EMERGING TECH CONFERENCE – Edge Intelligence

Volume 01, 2022, Page 169 – 176

Proceedings of Emerging Tech Conference:
Edge Intelligence 2022

An architecture for distributed decision making in a multi-agent UAV monitoring system

Niki Patrinopoulou[a], Ioannis Daramouskas[a,b], Dimitrios Meimetis[a], Vaios Lappas[c], Vassilios Kostopoulos[a]

[a]*University of Patras, Rio,Patra and 26504, Greece*
[b]*Computer Technology Institute and Press "Diophantus", Rio, Patra and 26504, Greece*
[c]*Greece National Kapodistrian University of Athens, Athens, Athens and 15772, Greece*

n.patrinopoulou@upnet.gr

Abstract

Unmanned aerial vehicles (UAVs) have proved to have a significant role in a plethora of applications (e.g., agriculture, search and rescue missions, infrastructure surveillance, area monitoring) and when clustered into collaborating groups are able to implement complex and cumbersome tasks efficiently. Systems of multiple autonomous UAVs can be seen as a system of moving sensors, which can be designed with design-making properties, in order to automate and optimize the gathering data process. If each UAV agent is provided with data processing functionalities in addition to data gathering, the system is transformed to a distributed system with great scalability capabilities. This paper presents a decentralized architecture for a UAV multi-agent system designed for area monitoring.

1. Introduction

Area monitoring and surveillance traditionally requires significant manpower, especially for large areas. By introducing unmanned agents, the process can become autonomous, demanding only one optional human supervisor and no additional manpower. This work introduces a decentralized, highly scalable architecture for a multi-agent UAV area monitoring system with a central entity acting as the sole point of truth. The decentralized architecture allows the allocation of the processing and the decision-making across the agents, minimising the quantity of data to be exchanged and ensuring the system's operation during temporary losses of connection.

[1]categorizes Multi-Agent Systems' architectures into centralized systems with a leader agent, federated systems with mediating agents, and decentralized systems with no hierarchy between the agents. [2] outlines some of the benefits of a decentralised architecture, including its invulnerability to the single-point-of-failure issue, its modular nature, and its capability of serving heterogenous multi-agent systems. The authors also suggest a hybrid architecture, which allocates the high-level swarm mission planning to a central agent, while the robotic agents conduct the planned missions using local sensing and communications. Their hybrid architecture was designed for small size multi-agent systems.

The proposed architecture is described in Section 2, while the required algorithms for the system are introduced in Section 3.An overview of the system's implementation and a comparison to more centralized approaches are given in Section 4.

2. System Overview

The system is designed to be scalable and autonomous. The system was originally designed as an area surveillance/monitoring system, but it can easily be adapted to fit a plethora of applications. The system includes n UAV agents, where the number of agents n may vary during the system's operation with no impact to the performance of the rest of the agents, along with a central entity agent responsible for gathering and redistributing data. It should be noted here that since a central entity exists, there is a bottleneck to the number of UAV agents that it can serve, but the limitations of the number of agents able to cooperate in the same system have been heavily relaxed in comparison to the fully centralized approach, as it is shown in Section 4.2. Each UAV agent of the system is a copy of the others and it is able to gather and process its own data and conclude to its own decisions based on those. The UAV agents communicate their processed data to the central entity, which gathers and groups relevant data. When a UAV agent needs the grouped data to make a decision, it will ask the central entity to transmit them. The central entity's data can also be used provide information to a human user/ supervisor.

The role of the UAV is to detect and follow intruders in the area of interest for a specified amount of time. To achieve that goal, the UAV must have the following abilities: detect intruders from optical camera feed, follow intruders, scan selected areas, select its next scanning area, and communicate its findings to the central entity.

The intruder detection process runs uninterrupted during the entire operation of the drone. On the other hand, the rest of the functionalities are enabled in different times only when the UAV agent activates the regarding mode. The agent is designed to have 3 modes; the scan mode, the follow intruder mode, and the communication with central mode, in which data transmission occurs. The three modes and the triggering events for transitioning between them are displayed in Fig. 1.

