The 2009 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 11-15, 2009 St. Louis, USA

Intelligent Pursuit & Evasion in an Unknown Environment

Jonathan Annas
Department of Computer Science
University of North Carolina at Charlotte
Charlotte, North Carolina 28223–0001
Email: jdannas@uncc.edu

Jing Xiao
Department of Computer Science
University of North Carolina at Charlotte
Charlotte, North Carolina 28223–0001
Email: xiao@uncc.edu

***Abstract*—This paper introduces a novel and flexible simulation platform for studying pursuit and evasion in unknown 2-D environments of arbitrary obstacles, in an effort to expand the practical application of pursuit-evasion research. The platform provides realistic simulation of the sensing capability of each robotic agent (either a pursuer or an evader). Each agent uses real-time local sensing to collect information from the environment while it simultaneously plans and executes its motion to best satisfy one or more objectives. The evader's objectives are to reach a specific goal location as quickly as possible and to avoid being caught by the pursuer. The pursuer's objectives are to locate and capture the evader whose motion is unknown, and when the evader is not seen, explore the environment and predict where the evader may be. Under a common real-time planning paradigm, each agent's planner dynamically adapts its goals and objectives to the agent's changing circumstances so that the agent can always choose the best course of action. Simulation results have shown that the introduced approach is an effective means to study sophisticated pursuit-evasion scenarios and accomplish objectives for both the pursuer and the evader in an unknown environment. The platform can be easily expanded to accommodate multiple agents in more complex pursuit-evasion tasks.**

I. Introduction

Pursuit and Evasion is a category of robotics and intelligent systems that encompasses a broad scope of problem formulations. The base context of these problems is that one or more agents must 'catch' another agent. The term catch has been defined as to chase, acquire visibility, maintain visibility, or to actually make contact with one or more opposing agents. Existing research in pursuit and evasion has been motivated by real life applications but often only addresses a highly constrained version of problem models. As a relatively young field, much of the work done has reduced agents to a point robot and environments are often represented as a connected graph or grid. The theory derived from even these simplified models has advanced security systems [6], military robotics simulations [9], emergency response simulations, and other important areas. This field has been gaining more attention in recent years with papers showing up in many major conferences. Research recently published in the category of pursuit-evasion has considered pursuit-evasion in an arena (the lion and the man game) where a point agent captures another using only bearing information [5]. Other research that has been applied to security makes use of sensor networks for tracking [7] and multiple robots to aid in surveillance [6]. The contributions of these methods have spanned from grounding theory through proofs in their respective environments in order to guarantee expected outcomes, to providing comparisons of specific approaches such as random search over exhaustive search in pursuit-evasion [4].

This paper introduces a novel and flexible simulation platform for studying pursuit and evasion in unknown 2-D environments of arbitrary obstacles. The platform provides realistic simulation of the sensing capability of each robotic agent (either a pursuer or an evader). It also provides a sensing-based real-time planning paradigm that can accommodate the navigation objectives of either agent based on circumstances during pursuit and evasion. In Section II, we give an overview of the motion planning paradigm in our approach. In Section III, we introduce the simulation of the physical robot agents equipped with sensors. In Section IV, we discuss the behaviors of the pursuer, including how the pursuer focuses on intelligent prediction of the evader's motion based on online sensing. In Section V, we discuss the behaviors of the evader and how they contribute to successful evasion. In Section VI, we explain the design of experiments using the simulation platform. In section VII, we present and discuss experimental results of our simulations. Finally, Section VIII concludes the paper and suggests possible future work.

II. Planning Paradigm

For both an evader and a pursuer to navigate in an unknown environment, each must be capable of real-time motion planning that is adaptive to sensory discoveries. Our planning paradigm is inspired by the past work of Xiao and Vannoy [10][8] addressing real-time adaptation in planning a robot's motion. Both borrowed the general *anytime* and *parallel planning* idea of evolutionary computation [1] by maintaining and repeatedly updating a set of solution candidates for the robot, i.e., paths [10] or trajectories [8], in iterations, called *planning cycles*. Planning is done in the continuous space of the robot's environment rather than on a discretized grid or graph[1]. A path is represented as a sequence of robot

[1]This becomes a clear advantage over traditional complete algorithms such as A* even in a known environment, see Figure 1.

configurations, called *path nodes*, from the robot's current configuration to a goal configuration. An *evaluation function* is used to code certain optimization criteria to evaluate how good a solution candidate is, and a number of simple mutation operators are used to modify the solution candidates. An initial set of solution candidates can be generated randomly, and in subsequent planning cycles, the solution candidates are repeatedly modified and evaluated to find the best one. As the planner runs, the robot simultaneously executes the best feasible solution at any time. As the robot moves, the starting configuration of the solution candidates are updated to the robot's current configuration. Since evaluation is based on the updated information of the environment from sensing, the best solution candidate found also changes as the circumstance changes. By following (or essentially switching to) the current best solution, the robot adapts its movement to the changing scenes.

