


# Autonomous trail-following unmanned aerial vehicle system based on resource partitioning of single hardware platform

Yoojin Lim,  Kyungil Kim, Jinah Shin, and Chaedeok Lim

Electronics and Telecommunications Research Institute, Daejeon, South Korea

As deep neural networks are spreading to almost all fields, flight systems in the unmanned aerial vehicle (UAV) domain are undergoing various transitions to intelligent systems. Among these transitions—in a bid to reduce flight risk—is the active research domain of autonomous navigation for intelligent UAVs. The autonomous trail-following flight system that this letter introduces can safely consolidate flight control and mission control within the latest commercial hardware platform. The resource usage and degradation of pass-through delay in vision-based convolutional neural network workloads show that virtualisation overhead is not significantly negative, and the overall performance of the introduced system is acceptable. Real-time cooperation is also verified as achievable—in that the workloads incur minimal communication delay—between the controls. Finally, the actual field test analysis demonstrates the applicability of our autonomous UAV system, whereby our system controls the UAV to follow the centre of a set trail.

**Introduction:** Recently, deep neural network (DNN) technologies have been actively studied and applied in order to raise the bar regarding intelligence in system control domains, such as unmanned aerial vehicles (UAVs). Prior works [1, 2] focus on enhancing the performance of specific missions such as tracking or monitoring within a UAV platform. Other research projects [3, 4] focus on improving UAV-related features, such as flight control or safety. With a focus on reducing flight risk or enhancing flight precision, autonomous navigation for UAVs is one of these actively studied domains. Research in [5] proposed a UAV guiding system based on image processing plans to include an artificial intelligence module for more accurate flight control in future work. Unlike prior research projects, which used a commercial flight controller (FC) device or did not use DNN-based missions, our work demonstrates a consolidated autonomous flight system that runs the partitioned controllers on one hardware platform.

The main functions of an autonomous trail-following flight system—the system that forms the focus of this study—are flight control and mission control. The FC manages a UAV's attitude and flight routes based on data gathered from its sensor and the external pilot, whereas the mission controller manages the purpose of the flight while also performing non-flight functions. The autonomous flight system that this work proposes has a cooperating and integrating architecture, in which a convolutional neural network (CNN)-based mission controller (MC) assists an FC, using the results gathered from processing real-time image data.

**System structure for autonomous trail-following UAV:** The proposed UAV system should be able to support the real-time cooperation between the FC and the MC on a single embedded hardware platform. The system should also be able to follow a trail in real-time to achieve autonomous flight while simultaneously adjusting its route to stay near the centre of the trail and avoid the trail edge. As demonstrated in Figure 1, to satisfy these requirements, the autonomous UAV, implemented on a partitioning hypervisor, performs CNN-based missions in the MC. Based on the results, it delivers the movement commands to the FC. Applying hypervisor technology to a consolidated flight system [6] is both reasonable and attractive because a hypervisor can both handle and independently operate the software and hardware features required for each system without conflicts. The MC uses micro air vehicle robot operating system (MAVROS), one of the robot operating system (OS) packages, for MC–FC communication through which it monitors the current status of the UAV and provides trail-following commands.

Trail navigation and position recognition algorithms are used for CNN-based missions, similar to the head direction and lateral offset of

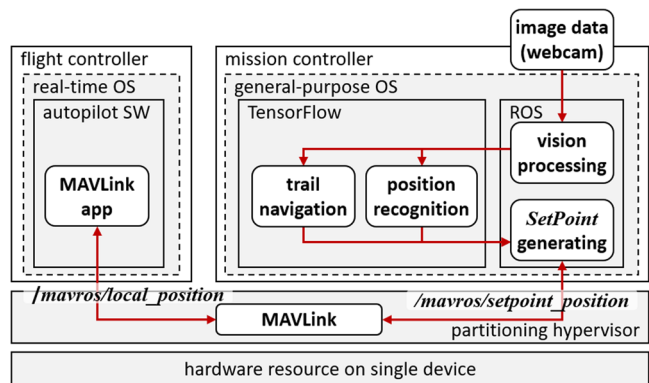


Fig 1 Conceptual data flow overview for the autonomous unmanned aerial vehicle (UAV) with vision-based trail-following flight

a vision-based bike trail-following approach [3, 4]. These algorithms control a UAV to follow a given trail while maintaining a position at near-centre of the trail. To explain in more detail, the trail navigation—a monocular vision system—is a CNN-based algorithm used to detect and track the trail according to clear distinctions. This algorithm allows the UAV to detect the direction in which a trail is located or whether a trail even exists, helping it to decide the direction to travel in. The MC performs the mission algorithm, detecting and tracking trails in real-time based on images obtained from the camera. The position recognition algorithm recognises the location of the UAV above the trail and guides its movement along the centre of the trail. These two CNN-based mission algorithms are performed both independently and in parallel through TensorFlow. The direction commands received from each mission are integrated into a MAVROS setpoint and delivered to the FC.