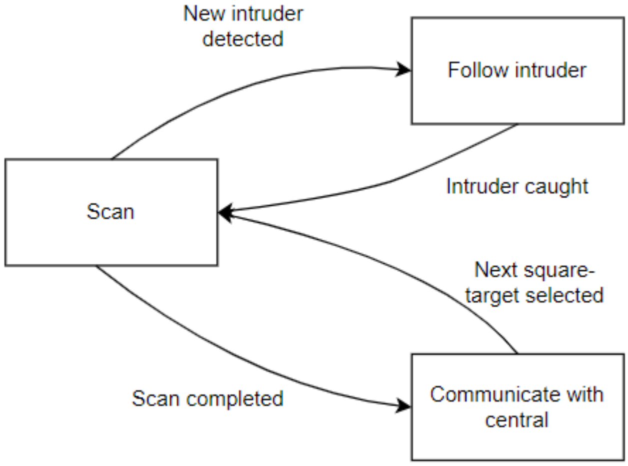

Figure 1: UAV agents' mode sequence and change triggers. The three modes of the agent "Scan", "Follow intruder" and "Communicate with central" are visualised as rectangles and the transitions between the modes are arrows, explaining the type of the cause that triggered the transition

We function under the assumption that the area to be monitored is a square horizontal plane. In order to manipulate that area in our decision-making algorithm, the world is treated as a grid containing equal size squares. Each of those squares is a potential target for the UAV agent and as such, they are referred as square-targets for the entirety of this paper. The central entity sees the grid as a 2D matrix, with each of the matrix's elements containing two values. The first value corresponds to the time that square-target was lastly scanned, while the second value is the estimated, based on gathered data, probability of finding an intruder in that square-target. The matrix values are updated regularly with a specific time step and during the update the recently received UAV data are integrated.

Fig. 2. shows the data exchange between the central entity and one UAV agent. The scan data message is sent from the UAV to the central entity every time a scan is completed and includes the number of detected intruders during scanning and the ID of the scanned square-target. The next square-target message contains the ID of the next selected target to be scanned by the agent, so that the central entity may flag it until its scanning is completed. The central entity sends the UAV agent a subset of the world matrix, so that the agent can make its next selection.

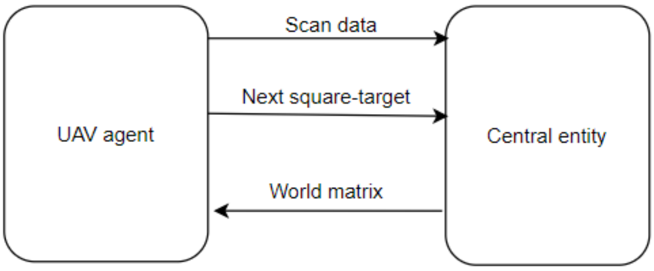

Figure 2: Central entity and UAV agent data exchange.

In our system we have drones equipped with optical cameras gathering large amount of data. If the camera's feed were to be transported from the UAV agents to the central entity for processing that would cause three potential problems; latency, bandwidth limitations and inability to scale the system upwards. By processing the camera's feed on-board we greatly minimize the amount of data to be transmitted within the system and allow the system to be easily scalable. So, in our application each UAV agent can be seen as an "edge". While the latencies added by a centralized approach of that system would not be disastrous for the system's function, the capability of being scalable is of high importance for such application.

3. Algorithms

The UAV agents utilize four algorithms to achieve their goals, an area coverage algorithm, a decision-making algorithm, an object detection algorithm, and an object tracking algorithm. The area coverage algorithm is used to guide the UAV during the scanning procedure of a square-target, while the decision-making algorithm is responsible for selecting the next square-target for the UAV agent. The object detection algorithm aims at identifying human-like intruders from the camera frames, and the object tracking algorithm is used to match detected intruders between sequential frames. The algorithms are briefly described in this section.

The coverage algorithm is enabled in the "scan" mode of the drone with the purpose of creating a path through which the whole square-target is covered by the field of view of the agent's camera.

To achieve this the drone follows a zig-zag pattern, concluding an alternative sequence of left-wise and right-wise movement interrupted by a shift in the other axis.

