**Instituto Tecnológico y de Estudios Superiores de Monterrey**

Modeling of Multi-Agent Systems with Computer Graphics (101)

# Evidence 2. Progress and presentation of the challenge. Review 4

**Professor:**

Iván Axel Dounce Nava

| | |
|---|---|
| Karen Valentina Mariel Villagrán | A01541014 |
| Guillermo Baltazar y Nungaray | A01621090 |
| Luciano Luna García | A01645870 |
| Daniel Eden Wynter González | A01645202 |
| Tarik Maina Gichuki | A01830431 |
| Mariano Sánchez Bermúdez | A01639413 |

Campus Guadalajara

## Challenge description

The mission consists of designing a multi-agent system that enables an autonomous UFO to locate, identify, and abduct specific cows in mountainous terrain.

Our project implements a UFO that patrols the terrain and detects cows using computer vision.
When a cow is identified, the UFO navigates to its position and descends above it.
This system combines autonomous movement with external AI detection.The mission consists of designing a multi-agent system that enables an autonomous UFO to locate, identify, and abduct specific cows in a mountainous terrain.

## Solution description

1. Receives and processes instructions through the FastAPI server, which loads the trained YOLOv8 model.
2. Navigates around the area using a sweeping pattern to explore the entire environment.
3. Identifies cows with probabilistic recognition based on the YOLOv8 trained dataset.
4. Follows the detected target and positions itself directly above the cow, adjusting alignment.
5. Communicates the detection results (label, bounding box, and confidence score) back to the server for further analysis.

The implementation of the system began with the installation process, carried out from scratch in a clean virtual environment. A Python virtual environment was created and activated to isolate dependencies and avoid conflicts with other projects. Once inside the environment, pip was upgraded to the latest version and the necessary libraries were installed. These included the Ultralytics package, which contains YOLOv8, as well as FastAPI, Uvicorn, OpenCV, and NumPy. After installation, the YOLOv8 version was verified to ensure that the environment was properly configured.

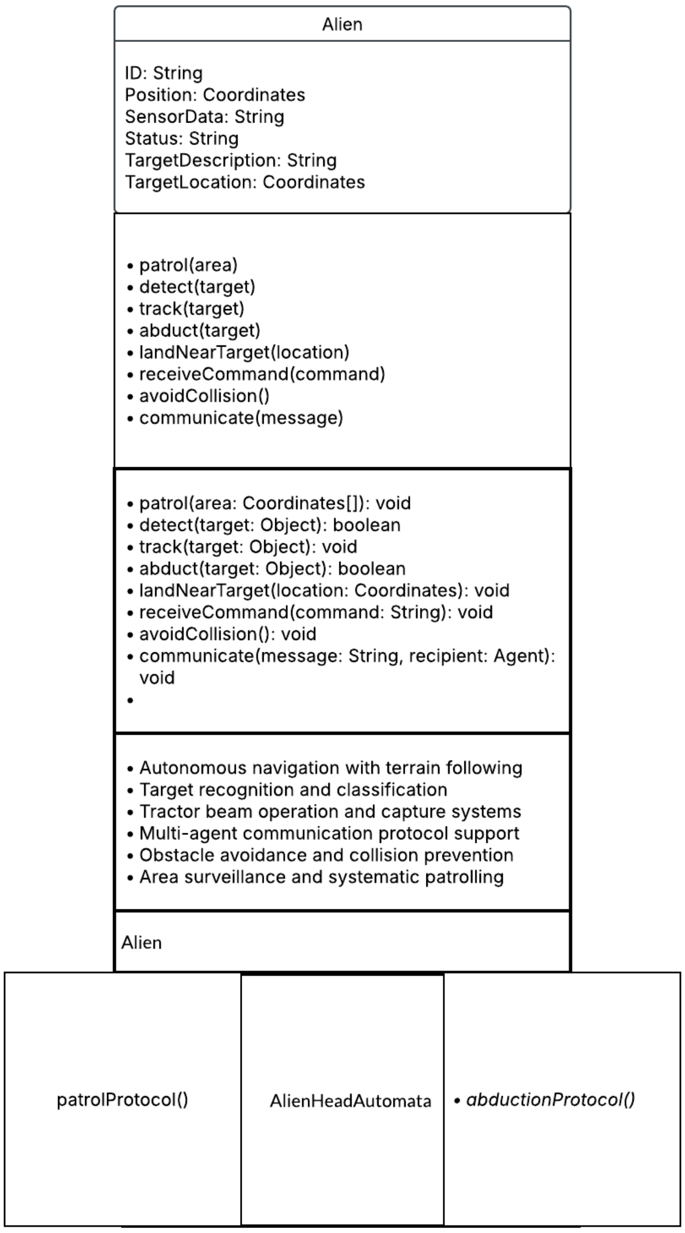
The training stage required preparing a dataset specifically designed for cow detection. Images were collected and carefully labeled with bounding boxes using a tool such as LabelImg. The dataset was organized following the standard YOLO format, with separate folders for training and validation images and labels. A configuration file, data.yaml, was created to specify the dataset paths and the list of classes, which in this case only included the class “cow.” Training was then carried out using transfer learning with a pretrained YOLOv8n model. Parameters such as the number of epochs and image size were adjusted to balance performance and accuracy. At the end of the training process, the best-performing weights were automatically saved in the directory runs/detect/train/weights under the name best.pt.