Our planning paradigm, in the context of pursuit and evasion, extends the above significantly by enabling more dynamic adaptation to deal with changing goals and objectives of the agents. For the evader, even though it has a fixed goal configuration, it may often need to change its goal temporarily in order to flee the pursuer during navigation. The pursuer has an even greater challenge. It may not even be able to see the evader, and when it can see the evader, it does not know how the evader will move. Therefore, the pursuer has to update its goal constantly by prediction.

Our planning paradigm enables dynamically changing an agent's goal and adapting its evaluation function based on the sensed circumstances to promote the most suitable behaviors. For example, when the evader has to flee the pursuer, its evaluation function changes to favor paths that allow the evader to turn. With such flexibility, the same planning paradigm, outlined in Algorithm 1, is able to be used for both the evader and the pursuer. Figure 2 shows the relationship among planning, sensing, and control (i.e., adaptation) cycles.

The same basic evaluation function is used by both agents, and they adapt it to their different needs at different times only by adjusting the weights of function. We use the following composite evaluation function F for both agents to evaluate a path, which consists of a sequence of path nodes:

$$F = \sum SegLengths + (C * C_w) + (\sum ArcLengths * A_w)$$

- $\sum$SegLengths is the summation of the individual segment lengths along the path.
- C is the number of collisions along the path. The C_w weight is set high and the $\sum$SegLengths is set to 0 for any path that has collisions. This effectively separates feasible from infeasible paths and allows for the adaptation of each with less likelihood of an agent being biased towards one infeasible path over another due to path length and therefore being stuck in local minima.
- C_w is a weight that determines the importance of collisions.
- $\sum$ArcLengths is the summation of the individual arc lengths from one path node to the next. An arc length captures the distance caused by an agent's rotation to change its orientation. By rotating the agent about its center from one orientation (at one path node) to the next, the distance traveled by the apex of the agent is the arc length.
- A_w is a weight that determines the importance of the ArcLength. This weight can be increased when an agent considers sharp turns to be more important than the length of a path. This is crucial for the concept of evasion.

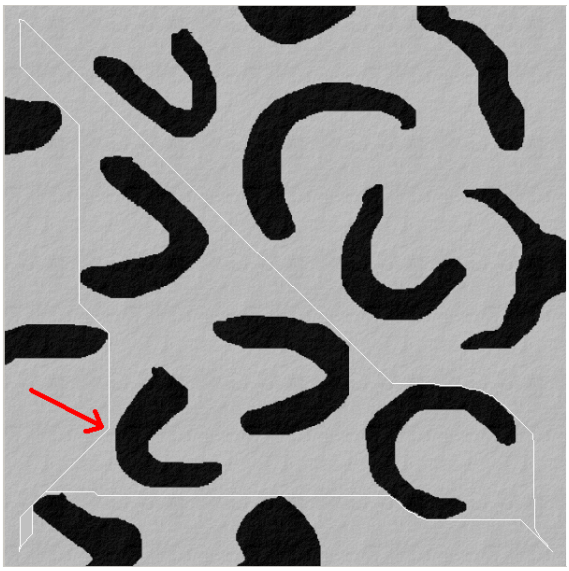

Fig. 1. A* results on a 600x600 grid with the red arrow highlighting a clearly non-optimal section of an optimal A* result due to the limited number of motion directions from one cell to a neighboring cell in a grid.

How each agent changes its goals or objectives is described in detail in Sections IV and V respectively.

Our planner also introduces a new concept of *progressive modification* of paths to deal with gradually discovered obstacles. Assume a path is currently infeasible because it is in collision with an obstacle and a repair operator has failed to adjust the path to be feasible in the current planning cycle due to incomplete knowledge of the colliding obstacle. That failed attempt and its associated information is accumulated to guide repairs in later planning cycles as the obstacle becomes more visible. In this fashion, a collided path may spiral outward off of an obstacle gradually.

