Tecnológico de Monterrey

Modeling of Multi-Agent Systems with Computer Graphics (Gpo 301)

Evidence 2 - Final Deliverable
11-29-2024
Link to presentation video:
https://youtu.be/chXu2F6mdLg

Students:

Omar Michel Carmona Villalobos | A01644146
Roberto Anwar Teigeiro Aguilar | A01643651
Luis Omar Olmedo Ortiz | A01643557
Catalina Pesquet | A01763937
Andres Barrera | A01763911

Professors:

Iván Axel Dounce Nava
Carlos Johnnatan Sandoval Arrayga
Obed Nehemías Muñoz Reynoso

Guadalajara, Jalisco

November 2024

1. Problem Description

This project aims to address the complex and critical challenge of coordinated patrolling in high-risk and resource-sensitive environments, where maintaining security and operational integrity is paramount. By leveraging a multi-agent system, the simulation will integrate drones, fixed cameras, and security personnel to create a realistic and functional surveillance network. The system is designed to detect and respond to potential threats, such as unauthorized intrusions, suspicious behavior, or equipment misuse, ensuring rapid response and effective incident resolution. Through this simulation, participants will apply principles of artificial intelligence, inter-agent communication, and autonomous systems to design and implement a cohesive and efficient security framework. The project not only focuses on developing technical capabilities in programming and simulation but also aims to highlight the practical applications of such systems in enhancing safety and operational efficiency in real-world scenarios like factories, construction zones, and agricultural areas.

For our project, we chose to simulate a barn scenario involving a wolf (representing undesirable behavior), a drone, three cameras, and a security agent (such as a farmer). This setup provided a manageable yet meaningful environment to apply and test the concepts learned in class. The wolf introduces a dynamic threat, while the cameras detect its movements, the drone investigates further, and the security agent makes the final decisions. This choice allowed us to implement key elements such as agent communication, decision-making frameworks, and graphical representation in a cohesive and realistic way.

2. Agents Description

2.1 DronAgent

- **setup():** Initializes the drone's attributes, such as beliefs (beliefs), intention (intention), plan steps (plan_steps), and patrol mode (patrol_mode). Prepares the BDI (Belief-Desire-Intention) architecture with initial values.
- **See(Environment):** Detects a robber within a specific range (10 units). Returns the detected robber or None if no robber is within range.
- **Brf (Detected_robber):** Updates the drone's beliefs based on perceived information. If a robber is detected, the robber's position is stored in the belief system. Previous beliefs are cleared to prevent outdated information.
- **Options():** Generates desires (goals) based on current beliefs. If a robber is detected, the goal is to inspect its location.
- **Filter():**Selects the most relevant or desirable goal from the generated options. In this case, the goal is to inspect a robber.
- **Plan():** Creates a plan to reach the robber's position by generating step-by-step movements along the x and y axes from the drone's current location to the target.
- **Execute():** Executes the current plan by taking one step at a time. If the plan is completed, resets the current intention.
- **Takeoff():** Simulates the drone taking off and entering patrol mode.
- **Land():** Simulates the drone landing, exiting patrol mode after completing its tasks.
- **Step():** Implements the complete BDI cycle. It handles perception, belief revision, goal generation, planning, and execution for each simulation step. If no more tasks are available, the drone lands.

2.2 RobberAgent

- **Setup():** Initializes the robber without special attributes.

- **move_ranomly():** Generates a random movement (including staying in place) within the environment.
- **step():** Calls move_randomly() to define the robber's behavior in each simulation step.

2.3 CamaraAgent

- **setup():**Initializes the camera with a detection range larger than the drone's (20 units) and a counter for the number of alerts sent.
- **detect_robber():** Identifies if a robber is within the camera's detection range. Returns the detected robber or None if no robber is found.
- **send_alert(robber):** Sends an alert to the system with the robber's location. Increments the counter of sent alerts.
- **step():** Detects robbers within range and, if one is found, sends an alert to the system.

2.4 SecurityPersonnelAgent

- **setup():** Initializes the agent's attributes, such as whether it is actively communicating with a drone and whether the current alert has been resolved.
- **respond_to_drone_signal(drone, robber_position):** Simulates taking control of the drone after it signals for assistance. Initiates communication with the drone and proceeds to confirm the robber's presence.
- **confirm_robber(drone, robber_position):** Validates the existence of a robber at the specified location and escalates the situation by issuing a general alarm.
- **simulate_general_alarm(drone):** Issues a general alarm, indicating the resolution of the alert. Ends communication with the drone and updates the alert status.

- **step():** Represents the agent's behavior during each simulation step. Monitors for alerts from the drone and reacts by responding to signals for assistance. After handling an alert, remove it from the simulation environment.

2.5 Agent interactions

- Drones detect and track robbers, updating their beliefs and executing plans to inspect their locations.
- Cameras detect robbers and send alerts to the system.
- Security personnel respond to alerts, coordinate with drones, confirm threats, and issue general alarms if necessary.

3. Ontology

3.1 Core Classes

- **Entity (Thing):** Superclass for all agents and entities.
- **Drone (Entity):** Represents a patrolling surveillance drone.
- **Camera (Entity):** Represents a stationary surveillance camera.
- **Robber (Entity):** Represents a suspicious individual.
- **SecurityPersonnel (Entity):** Represents security personnel tasked with handling threats.
- **Place (Thing):** Represents specific areas where events occur.