**Implementation of consolidated UAV system:** The details regarding the FC, the MC, and the partitioning hypervisor are as follows: Qplus-AIR is an avionic 64-bit real-time OS that supports temporal and spatial partitioning based on ARINC653 interfaces. It is DO-178B Level A certified and has been successfully tested for unmanned aircraft system flying [7]. Linux for Tegra (L4T) is an Ubuntu-based 64-bit general-purpose OS and NVIDIA's officially distributed board support package for the NVIDIA Jetson board—the kernel version is 4.4.8. The research prototype of a proprietary hypervisor (named EARTH) is a type 1 full-virtualisation technique based on architecture (ARM) virtualisation extensions. EARTH is designed to support the reusability of a guest OS and minimise the overhead for hardware resource virtualisation. It isolates various resources such as the central processing unit (CPU) core, memory region, and peripheral input/output devices while booting the system and allocates them to guest OSs statically. Furthermore, it provides the functionality for shared memory-based communication among guest OSs. Additionally, EARTH is certifiable with Radio Technical Commission for Aeronauts DO-178C Level A, such as PikeOS and VxWorks.

The hardware platform used for the target UAV system is NVIDIA Tegra X2 (TX2), which is a common embedded platform for artificial intelligence-based applications. TX2 has six cores in two ARM-based CPU clusters. It provides a physical memory of 8 GB and a graphics processing unit (GPU) of 256 Compute Unified Device Architecture cores. It also includes communication buses, such as universal asynchronous receiver/transmitters (UART) and serial peripheral interfaces (SPI), through a connected developer kit. For the sensor board, which is essential to the flight operation of UAVs, Emlid's NAVIO2 is used. This device has sensors for an inertial measurement unit, global positioning system, and barometer, and it provides pulse-width modulation outputs and extension ports independently. The camera for the vision-based CNN workloads is a Microsoft HD-3000, and the video frame is 720p at 30 fps. Also, the UAV consists of a Tarot X6 hexacopter frame, Hobbywing XRotor Pro 50A electronic speed controllers, Tarot 5010 300KV motors, an OrangeRx R620X V3 6Ch receiver, a Holybro telemetry radio 915 Mhz transceiver, and the implemented TX2.

In the architecture of the autonomous UAV system, EARTH virtualises and partitions the hardware resources of TX2 as shown in

