Workshop - Integrating LLMs in Robotics & Drones

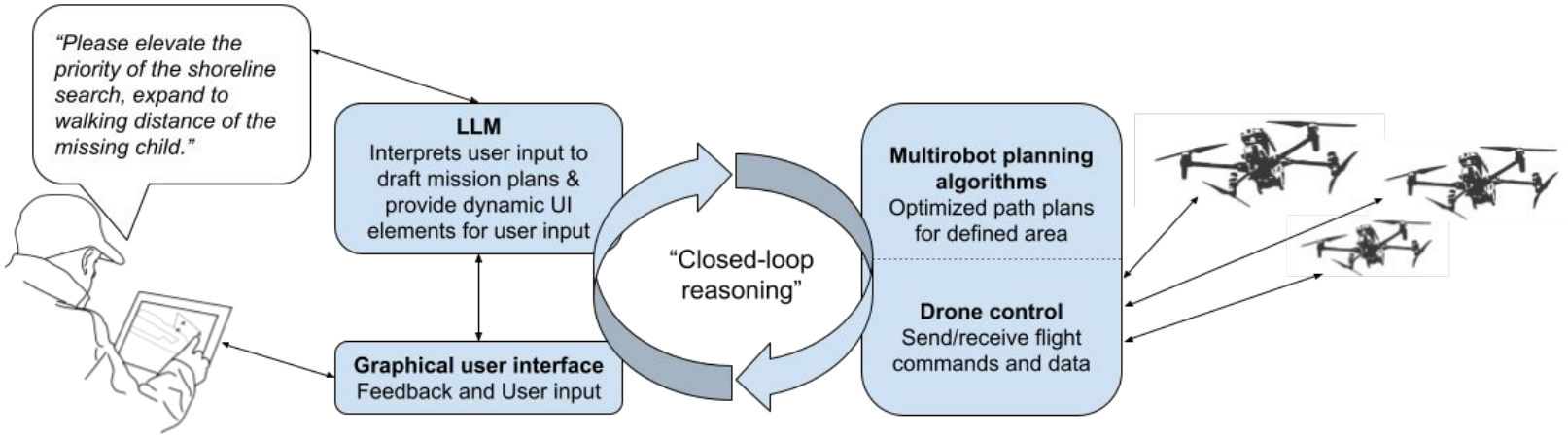
By: Alejandro Jarabo-Peñas **Supervisor:** Anders L. Christensen
UAS Drone Center, University of Southern Denmark (SDU), Odense Denmark

NAMUR Project: **N**atural-language **A**ssisted **Mu**lti-**R**obot Interaction and Planning
Funded by: Independent Research Fund Denmark under grant 10.46540/4264-00105B

SDU

NAMUR : **N**atural-language **A**ssisted **Mu**lti-**R**obot Interaction & Planning

Challenge	Multi-robot teams are hard to manage, especially for non-experts.
Opportunity	LLMs enable simple, natural team control.
Vision	Operators speak, drones act.



First test with real drones: the power of WildBridge

github.com/WildDrone/WildBridge

Part 1:

LLM APIs
Prompt Engineering
Tool Calling
LLM Agents

https://medium.com/data-science-at-microsoft/how-large-language-models-work-91c362f5b78f
https://www.youtube.com/watch?v=LPZh9BOjkQs