3.2 Relationships (Object Properties and Data Properties)

- **is_in_place (ObjectProperty):** Links entities to specific places.
    - **Domain:** Entity
    - **Range:** Place

- **at_position (DataProperty, FunctionalProperty):** Specifies the position **of a place as a string.**
    - **Domain:** Place
    - **Range:** str

- **detects_suspicion (ObjectProperty):** Links cameras or drones to robbers they detect.
    - **Domain:** Camera, Drone
    - **Range:** Robber

- **alerts (ObjectProperty):** Links cameras to drones or security personnel when an alert is issued.
    - **Domain:** Camera
    - **Range:** Drone, SecurityPersonnel

- **patrolled_by (ObjectProperty):** Links a place to a drone patrolling it.
    - **Domain:** Place
    - **Range:** Drone

- **status (DataProperty, FunctionalProperty):** Indicates the current status of an entity as a string.
    - **Domain:** Entity
    - **Range:** str

3.3 DroneAgent Class

Represents the drone with a BDI (Belief-Desire-Intention) architecture.

3.3.1 Attributes

- **beliefs:** Holds current knowledge about the robber's position.
- **intention:** Current goal, such as verifying an alert.
- **plan_steps:** Steps the drone takes to achieve its goal.
- **patrol_mode:** Indicates if the drone is patrolling.
- **landing_position:** The fixed landing location for the drone.

3.4 CameraAgent Class

Represents stationary cameras detecting robbers within their range.

3.4.1 Attributes

- **detection_range:** The range within which the camera can detect robbers.
- **alerts_sent:** Counter for the number of alerts issued by the camera.

3.5 Robber Class

Represents a suspicious entity moving randomly within the grid.

3.5.1 Attributes

- **agent_type:** Unique identifier for visualization.

3.6 SecurityPersonnelAgent Class

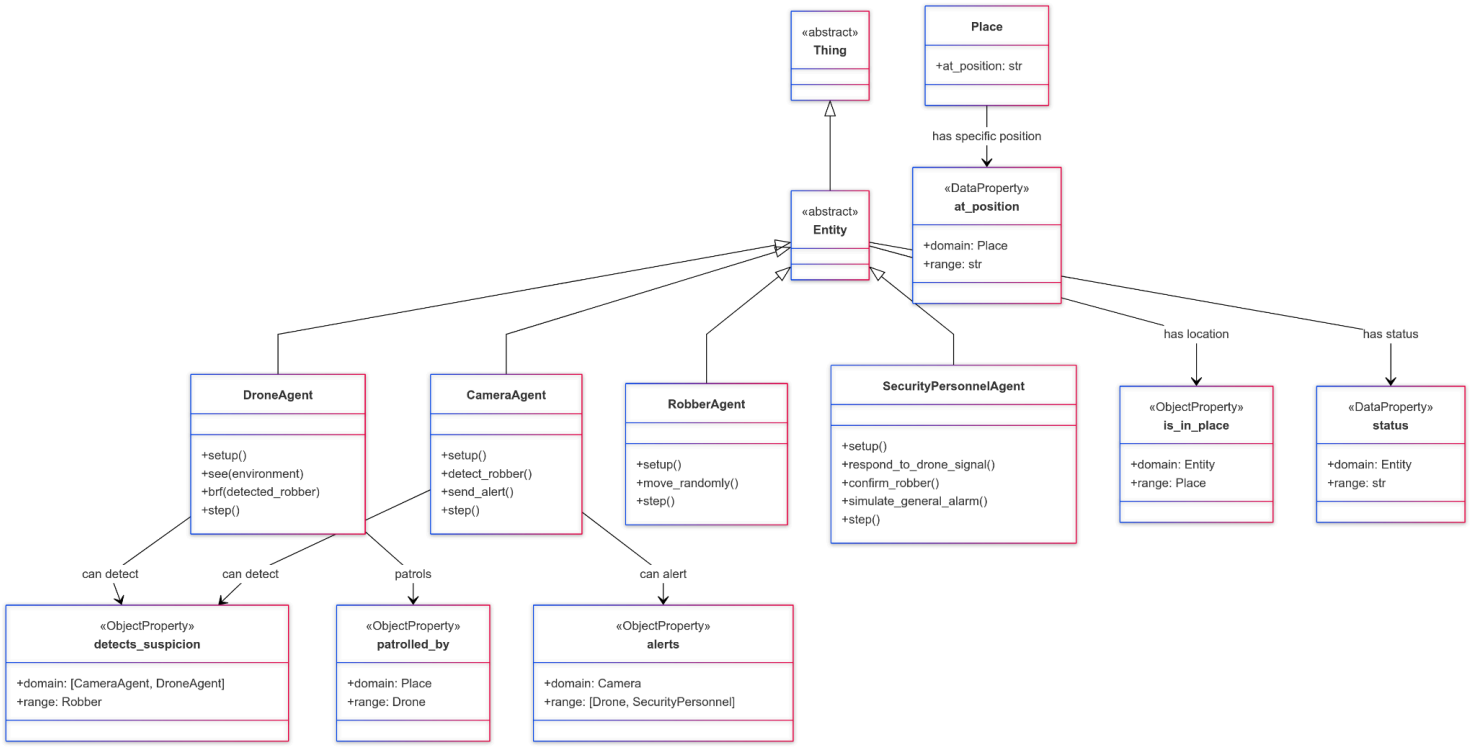
Represents security personnel managing alerts and drone control.

3.6.1 Attributes

- **in_communication:** Tracks active communication with the drone.
- **alert_handled:** Tracks if an alert has been addressed.