The decision-making algorithm is activated after the scanning process is completed. Its aim is to find the next square-target for the UAV agent; this decision is based on data received from the central entity concerning the area of interest. From the data, the agent computes margins for these square-targets based on the number of intruders detected in each of them and the time they were last visited. One of the square-targets is selected in a stochastic manner based on the margin assigned to it and the neighboring square-targets. The algorithm can be seen as a task allocation algorithm since it is solving an assignment problem in order to assign one of the square-targets to an agent. There is a large set of algorithms to choose from for this application, we selected to use a swarm intelligence stochastic optimization algorithm.

The proposed object detection module is based on the YOLO model and more specifically is a custom YOLOv4 model, as it is capable of multi-object detection in real-time. Multiple object detection is essential for this specific application, as agents should be capable of detecting other intruders while following one of them. The YOLO model returns the computed bounding boxes of the intruders in the image frame. The bounding box is fed to the target tracking algorithm, the confirmed bounding box coordinates are fed to the guidance algorithm to guide the UAV while following an intruder. The target tracking is achieved by the Deep SORT algorithm. The object detection and target tracking are described in detail in [3].

4. Results

4.1 Implementation

The described system was implemented and tested in Software-In-The-Loop (SITL) simulations. The selected UAV agent is a quadcopter equipped with a downwards looking optical camera. The SITL simulations were conducted using the GAZEBO [4] physics engine. The selected drone is an iris px4 [5] drone, presented in Fig. 3., with an optical camera.

Figure 3: The iris drone as it is visualized in the GAZEBO simulator.

Although the system was developed and tested in SITL simulation for the purpose of this work, for a real-world application the drones would have to be equipped with some onboard system capable of running the object detection and tracking algorithms and the decision-making and guidance algorithms (including coverage). The researchers in [6] introduce a drone equipped with a Jetson TX2 board, capable of running visual tracking on board, exploiting a stereo camera's feed and visual odometry. In [7] a Raspberry Pi with Intel Neural Compute Stick Movidius VPU is used as the edge computing device mounted on UAV to perform the object detection in real-time.

The present system was developed in C++ and python using the Robotic Operation System (ROS) [8] in the version ROS melodic. Fig. 4. shows the drone's software architecture in ROS, while Fig. 5. shows the communication between UAV agent and the central entity in ROS. The UAV's guidance algorithms are written in C++ and the central entity code is developed in python. As depicted in Fig. 4. two ROS nodes were developed for every UAV agent (identical for all of them);

one executing the object detection and tracking algorithms with input the camera frames published from the camera node, and one named “Guidance node” executing all the guidance and decision-making processes. The “Guidance node” computes the velocity commands for the aircraft and publishes them, for the “mavros node” to receive them. The purpose of the “mavros node” is to receive the control commands and translate them from the ROS environment to the format the autopilot understands using the MAVLink communication protocol [9].

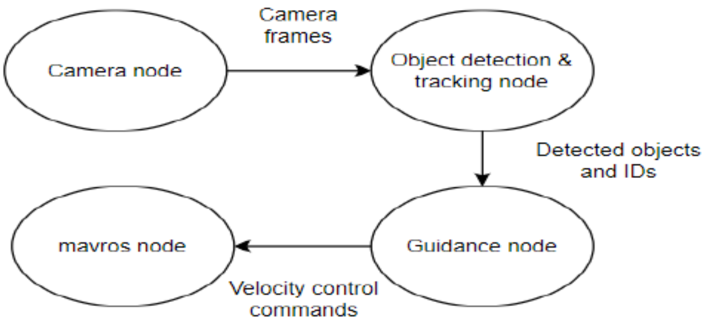

Figure 4: ROS architecture in the UAV agent. The nodes are presented as ellipses, while the messages published in the ROS Topics are presented in the arrows connecting the nodes.

The “Guidance node” receives input from the “Object detection & tracking node” in the UAV’s system and is also responsible for the communication with the “central entity node” (Fig. 5). The communication between the central entity and the UAV agents occurs using ROS Topics and custom messages, one per each type of data messages exchanged between the central entity and the UAVs.