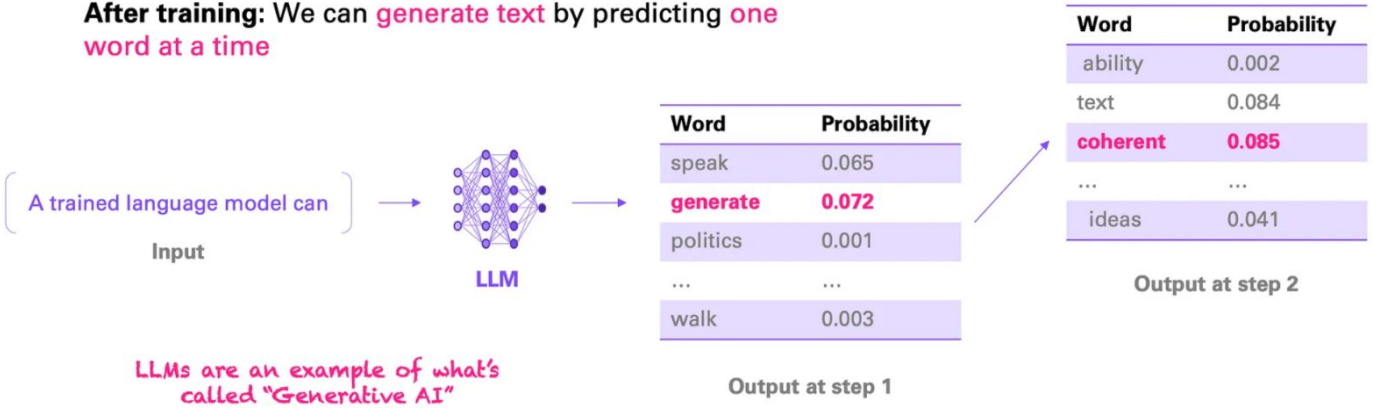
What are LLMs

Stochastic Parrots

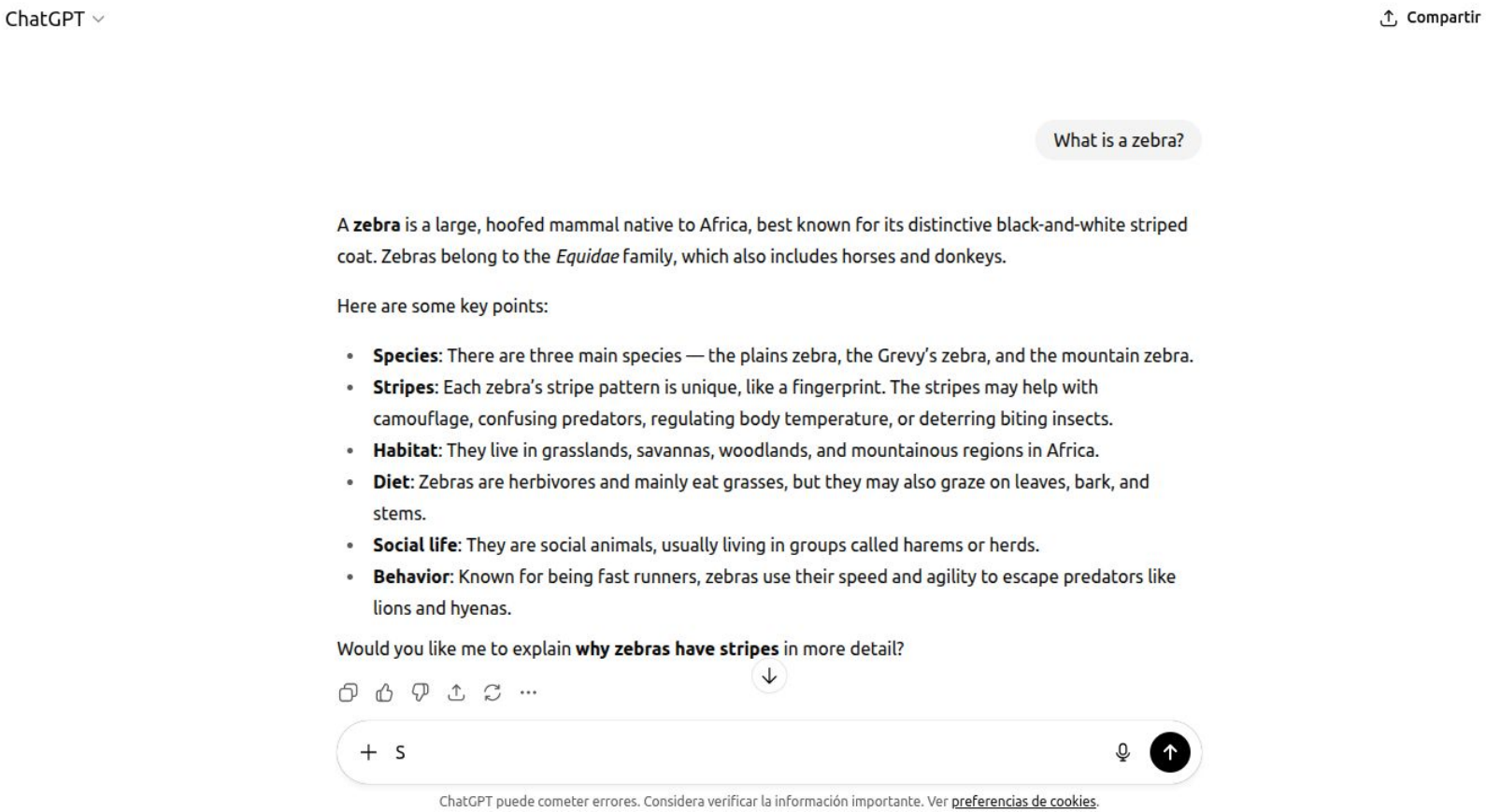
AI system trained on vast text data that generates language by predicting the next word in a sequence based on context