4. Agent classes and protocol diagram

4.1 Agent classes and ontology diagram



UML Class Diagram with Ontological Extensions

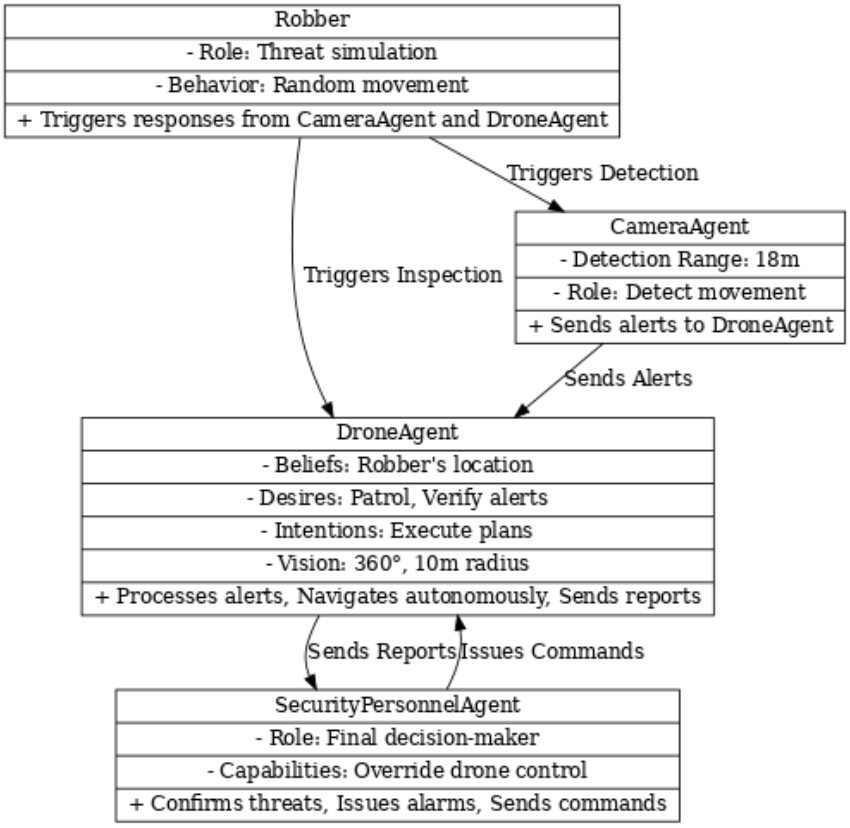
4.1.1 Diagram description:

The diagrams represents the class hierarchy and relationships within the security system, modeling the different agents and their interactions.

1. **Entity Hierarchy:**

- The diagram starts with the abstract Thing class, which is the parent class for all entities in the system.
- The Entity class is also abstract and serves as the base class for the specific agent classes.

2. Agent Classes:



Agent UML Diagram

- **Drone:** Represents the autonomous surveillance drone agent. It has methods for setup, perception (seeing the environment), belief revision, and execution of the main logic.
- **Camera:** Represents the stationary surveillance camera agent. It has methods for setup, detecting robbers, and sending alerts.
- **Robber:** Represents the suspicious individual (robber) agent. It has methods for setup and randomly moving within the environment.
- **SecurityPersonnel:** Represents the security personnel agent responsible for responding to the drone's alerts, confirming the robber's presence, and issuing a general alarm.

- **Place:** Represents a specific location or position in the environment, with the at_position data property storing the position.

3. **Inheritance Relationships:**

   All the specific agent classes (Drone, Camera, Robber, SecurityPersonnel) inherit from the abstract Entity class.

4. **Object Properties (Relationships):**
   - **is_in_place:** An object property that links an Entity (such as the Drone, Camera, Robber, or SecurityPersonnel) to a Place.
   - **detects_suspicion:** An object property that represents the ability of a Camera or Drone to detect a Robber.
   - **alerts:** An object property that models the ability of a Camera to alert a Drone or SecurityPersonnel.
   - **patrolled_by:** An object property that associates a Place with the Drone agent responsible for patrolling that area.

5. **Data Properties:**
   - **at_position:** A data property that stores the specific position of a Place.
   - **status:** A data property that represents the status of an Entity.

6. **Relationships:**
   - The diagram shows the relationships between the different classes, indicating the capabilities and interactions between the agents.

- For example, the Drone and Camera can detects_suspicion of a Robber, the Camera can alert the Drone or SecurityPersonnel, and the Drone patrols the Place.
- The Entity has a is_in_place relationship with the Place, and the Place has an at_position data property.

