



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: Natural-language Assisted **Multi-Robot** Interaction and Planning
Funded by: Independent Research Fund Denmark under grant 10.46540/4264-00105B

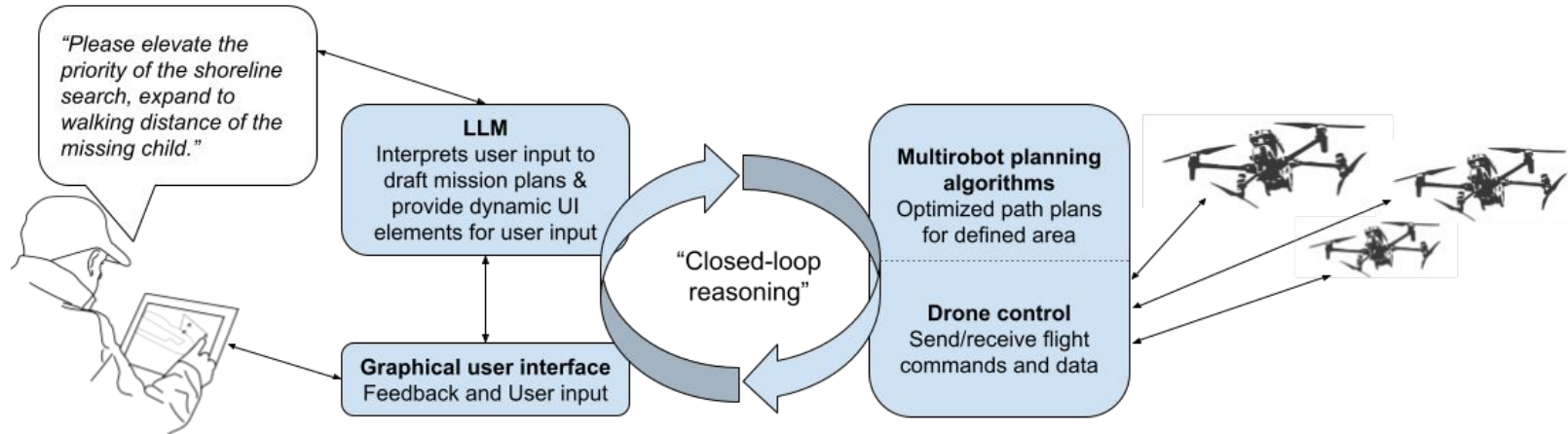


NAMUR : Natural-language Assisted Multi-Robot Interaction & Planning

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](https://github.com/WildDrone/WildBridge)



Part 1:

LLMs & APIs

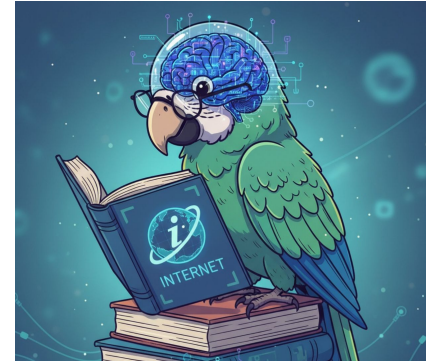
Prompt Engineering

Tool Calling

LLM Agents

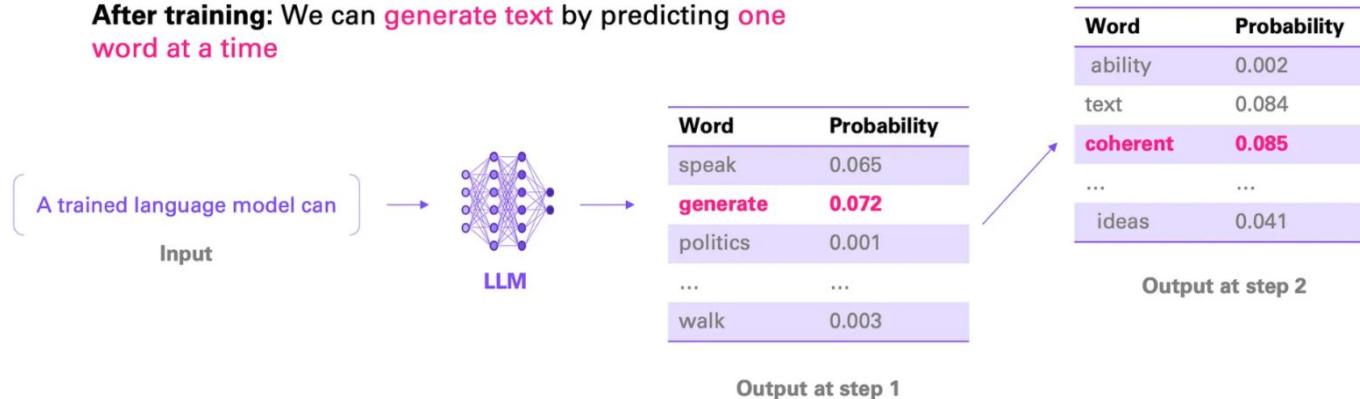
What is a Large Language Model (LLM)