Stochastic parrot during training

After training: We can generate text by predicting one word at a time



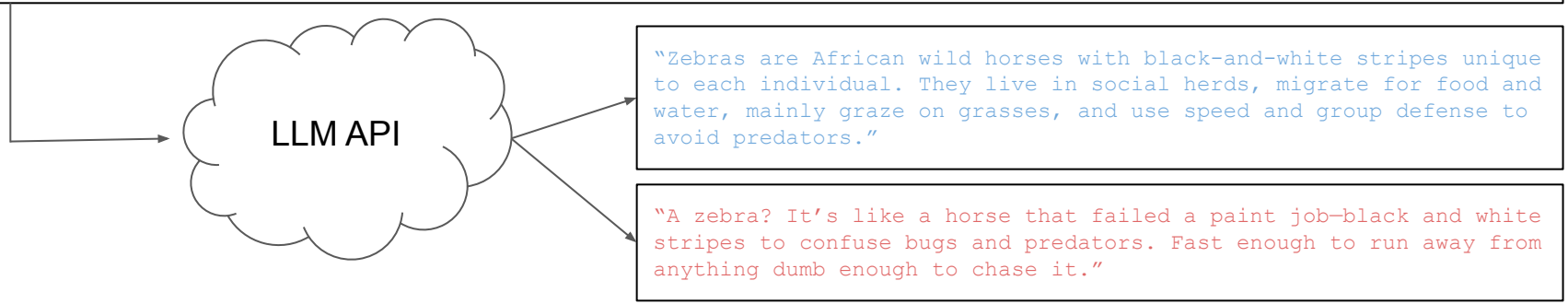
How we usually interact with LLMs



How to interact with LLMs via APIs

```
# Model options - Different sizes and capabilities
models = ["openai/gpt-4o-mini", "openai/gpt-4o", "google/gemini-2.5-flash", …]

response = completion(
  model="openai/gpt-4o-mini",    # Replace this with your chosen model from above
  messages=[
    {"role": "system", "content": "You are a helpful and very concise assistant."},    # System prompt #1
    {"role": "system", "content": "You are a rude, ironic and very concise assistant."},    # System prompt #2
    {"role": "user", "content": "What is a zebra?."}   # User prompt
  ],
  temperature=0.5,     # Controls creativity: lower is focused/deterministic, higher is creative/random
  max_tokens=200,      # Limits the length of generated output to 200 tokens (around 150 words)
)
```



Limitations of LLM Models

Problem #1: Outdated Knowledge

- Trained only on available data at the time of training
- Cannot access real-time updates (e.g., weather, stock values)

Problem #2: Weak in Math

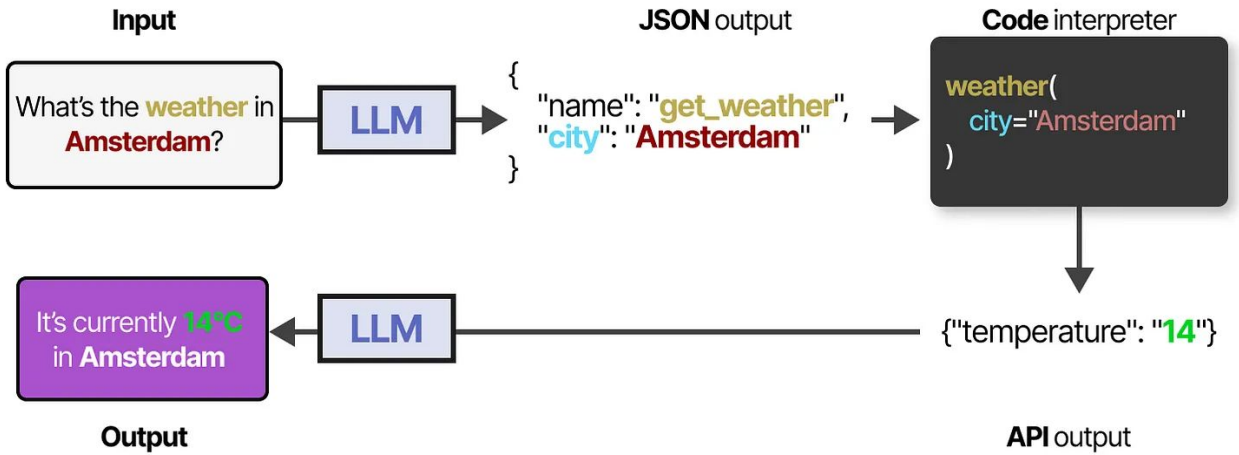
- Strong at writing and summarizing text
- Prone to errors with simple math operations