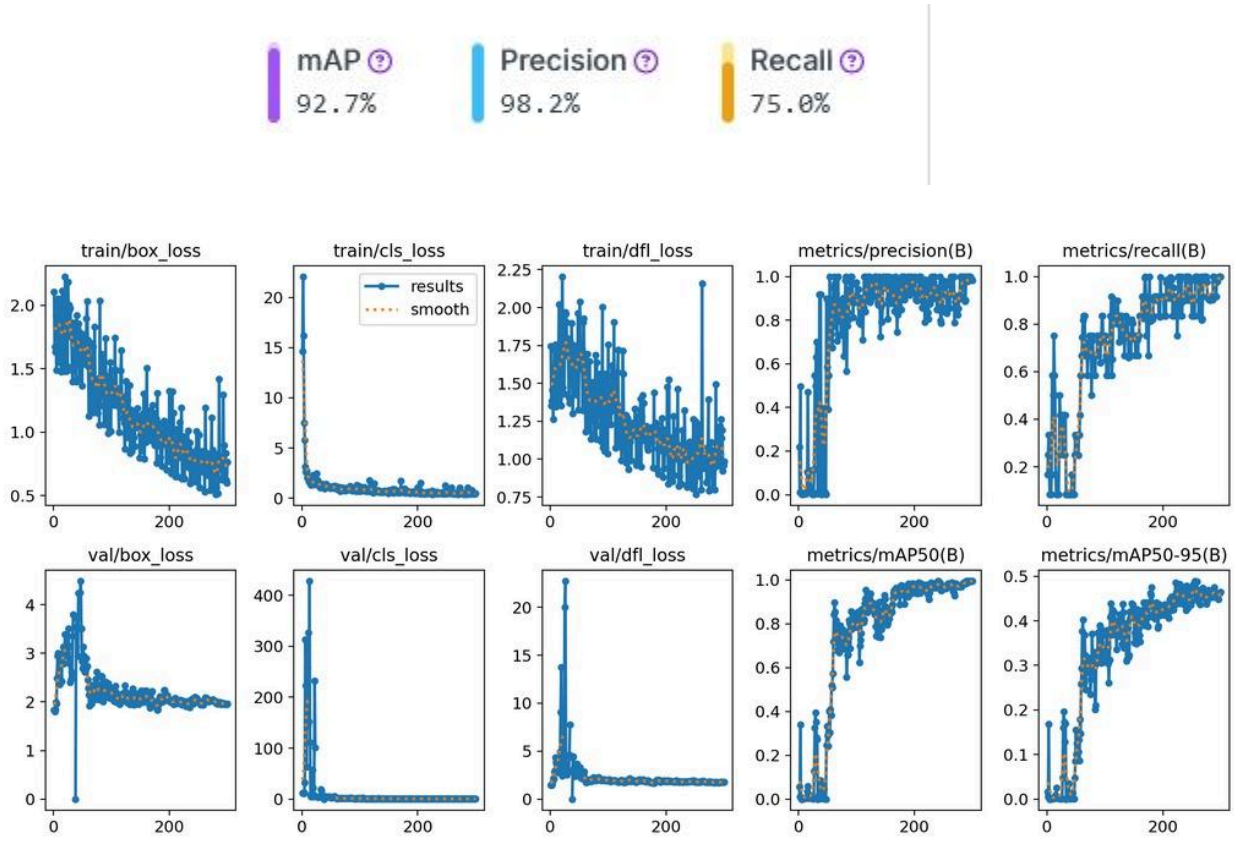
The diagram provides an overview of the class hierarchy, properties, and relationships within the security system, serving as a valuable reference for understanding the structure and interactions between the different agents that were created.

