

# A Framework for Learning Predator-prey Agents from Simulation to Real World.

Jiunhan Chen, Department of Computer Science, Vrije Universiteit Amsterdam, the Netherlands  
 Zhenyu Gao, Department of Computer Science, Vrije Universiteit Amsterdam, the Netherlands

**Abstract**—In this paper, we propose an evolutionary predator-prey robot system which can be generally implemented from simulation to the real world. We design the closed-loop robot system with camera and infrared sensors as inputs of controller. Both the predators and prey are co-evolved by NeuroEvolution of Augmenting Topologies (NEAT) to learn the expected behaviours. We design a framework that integrate Gym of OpenAI, Robot Operating System (ROS), Gazebo. In such a framework, users only need to focus on algorithms without being worried about the detail of manipulating robots in both simulation and the real world. Combining simulations, real-world evolution, and robustness analysis, it can be applied to develop the solutions for the predator-prey tasks. For the convenience of users, the source code and videos of the simulated and real world are published on Github<sup>1</sup>.

**Index Terms**—Evolutionary robotics, Predator-prey, NEAT, Robot operating system.

## I. INTRODUCTION

Predator-prey is a classical pursuit-evasion problem. A typical scenario is that there are a predator and a prey in a square arena, the predator must catch the prey within a certain time. The game ends when the predator catches the prey or time is beyond the limitation. The application of the solution of predator-prey problem might be used for searching, rescuing, exploration...etc. If we extend the predator-prey problem as multiple predators to catch a prey. The collaboration among multiple agents strengthen the reliability and scalability of completing a task.

The training for robots in the real world can be a time-consuming and money expensive task. An alternative way can be that we train the robots in the simulation world, and then we transfer the well-trained robots to the real world. Also, in the real world, we can even train the trained robots for a shorter time. The simulation environment can be created by Gazebo<sup>2</sup>, which is a widely used 3D real-time simulation physics engine.

There are different methods to solve the predator and prey problem. An early work from Bryson and Baron considered the pursuit-evasion problem as a differential game, it means the solution of the pursuit-evasion problem can be solved analytically when the map and the pose of the predator and the prey are known [1]. Another the work was from Raboin et al., they invented an algorithm to solve partially observable pursuit-evasion problem with optimizing the uncertainty of the prey [2]. In the meantime, some researchers tried to solve this

problem with Evolutionary Algorithms for various scenarios. We followed their steps and tried to explore interesting things from that point.

Our research topics includes that we implement different experimental settings to evolve the predators and show their advantages and disadvantages. Also, we aim to provide an simulated experimental environment for the predator-prey task. From an experimental point of view, many aspects should be considered in this task. We can ask ourselves: "What's the number of predators? Can we change the number dynamically? Predators are controlled by a single controller or multiple controllers? Do we evolve the predators and the prey at the same time or we use a fixed strategy for one party? Is the environment fully observable or partially observable?" These questions help us to develop our research.

we expand our work to a partially observable environment. Instead of using an overhead camera, the sensors from the robots are used. For the more powerful sensors, in this paper we use the wheeled robots, Robobo<sup>3</sup> that combines a simple mobile base with your smartphone to create the next generation of educational robots. The robot has short-range (20cm) IR sensors and a camera by carrying a mobile phone on itself. In order to reduce the reality gap, we must simulate the camera and IR sensors in Gazebo. After finishing the evolution in simulation, we can transfer the evolved agents into the real world to further evolve or examine the performance. Both the predators and the prey co-evolved, even though the "arms race" might be not easy to be triggered, which is a key point for competitive coevolution. Fortunately, "Hall of Fame" is a technique that can stabilize the competitive coevolution process. We choose to co-evolve the sensor-based predators with an all-knowing prey in the simulation world, and the sensor-based predators can be transferred to the real world immediately. In the simulation world, we can access the data which is hard to get from the real world to help the evolution process. An all-knowing prey might encourage predators to develop better performance. Also, the fitness function for the predators is computed by the distance between agents, which is hard to get in the real world.

## II. RELATED WORK

Most of the existing studies in evolutionary robotics address the co-evolution of pursuit and evasion strategies. That is, both the predator(s) and the prey(s) evolve their behaviour. One of the early works is from Nolfi and Foreano [3] who perform

<sup>1</sup>[https://github.com/chenjiunhan/Predators\\_and\\_Prey/](https://github.com/chenjiunhan/Predators_and_Prey/)

<sup>2</sup><http://gazebosim.org/>

<sup>3</sup><https://theroboboproject.com/en/>

co-evolution of one predator and one prey to investigate the "arms races" phenomenon in a real world setting. The predator and prey are placed in a square world  $47\text{cm} \times 47\text{cm}$ . Several researchers simulated the predator-prey environment with the software. The article from Yong and Miikkulainen [4] shows that three predators corporately catch a prey, which is achieved in a  $100 \times 100$  toroidal grid simulated world. Haynes and Sen [5] put the prey and the predators in  $30 \times 30$  grid simulated world. Using such simple settings for simulation helps the research on different algorithms and strategies for predators. However, if the goal is to have robots applied in the real world then more realistic simulation environments are needed and even that may not be enough because of the infamous reality gap [6].