III. Agent and Sensor Models

We focus this paper on pursuit and evasion in an unknown, planar environment. We use two robotic agents, one as the pursuer and the other as the target or evader. Each agent has the shape of a triangle with three degrees of freedom (x, y, and θ). It is driven by two back motors, and its apex points to the forward moving direction (see Figure 3). We simulate the environment as consisting of an arbitrary number of obstacles of arbitrary size and shape, which are unknown to the robotic agents. Each agent senses the environment by virtual vision, which uses color to detect obstacles and convert those to a

Algorithm 1 Planning Paradigm

Initialize a set S of agent paths with randomly generated intermediate nodes connecting the agent's current configuration to a (current) goal configuration.
Evaluate all paths in S. $P_{best} \leftarrow$ best path.

while Goal is not reached **do**
Simultaneously do **Move** and **Plan**:

Move along P_{best} unless forced to stop to avoid collision.

Plan:
Modify paths in S.

if end of current control cycle **then**
Update starting configuration of paths in S.
Evaluate S and update P_{best}.
end if

if new sensing cycle **then**
Update own view of world (i.e., map).
Update goal and adjust evaluation function to suit the current agent objective.
Evaluate S and update P_{best}.
end if
end while

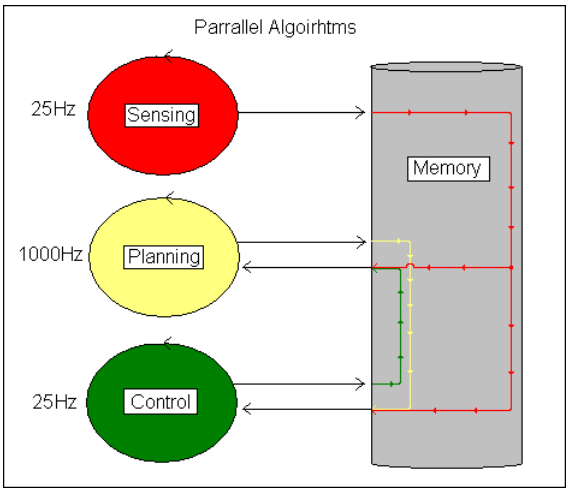

Fig. 2. Planning, Sensing, and Control Cycle Relationship

local representation in the agent's memory. Each agent is also equipped with virtual collision detectors. The details of these simulated sensors are explained below.

A. Vision Scanning

Each robotic agent is equipped with a virtual vision scanner that detects obstacles and agents. The vision scanner shoots out a ray from the robot in a circular fashion at each interval of ω degrees until it completes a 360 degree scan. At each interval, the ray is traced from the agent's location outward until a change of color occurs that indicates collision with an obstacle

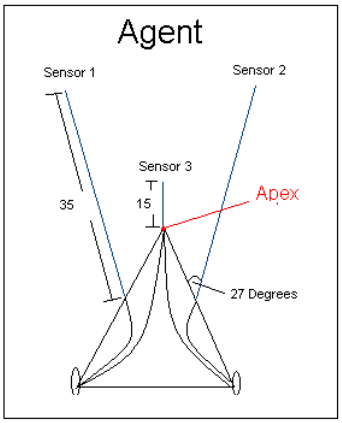

Fig. 3. Agent & Sensors

or agent. This point is then discretized into an obstacle pixel and mapped into a two dimensional array which represents the agent's memory. The mapped point is expanded in size to allow for sensor noise (line rasterization rounding error). We found the ideal sensor mapping to be one point in real space to a 4x4 grid of points in the discretized agent memory space. The entire process of finding all visible obstacle pixels can be done at 25Hz using less than 2% of one core of the available CPU, and is therefore very efficient.

B. Obstacle Avoidance

Each agent also has a sensor array of three "'whisker"' sensors for avoiding collisions. Figure 3 shows the location of two sensors angled at 27 degrees outward from the center of the robot, and one sensor facing directly ahead. These sensors are each a single ray traced in the same way as the vision scanner rays described in the section III-A. Each angled sensor is connected to the motor at the same side and can override that motor's current force when an obstacle is sensed. If an obstacle is detected, that sensor will cause it's motor to speed up proportionally with the distance to the detected obstacle to force the agent to turn away. As an object gets closer, the agent will turn faster. Similarly, the middle sensor controls both motors and will slow them both according to the distance to an obstacle detected. This local reactive control allows for correction in the event of any poor estimate made in the discretization of the map by the vision scanner due to the scan resolution. The local sensing may be in conflict with the motion planner's current path. To resolve this we allow the reactive local sensing to subsume or override the deliberative planned path as the agent approaches obstacles [2]. We use Algorithm 2.