Once the model was trained, it was integrated into a FastAPI backend to provide detection as a service. The server script, named yoloe_server.py, was configured to load the trained weights, receive images through HTTP POST requests, and return detection results in JSON format. The only required adjustment to the configuration was ensuring that the absolute path of the weights file was correctly specified in the script.

Execution of the system was carried out by launching the FastAPI server through Uvicorn, which made the detection service available at localhost on port 5000. When running, the server accepted image inputs and returned the predicted bounding boxes, confidence scores, and labels. For example, uploading an image containing a cow resulted in a JSON response that identified the detected object as “cow” along with its position in the frame.

In summary, the process consisted of installing the necessary libraries in a controlled environment, training a YOLOv8 model on a custom dataset, configuring a backend to host the model, and executing the server to provide real-time inference. This workflow ensured reproducibility, scalability, and practical usability of the developed solution.

## Final UML agent class diagrams and interaction protocols.



Alien

ID: String
Position: Coordinates
SensorData: String
Status: String
TargetDescription: String
TargetLocation: Coordinates

- patrol(area)
- detect(target)
- track(target)
- abduct(target)
- landNearTarget(location)
- receiveCommand(command)
- avoidCollision()
- communicate(message)

- patrol(area: Coordinates[]): void
- detect(target: Object): boolean
- track(target: Object): void
- abduct(target: Object): boolean
- landNearTarget(location: Coordinates): void
- receiveCommand(command: String): void
- avoidCollision(): void
- communicate(message: String, recipient: Agent): void
-

- Autonomous navigation with terrain following
- Target recognition and classification
- Tractor beam operation and capture systems
- Multi-agent communication protocol support
- Obstacle avoidance and collision prevention
- Area surveillance and systematic patrolling