Concerning the behaviour of predators, we can distinguish two cases: the predators can be homogeneous or heterogeneous, that is, the predators have the same controller or not. The work from Bryant and Miikkulainen [7] notes that homogeneous agents can have some advantages compared to heterogeneous ones in some cases, for example, when the size of the team changes, although Haynes and Sen [5] reports that heterogeneous agents are better in the symmetrical setting like predator-prey. Most of them select neural networks as controllers. Yong and Miikkulainen [4] set inputs as the offset between "predator and prey" and "predator and predators" and choose heterogeneous agents, which makes the controllers able to coordinate different predators. We choose to use homogeneous agents to make our system easily scalable, i.e., our system works with the different number of predators. With features selection, we can reduce the dimensions of weights in the neural network, also make predators show obvious behaviour of the collision avoidance, which is similar to bird flock avoid the collision from each other in swarm intelligence [8].

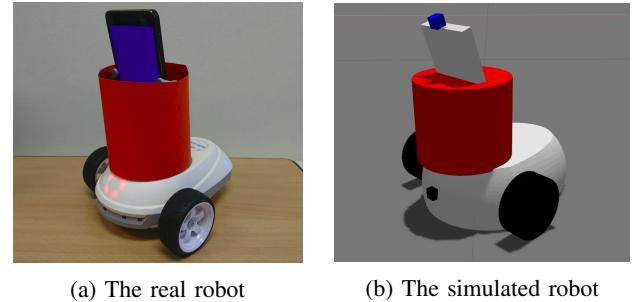
Predators need a good opponent for evolving a complex strategy to catch a skillful prey. There are different strategies for prey in the literature. Jeong and Lee design a fuzzy logic controller using genetic algorithm [9]. Nolfi and Foreano [3], [10] evolved both the predator and the prey, which can suffer from "arms races" and the "red queen effect" [11]. But the state-of-the-art shows that it can be improved by Cooperative Multi-objective Evolutionary Algorithms [12]. In the work from Yong and Miikkulainen [4], the evasion strategy of prey is directly away from the nearest predator. In this work, we design a 'smart' strategy based on the gradient of the Gaussian function, which will be introduced in our previous study [13]. It is easy to implement and applicable in both the simulated world and the real world.

### III. METHODOLOGY

#### A. Robots

In the paper, Robobo is our new robot to perform our experiments. Robobo is capable of carrying a mobile phone as a camera. We can use the camera to locate the relative position of the predators and the prey. We choose to use the front camera of the mobile phone to reduce the delay of data processing, because the resolution of the front camera is

relatively low. Robobo has five front IR(infrared) sensors and three back IR sensors. The maximal range of IR sensors are only 20 cm. so we merely use one front IR sensor in the middle for collision detection. Robobo has five front IR(infrared) sensors and three back IR sensors, but we merely use one front IR sensor in the middle to avoid from collision. The blue cube in Figure 1b represents the camera of Robobo. The red cylinder is for the camera to detect the robot, and the prey can be distinguished by using green cylinder.



(a) The real robot (b) The simulated robot

Fig. 1: Robobo in the real world and the simulation.

#### B. The Simulation Environment

1) *Integrating Gym of OpenAI, ROS, and Gazebo:* Two of the most popular simulation environments are Gazebo and VRep. But it spends time to build an environment for research requirement. That's why there are platforms such as Gym<sup>4</sup> from OpenAI to encapsulate the environment with an interface to perform action such that users can only focus on algorithms. So we integrate Gym, ROS, and Gazebo together to process all the details including controlling robots, resetting the experiment and getting world information. Figure 2 shows the architecture of the integration.

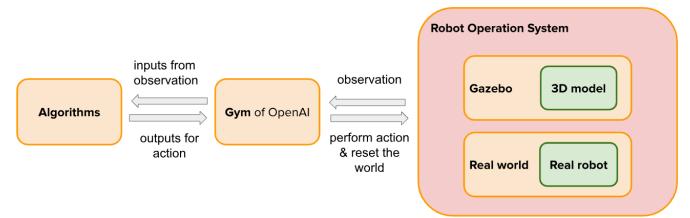


Fig. 2: Integrating Gym, ROS, and Gazebo. The users only need to focus on algorithms by receiving observed world data and generating output for action. The rest of things will be processed by Gym and ROS. RROS encapsulate Gazebo and the real world to provide the control interface for Gym.

In the beginning of one evaluation, the predators are placed parallel to one side of the wall, the prey is placed at the center. The arena remains a square field, but the size is double as  $4\text{m} \times 4\text{m}$  to allow that the robots move more freely. Figure 3 shows the environment in the simulation world and the viewpoints from the different agents. When

<sup>4</sup><https://gym.openai.com/>

the robot detects the other robots, a bounding box is created to select the region of detected agent by color detection based on hue difference among colors[14]. The gazebo allows user to simulate IR sensor and camera, so we try to make the difference between the simulation and the real world as small as possible. The 3D model of the robots were created by FreeCAD, which is an open source technical drawing software to create 3D models. Figure 4 shows the sketch of the robot and its 3D model in FreeCAD.