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**

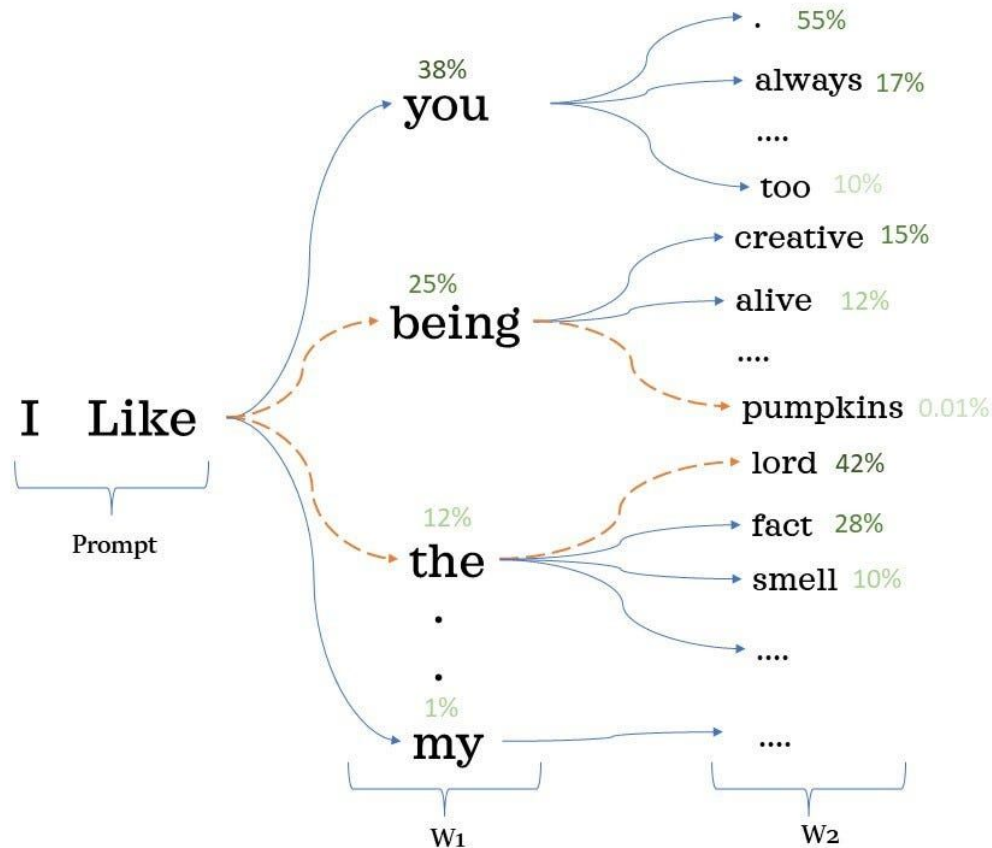


Temperature and how the next word is chosen



2.0 More Randomness/Creativity

0.0 More Deterministic



How we usually interact with LLMs

ChatGPT ▾

↑ Compartir

What is a zebra?

A **zebra** is a large, hoofed mammal native to Africa, best known for its distinctive black-and-white striped coat. Zebras belong to the *Equidae* family, which also includes horses and donkeys.

Here are some key points:

- **Species:** There are three main species — the plains zebra, the Grevy's zebra, and the mountain zebra.
- **Stripes:** Each zebra's stripe pattern is unique, like a fingerprint. The stripes may help with camouflage, confusing predators, regulating body temperature, or deterring biting insects.
- **Habitat:** They live in grasslands, savannas, woodlands, and mountainous regions in Africa.
- **Diet:** Zebras are herbivores and mainly eat grasses, but they may also graze on leaves, bark, and stems.
- **Social life:** They are social animals, usually living in groups called harems or herds.
- **Behavior:** Known for being fast runners, zebras use their speed and agility to escape predators like lions and hyenas.