IV. The Pursuer

The pursuer's main objective is to catch the evader. We define catch in this paper as the act of the pursuer physically touching the evader. To achieve this the pursuer must first gain knowledge of the environment by exploring areas in its mental map (acquired from sensing) in order to determine where the

Algorithm 2 ObstacleAvoidance

Let R_{ctr}, R_{right}, R_{left} be the max sensor ranges.

d_{ctr} = distance from sensor S_{center} to an obstacle O
d_{right} = distance from sensor S_{right} to an obstacle O
d_{left} = distance from sensor S_{left} to an obstacle O

if ($d_{ctr} < R_{ctr}$) and ($d_{ctr} > 0$) **then**
    Set right and left motor speeds to 0.
end if

turnRight = (R_{right} / d_{right}) * 2 degrees
turnLeft = (R_{left} / d_{left}) * 2 degrees

if (turnRight $>$ turnLeft) **then**
    Increase right motor speed to turn agent by turnRight degrees.
else
    Increase left motor speed to turn agent by turnLeft degrees.
end if

evader is most likely to be spotted. After the evader is visible, the pursuer must maintain visibility by predicting where the evader will move next and setting its goal to that location. The following subsections explain these core ideas in depth.

A. Intelligent Exploration

If the pursuer has no line of sight or knowledge of where the evader is (or last was), he must begin the search by exploring its map. As feedback comes in from his sensors, the pursuer determines which area of the map he is most likely to spot the evader from and begins moving towards that area. This area is defined as a maximum visibility area, and the maximum visibility is computed by estimating the total amount of the map that is visible from that location (i.e., discretized integration). Keep in mind that the pursuer only knows about the map from sensor data. This means that the computation of the maximum visibility area takes place inside his own world view and is an estimate based on current knowledge. The maximum visibility area is likely to change as the pursuer begins to reach that location or as sensor data unveils new obstacles. At this time, a new maximum visibility location is determined. This allows the pursuer to explore the environment if he cannot spot the evader because an area that the pursuer has no knowledge of will always be viewed as more visible since obstacles can only be added to the map as he discovers them.

The process for finding the maximum visibility area is outlined in Algorithm 3. The agent uniformly places n seed configurations over his mental model (i.e., map) of the environment. He then traces rays from each seed (max visibility candidate) until an obstacle is hit. This information is used to estimate the total area visible from that location. The seed with the most area visible to it is then chosen. If the pursuer already has sensor data of a past location where the evader is known to have recently been, the algorithm is changed slightly.

Algorithm 3 MAXVISIBILITY

1: uniformly place n seeds in agent's current map knowledge
2: **for** each seed **do**
3: perform visibility scan
4: visibility $\leftarrow \sum$ray lengths
5: **end for**
6: **return** Max(seeds)

Now the uniform seed distribution is done in a tight pattern over the last known location of the evader. This translates to the situation where the pursuer has lost sight of the evader but knows he is close by. The pursuer then uses his accumulated knowledge of the map to decide which area close by would be most likely for him to spot the evader again. We call this *Smart Seeding*.

B. Intelligent Motion Prediction

Since the evader's motion is unknown to the pursuer, the pursuer must make predictions based on the history of sensed evader motion. Algorithm 4 shows this process. The previous locations of the evader sensed by the pursuer's vision scanner were recorded and also time stamped. The motion prediction system uses these time stamps to create a time indexed list of estimated velocities along with the locations. Next it processes the list into different motion patterns of the evader based on the information from the most recent time t_i all the way back to t_i - Δt. Currently, the system identifies three motion patterns of the evader: left and right turns, left and right circular motions (concurrent small turns), and linear movement. These patterns are used to predict the evader's motion and where the evader is likely to be in the near future.

Algorithm 4 PredictMotion

1: Sensors provide new points of timestamped evader positions.
2: Determine the direction and speed changes at each point.
3: Plot points on internal environment model.
4: Match point pattern to either a line, sharp turn, or circular turn.
5: Assume that the evader will continue moving along that shape.
6: Set goal to a location on that shape at a distance $d_{prediction}$ in front of the evader based on the current estimated evader velocity and pursuer's current distance from the evader.