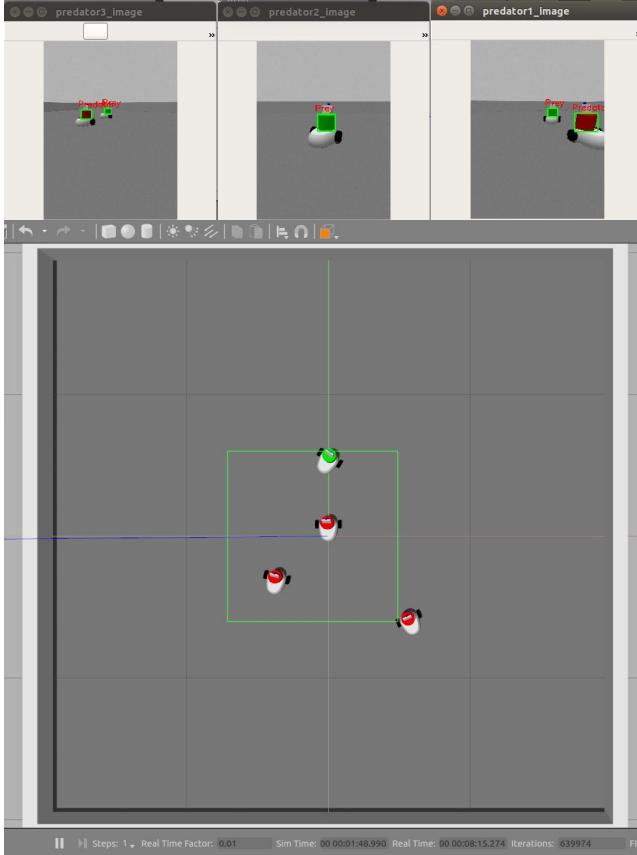


Fig. 3: The simulation environment. Three windows on the top are the views from different agents. The detected objects are selected by the bounding boxes.

### C. The Real World Environment

The real world environment is a  $4m \times 4m$  arena. In this scenario, we use the embedded camera instead of overhead camera system. The combination of camera and IR sensors are used to localize the coordinate of a robot itself and other robots. The real-time robot vision [14], [15] are partly applied to recognize other robots with colorful wrap. Three predators and one prey are connected with a computer via Wifi. A ROS master server is running on the computer. So robots pass their information from sensors to the computer via ROS, and then the controllers on the computer take data from sensors as inputs to generate outputs. In the final, the computer send control command back to the robots. Figure 5 shows the viewpoints from the agents and the experimental setting in the real world.

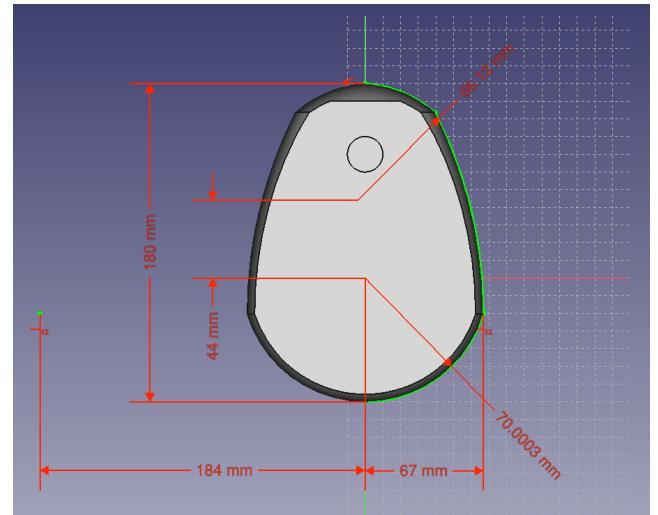


Fig. 4: Sketch of the robot in FreeCAD

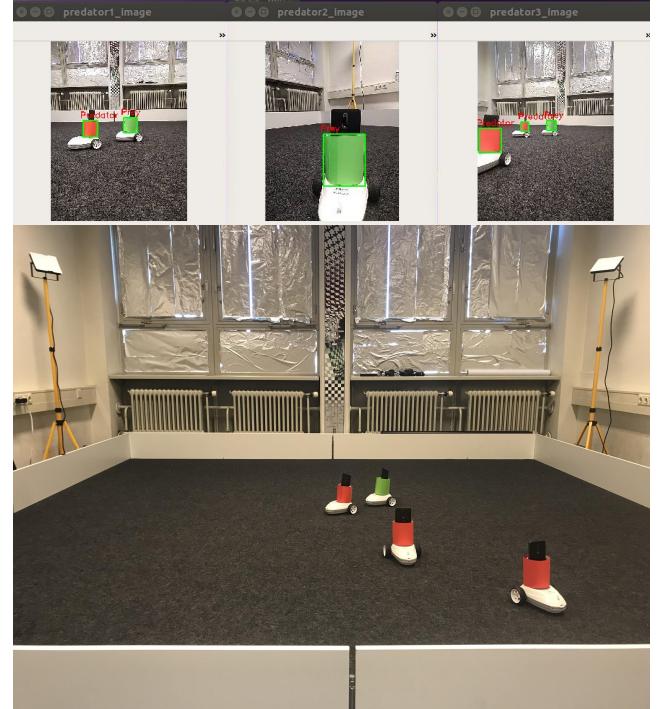


Fig. 5: The real world experiment. Three windows on the top are the views from different agents. The detected objects are selected by the bounding boxes.

### D. Partially Observable

Unlike other systems with an overhead camera, we merely rely on the camera and the IR sensors on the robots themselves. This choice can extend the application of the predator-prey evolution system. However, we must face another challenge when we merely rely on the sensors of robots, which is that the world becomes partially observable for robots. When a robot is placed at a new place, it can't be sure where it is in this world and how it should move until the robot has detected the environment nearby itself. Take self-driving vehicles as an example, a self-driving system contains multiple deep neural

networks to detect the environment, including the model to detect pedestrians, the model to detect traffic light...etc. Based on that environment information, the computer of the vehicle can make the decision of the next action.