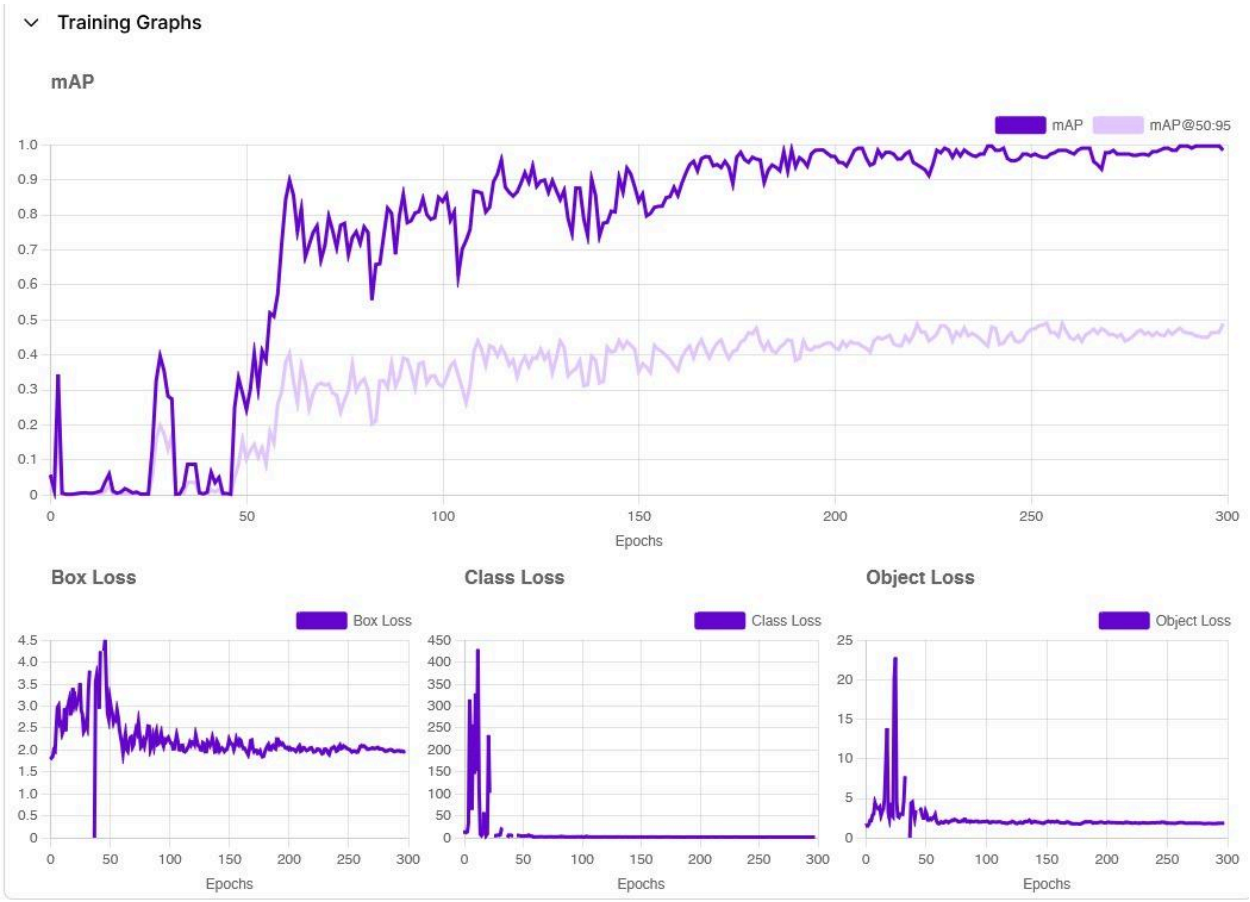
5. Utility and success of agents

5.1 First metric: Precision in the detection of anomalies in YOLO

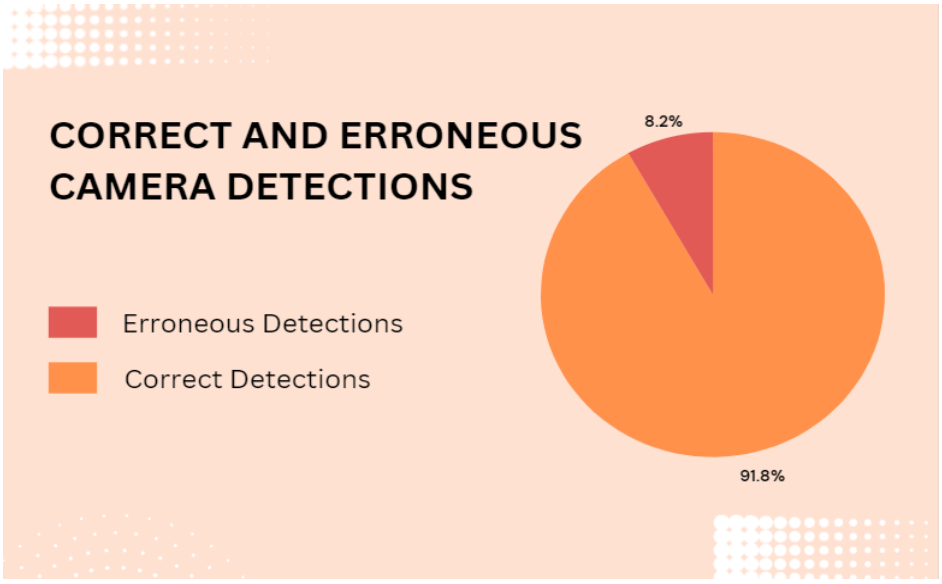
For this first metric, we measured the effectivity that it was specifically for the cameras at the moment that they can detect the anomalies (wolf) in the environment using the detection tool YOLO.

In the first test, we got an effectivity of 98.20 %, as is shown in the next image:





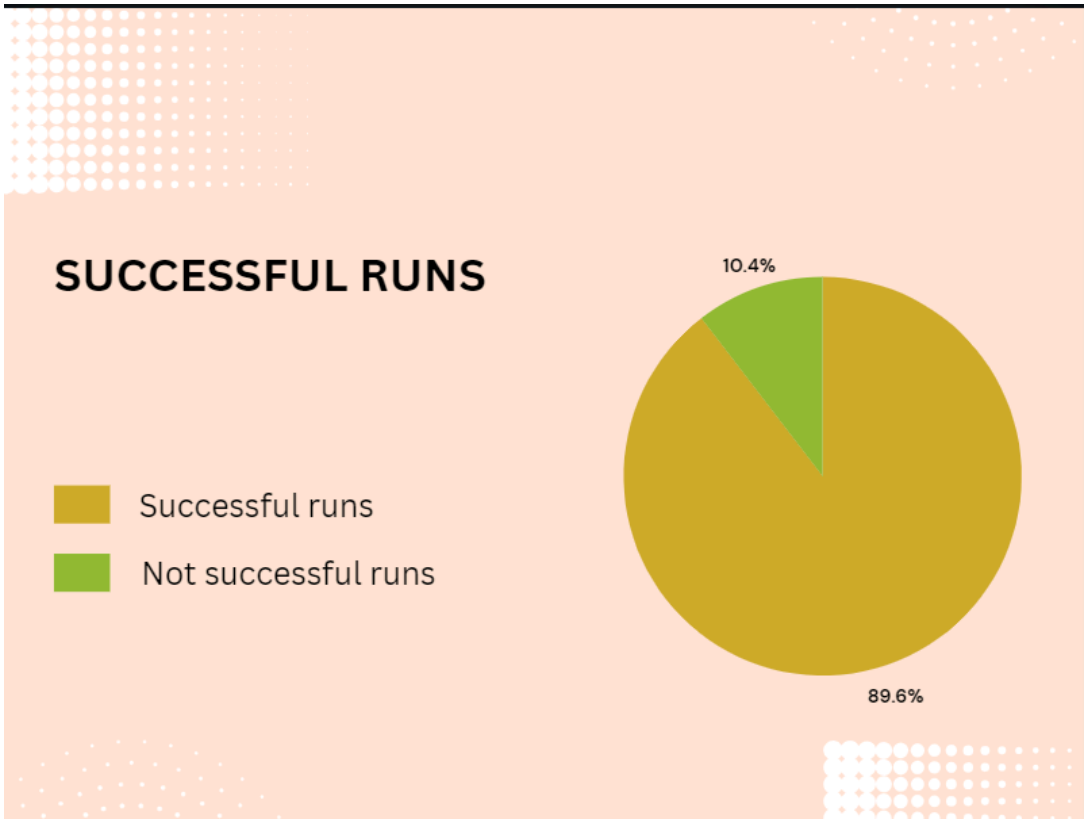
However, after doing another 9 tests with the anomalies being in different positions inside the environment we got a new average of effectiveness of 91.80 %, which is shown in the next image:



5.2 Second metric: Success rate in simulated runs

The second metric that we used to measure the success rate of our agents was through simulating the scenario 50 times. In each of the runs, our agents had the task of detecting and reacting to every anomalie in the environment. We considered a successful rate when all of the agents are able to identify and stop the anomalies avoiding any risk situation.