https://newsletter.maartengrootendorst.com/p/a-visual-guide-to-llm-agents

Augmented LLMs with Tool Calling: **LLM Agents**

Solution #1: Lack of Current Information

Connect to APIs (e.g., weather, stock prices) for real-time updates

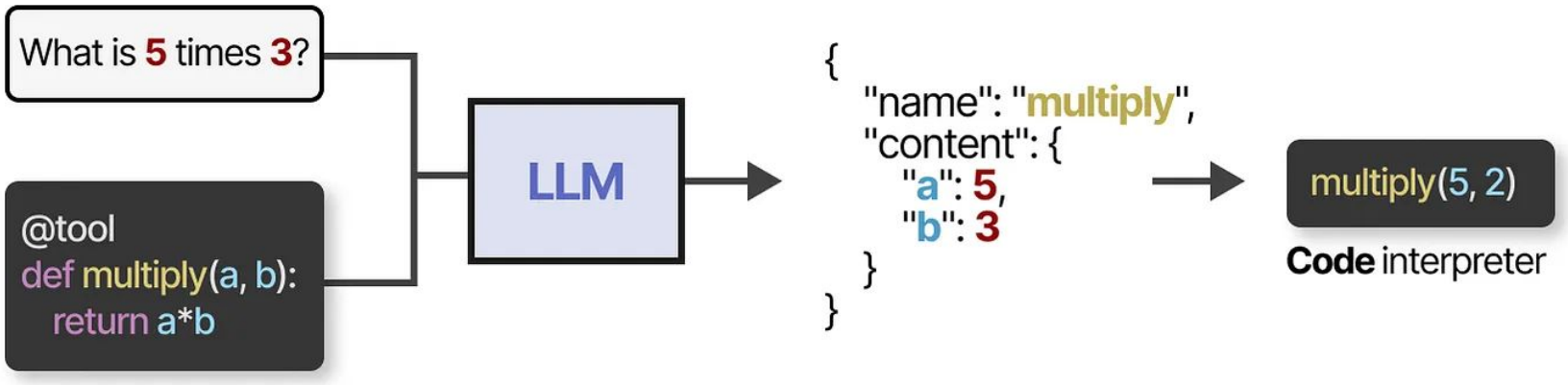


https://newsletter.maartengrootendorst.com/p/a-visual-guide-to-llm-agents

Augmented LLMs with Tool Calling: **LLM Agents**

Solution #2: Math Calculations

Allow the LLM to use a digital calculator for accurate results



How do we code LLM Agents

```python
# TOOLS DEFINITION
@add_parameters_schema(  # Parameters description for LLM Agent
    city={"type": "string", "description": "City name"},
)
def get_weather_forecast(city: str, days: int = 3) -> str:
    """Get current weather for a city using OpenWeatherMapAPI"""  # Tool description for LLM Agent

    url = f"http://api.openweathermap.org/data/2.5/weather?q={city}&units=metric"  # Weather API URL

    data = requests.get(url).json()
    weather = data["weather"][0]["description"].capitalize()
    temp = data["main"]["temp"]

    return f"Current weather in {city}: {weather}, {temp}°C"

# LLM AGENT DEFINITION
weather_llm_agent = LLMAgent(model="openai/gpt-4o-mini")  # Create an instance of an LLM Agent
weather_llm_agent.tools = [get_weather_forecast]  # Bind the tool to the agent

# Set the system prompt defining what the agent task is
weather_llm_agent.system_prompt = """You are a helpful assistant answering questions about the
weather. Use the tools provided to you to get updated information."""

# Test our weather agent
response = weather_llm_agent.chat("""Is the weather better in Zurich or Basel today?""")
```