#### E. Controller for prey

The controller of the prey and the controllers for the predators are both evolved by NEAT. Our purpose is to evolve the predators with the sensors to catch the prey. One of the advantage of the simulation environment is that we can access specific data easily, compared to the real world. For example, we are able to directly get the pose of robots from Gazebo, however, in the real world, we must rely on some techniques like SLAM or GPS to locate the pose of robot. This fact allows us to create a all-knowing prey to help the evolution of predators. Here we list the inputs and the outputs of the prey:

#### Input Layer:

- $\Delta\theta_1, \Delta\theta_2, \Delta\theta_3$ : The angle difference between the orientation of the prey and the direction of the predators with relative to the prey.
- $d_1, d_2, d_3$ : The distance between prey and predators.
- $x, y$ : The coordinates of the prey.

#### Output Layer:

- $\omega_L$ : Angular velocity of the left wheel
- $\omega_R$ : Angular velocity of the right wheel

Using hyperbolic tangent as the activation function.

To be aware of that the inputs are much more than the predators controller, which we can see in next subsection. This fact implies the mission is much harder for the prey, because the difference in the number of agents. Also, in most of situations, the predators only need to focus on moving toward the prey, however, the prey can't just simply run away from the predators, instead the prey has to turn 90 degrees to avoid from hitting the wall or being trapped from multiple predators. To evolve a better prey, we need a robot with wide-ranging sensors, otherwise, we have no choice but to use an all-knowing prey to make predators better.

#### F. Heterogeneous controllers for predators

Instead of the all-knowing prey, the predators can merely rely on their cameras and IR sensors. Using the heterogeneous controllers for each predator can increase the diversity of the predators and prevent from evolving into monotonous behaviour like only trailing the prey. We tried various inputs, but the simple inputs have a better effect. Here we list our inputs and outputs for the controllers of the predators:

#### Input Layer:

- $x_{image}$ : The horizontal position of prey shown in the image coordinate, the center is defined as zero.
- $A$ : The area of the prey in the image, and -1 if there is no prey in the image.
- $c$ : +1 if middle front IR sensor detects an object, otherwise -1.

#### Output Layer:

- $\omega_L$ : Angular velocity of the left wheel
- $\omega_R$ : Angular velocity of the right wheel

Using hyperbolic tangent as the activation function.

#### G. Standard Coevolution Framework

In the paper, we want to try heterogeneous controllers for the predators, therefore, there are four targets that need to be evolved, three predators and one prey. Assuming one of the predators is evolving, what are the teammates and the opponent that we should pick for an evolving target? For the teammates and the opponent we both pick the individuals with the best fitness from the previous generation, and randomly pick for initialization. It's more meaningful to cooperate with better teammates, because if we randomly pick two teammates for the evolved target, which leads to the performance of the team more depends on the teammates but not evolved target. The noise of fitness could be huge and more randomly. Also, it's more meaningful to compete with the best opponent, if the evolving predator can raise the performance even against the best opponent and the teammates are fixed comparing to the individuals in the same generation, which means the performance should be credited to the evolving predator. Otherwise, we don't know a good or bad performance should be credited to the evolving predator or a randomly chosen prey. Potter and De Jong talked about the architecture for the standard coevolution framework[16]. We apply the standard coevolution framework to our predators and prey task, and the architecture of coevolution can be drawn as Figure 6

we evolve all the agents alternatively. Except for the evolving target, the other agents are usually controlled by the best controllers from the previous generation. But we also implemented the technique "Hall of fame" for competitive coevolution, which means that the agent must play against the best opponents from each earlier generation. To reduce the evolution time, we make the evolving target to play against the best opponents from the randomly previous 10 generations. The parameters of NEAT is given in Table I

parameter	value
Number of generation	100
Population size	20
Weight mutate rate	0.8
Bias mutate rate	0.7
Probability of adding(deleting) connection	0.1
Probability of adding(deleting) node	0.1
Number of elites	4

TABLE I: The main parameters of NEAT

#### H. Fitness Function

We take the fitness function from [17] as a reference. For the prey, the fitness is computed by the survival time, the longer the prey survives, the higher the fitness can be. So the upper bound of fitness is when the prey never get caught