A prediction is made along the pattern shape at a distance:

$$d_{prediction} = \max(\frac{1}{2}d_{evader}, d_{prediction\_min})$$

where d_{evader} is the pursuer's distance to the last point the evader was sensed at and $d_{prediction\_min}$ is a small number based on the perceived evader speed. It is necessary that there

be a minimum distance because we assume that the evader is always moving. Therefore if the prediction distance was zero, the pursuer would never catch the evader because he would always set his goal to the last known location and the evader would be long gone by the time that point was reached. When the evader is further away, the prediction point is further along the predicted pattern because the evader will continue moving during the time it takes to reach the prediction point.

V. The Evader

The evader begins its motion with the knowledge of his fixed goal location. His primary objective is to reach this goal as quickly as possible while navigating through the unknown environment. However, if the evader's sensor data relays that he is within the pursuer's range of sight and the pursuer is closer than the goal, his primary objective changes to avoid the pursuer. The evader uses the following behaviors to avoid the pursuer:

- Flee - Upon first seeing the pursuer, the evader must assume the pursuer has also seen him. The evader begins to flee in a panic. We define fleeing as the optimization of paths that provide for immediate turns with the purpose being to break line of sight from the pursuer. This is easily accomplished by adapting the path evaluation function through increasing the weight/importance of the direction change at each knot configuration in each path. Recall from Section II that the direction change is measured by the ArcLength change between two knot configurations.

- Evade - After the evader has successfully fled from the pursuer (broken direct line of sight), he attempts to further evade the pursuer by navigating towards a point in the environment which is likely to be the least visible. First, the evader calculates the *min visibility location* based on his current knowledge of the map. This is accomplished using the same Algorithm 3 that the pursuer uses except the minimum area location is chosen instead of the maximum area. After this location is found, it is temporarily sent to the planner as the new goal location. The original goal will resume after the evader has successfully prevented line of sight to the evader for some fixed amount of time (e.g., 5 seconds).

- Random Walk - This is the same method that the pursuer uses to help escape local minima by allowing reactive control based on local sensing to override deliberate path planning [2].

Therefore the evader's strategy uses the same algorithms as the pursuer, but utilizes different objective functions. One difference to note is that the evader never has the explicit objective to explore the environment, but still uses all of his sensor feedback from t_0 to $t_{current}$ in order to effectively flee from and evade the pursuer.

VI. Simulation Approach

We created our simulation platform from scratch in C++ using OpenGL for graphics. While other flexible simulation platforms exist such as Player/Stage [3], our platform is specifically designed to handle high computational requirements and parallel algorithms, such as evolutionary algorithms, running for multiple agents. Recall from Section III that the only pre-existing knowledge each agent needs is the color (or color range) of obstacles and other agents. This approach effectively allows the agents to navigate a picture as opposed to navigating the geometric representation of a map. After inputing the required color ranges of obstacles and agents our agents can navigate any environment that is (or can be abstracted to) a 2-D picture. This includes blueprints of a building, a Google map, or even your desktop background picture.

While our method of sensing and map representation allow for flexibility in *where* the agents act, it is equally important to allow for flexibility in *how* the agents act. In our platform, we employ the evolutionary algorithm explained in Section II. This permits the easy addition of new future behaviors for any agent simply by adding a variable the evaluation function and deciding which environment changes/sensor input should be linked to the weight of that variable.

This flexible nature of our platform allows for a multitude of different experimental environments and configurations. We first analyze which characteristics of the environment will have the greatest impact on our agents' behaviors and attempt to set up an experiment to maximize these characteristics.

First, consider the layout of the environment's obstacles. Our vision scanning system allows us to model an environment that contains many complicated obstacles that would be difficult to represent geometrically. One characteristic of the environment is the density ratio of obstacles to free space. A higher density limits visibility and therefore is more complex to online pursuit-evasion. Next, we identify the presence of a *trap* by the following: 1. an obstacle whose concavity is larger than the largest dimension of a robot, 2. two or more of convex obstacles together such that the gap between them is smaller than the smallest dimension of a robot. The difficulty in navigating amongst these types of obstacles which are initially unknown is that the robot must often explore the complete obstacle before realizing that it is impassible. This is why so much previous work has excluded this case in order to maintain theoretical limits on time or space complexity in finding optimal paths. We believe the presence of traps to be a key component in determining how robust a model for pursuit and evasion is and therefore the environment in our experiment contains many traps.