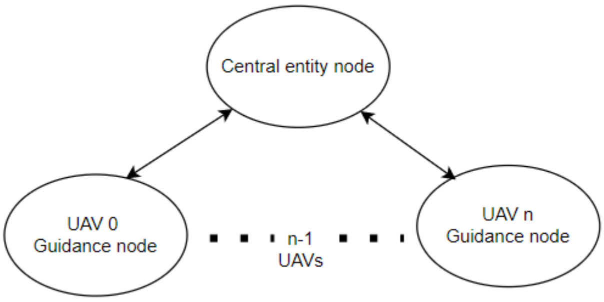

Figure 5. The central entity and UAV agents communication architecture.

4.2 Bandwidth benefits

In the current section we compare the decentralized system presented above with two centralized implementations of a system using the same logic and algorithms. The comparison is used to show the positive impact of the decentralized architecture on the system’s scalability.

The decentralized approach implemented in this work enables the distribution of the processing weight among the agents and requires minimum data exchange. The UAV agents only need to send the number of the detected intruders in a scanned square, and a message containing the ID of the next selected square-target for the UAV agent. While central entity sends the gathered data for each square-target in the area of interest for the UAV agent. The necessary data for each square-target are its unique ID, the last time it was scanned by a drone, and the probability of containing intruders in the present time.

For the centralized system, which we use for comparison, we assume the same architecture and algorithms with the only difference that the object detection algorithm runs in the central entity, which requires each UAV agent to transmit its camera feed to the central entity agent for processing. The rest of the transmitted data remain the same. As we centralize the processing of the data, it is reasonable to think about the centralization of the decision-making process as well. So, we introduce a second centralized system which implements the decision-making and guidance processes in addition to the object detection through the central entity agent, albeit it functions with the same algorithmic logic.

The data exchanged between the central entity and a single UAV agent in the decentralized approach are listed here:

- Scan message: sent from the UAV agent to the central entity every time a scan is completed and includes two integer values; the number of detected intruders and the scanned square-target's ID.
- Next square-target message: sent from the UAV agent to the central entity every time a new target is selected and includes two integer values; the ID of the UAV and the selected square-target's ID.
- World map message: sent from the central entity to the UAV agent every time the agent needs to select its new target and includes two long float values and one integer for each of the square-targets of the requested map. In our algorithm the agent requests the map of its immediate neighborhood, including 9 square-targets, that value may vary depending on the chosen size of squares, the flight altitude and the decision-making algorithm in use.

Those messages are sent sequentially every time a scan is completed. In our experiments the squares' size was 10x10 m and the average time interval between sequential scans for an agent was 40.9 seconds. The data exchanged between the central entity and a single UAV agent in the centralized object detection and tracking approach are inherited from the decentralized approach with two additional messages, listed here:

- Camera feed: streaming from the UAV agent to the central entity.
- Detected intruders message sent from the central entity to the UAV agent for every detected intruder and it includes the bounding box of the detected object and its tracking ID. It is difficult to estimate the frequency required for that message, as it is dependent to the density of the intruders in the camera frame.

Finally, the data exchanged between the central entity and a single UAV agent in the fully centralized approach are listed here:

- Camera feed: streaming from the UAV agent to the central entity.
- MAVLink control messages: streaming from the central entity to the UAV agent.

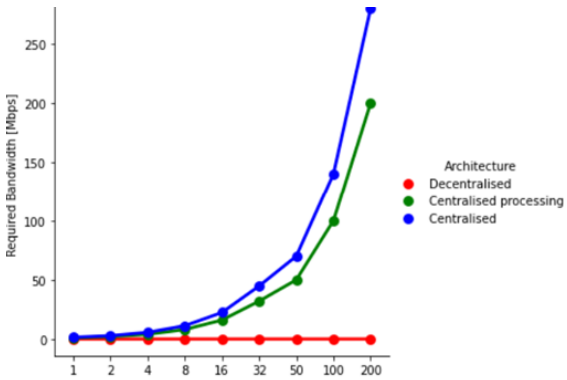

Figure 6: Graph showing the bandwidth requirements of each of the three approaches based on the number of UAV agents in the system.

Based on [10] the required bandwidth for streaming FPV camera video is 1 to 2 Mbps, while C2-MAVLink control stream requires 0.4 Mbps. It is evident how centralizing the image processing sets hard limitations to the system's scalability. The limits get stricter when the guidance algorithms of the agents are also assigned to a central entity.