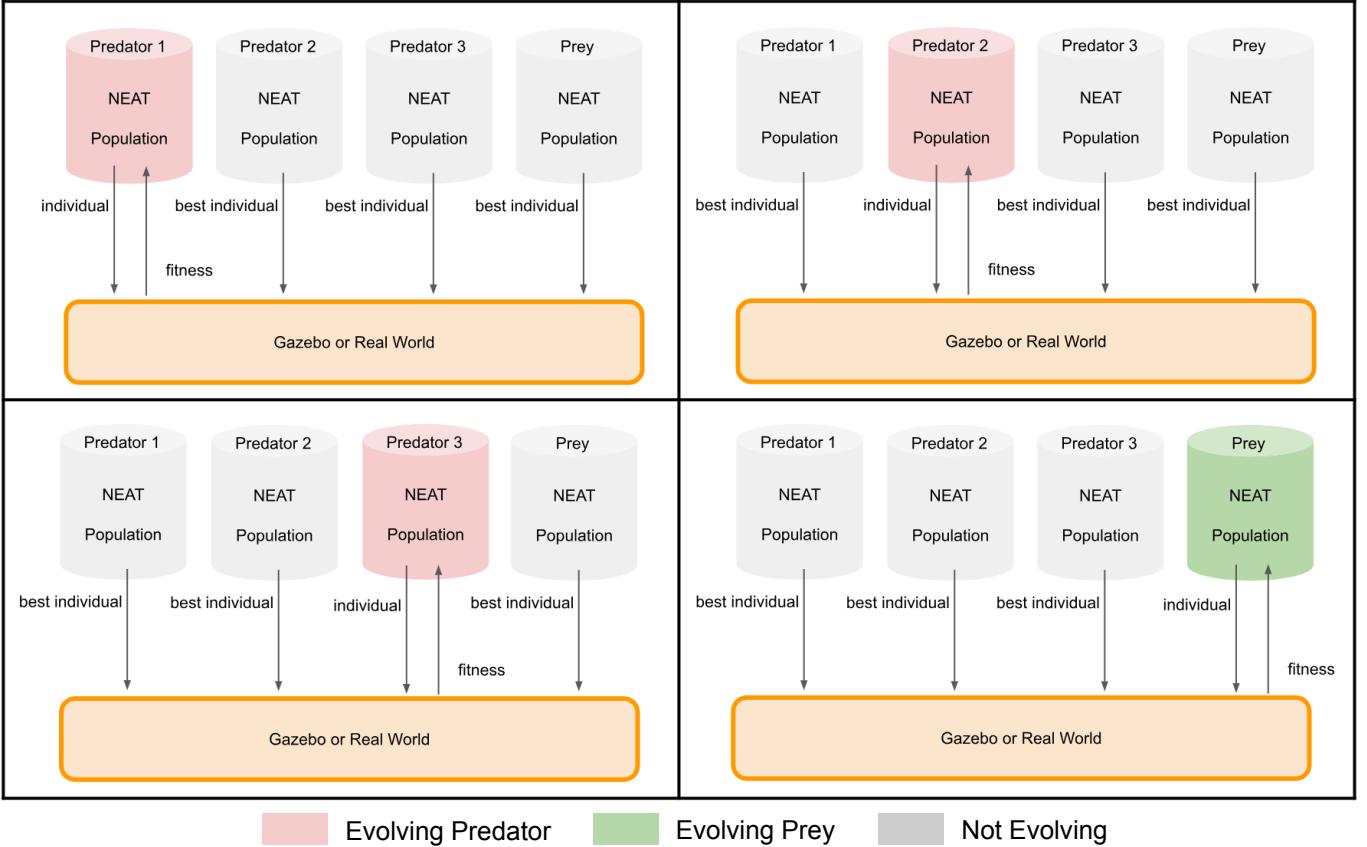


Fig. 6: The standard co-evolution framework. Each agent evolves alternatively. The evolving agent(pink or green) tests its individual and get fitness as feedback. The other agents(grey) send the best controller from the previous generation to form the team or be the opponent(s).

during the evaluation time 30 seconds. For the predators, we choose to use selfish fitness function for each predator, in other words, only the final distance between evolved predator and the prey is considered. We want to emphasize that one evaluation is ended when one of the predators catches the prey. If there is an outstanding predator, it may lower the fitness of the other predators, which can motivate the other predators to become the first one who can catch the prey. But if all the predators are so selfish that they merely chase behind the prey, which may fail to catch the prey. We also tried to share the same fitness between all the predators, however, it seems that an outstanding predator usually makes the other predators lazy. The outstanding predator confuses others so that the other predators don't know if they are good. Here we display the mathematical representation of the fitness functions:

#### Fitness function for prey:

$$f_{prey}(\mathbf{t}) = \frac{1}{K} \sum_{i=1}^K \frac{t_i}{T} \quad (1)$$

$T$  is a constant for evaluation time 30.  $t_i$  is caught time at  $i_{th}$  evaluation,  $t_i = T$  if the prey is not caught.  $K$  is the number of evaluations. The opponents are selected by randomly previous 10 generations.

#### Fitness function for a predator:

$$f_{predator}(d) = \frac{1}{K} \sum_{j=1}^K \frac{1}{d_j} \quad (2)$$

$d_j$  is the distance between the evolving predator and the prey at  $j_{th}$  evaluation.  $K$  is the number of evaluations. The opponents are selected by randomly previous 10 generations. The reason that we use the inverse of distance instead of that a constant minus  $d$  is because there is a huge difference between "close" and "extremely close". Otherwise, the predators will be inclined to get easy points by staying somewhere close to the prey but never catch it.

## IV. RESULTS

The experiments include the evolution process and the evaluation in the simulation world, and evaluation in the real world.

#### A. Evolution and Evaluation in simulation

As we introduced the standard coevolution framework in subsection III-G, we evolved both the predators and the prey alternatively with 100 generations and relatively small population size 20. In our previous study [18], [13], the small

population size, even 13 is enough to evolve the agents. After the evolution process is finished, we tried to evaluate the controllers with Master Tournament, which means that we select the controllers with the highest fitness in every generation, and then we make them play against each other. So there will be  $100 \times 100$  evaluation times for all the 100 generations of both the predators and the prey.

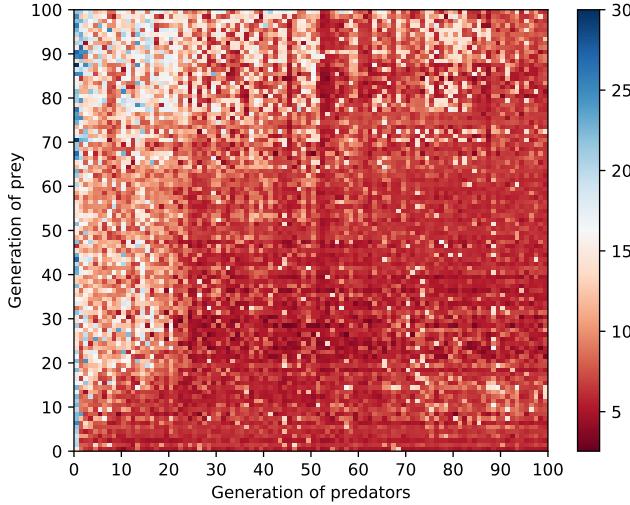


Fig. 7: The best prey from 100 generations versus the best predators from 100 generations. It shows a trend that the agent from the later generation performs better when it plays against the agent from the early generation.