1st Interactive Exercise
Prompt Tuning and LLM Agents

- Experiment with system and user prompts, and adjust temperature settings
- Build your own travel assistant agent that leverages external tools

- Form groups of 2-3 people, ideally including someone familiar with programming
- Spend about 30 minutes working through the notebook together
- Ask questions anytime something isn't clear

1st Google Colab Notebook: https://tinyurl.com/3yw7ufe6

Initial Setup:

- **Create a copy of the notebook:** Click File -> Save a copy in Drive
- **Get a free API key for Gemini:** Click on 🔑 icon on left menu -> Gemini API Keys -> Import key from Google AI Studio
- **Run the setup cells**

Part 2:

Using LLM Agents to Control Drones

Drone Safari Game - Overview & Purpose

Navigate a drone on a 12x12 grid to photograph three animals — Zebra, Elephant, and Oryx — while avoiding crashes and not scaring the animals away.

Game Rules:

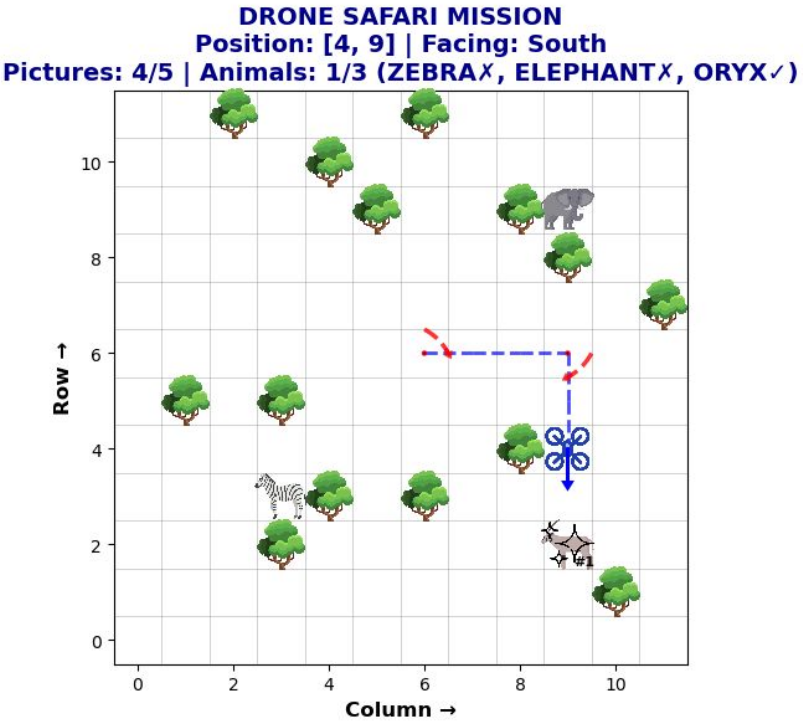
- Start at position (6,6) facing North
- You can take 5 photos in total (including any missed shots)
- Must be 2 cells away from an animal when taking photos
- Getting adjacent to animals scares them away — game over
- Crashing into trees, animals, or grid boundaries — game over
- Trees block your camera view when photographing

The goal is to capture pictures of all three animals with the limited # of shots available.

Purpose: Create an LLM Agent to play the game with natural-language commands:

- “Move forward”, “Move backwards”, “Move left”, or “Move right”
- “Turn right” or “Turn left”
- “Take picture”

This enables intuitive drone control through simple conversational interaction.



Break

2nd Interactive Exercise
Build your Drone Control Agent

- Familiarize yourself with the Drone Safari game
- Understand the tools available to interact with the game
- Connect these tools to your LLM Agent
- Craft a system prompt that enables your agent to perform the desired tasks
- Verify that natural language commands translate into correct in-game actions

Extra Challenge: Create a system prompt that allows multiple actions in a single command, for example: "Move forward 5 times, turn right, and take a picture"

2nd Google Colab Notebook: https://tinyurl.com/4tbwztwa

Initial Setup (same as before):

