Cops and Thieves

Project for the Agent Systems course

Tomasz Kawiak, Mateusz Mazur

Faculty of Electrical Engineering, Automation, Computer Science and Biomedical Engineering, AGH
Field of study: Computer Science and Intelligent Systems
Specialization: Artificial Intelligence and Data Analysis

March 31, 2025

Introduction

Introduction

Cops And Thieves (*cops and robbers*) is a strategic pursuit-and-evasion game where two opposing agent types operate in a shared environment. Thieves aim to evade capture, while cops patrol, chase, and arrest thieves to maintain order. The game mechanics involve agent coordination, pathfinding, and adaptive decision-making.

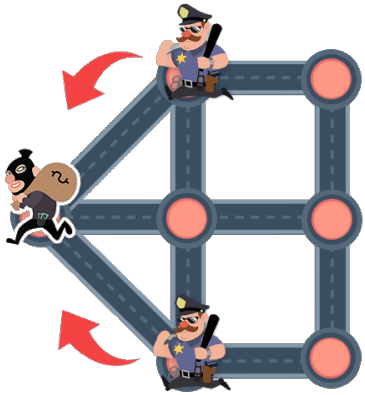

Figure 1: *Cops and Thieves* game depiction.
Source: *Catch The Thief: Help Police* by MicroEra

Motivation

Comparison of other approach for a problem considered previously on engineering studies course *Development Workshop*. Our project – *Chase model* – was also implementation of the cops and thieves game. This project aims to hopefully improve our earlier attempt.

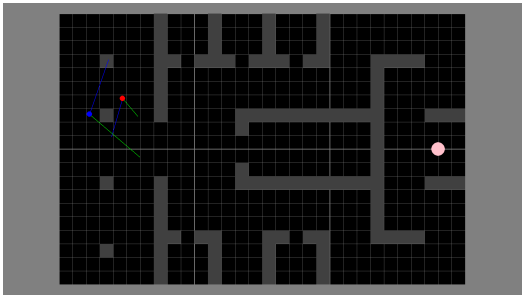

Figure 2: *Chase model* – game area.

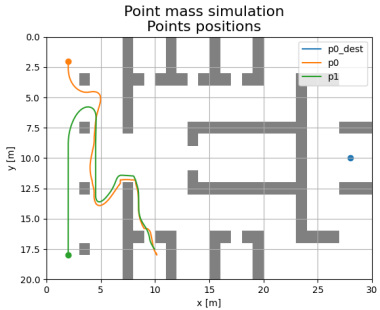

Figure 3: *Chase model* – movement chart.

Figure 4: We are back

As depicted in fig. 4, we are back to face the challenge of the cops and thieves problem, but this time with a different approach.

Goal

The main goal of this project is to train *environment-agnostic* agents:

- *Cops*: search and chase the thief.
- *Thief*: hide and try not to get caught.

A key aspect for both agents will be their ability to analyze their surroundings and act based on past observations. For our project we'll initially focus on only 2 agents (1 cop and 1 thief), with possibility of increasing the number of cop agents to observe cooperative behavior and more sophisticated search patterns.

We expect to achieve the following behaviors from agents:

Agent Type	Expected Description
Cop	Search and chase (if more cops, cooperation)
Thief	Evade capture and hide efficiently

Technological Stack

In our project we intend to use the following frameworks:

- skrl (fig. 5) for MARL implementation.
- PettingZoo (fig. 6) to guarantee MARL environment standards.
- pymunk (fig. 7) as 2D physics engine, complemented by:
  - pygame (fig. 8) for visualization.

Figure 5: skrl logo

Figure 6: PettingZoo logo

Figure 7: pymunk logo

Figure 8: pygame logo

1st progress update

Change log

Agents

As stated in our first presentation, we have 2 agent types, namely, cops and thieves. Because this part is critical for our project to move forward, we have implemented agent functionality, which covers:

- observation space implemented using vision controller (ray casting),
- action space (enabling movement of our agents in 2D space),
- initial reward function.

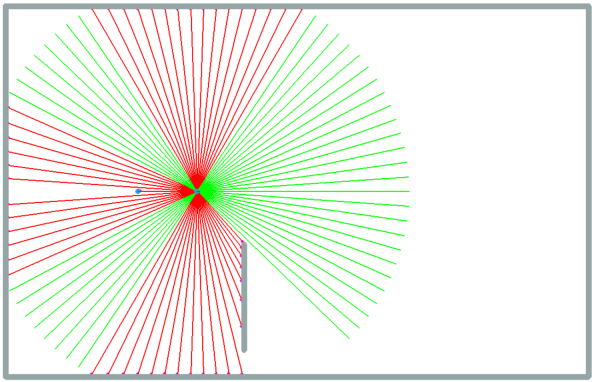

Figure 9: Visualization of Vision controller

Environment

Using *PettingZoo* library we have implemented initial environment, compliant with `Parallel API` in order for all agents to simultaneously perform action and observations. As part of this we have implemented:

- rendering together with map generation (explained in-depth later on).
- Basic methods needed for *Reinforcement Learning* algorithms (`reset`, `step`, etc.).
- Shared observation space:
  - needed for learning loop, where agents from the same category share a common observation that represents global information about the environment,
  - constructed based on their intrinsic priority (i.e. cops wanting to catch a thief), shared observation holds the most important information about their observations.

GUI & Visualization

Apart from a command line interface (CLI) "visualization" (suited for training the agents), we have also implemented a graphical user interface (GUI) that allows us to visualize the environment and the agents' actions. The GUI is presented in fig. 10.

Environment map generation

We have implemented a simple map generation tool that allows us to create a *map* file based real-world data from OSM. It generates obstacles for all the buildings in the area and places agents in given locations. Moreover, it generates a `png` file with a depiction of the area to be used as a background for the GUI. Figure 10 presents a screenshot of the GUI with a map of the AGH University of Science and Technology in Kraków, Poland.

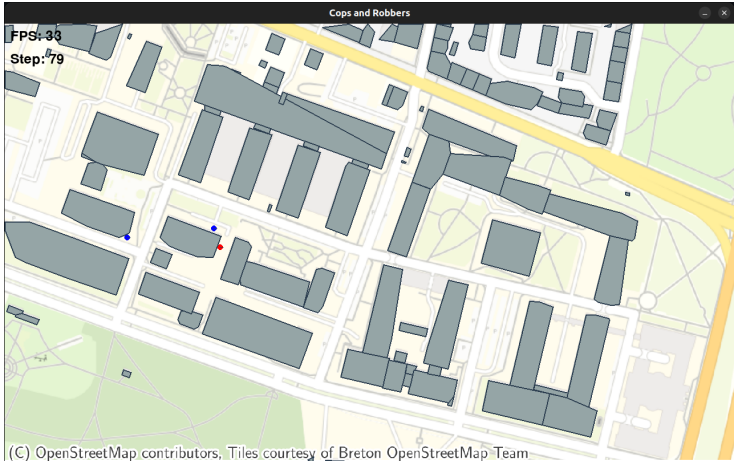

Figure 10: GUI of the application

Thank you for your attention

Questions?