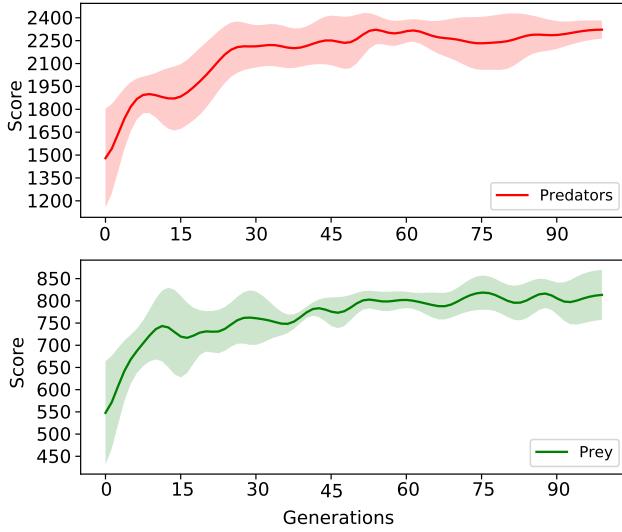


Fig. 8: The best prey from 100 generations versus the best predators from 100 generations. The graph shows the accumulated score for each generation and the final performance is stabilized. The shadow represents the range within one standard deviation. The line represents the average from three repeated experiment.

Figure 7 shows the caught time of the master tournament. The later generation inclines to perform better when it competes against early generation. However, we can see that the task is more difficult to the prey from that most of values are

below 10 seconds. Also the instability happened nearby 90th generation of predators and 13th generation of prey.

Figure 8 is drawn by accumulated scores of agents. The score of prey is defined by caught time, and it accumulates a prey from a generation to play against all generations of predators. The score of predators is defined by that a constant minus the caught time, and the calculation of the accumulated score is similar. The graph shows that the coevolution can be stable in this evolution framework, although it seems that the upper limit of performance is hit around the 50th generation of predators.

To see that the behaviour was getting more complicated, we cherry-picked the predators from the 41st generation play against the preys from all the generations, and draw their trajectories as shown in Figure 9. During the generations between 0 to 19, the best strategy for the prey is to move to the corner for prolonging the survival time. From the 20th to the 39th generation, the prey developed the strategy to slip away from the gap between the predator and the wall. The prey successfully went through the gap at the 24th generation. The generations from 40 to 100, Although, the performance for prey didn't change too much, the prey still tried to run away from the predators along different paths.

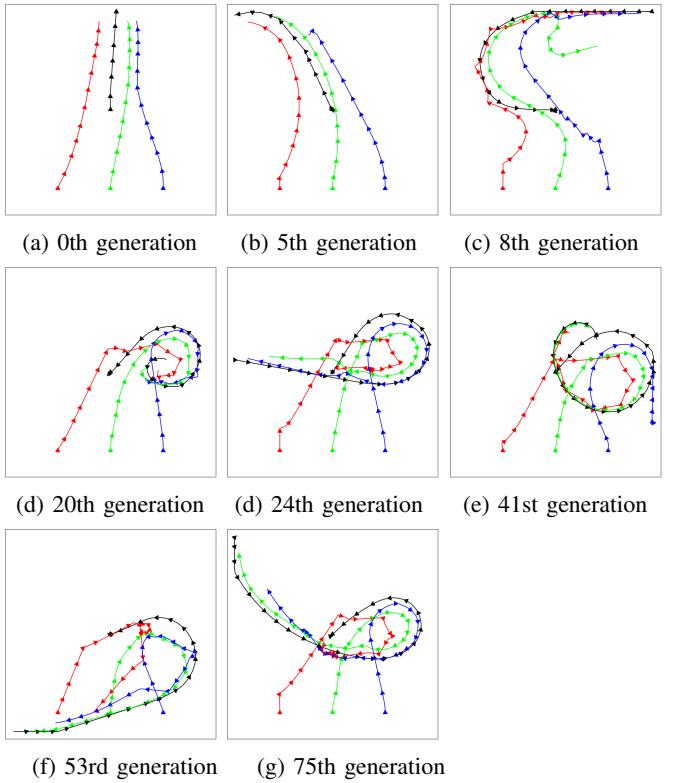


Fig. 9: The trajectories of the predators from 41th generation versus the prey from the other generations

### B. Evaluation in the real world

In the real world, the prey cannot be an all-knowing player, but we are still interested in evaluating the performance of the evolved predators when the controllers are transferred from

the simulation world to the real world. So we use multiple human prey players to evaluate the performance of evolved predators in both simulation and the real world.

We created an interface for the human to control the predator with keyboard for both the simulation world and the real world. However the human players may introduce the bias, taking the average from multiple human players is necessary. The time spent to catch the human prey player is the criterion. The result can be shown in Figure 10. The average survival time in the simulation world is 8.43 seconds, however in the real world the average survival time is 21.59 seconds. Even though the predators in the real world show the behaviour of pursuit, they still need further evolution to be applied in the real world. The reality gap will be further discussed in the section V.