By doing 50 runs of our project we were able to achieve a successful rate of 89.6% and a non successful rate of the remaining 10.4%. This outcome let us evaluate the capacity of the system to work in a efficient way in different scenarios and assure that the agents were able to protect the environment through different conditions.



6. Installation, configuration and execution

6.1 Process of installation: The first step is to download the Unity Hub App from this link: https://unity.com/es/unity-hub

Then, to create the project we had to install one of the Unity most recent versions, the 6000.0.25f1 in case you don't have it. Open the project and you are good to go.

6.2 Configuration: No configuration was required for this project.

7. Analysis and Reflections

7.1. Luis Olmedo:

In this project, we explored how multiple agents—drones, cameras, and security personnel—can collaborate to enhance security in high-risk environments. Using the AgentPy library, we modeled the agents' behaviors and interactions, while Unity provided the graphical simulation of the environment. A Flask server facilitated communication between the agents, ensuring smooth data exchange and control. Additionally, we trained a YOLO model for real-time object detection, enabling the drone and cameras to identify suspicious activity accurately.

One of the biggest challenges was ensuring seamless coordination between the drone, cameras, and human operators. We also had to ensure the drone could

autonomously navigate and respond effectively to alerts, even when faced with environmental factors like wind. These challenges required integrating various technologies and refining our approach through iterative testing.

This project not only enhanced our technical skills in multi-agent systems, computer vision, and simulation design but also demonstrated the importance of using different tools in harmony. The combination of AgentPy, Unity, Flask, and YOLO allowed us to create a robust system that shows how advanced technologies can work together to improve safety in real-world scenarios.

Why did you select the multi-agent model used?

The multi-agent model was selected because it combines autonomous behavior, semantic reasoning, and human-like decision-making in a way that closely mirrors real-world surveillance systems. It allows for adaptive, decentralized problem solving, making it highly suitable for exploring and optimizing coordinated patrol and monitoring strategies.

What were the variables that were taken into account at the time of making the decision?

When deciding to pursue the robber, key variables shaped the drone's actions. Its belief system is continuously updated based on its detection range, ensuring awareness of potential threats. The robber’s position and distance became central, driving the drone’s desire to pursue. The decision also depended on the drone’s patrol mode, ensuring readiness, while its path was carefully planned in discrete steps. These variables highlighted a balance between awareness and precision, enabling the drone to act purposefully and strategically.

What is the interaction of these variables with respect to the simulation result?

The interaction of these variables, belief system, detection range, robber's position, patrol mode, and discrete step ensured dynamic adaptability in the simulation. Together, they allowed the drone to efficiently locate, pursue, and track the robber, leading to a more realistic and strategic outcome where both awareness and pursuit were balanced for optimal performance.

Why did you select the graphical design presented?

The graphical design was chosen to enhance clarity and user engagement while presenting complex information. It was designed to be intuitive, with clear visual cues that help users quickly understand agent behavior and interactions in the simulation. The graphical design was chosen to enhance clarity and user engagement while presenting complex information. It was designed to be intuitive, with clear visual cues that help users quickly understand agent behavior and interactions in the simulation.

What are the advantages you find in the final solution presented?

The final solution offers numerous advantages, especially with the integration of the BDI (Belief-Desire-Intention) model for the drone. This model enables the drone to operate autonomously, as it can form beliefs about its environment, set goals based on those beliefs, and take actions to achieve them. For example, the drone can decide whether to investigate a potential threat or return to its station based on the situation at hand. This dynamic decision-making process enhances the realism and adaptability of the drone's behavior, allowing it to respond intelligently to ever-changing conditions.

What are the disadvantages that exist in the solution presented?

While the solution has many advantages, it also has some disadvantages. The BDI model, though powerful, can be complex to design and may demand high computational resources, potentially affecting performance. Additionally, the absence of environmental physics limits the realism of the simulation, as factors like wind or obstacles aren't considered. Lastly, the drone's autonomy, while enhanced, might still struggle in highly dynamic or unexpected scenarios without human intervention.

What modifications could you make to reduce or eliminate the disadvantages mentioned?

To address the disadvantages, we could simplify the BDI model to improve performance and reduce computational load. Adding environmental physics, such as wind or terrain, would enhance realism and dynamic behavior. Additionally, incorporating machine learning or advanced sensors would improve the drone's adaptability to unforeseen situations, making it more resilient. Lastly, optimizing resource allocation would help balance performance, especially in complex simulations.

7.2. Michel Carmona:

Why did you select the multi-agent model used?

We selected this model because it was the one that help us to implement the communication between all the agents of the system in a simplest way. The agent model that we selected are able to manage the interactions when an anomaly is detected.

What were the variables that were taken into account at the time of making the decision?

Some of the variables that we are taking into account are the efficiency of them at doing their respective tasks. Also their roles that each of the agents have they cover that goes along with our project.

What is the interaction of these variables with respect to the simulation result?

-Detection Capabilities: The agents' capacity to detect threats was evaluated based on precision and range, aided by computational vision tools like YOLO. The simulation operates on a fixed 100x100 grid for consistency.

-Communication Mechanisms: Collaboration among agents, such as drones reacting to alerts from cameras, was essential for efficient coordination.

-Agent Roles: Each agent had distinct tasks, including monitoring, decision-making, and issuing alerts.