Now that the environment characteristics are decided, we must consider the starting agent configurations. We found the relative speed of the agents to be of paramount importance. We run our simulation on the same complex environment with three variations of relative speed: 1. both agents have the same speed 2. the pursuer is 10% faster than the evader, and 3. the pursuer is 20% faster than the evader. Coincidently, the starting

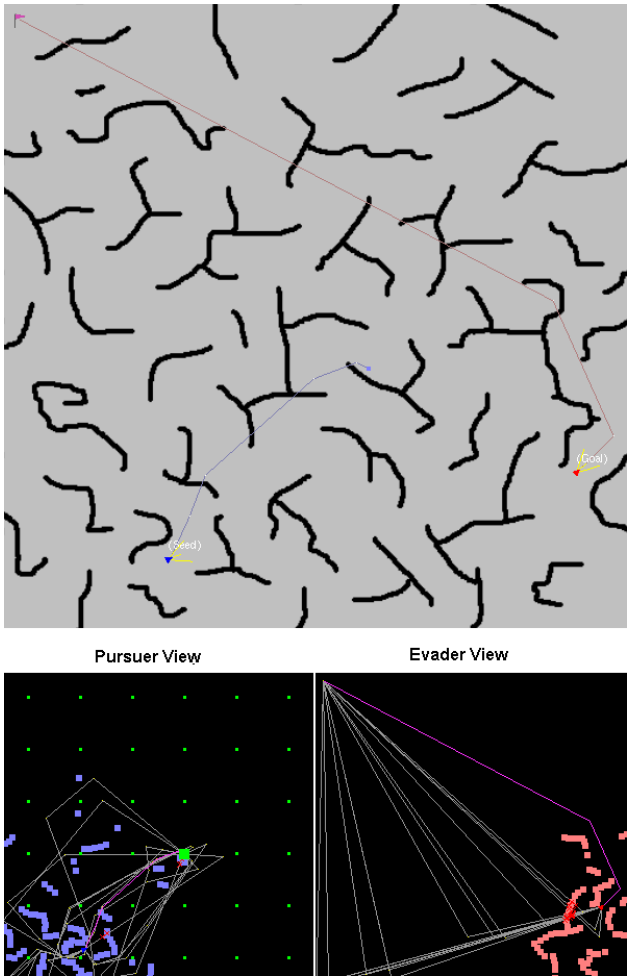

Fig. 4. Simulation map containing many obstacles and traps.

location and visibility of each agent is equally important. In the context of pursuit and evasion, however, it makes little sense to consider simulations in which the agents are unlikely to ever see each other, and therfore pursuit and evasion could never take place. In response to this, we simulate only the situation in which the agents begin in two different corners with no visilibity of each other, but are likely to see each other in the center because the evader's goal is in his opposing corner. Figure 4 shows our environemnt, the agent and goal starting locations, and each agent's internal representations of the environment shortly after a simulation has started. The evader is likely to cross the center of the environment on his way to his known goal, and the pursuer is likely to cross the center of the map in his exploration done through visibility seeding explained in Section IV-A.

VII. Results

The three scenarios of the pursuer having equal speed, 10% greater speed, and 20% greater speed than the evader discussed in Section VI were simulated 50 times each and the average results from each scenario were recorded. We define a pursuer win to be the puruser physically touching the evader before the evader reaches his goal. We define an evader win as the evader reaching his goal location without being touched by the pursuer. A visual summary of the actual paths traveled by each agent in the 50 simulations from scenario 3 is shown in Figure 5. The blue dashed lines represent the actual paths traveled by the pursuer with a blue X to show his end location for that simulation. The red dashed lines show the actual paths for the evader with a red X at its end location. We have labled the following areas in the image to help the reader interpret it.

- Start - The labels for Pursuer Start and Evader Start specify the beginning location of each agent in the bottom left and bottom right corners respectively. The evader's goal is static and in the top left corner.
- Initial Confrontation - This shows where the agents often saw each other for the first time. At first sight, the evader would go into flee mode and the pursuer would begin using his prediction algorithm and/or smart seeding.
- Evader Trap - This concave region which we call a *trap* causes the evader to get temporarily stuck several times as shown by the amount of path lines drawn there.
- Close Calls - These are particularly interesting because the pursuer's ending location was very close to the evader's goal at the end of the simulation. This typically means that the pursuer was in "'hot pursuit"' at the time of the evader's success.
- Pursuer Success - Areas that contain red X's that are not in the top left goal location show that the pursuer successfully caught the evader at that point.
- Evader Success - This is the static goal location that the evader reached without being caught by the pursuer.