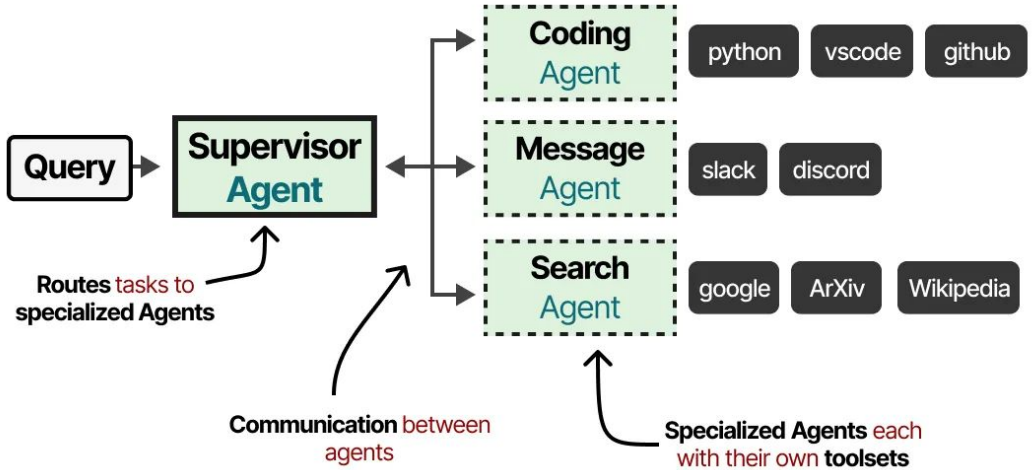
- **Create a copy of the notebook:** Click File -> Save a copy in Drive
- **Get a free API key for Gemini:** Click on 🔑 icon on left menu -> Gemini API Keys -> Import key from Google AI Studio
- **Run the setup cells**

Intro to LLM Multi Agent Systems

What? LLM agents with different roles working together

How? They communicate, coordinate, and split tasks

Why? More scalable, flexible, and effective than one agent alone



What we did in Malaga

Deployment of voice-controlled UGV (Rover J8)
in realistic SAR scenarios:

- 150+ first responders
- 3 real missions delivering critical materials

Non-expert operator control with natural language:

- Specifies missions for the UGV
- Multi-Agent LLM identifies where the robot should go
- Plans a safe route to reach that location, and makes UGV move
- Provides visual feedback on the completion status

Commands Interpretation - Our LLM Multi-Agent

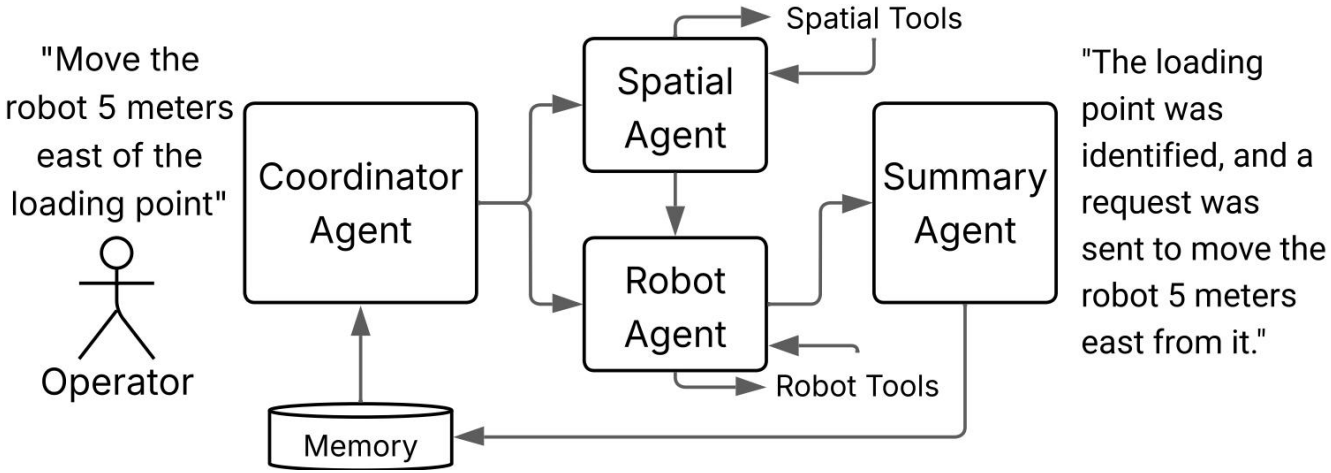
Coordinator → decomposes tasks

Spatial agent → spatial awareness

Robot agent → movement requests

Summary agent → conversational memory

Operator preview before confirmation = safety and correctness



Live Demo?

Thank you :)

Code used:
github.com/alejp1998/wilddrone-llm-workshop

SDU

Photos

More photos