Why did you select the graphical design presented?

The graphical design we choose to replicate a real world scenario is a farm, where the wolf, that is our intruder, can harm all of the animals from the farms and also the field. We decorate the scenario like a real big farm.

What are the advantages you find in the final solution presented?

That the outcome presented has a really good successful rate on average, also that each of the agents presented they accomplished their roles as requested. Son in general, the result that we had in our project was really good.

What are the disadvantages that exist in the solution presented?

Currently, the communication between the backend and Unity is slow, resulting in lower frame rates and the need to introduce artificial delays to manage performance issues.

What modifications could you make to reduce or eliminate the disadvantages mentioned?

Maybe using third party tools so it can be connected in a better way.

7.3. Roberto Teigeiro:

Why did you select the multi-agent model used?

The multi-agent model was chosen because it mirrors real-world scenarios involving autonomous and interdependent entities, such as drones, cameras, and personnel. It allows for distributed decision-making and coordination, enabling effective surveillance in a dynamic and complex environment. The model's modular nature made it an excellent choice for exploring and optimizing strategies for patrolling and threat detection.

What were the variables that were taken into account at the time of making the decision?

Detection Capabilities: The agents' ability to detect threats based on range and precision, so computational vision, the size of the place (in this case it’s a constant 100x100 simulated grid), the

Communication Mechanisms: Coordination between agents, such as drones responding to alerts from cameras.

Agent Roles: Specific tasks like tracking, decision-making, and issuing alerts.

Scalability and Modularity: Ensuring the system could be expanded or modified for more complex scenarios.

Computational Efficiency: Balancing model complexity with performance constraints, we have also added that.

What is the interaction of these variables with respect to the simulation result?

Detection capabilities determined how effectively threats were identified, this using yolo model when training it will return us some metrics on efectiveness

Why did you select the graphical design presented?

the model was chosen to mimic a real world scenario where someone would like to implement this into a real scenario that works like this, it would need some fine tuning

What are the advantages you find in the final solution presented?

from the BDI model allowed for autonomous, context-driven decisions by the drone, so we can leave the agents to do the work by themselves, so we think that this is a very efficient way to do this

What are the disadvantages that exist in the solution presented?

the backend-unity communication is very slow at the moment, and we can't process all of it in real time with much frames per second, so we added some delays to them.

What modifications could you make to reduce or eliminate the disadvantages mentioned?

change how it communicates with it, maybe do it via websockets instead of an api. find some other approach on how to communicate with unity and back.

7.4. Catalina Pesquet:

To begin with, this project allowed me to explore the graphical aspect of projects (in this case, Unity) and drastically improve my programming skills and my ability to visualize agents. I have also previously studied topics related to reinforcement

learning, which often relies on the MAS (Multi-Agent System) approach. I had experience in 3D modelling in the context of designing complex mechanical objects such as reductors but I never had the occasion of linking my objects to any code or any behavior as we did in this class. Thus, regarding the theoretical aspect and agent coding, I was able to discover the AgentPy library, which I had never used before and which I am certain will be very useful to me in the future. It was also very interesting because the project is based on the real requirements of a company and demands more than just theoretical knowledge—it requires a more comprehensive implementation.

Why did you select the multi-agent model used?

The choice of a multi-agent model for this project is justified by the need to coordinate multiple autonomous entities (drone, cameras, personnel) that must work together. The document demonstrates that the system requires constant communication and distributed decision-making, where each agent has specific but interdependent roles. The multi-agent model enables handling this complexity in a modular and scalable manner.This model serves as a well-balanced starting point, considering the time constraints we had. We deliberately chose a setup with three cameras, a robber, and a security agent to ensure we could develop both the programming and graphical components in parallel. This approach allowed us to fully apply the concepts from both parts of the course. While the model is intentionally straightforward, it lays a solid foundation for future enhancements, such as adding bystanders with potentially ambiguous behaviors or enabling the drone to adjust its speed dynamically based on proximity to the target. This choice allowed us to deliver a complete and polished project within the given timeframe.

What were the variables that were taken into account at the time of making the decision?

The decision-making process took into account several variables. First, the need to balance the development of both the programming and graphical components was a key consideration. This ensured that we could effectively apply the theoretical and practical concepts covered in the course, such as BDI (Belief-Desire-Intention) models, ontologies for knowledge representation, and agent communication through messages.

Additionally, we aimed to design a manageable yet meaningful model that allowed us to explore these themes in a real-world-inspired context. The choice of including three cameras, a robber, and a security agent provided a functional scenario to implement and test these concepts, while also leaving room for graphical development. This approach enabled us to integrate these theoretical frameworks into a cohesive and practical implementation.

What is the interaction of these variables with respect to the simulation result?

The interaction of these variables shaped the simulation's effectiveness:the BDI enabled realistic decision-making, with agents adapting their behavior dynamically. The ontologies allowed structured knowledge representation and ensured clear communication and coordination between agents. Together, these variables created a cohesive, functional simulation that effectively demonstrated the interplay of autonomy, coordination, and responsiveness.

Why did you select the graphical design presented?

We selected the presented graphical design to reflect a real-world problem scenario in a visually clear and intuitive manner. The barn setting, featuring a wolf (an undesirable behavior), a drone, cameras, and a security agent, represents a simplified yet realistic model of modern surveillance challenges. Similar situations arise in agriculture, security, and industrial settings, where multiple entities must coordinate to detect and respond to potential threats effectively.

several advantages, particularly the implementation of a BDI (Belief-Desire-Intention) model for the drone. This approach allows the drone to make autonomous decisions based on its understanding of the environment, such as investigating detected threats or returning to its station. The BDI framework enhances the realism and adaptability of the drone's behavior

What are the advantages you find in the final solution presented?