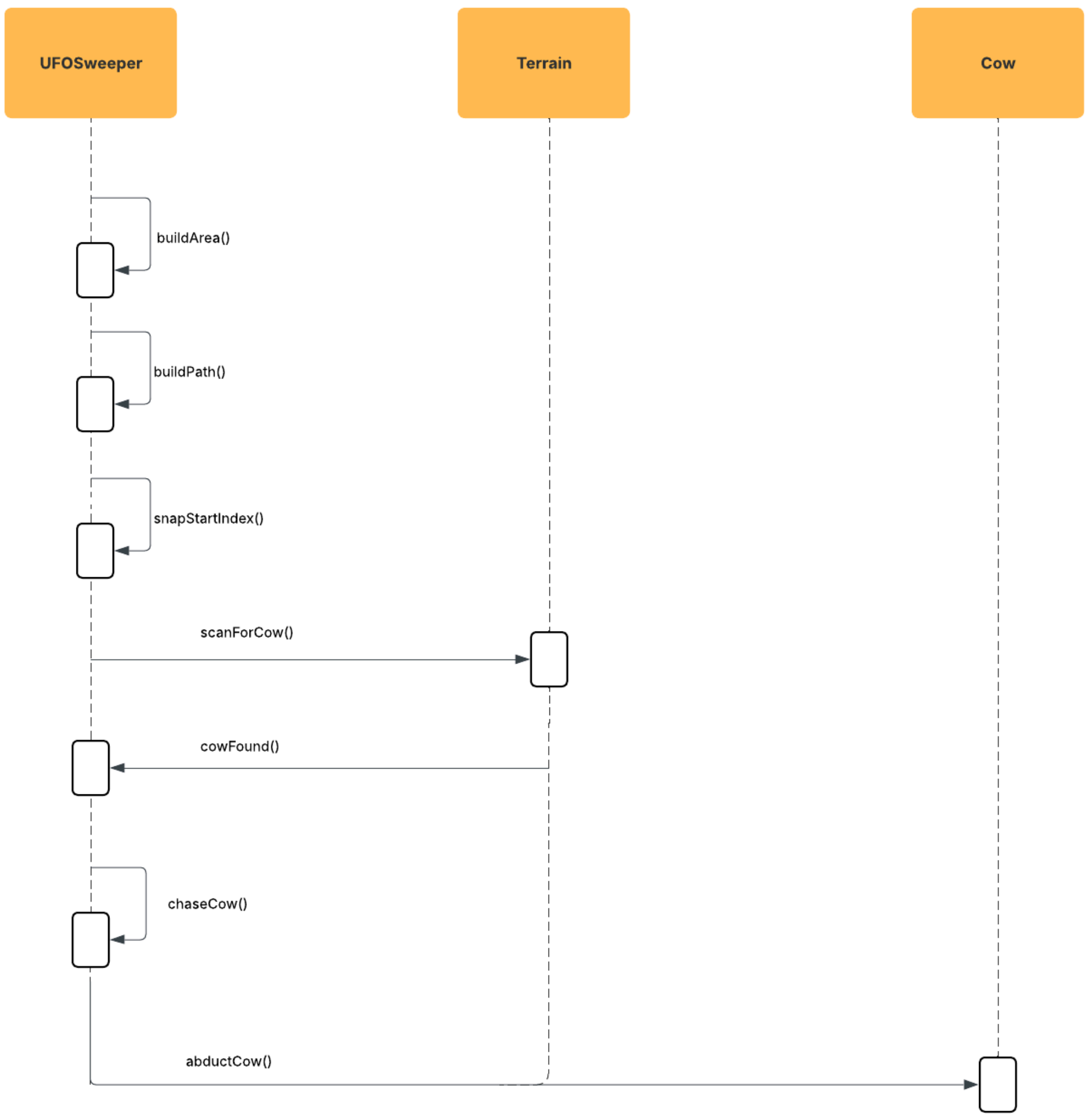
Alien

patrolProtocol()

AlienHeadAutomata

- *abductionProtocol()*

**UML Diagrams to explain the simulation and its elements.**



**Installation process**

Unity package installation:

To import the unitypackage file you only need to create a new 3d universal project, go to assets and import the package from your computer.

Code in the terminal for the installation of the YOLOV8:

```
python -m venv venv

venv\Scripts\activate        # (Windows)
```

```
source venv/bin/activate     # (Linux/Mac)

pip install --upgrade pip

pip install ultralytics

pip install fastapi uvicorn opencv-python-headless numpy

yolo detect train data=data.yaml model=yolov8n.pt epochs=60 imgsz=640

uvicorn yoloe_server:app --reload --host 127.0.0.1 --port 5000
```

## Individual documentation

**Daniel Eden Wynter González**

1. Analysis of the Developed Solution

I selected the multi-agent model because the problem required autonomous entities that could act independently while coordinating to achieve a shared goal. The main variables considered were autonomy, coordination, scalability, environment complexity, and robustness. Their interaction allowed the system to remain flexible, resilient, and efficient.
The chosen system architecture ensured modularity and effective communication between agents. Its main advantages were parallelism, adaptability, scalability, and fault tolerance. However, disadvantages included communication overhead, higher implementation complexity, and coordination challenges. These could be reduced through optimized communication protocols, better coordination rules, and monitoring tools.

2. Reflection on My Learning Process

At the beginning, I expected to only learn the basics of agent programming. Instead, I gained a broader understanding of how to design and analyze multi-agent systems, including communication and coordination strategies. I also learned to evaluate trade-offs between centralized and decentralized approaches. Overall, the experience went beyond my expectations and helped me develop both technical and analytical skills.