Would you like me to explain **why zebras have stripes** in more detail?



+ S



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
```

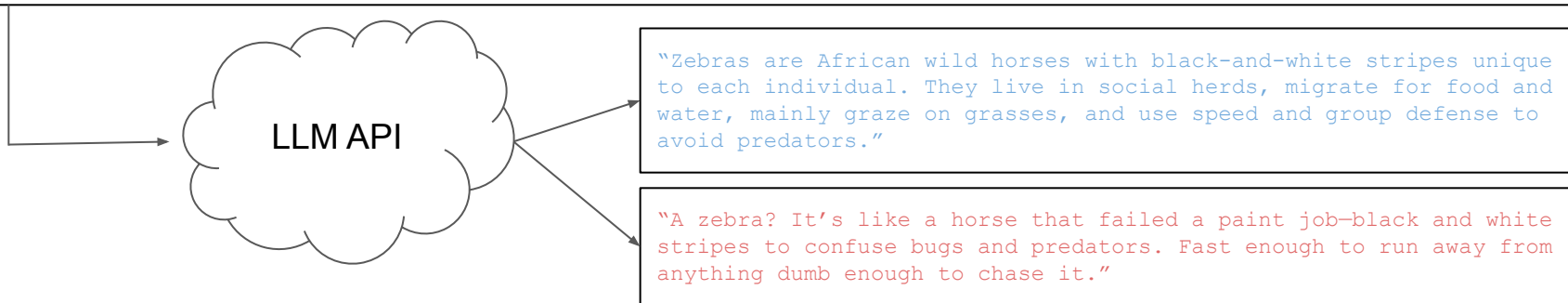
```
        {"role": "user", "content": "What is a zebra?."} # User prompt
```

```
    ],
```

```
    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)