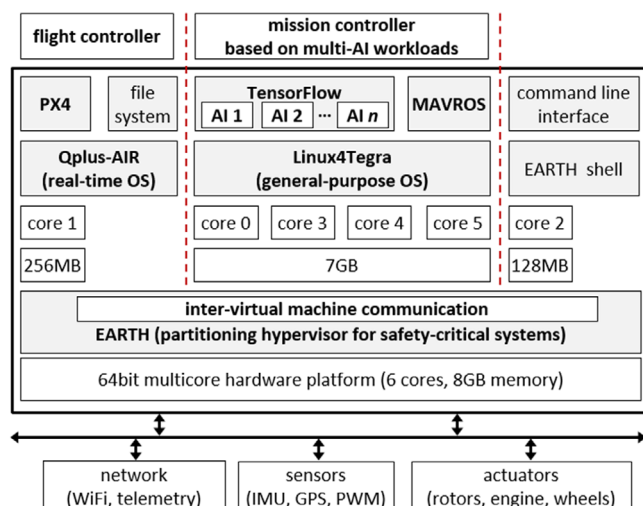


Fig 2 Implemented system architecture for autonomous UAV

Figure 2. EARTH allocates partitioned resources and manages access to support the independent execution of guests, including the FC and the MC. It also provides inter-virtual machine communication (IVC) to allow for inter-guest communication. For the FC, Qplus-AIR was used with one core and 256 MB. It creates a partition, operating the well-known FC application PX4, version 1.5.4. The Qplus-AIR kernel provides a special-purpose function that allows an application kernel-level access to memory. This means that PX4 can directly access the shared memory and perform IVC with the MC. EARTH allocates SPI buses for NAVIO2 to the FC, and the FC uses these to acquire sensing data and control the motors. For the MC to locate the UAV position and run the trail navigation, NVIDIA's L4T was used with four cores and 7 GB allocated. In the L4T, TensorFlow is installed to execute the CNN-based mission algorithms, and the MAVROS package and IVC driver provided by EARTH are installed to send generated setpoint commands to the PX4 of the FC through IVC. EARTH allocates most of the device resources such as USB and communication ports to the MC, and the MC uses these to obtain vision data from the camera and communicates with the ground control station. In the remaining domain, a 32-bit shell guest is enabled that supports UART communication and is used for monitoring statistical measurement and functional tests to develop the UAV system.

**Measuring overhead from CNN workloads:** The results of this test are based on a performance analysis of the MC with regards to operating mission algorithms simultaneously. The results indicate that the virtual-

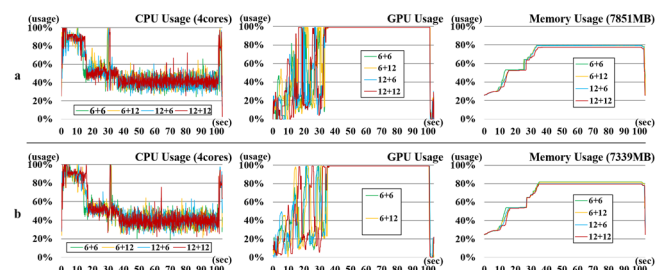


Fig 3 Central processing unit, graphics processing unit (GPU), and memory usage of native Linux for Tegra (L4T) and L4T on EARTH during convolutional neural network workloads run; in GPU cases, the median value was used to simplify the fluctuating usage line (a) resource usage of native L4T, (b) resource usage of L4T on EARTH

isation overhead of EARTH does not have a significant negative effect on the overall system performance.

While running the CNN workloads with trail navigation and position recognition algorithms, the resource usage of CPU, GPU, and memory in the native L4T and EARTH-based L4T environments were checked to evaluate the performance overhead of EARTH. Both algorithms used either six or 12 layers, and four combinations for the CNN workload cases were used. In this test, to measure the overhead of EARTH for the CNN workloads, hardware resources were allocated to a guest L4T, similar to the native L4T. Thus, four CPU cores and 7.75 GB memory were allocated to the guest L4T, except for in the regions used by the test features of EARTH. To check the usage of memory, CPU, and GPU, the NVIDIA utility tegrastats was used, and each item was measured at 10 Hz.

Figure 3 shows the usages of CPU, GPU, and memory when performing the CNN workloads in native L4T and L4T on EARTH. It was found that the usages of the two cases were extremely similar in all combinations of layers. Once the CNN workloads were executed, the resource usages of GPU and memory rapidly increased for 35 s, as the neural network expanded, after which the usages stabilised. Table 1 shows the average and standard deviation of each resource usage in the peak section of the CNN workloads between 35 and 100 s. CPU and GPU resource usages in the EARTH environment were slightly lower than those in the native environment, primarily due to the overhead of hardware resource virtualisation. Another reason is the difference in allocated physical memory size. The memory resource usage of L4T on EARTH was approximately 1.7% higher on average, but the amount of available data per unit time was relatively small. This is why the CPU and GPU usages were relatively low.

Even if each resource generates only a small overhead, it is the combined overhead of all resources that affects the workload execution time. Table 2 shows the neural network pass-through time in relation

Table 1. Average hardware resource usage of native Linux for Tegra (L4T) and L4T on EARTH during the peak period of convolutional neural network workloads

Mission controller	TN+PR layer	Central processing unit usage		Graphics processing unit usage		Memory usage		
		AVG	STD	AVG	STD	AVG	STD	Size
Native L4T	6+6	40.32	5.77	88.48	27.26	79.64	0.02	6252
	6+12	40.93	5.62	87.83	28.14	79.27	0.02	6223
	12+6	39.28	5.71	87.63	28.16	79.34	0.02	6228
	12+12	41.26	5.51	88.40	27.02	77.41	0.02	6075
L4T on EARTH	6+6	40.10	6.43	87.61	27.85	81.85	0.05	5999
	6+12	39.36	6.39	86.65	29.27	81.43	0.13	5975
	12+6	39.30	6.31	87.46	28.10	79.77	0.05	5854
	12+12	39.29	6.21	87.31	27.96	79.46	0.02	5831

Notes: Unit: percent and megabyte.