## Luciano Luna García

1. Analysis of the Developed Solution

I chose the multi-agent model because the task required independent entities that could still collaborate toward a shared mission. The most relevant factors were autonomy, scalability, environmental complexity, and coordination. This allowed the system to remain flexible and adaptable under different conditions. The modular architecture supported clear communication between agents, making the system fault tolerant and easier to scale. However, drawbacks such as increased coordination complexity and communication overhead were present. These can be mitigated with more efficient protocols and better synchronization mechanisms.

2. Reflection on My Learning Process

Initially, I expected to focus mainly on developing the interface and graphical components. Over time, I realized the importance of designing systems that balance communication, modularity, and decision-making. I gained a deeper understanding of multi-agent systems, including how agents interact and share information in decentralized environments. This helped me strengthen my technical and problem-solving skills while also improving my ability to analyze trade-offs in system design.

## Karen Valentina Mariel Villagrán

1. Analysis of the Developed Solution

I selected the multi-agent approach because it allowed for distributing the workload across autonomous agents while still coordinating actions to meet the global objective. Key considerations included autonomy, robustness, coordination, and scalability, which enabled the system to operate effectively even in complex terrains. The modular architecture facilitated communication and improved flexibility. On the other hand, this approach also introduced challenges such as synchronization overhead and added design complexity. These issues can be improved with more robust coordination rules and monitoring tools.

2. Reflection on My Learning Process

At first, I thought my main role would be limited to preparing diagrams and representations. However, I quickly realized how these models influenced system design and communication between agents. This broadened my perspective, helping me to connect theoretical models with their practical implementation. The experience allowed me to strengthen not only my technical knowledge but also my capacity to structure ideas and visualize interactions in complex systems.

## Guillermo Baltazar y Nungaray

1. Analysis of the Developed Solution

For this project, I opted for a multi-agent model because the mission demanded autonomous agents capable of working independently but with the ability to coordinate effectively. The critical factors were scalability, robustness, coordination, and environmental complexity. The modular architecture enabled adaptability and fault tolerance, which made the system more reliable. However, I also noticed disadvantages, such as communication delays and higher implementation costs. These can be reduced through optimized agent communication strategies and clearer coordination mechanisms.

2. Reflection on My Learning Process

I initially expected to focus only on the implementation of navigation and perception modules. Instead, I gained valuable insight into how these components integrate within a larger multi-agent system. I learned to evaluate the trade-offs between independent operation and coordinated behavior. This process improved my technical skills in module development as well as my analytical ability to anticipate challenges and propose improvements to ensure smooth collaboration among agents.

### Tarik Maina Gichuki

1. Analysis of the Developed Solution

I chose the multi-agent system because it supports distributed decision-making, which is crucial in dynamic and uncertain environments. Important variables considered were autonomy, coordination, and robustness. These aspects helped ensure that agents could work both independently and collectively. The architecture's modularity enhanced flexibility and made the system scalable. Nevertheless, drawbacks included coordination difficulties and unpredictable behaviors when multiple agents interacted simultaneously. These challenges can be minimized with better communication frameworks and monitoring.

2. Reflection on My Learning Process

At the beginning, I expected to work mainly on the coding side of the project. However, I discovered the importance of system architecture and how design decisions directly impact performance. I learned more about communication strategies among agents and the balance between centralized and decentralized approaches. This experience not only expanded my technical knowledge but also improved my critical thinking in problem-solving scenarios involving multiple autonomous entities.

### Mariano Sánchez Bermúdez:

I selected the multi-agent model because it allowed each agent to operate autonomously while still coordinating toward a common goal, something a single-agent approach could not capture as effectively. The key variables considered were autonomy, communication, and environmental constraints, all of which influenced how efficiently tasks were distributed and completed. I chose the system architecture for its modularity and scalability, which made adaptation easier. The solution's main advantages were robustness and adaptability, while the disadvantages were added complexity in communication and potential uncontrolled behaviors. These could be reduced with clearer protocols and better monitoring tools.

Looking back at my learning process, I see how much my perspective has shifted. At first, I thought the work would mainly involve coding functionality, but I came to understand the importance of design decisions, role definitions, and interaction analysis. This experience strengthened not only my technical skills but also my ability to evaluate trade-offs and anticipate challenges in building multi-agent systems.

## Complete implementation of the agents, graphical interface, and video of execution

In the GitHub repository: https://github.com/Lucianogrc/Person-of-Interest