Fig. 6. presents the relationship between the required bandwidth and the number of UAV agents in the system for each of the implementations. While the demanded bandwidth is almost constant for the decentralized approach proposed in this paper, that is not the case for the more centralized approaches, for which a quite significant increase is depicted in the graph, showing a major problem in scaling such a system up. The required bandwidth for the two centralized systems was estimated based on the values provided above, while for the decentralized system the bandwidth was measured for each of the three ROS Topics used for the communication of the UAV agent and the central entity and was summed to calculate the overall bandwidth. For each of the scan and next square-target messages the bandwidth was measured at 0.35 B/s, and for the world map message the bandwidth was measured at 8 B/s.

It is also obvious that the scalability of the system is not only deprecated by the bandwidth limitations, but from the processing power of the central entity as well, as it is unwise to assume that it would be able to keep up with the processing requirements for executing real-time object detecting in dozens (or even more) of camera feeds.

5. Conclusions

The presented system above was initially designed as an area monitoring system, albeit the scanning and task allocation algorithms can become quite elaborate in order to increase the system's efficiency and may be adjusted depending on the environment and the specific application it is intended for. The paper focuses on proposing a decentralized architecture which may be utilized in a variety of applications. The centralization of the system's architecture would impose severe bandwidth and processing constraints. So, for such systems in which scalability is important, we propose the use of a decentralized architecture, distributing the processing and decision-making demands among the agents. Adding a centralized agent to which minimum processing is allocated and it solely acts as single point of truth, provides the system with global optimization capabilities.

Acknowledgments

This research is based upon work supported by the Air Force Office of Scientific Research under award number FA9550-19-1-7032 (Real-time Decision Making for Autonomous Systemsusing AI Methods).

References

[1] A. Rasheed, M. Abdullah and A. Al-Araji, “A review of multi-agent mobile robot systems applications,” International Journal of Electrical and Computer Engineering (IJECE), vol. 12, no. 4, pp. 3517-3529, 2022.
[2] M. Abdelkader, S. Güler, H. Jaleel and J. Shamma , “Aerial Swarms: Recent Applications and Challenges,” Current Robotics Reports, vol. 2, p. 309–320, 2021.
[3] I. Daramouskas, N. Patrinopoulou, D. Meimetis, V. Lappas and V. Kostopoulos, “A design and simulation of a target detection, tracking and localisation system for UAVs,” in 2022 30th Mediterranean Conference on Control and Automation (MED), Athens, Greece, 2022.
[4] N. Koenig and . A. Howard, “Design and Use Paradigms for Gazebo, An Open-Source Multi-Robot Simulator,” in Proceedings 01 2004 IEEE!RSJ International Conference on Intelligent Robots and Systems, Sendai, Japan, 2004. doi: 10.1109/IROS.2004.1389727.
[5] “PX4 autopilot,” Dronecode, [Online]. Available: Open-source embedded framework for Unmanned. [Accessed 12 11 2020].
[6] P. Wyder, Y.-S. Chen, A. Lasrado, R. Pelles, R. Kwiatkowski, E. Comas, R. Kennedy, A. Mangla, Z. Huang, X. Hu, Z. Xiong, T. Aharoni, T.-C. Chuang and H. Lipson, “Autonomous drone hunter operating by deep learning and all-onboard computations in GPS-denied environments,” PLOS ONE, vol. 14, no. 11, 2019.
[7] M. S. Alam, . B. V. Natesha, . T. S. Ashwin and R. M. R. Guddeti , “UAV based cost-effective real-time abnormal event detection using edge computing,” Multimedia Tools and Applications volume, vol. 78, p. 35119–35134, 2019.
[8] M. Quigley , K. Conley , B. Gerkey, J. Faust , T. Foote , J. Leibs, R. Wheeler and A. Ng, “ROS: an open-source Robot Operating System,” in ICRA Workshop on Open Source Software, Kobe, Japan, 12-17 May 2009.
[9] “mavlink-ROSWiki,” [Online]. Available: http://wiki.ros.org/mavlink. [Accessed 2021].
[10] K. Kabbabe Poleo, W. Crowther and M. Barnes, “Estimating the Impact of Drone-based Inspection on the Levelised Cost of Electricity for Offshore Wind Farms,” Results in Engineering, vol. 9, 2021.