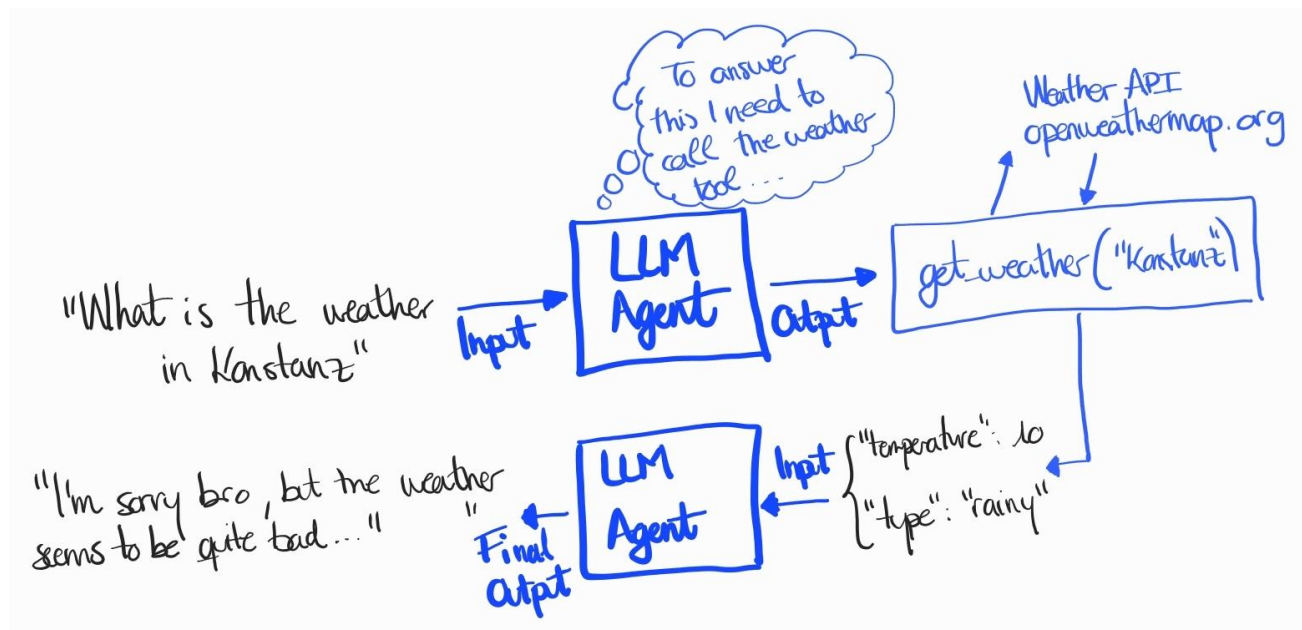
Problem #2: Weak in Math

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

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

Solution #1: Lack of Current Information

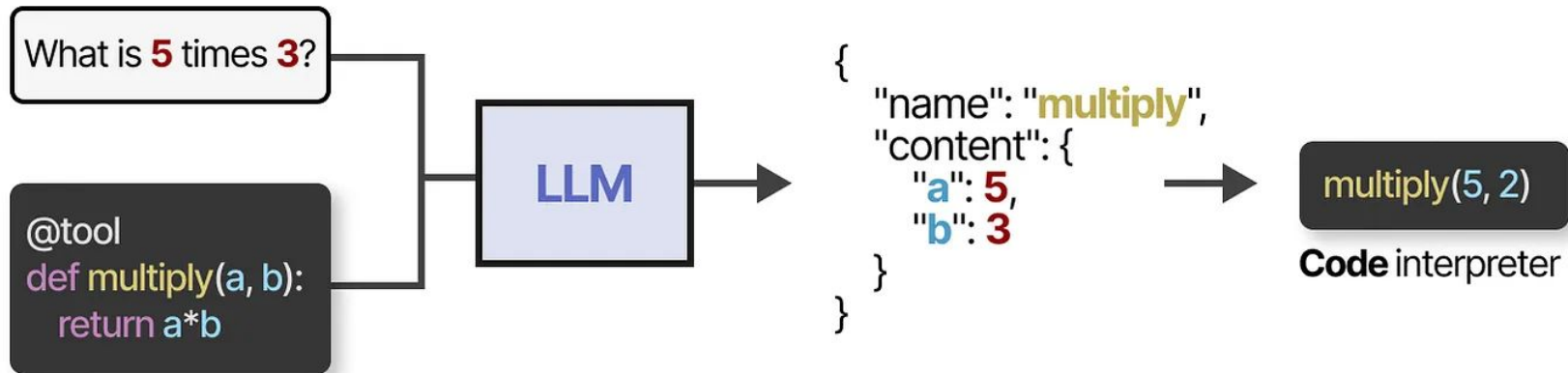
Connect to APIs (e.g., weather, stock prices) for real-time updates