TN: trail navigation; PR: position recognition; AVG: average; STD: standard deviation.

Table 2. Average pass-through time per layer under CNN workloads of native L4T and L4T on EARTH

TN+PR layer	Native L4T		L4T on EARTH	
	TN	PR	TN	PR
6+6	10,381	10,089	11,190	10,576
6+12	10,237	10,724	11,191	11,774
12+6	10,871	10,328	11,515	11,082
12+12	10,599	10,413	11,264	10,880

Notes: Unit: microsecond.  
TN: trail navigation; PR: position recognition.

Table 3. Micro air vehicle robot operating system (MAVROS) communication frequency on PIXHAWK2 and PX4 on EARTH

MAVROS topic	PX4 on EARTH		Pixhawk2	
	W/ MC	W/o MC	W/ MC	W/o MC
Mavros/altitude	10	10	10	10
Mavros/imu/data_raw	46	46	50	50
Mavros/imu/mag	46	46	50	50
Mavros/local_position/pose	28	28	30	30
Mavros/local_position/velocity	28	28	30	30
Mavros/state	1	1	1	1
Mavros/target_actuator_control	10	10	10	10
Mavros/setpoint_position/local	7.5	7.5	8	8
Mavros/timesync_status	10	10	10	10

Notes: Unit: hertz.

to the number of CNN layers with or without EARTH. Regardless of the combination of layers, it took a similar time for all resources to process sensing data, though there was an average delay of 6.5%. This delay can be understood as a 6.5% average decrease, compared with the native MC, in terms of response speed for processing the sensing data of the EARTH-based UAV. However, this occurs merely on a microsecond level. Thus, given the requirements and the overall performance of the target system, it has little impact and is acceptable.

**Measuring overhead of data communication:** The results of this test were used to analyse the effect of EARTH's FC–MC communication frequency and workloads on IVC function. The communication frequency of Pixhawk2—a commercial FC board—was measured to compare and analyse the communication frequency of PX4, the target FC. The version of PX4 on EARTH is 1.5.4, while the version of PX4 on Pixhawk 2 is 1.8.2. The MAVROS topic frequency was measured for each case according to the presence or absence of the CNN workload operation on the MC. According to the results displayed in Table 3, the MAVROS topic frequency of PX2 installed on the MC was either equivalent to the MAVROS frequency of Pixhawk2 or degraded by 8% or less, with the degradation being adjustable. The consistent communication frequency of the MAVROS topic, regardless of whether the MC operated, indicates that the CNN workloads operating on the MC do not affect FC–MC IVC. Additionally, this consistency demonstrates that the application of IVC for the cooperation of the MC and the FC does not lead to negative impacts caused by the CNN missions, such as latency.

**Field test of FC–MC cooperated autonomous flight:** The final test results demonstrate that the guest FC on EARTH can facilitate UAV flight by adjusting the flight route through trail navigation and position recognition with the MC of other guests. Such cooperation sig-

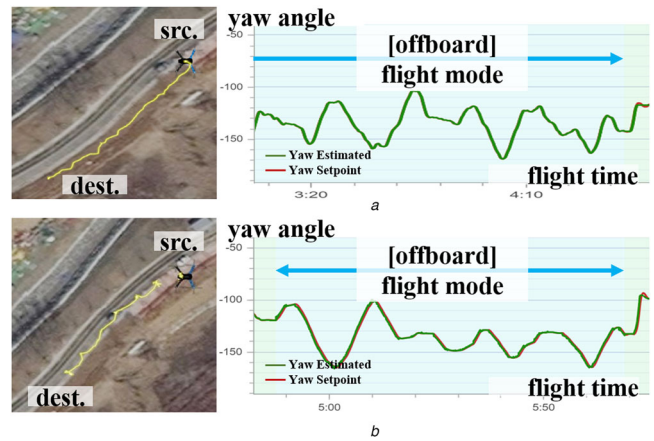


Fig 4 Flight route and yaw graph of two assembled autonomous UAVs (a) field test result from Pixhawk2 connected to native L4T; (b) field test result from flight system with PX4 and L4T on EARTH

nifies that the two isolated FC and MC systems can perform their mission independently while also sharing the results of the mission in real-time through IVC. In other words, the results demonstrate the proper functionality and applicability of the autonomous flight system.