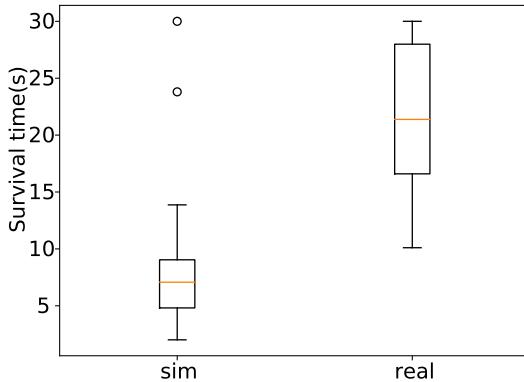


Fig. 10: The performance of the human controlled prey versus the evolved predators in both the simulation world and the real world.

## V. DISCUSSION

### A. The reality gap

The number of frames per second(FPS) could be a source which leads to increase the reality gap. FPS in the simulation world is around 10(because of the computation of simulation, image processing, and running algorithm), however due to the transmission of high quality of images via local WiFi, ROS and a router, the FPS in the real world can only be 5-7. The lower FPS has an impact on the object detection, some frames can be missed which leads to over rotate, even though during the rotation the object can be detected as shown in Figure 11.

### B. Triggering the arms race

The arms race is not easy to be triggered. Even the observed behaviour becomes more complicate, it's not necessary that the performance is always better. However, the advantage of coevolution is to evolve various opponents without delicate human design. With the help of "Hall of Fame", the predators are able to catch various opponents and get higher average fitness. The variety of the evolution process makes the predators complete their evolution target in a certain degree. "Hall of Fame" can be implemented by choosing the opponent by random sampling. We also tried to select opponents from 10 previous

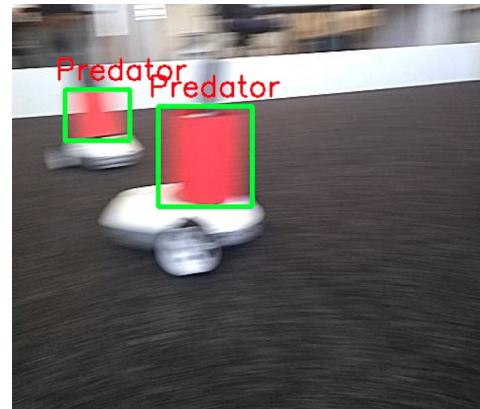


Fig. 11: The viewpoint from a rotating robot. Two predators are detected.

generations, but it cannot stabilize the coevolution process. A sampling method for selecting opponents is necessary to stabilize coevolution. However, if we select opponent random sampling, it means that when the generation increases, it's possible that the sampling method, unfortunately, selects only bad performance controllers, which may lead to misestimating fitnesses, and it may further lead to hitting the upper limit of performance. In our experiment, the predators hit the upper limit of performance about 50th generation, so we may improve the performance with performance-based methods as mentioned in the work from Rosin and Belew [19].

### C. Fitness function

In a few preliminary experiments, different fitness functions are tested. For the predator, fitness function  $f(d) = \text{constant} - d$  and  $f(d) = \frac{1}{d}$ ,  $d$  is the distance between the prey and the evolving predator, have a huge difference, because it distinguishes the difference between "close" and "extremely close". The fitness function also can be defined by team performance, instead of only individual performance. In our other preliminary experiments, the fitness which defined by full team performance usually generates a lazy individual. It may be caused by that the difficulty of this task for multiple predators is relatively lower to the prey. The capitalism styled fitness function  $f(d) = \frac{1}{d}$  works well at least in the simulation world, in other words, every predator works for itself to maximize the resource on its hands, but the whole team can be benefited from the result too. The predators are also not that selfish to just follow behind the prey, otherwise the severe collision should be observed, which means that they must collaborate in a certain degree.

### D. The Heterogeneous Controller

The heterogeneous controller indeed shows the diversity compared to the homogeneous controller. According to our observation, some of the predators can rotate in a fixed direction(e.g. clockwise) to search the prey, and move toward prey directly when they find the prey, another observed controller can follow the prey smoothly and rotate in different directions which depends on where the prey disappeared in the views of

predators. The various types of predators make they wouldn't just follow behind the prey as single homogeneous controller.

### E. Evolution Time

The processing of evolution takes a lot of computing time. There are two main reasons, the first one is because we need "Hall of Fame" to stabilize the process of coevolution, which makes our evolution n times longer than simple evolution, and n depends on how many previous generations to be competed with. Another reason is the time for simulating the camera and IR sensors to reduce the reality gap, it largely slows down the simulation speed of Gazebo. The whole evolution process needs 100 hours for 100 generations and population size as 20, which is around 300 times slower than using a homogeneous controller and a fully observable environment.

## VI. CONCLUSION

We presented a framework for evolutionary robotics in both simulation world and the real world by integrating Gym, Gazebo, and ROS. The interface provided by Gym allows users and researchers to focus on the learners, instead of wasting time on the setup with Gazebo in the simulation world or hardware in the real world. The framework can be used to investigate the tasks of cooperative coevolution and competitive evolution. Furthermore, reinforcement learning algorithms can be also applied as the leaner easily because our framework integrated the Gym package.

Our experiment shows that the standard coevolution framework can be also applied for sensor-based 3-versus-1 predator(s) and prey scenario. However, the transferability from the simulation to the real world can be limited by the hardware resource. The evolved robots still need to further evolve in the real world. The sampling for selecting opponents is the key step to stabilize the coevolution. At least the random sampling must be used.

The processing of camera and IR sensors in simulation consumes large computational resource and therefore the evolution process takes a lot of computing time. We note that the requirement of computation resources enlarges the reality gap, because usage of computation resource and FPS cannot be consistent.