The final solution has several advantages, particularly the implementation of a BDI (Belief-Desire-Intention) model for the drone. This approach allows the drone to make autonomous decisions based on its understanding of the environment, such as investigating detected threats or returning to its station. The BDI framework enhances the realism and adaptability of the drone's behavior, making it more effective in dynamic scenarios.

What are the disadvantages that exist in the solution presented?

The solution lacks realism, as it does not include environmental physics like wind, which could affect the drone's movement. Similarly, the wolf's behavior could be enhanced by adding slower movement zones, such as mud or slippery surfaces, to simulate environmental challenges. These additions would make the simulation more realistic and robust.

What modifications could you make to reduce or eliminate the disadvantages mentioned?

7.5. Andrés Barrera

- **Why did you select the multi-agent model used?**

The multi-agent model was chosen because it simulates the interaction of multiple autonomous agents within a shared environment, allowing for a realistic and complex simulation of the surveillance process. Each agent operates independently and makes decisions based on its own behavior

- **What were the variables that were taken into account at the time of making the decision?**

Agent behavior: How each type of agent (drone, camera, wolf, security) behaves within the simulation. For instance, the drone detects robbers, the camera sends alerts, and the security personnel confirm the robber's presence.
Environmental factors: The agents exist on a grid where they must interact with their surroundings. The grid size, agent positions, and movement patterns affect agent interaction and decision-making.
Agent communication: How agents communicate with each other. For example, drones send alerts, and security responds to those alerts.

- **What is the interaction of these variables with respect to the simulation result?**

The interaction between these variables directly influences the outcomes of the simulation. The grid layout allows agents to detect and respond to changes in their environment, while the agents' behaviors are modeled to reflect real-world scenarios. For example, when a drone detects a wolf, it triggers an alert that activates the response of security personnel. This interaction chain illustrates the decision-making processes within the system, showing how one agent's action leads to a series of reactions in others. The interconnectedness of these variables is essential to creating an accurate simulation where agents work together or against each other to reach different goals, making the system dynamic and complex. The more variables and behaviors included, the more realistic the simulation becomes, demonstrating how environmental changes affect the agents' decisions and actions.

- **Why did you select the graphical design presented?**

The graphical design was chosen to clearly represent the simulation of a multi-agent system within an agricultural zone. Given that the environment includes factors such as crops, fencing, and wildlife, visualizing this on a grid allows for easy tracking of agent movements, such as the wolf and the security agents. This layout provides clarity by distinguishing between different types of agents and objects in the environment. For example, the wolf is shown as a specific type of agent with a distinct icon, while the crops, fences, and other environmental features are represented to emphasize the agricultural context. The grid design also makes it easier to simulate interactions. Furthermore, this design is adaptable and can be modified to show more complex behaviors or larger landscapes if needed. It helps keep the simulation understandable while also providing a straightforward way to modify and test the agents' actions within this specific context.

- **What are the advantages you find in the final solution presented?**

One of the main advantages of the final solution is the ease with which it models and visualizes the interactions between agents in an agricultural setting. The grid-based environment helps ensure that the agents' actions, such as detecting the wolf or guarding specific areas of the farm, can be tracked and understood clearly. The separation of the agents by type, such as drones, farm workers, and wolves, helps clarify their roles and allows us to see the specific interactions between these agents in the agricultural zone. This also simplifies the debugging and optimization process, as we can identify which agent's behavior needs adjustment. Moreover, the solution is highly adaptable, meaning that new behaviors, agents, or elements can be added easily to the environment. For instance, more wolves or different types of farm workers with varying tasks could be introduced to further test the system's response. The visual aspect of the design allows the results to be presented in a way that is both intuitive for users and useful for analyzing agent behaviors in this unique environment.

- **What are the disadvantages that exist in the solution presented?**

There are a few potential disadvantages:
The agents, while autonomous, follow fairly simple behaviors like wolves moving randomly. This simplification might limit the realism of the simulation, especially in more complex scenarios. Also limited environmental interaction, cause the agents' environment is static, and there’s no advanced environmental interaction (like obstacles or dynamic changes) which could make the simulation more complex and realistic.

- **What modifications could you make to reduce or eliminate the disadvantages mentioned?**

Enhance agent behavior: Introduce more complex behaviors for wolfs like strategic movement based on environmental factors or previous alerts, or add additional decision-making algorithms for the agents.
Improve error handling and communication: Implement failure scenarios for communication, such as loss of signal or delayed responses, which would add complexity and realism.
Dynamic environment: Introduce elements like obstacles, changing environmental conditions, or even weather that could affect agent movement or detection.

- **My Reflection on Learning**

Throughout this project, I took on the responsibility of designing and creating the graphical elements because this is my area of expertise. I felt confident in handling the visual aspects since I've had experience with design and graphics, and I wanted to ensure that the simulation was both functional and aesthetically clear. While I focused on the visual representation, my teammates handled most of the coding, which allowed us to divide the tasks based on our strengths. However, looking back, I realize that I didn't gain much from the class itself. The concepts and skills we were supposed to learn didn't feel very connected to the project, and I ended up figuring out a lot of the design elements and the simulation setup on my own, outside of class. In fact, much of the learning was self-driven, as I had to rely on resources like online tutorials and trial-and-error to get things right. This

independent learning process was valuable, but a bit frustrating. Despite this, I'm proud of the work I did on the project, particularly with the graphical design, as it allowed me to use my skills in a meaningful way and contributed to a well-rounded final product.