The assembled UAVs flew a one-way distance of approximately 70 m at the speed of human walking. Flight data was recorded using uLog—a log file officially supported by PX4—to visualise the flight route adjustments of the two flight systems, the consolidated FC–MC system on EARTH and a commercial Pixhawk2 connected to MC. The created logs included data transmitted as uORB messages and flight motion. In addition, PX4 provides a tool to analyse the log at <https://logs.px4.io>, and the route maps and graphs in Figures 4(a) and (b) were obtained via this online tool. In the satellite maps, the black quadcopter icon indicates the location of the source, and the yellow line shows the flight route to the destination. Neither flight route was a straight line between the source and the destination. As shown on the maps, the UAV flew within the trail and adjusted its route so as not to deviate from the trail when it reached the trail boundaries.

While travelling to the destination—according to the results of trail navigation and position recognition on the MC—if the UAV reached the edge of the trail, it adjusted its yaw to travel as trained. This is why the flight route was not straight. This adjustment can also be clearly seen in the yaw setpoint graphs. The area of the offboard flight mode is the period in which the FC guest adjusted the route by the MC guest and where yaw setpoint values were not consistent. Such fluctuating yaw setpoint values indicate that the MC sent the setpoint message to the FC to adjust the forwarding angle to prevent deviation. If there were no MC, the yaw values would remain consistent. Furthermore, though the two systems have a different FC and hardware structure, the fluctuations of the tendency and degrees on the yaw graphs are similar. It represents the proper functionality of the demonstrated autonomous flight system.

**Conclusion:** This work designed and implemented an autonomous trail-following flight system that safely consolidates an MC and FC on the latest single high-performance commercial hardware platform as opposed to separate boards. The flight system, which is implemented based on EARTH, demonstrated its functionality using the CNN-based trail-following application. When performing the CNN workloads in the implemented system, the degradation of resource usage in the Linux guest was 1.7% or lower, on average, compared with the usage and the pass-through time of the native Linux. The pass-through delay in each layer of the neural network was, on average, 6.5% or less. In addition, real-time flight control was achieved in an environment in which the speed of the IVC between the FC and the MC was satisfied, and the workloads on the MC did not affect the data communication frequency. The flight test results from the two different flight systems demonstrated cooperation between the FC and the MC, both practically and visually, based

on the CNN workload and represented the proper functionality of the demonstrated system.

**Acknowledgements:** This work was supported by the Institute of Information and Communications Technology Planning and Evaluation (IITP) Grant (No.2017-0-00067, Development of ICT Core Technologies for Safe Unmanned Vehicles) and the National Research Foundation of Korea (NRF) Grant (No.NRF-2020M3C1C2A01080819, DNA+ Drone Technology Development Program) that are funded by the Ministry of Science and ICT (MSIT), Republic of Korea.

© 2021 The Authors. *Electronics Letters* published by John Wiley & Sons Ltd on behalf of The Institution of Engineering and Technology

This is an open access article under the terms of the Creative Commons Attribution License, which permits use, distribution and reproduction in any medium, provided the original work is properly cited.

Received: 13 November 2020 Accepted: 12 January 2021

doi: 10.1049/ell2.12099

## References

- 1 Nie, J., Luo, T., Li, H.: Automatic hotspots detection based on UAV infrared images for large-scale PV plant. *Electron. Lett.* **56**(19), 993–995 (2020)
- 2 Han, Y., et al.: Boundary-aware vehicle tracking upon UAV. *Electron. Lett.* **56**(17), 873–876 (2020)
- 3 Back, S., et al.: Autonomous UAV trail navigation with obstacle avoidance using deep neural networks. *J. Intell. Rob. Syst.* **100**, 1195–1211 (2020)
- 4 Smolyanskiy, N., et al.: Toward low-flying autonomous MAV trail navigation using deep neural networks for environmental awareness. In: IROS IEEE/RSJ 2017, Vancouver, BC, Canada, pp. 4241–4247 (2017)
- 5 Basso, M., de Freitas, E.P.: A UAV guidance system using crop row detection and line follower algorithms. *J. Intell. Rob. Syst.* **97**, 605–621 (2020)
- 6 Gaska, T., Werner, B., Flagg, D.: Applying virtualization to avionics systems - the integration challenges. In: DASK IEEE/AIAA 29th, Salt Lake City, UT, USA, pp. 5.E.1-1–5.E.1-19 (2010)
- 7 Yoon, S., et al.: Timed model-based formal analysis of a scheduler of Qplus-AIR, an ARINC-653 Compliance RTOS. *IEICE Trans. Inf. Syst.* **E100.D**(10), 2644–2647 (2017)