Although the framework of evolutionary robot works well from simulation to the real world, there are still many open issues to be further studied. In the future, we aim to apply a more effective learner for the evolution that can be done in a shorter computing time, for instance, the state-of-the-art data efficient learners, Bayesian optimization and Bayesian-Evolutionary Algorithm [20]. This is always important issue for experiments in the real world. In addition, there are many other types of robots can be applied in this framework. We used the wheeled robots that is simple to be controlled but not so interesting. The tasks with other robots such as the modular robots with evolvable shapes [21], [22], could be significant interesting. Last, the semantic new technology, knowledge graphs [23], [24], can be applied for the human-computer interaction in future.

## REFERENCES

- [1] Y. Ho, A. Bryson, and S. Baron, "Differential games and optimal pursuit-evasion strategies," *IEEE Transactions on Automatic Control*, vol. 10, no. 4, pp. 385–389, 1965.
- [2] E. Raboin, U. Kuter, and D. Nau, "Generating strategies for multi-agent pursuit-evasion games in partially observable euclidean space," in *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 3*, 2012, pp. 1201–1202.
- [3] S. Nolfi and D. Floreano, "Coevolving predator and prey robots: Do "arms races" arise in artificial evolution?" *Artificial life*, vol. 4, no. 4, pp. 311–335, 1998.
- [4] C. H. Yong and R. Miikkulainen, "Coevolution of role-based cooperation in multiagent systems," *IEEE Transactions on Autonomous Mental Development*, vol. 1, no. 3, pp. 170–186, 2009.
- [5] T. D. Haynes and S. Sen, "Co-adaptation in a team," *International Journal of Computational Intelligence and Organizations*, vol. 1, no. 4, pp. 231–233, 1997.
- [6] N. Jakobi, P. Husbands, and I. Harvey, "Noise and the reality gap: The use of simulation in evolutionary robotics," in *European Conference on Artificial Life*. Springer, 1995, pp. 704–720.
- [7] B. D. Bryant and R. Miikkulainen, "Neuroevolution for adaptive teams," in *Evolutionary Computation, 2003. CEC'03. The 2003 Congress on*, vol. 3. IEEE, 2003, pp. 2194–2201.
- [8] G. Beni, "From swarm intelligence to swarm robotics," in *International Workshop on Swarm Robotics*. Springer, 2004, pp. 1–9.
- [9] I.-K. Jeong and J.-J. Lee, "Evolving fuzzy logic controllers for multiple mobile robots solving a continuous pursuit problem," in *Fuzzy Systems Conference Proceedings, 1999. FUZZ-IEEE'99. 1999 IEEE International*, vol. 2. IEEE, 1999, pp. 685–690.
- [10] S. Nolfi, "Co-evolving predator and prey robots," *Adaptive Behavior*, vol. 20, no. 1, pp. 10–15, 2012.
- [11] J. Paredis, "Coevolving cellular automata: Be aware of the red queen!," in *ICGA*, 1997, pp. 393–400.
- [12] L. M. Antonio and C. A. C. Coello, "Coevolutionary multiobjective evolutionary algorithms: Survey of the state-of-the-art," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 6, pp. 851–865, 2018.
- [13] G. Lan, J. Chen, and A. Eiben, "Simulated and real-world evolution of predator robots," in *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2019, pp. 1974–1981.
- [14] G. Lan, J. Benito-Picazo, D. M. Roijers, E. Domínguez, and A. E. Eiben, "Real-time robot vision on low-performance computing hardware," in *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, Nov 2018, pp. 1959–1965.
- [15] G. Lan, L. De Vries, and S. Wang, "Evolving efficient deep neural networks for real-time object recognition," in *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2019, pp. 2571–2578.
- [16] M. A. Potter and K. A. D. Jong, "Cooperative coevolution: An architecture for evolving coadapted subcomponents," *Evolutionary computation*, vol. 8, no. 1, pp. 1–29, 2000.
- [17] A. Rawal, P. Rajagopalan, and R. Miikkulainen, "Constructing competitive and cooperative agent behavior using coevolution," in *Proceedings of the 2010 IEEE conference on computational intelligence and games*. IEEE, 2010, pp. 107–114.
- [18] G. Lan, J. Chen, and A. Eiben, "Evolutionary predator-prey robot systems: From simulation to real world," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2019, pp. 123–124.
- [19] C. D. Rosin and R. K. Belew, "New methods for competitive coevolution," *Evolutionary computation*, vol. 5, no. 1, pp. 1–29, 1997.
- [20] G. Lan, J. M. Tomczak, D. M. Roijers, and A. Eiben, "Time efficiency in optimization with a bayesian-evolutionary algorithm," *arXiv preprint arXiv:2005.04166*, 2020.
- [21] G. Lan, M. van Hooft, M. D. Carlo, J. M. Tomczak, and A. E. Eiben, "Learning locomotion skills in evolvable robots," 2020.
- [22] G. Lan, M. De Carlo, F. van Diggelen, J. M. Tomczak, D. M. Roijers, and A. E. Eiben, "Learning directed locomotion in modular robots with evolvable morphologies," *arXiv preprint arXiv:2001.07804*, 2020.
- [23] T. Liu, X. Pan, X. Wang, K. A. Feenstra, J. Heringa, and Z. Huang, "Exploring the microbiota-gut-brain axis for mental disorders with knowledge graphs," *Journal of Artificial Intelligence for Medical Sciences*, vol. 1, pp. 23–41, 2020.
- [24] T. Liu, X. Pan, X. Wang, A. Feenstra, K. A. Feenstra, J. Heringa, and Z. Huang, "Predicting the relationships between gut microbiota and mental disorders with knowledge graphs," *Health Information Science and Systems*, vol. 8, pp. 1–9, 2020.