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

TOOLS DEFINITION

```
@add_parameters_schema( # Parameters description for LLM Agent
    city={"type": "string", "description": "City name"},
)
def get_weather_forecast(city: str) -> 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 20 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

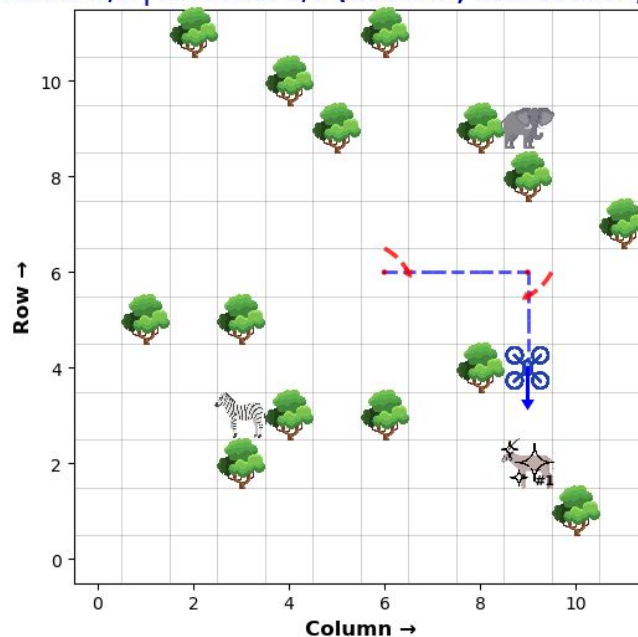
The goal is to capture pictures of all three animals with the limited # of shots available.

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.

DRONE SAFARI MISSION
Position: [4, 9] | Facing: South
Pictures: 4/5 | Animals: 1/3 (ZEBRA~~X~~, ELEPHANT~~X~~, ORYX✓)



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”

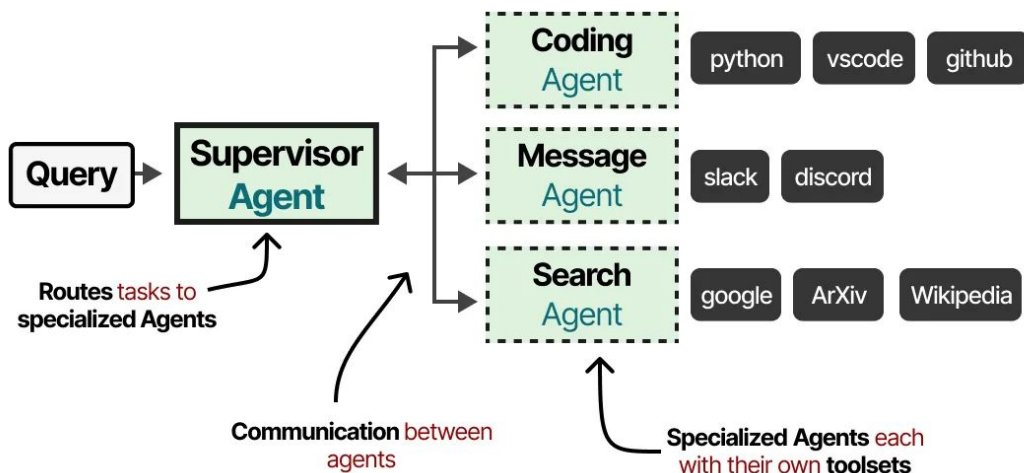
2nd Google Colab Notebook: <https://tinyurl.com/4tbwztwa>

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 ground robot in realistic SAR scenarios:

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

Non-expert operator control with natural language:

“Deliver the extra batteries to the drone pilot”

- Specifies missions for the robot
- Multi-Agent LLM identifies where the robot should go
- A safe route is planned to reach that location, robot starts moving there
- Provides visual feedback on the completion status



Commands Interpretation - Our LLM Multi-Agent

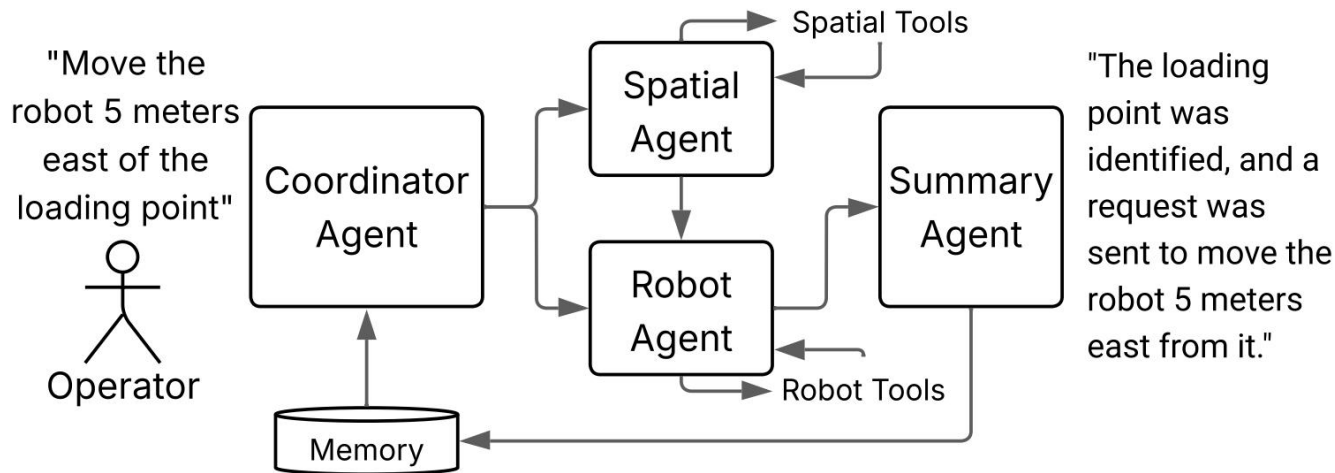
Coordinator → decomposes tasks

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



Photos



More photos