TABLE I
Average results of 3 scenarios which were simulated 50 times each.

Scen #	*Pursuer Speed Advantage*	*Evader Wins*	*Pusuer Wins*	*Avg Time (s)*	*Evader Distance To Goal (in)*	*Pusuer Distance To Evader (in)*
1	0%	33	17	23.548	207.764	151.894
2	10%	29	21	32.779	218.489	192.051
3	20%	32	18	29.542	174.968	179.980

We divide our results into two categories. The first, shown in Table I, are the average outcomes of each simulation with respect to wins/losses. The distance of each agent to his respective goal at the end of a simulation is reported to indicate to what degree an agent won or lost. Finally, the average time in seconds of each simulation is also reported. The second category, depicted in Figure 6, is the breakdown of each agent's time allocated to each behavior over the course of the simulations. A sample video with excerpts from some of the simulations is provided with this paper.

Given the initial setup as discussed previously, we expected the outcome to be fairly balanced. For instance, if the pursuer and evader often meet in the center of the map, it may be equally likely that the pursuer successfully catches the evader using his prediction algorithm and smart seeding compared

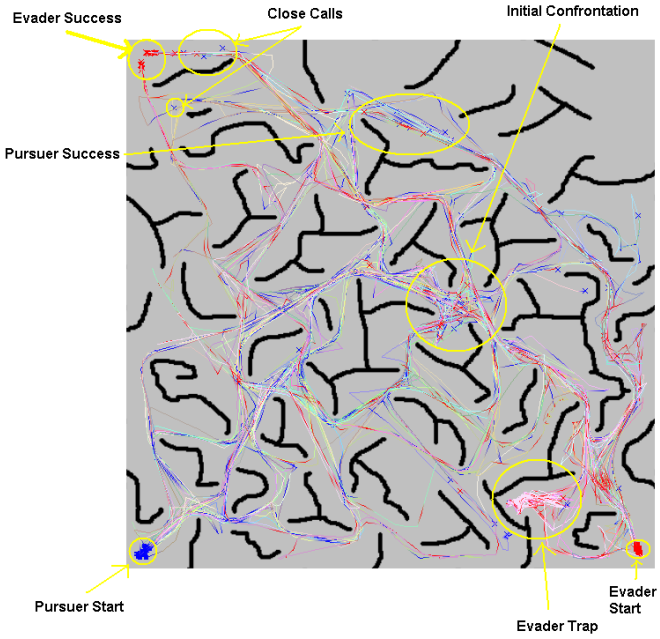

Fig. 5. Overlay of actual paths traveled by each agent over 50 simulations of scenario #3.

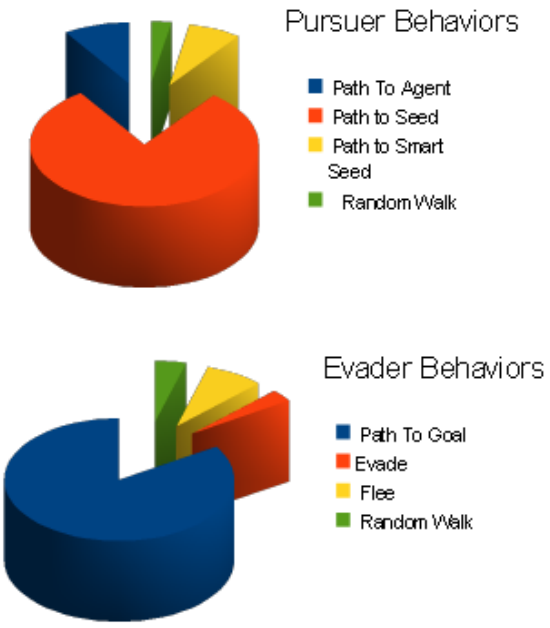

Fig. 6. Breakdown by percentage of time each agent spends acting under their respective behaviors.

to the evader properly utilizing his flee and evade objectives. The results indicate that even when the pursuer has up to a 20% speed advantage over the evader, the evader still wins a significantly higher number of simulations. This makes sense when we consider the pursuit and evasion of humans and that it is faster to act than react. On the other hand, even when the pursuer has the same speed as the evader, ie. lacking speed advantage (scenario 1), the pursuer can still win a significant amount of times. This is due to the efficiency of the pursuer's prediction algorithm and also to the presence of traps.

