2] J. Latombe, *Robot Motion Planning*. Kluwer Academic Publishers, 1998.
[3] Z. Sun and J. Reif, "On robotic optimal path planning in polygonal regions with pseudo-eucledian metrics," *IEEE Transactions on Systems, Man, and Cybernetics - Part A*, vol. 37, no. 4, pp. 925–936, 2007.
[4] D. E. D. Culler and M. Srivastava, "Overview of sensor networks," *Computer*, vol. 37, no. 8, pp. 41–49, 2004.
[5] S. Megerian, F. Koushanfar, G. Q. amd G. Veltri, and M. Potkonjak, "Exposure in wireless sensor networks: Theory and practical solutions," *Wireless Networks*, vol. 8, pp. 443–454, 2002.
[6] V. Phipatanasuphorn and P. Ramanathan, "Vulnerability of sensor networks to unauthorized traversal and monitoring," *IEEE Transactions on Computers*, vol. 53, no. 3, 2004.
[7] S. M. Lucas, "Ms pac-man competition," *SIGEVOlution*, vol. 2, no. 4, pp. 37–38, Dec. 2007.
[8] N. Wirth and M. Gallagher, "An influence map model for playing Ms. Pac-Man," in *IEEE Symposium On Computational Intelligence and Games*, dec 2008, pp. 228 –233.
[9] D. Robles and S. M. Lucas, "A simple tree search method for playing Ms. Pac-Man," in *Proceedings of the 5th international conference on Computational Intelligence and Games*, 2009, pp. 249–255.
[10] 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 *Proceedings of the Asia Simulation Conference*, oct 2009.
[11] R. Thawonmas and T. Ashida, "Evolution strategy for optimizing paramters in Ms. Pac-Man controller ICE Pambush 3," in *Computational Intelligence and Games, IEEE Symposium on*, 2010, pp. 235 – 240.
[12] N. Ikehata and T. Ito, "Monte-carlo tree search in ms. pac-man," in *Computational Intelligence and Games (CIG), 2011 IEEE Conference on*, sept 2011, pp. 39 –46.
[13] *Ms. Pacman Game*. http://webpacman.com/.
[14] D. Zhu and J. C. Latombe, "New heuristic algorithms for efficient hierarchical path planning," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 1, pp. 9–20, 1991.
[15] K. Kadem and M. Sharir, "An efficient motion planning algorithm for convex polygonal object in 2-dimensional polygonal space," *Courant Institute of Mathematical Science, New York, NY, Tech. Rep. 253*.
[16] S. Ferrari and C. Cai, "Information-driven search strategies in the board game of clue," *Trans. on Systems, Man, and Cypernetics*, vol. 39, no. 3, 2009.
[17] G. Foderaro, V. Raju, and S. Ferrari, "A model-based approximate λ-policy iteration approach to online evasive path planning and the video game Ms. Pac-Man," *Journal of Control Theory and Applications*, vol. 9, no. 3, pp. 391–399, 2011.