We can also reason about the behavior distribution over time. Notice from Figure 6 that there is a positive correlation between how often the pursuer is making an intelligent guess on where the evader might be (smart seeding) and how often the evader is immediately fleeing the pursuer. This makes sense because both agents must modify their behaviors after one agent has acquired visibility of the other. The pursuer spends a majority of the time exploring (path to seed) while the evader spends a majority of the time on a deliberate path to his goal according to his planning algorithm. It is interesting to see that even though the evader spends time on a deliberate path to a known goal, the pursuer still manages to catch the evader about 31% of the time. This shows the effectiveness of the pursuer's method of exploration based on estimation of high visibility.

VIII. Conclusion & Future Work

The presented experimental results in Section VII shows that we have created a robust platform for testing different behaviors of agents and determining how effective those behaviors are in the context of pursuit and evasion. We have utilized an adaptive planning paradigm to accommodate the changing objectives of both the evader and the pursuer in an unknown environment. The agents created a mental representation of the environment as real-time sensing data were made available to them and often navigated intelligently.

While our approach does provide a new robust framework for research in the area of pursuit and evasion with impressive results, there is much future work to be done. The existing strategies and behaviors can be improved and extended (e.g. adding more patterns for prediction). In addition, more realistic sensing that takes into account uncertainty could be modeled. A greater variety of simulations can be experimented, including real-world environments abstracted to 2-D pictures/maps. More future work may include the addition of multiple agents in cooperation and competition using team communication to pool map knowledge and make more accurate predictions of the location of the evader(s). Other extensions could include adding additional intelligent behaviors to the agents. Beyond these multiple uses of the existing architecture presented, a complete shift into a three dimensional simulation environment is the next step in advancing the practical applications of this pursuit and evasion research. Finally, a vast area for future work is to take into account realistic vehicle dynamics (with realistic accelerations) to plan true trajectories rather than paths. With trajectories being planned, they could also take

into account dynamic objects in the environment, where not only is the environment unknown. All of these extensions are fertile ground for new research.

ACKNOWLEDGMENT

The first author gratefully acknowledges the support of the GAANN Fellowship from the U.S. Department of Education.

REFERENCES

[1] P. Bonissone, R. Subbu, N. Eklund, and T. Kiehl, "Evolutionary algorithms + domain knowledge = real-world evolutionary computation," *Evolutionary Computation, IEEE Transactions on*, vol. 10, no. 3, pp. 256–280, June 2006.

[2] R. Brooks, "A robust layered control system for a mobile robot," *Robotics and Automation, IEEE Journal of*, vol. 2, no. 1, pp. 14–23, Mar 1986.

[3] B. P. Gerkey, R. T. Vaughan, and A. Howard, "The player/stage project: Tools for multi-robot and distributed sensor systems," in *In Proceedings of the 11th International Conference on Advanced Robotics*, June 2003, pp. 317–323.

[4] V. Isler, S. Kannan, and S. Khanna, "Randomized pursuit-evasion in a polygonal environment," *Robotics, IEEE Transactions on*, vol. 21, no. 5, pp. 875–885, Oct 2005.

[5] N. Karnad and V. Isler, "Bearing-only pursuit," in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, May 2008, pp. 2665–2670.

[6] A. Kolling and S. Carpin, "The graph-clear problem: definition, theoretical properties and its connections to multirobot aided surveillance," in *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, Oct 2007, pp. 1003–1008.

[7] S. Oh, L. Schenato, C. P., and S. Sastry, "Tracking and coordination of multiple agents using sensor networks: System design, algorithms and experiments," in *Proceedings of the IEEE*, Jan 2007, pp. 234–254.

[8] J. Vannoy and J. Xiao, "Real-time adaptive motion planning (ramp) of mobile manipulators in dynamic environments with unforeseen changes," *Robotics, IEEE Transactions on*, vol. 24, no. 5, pp. 1199–1212, Oct 2008.

[9] R. Vidal, O. Shakernia, H. Kim, D. Shim, and S. Sastry, "Probabilistic pursuit-evasion games: theory, implementation, and experimental evaluation," *Robotics and Automation, IEEE Transactions on*, vol. 18, no. 5, pp. 662–669, Oct 2002.

[10] J. Xiao, Z. Michalewicz, L. Zhang, and K. Trojanowski, "Adaptive evolutionary planner/navigator for mobile robots," *Evolutionary Computation, IEEE Transactions on*, vol. 1, no. 1, pp. 18–28, April 1997.