Unmanned Systems, Vol. 1, No. 2 (2013) 211–245

DOI: 10.1142/S2301385013400049

UNMANNED SYSTEMS
http://www.worldscientific.com/worldscinet/us

Development of an Unmanned Coaxial Rotorcraft for the DARPA UAVForge Challenge

Feng Lin*, Kevin Z. Y. Ang†, Fei Wang‡, Ben M. Chen§, Tong H. Lee¶, Beiqing Yang, Miaobo Dong, Xiangxu Dong, Jinqiang Cui, Swee King Phang, Biao Wang, Delin Luo, Kemao Peng, Guowei Cai, Shiyu Zhao, Mingfeng Yin, Kun Li

UAV Research Group, the National University of Singapore, Singapore 117576

In this paper, we present a comprehensive design for a fully functional unmanned rotorcraft system: GremLion. GremLion is a new small-scale unmanned aerial vehicle (UAV) concept using two contra-rotating rotors and one cyclic swash-plate. It can fit within a rucksack and be easily carried by a single person. GremLion is developed with all necessary avionics and a ground control station. It has been employed to participate in the 2012 UAVForge competition. The proposed design of GremLion consists of hardware construction, software development, dynamics modeling and flight control design, as well as mission algorithm investigation. A novel computer-aided technique is presented to optimize the hardware construction of GremLion to realize robust and efficient flight behavior. Based on the above hardware platform, a real-time flight control software and a ground control station (GCS) software have been developed to achieve the onboard processing capability and the ground monitoring capability respectively. A GremLion mathematical model has been derived for hover and near hover flight conditions and identified from experimental data collected in flight tests. We have combined H_∞ technique, a robust and perfect tracking (RPT) approach, and custom-defined flight scheduling to design a comprehensive nonlinear flight control law for GremLion and successfully realized the automatic control which includes take-off, hovering, and a variety of essential flight motions. In addition, advanced mission algorithms have been presented in the paper, including obstacle detection and avoidance, as well as target following. Both ground and flight experiments of the complete system have been conducted including autonomous hovering, waypoint flight, etc. The test results have been presented in this paper to verify the proposed design methodology.

Keywords: Unmanned coaxial rotorcraft; unmanned system design; modeling; flight control; robust control; vision-based obstacle detection and avoidance; target tracking.

US

1. Introduction

In the last two decades, unmanned aerial vehicle (UAV) systems aroused great interests worldwide[1–3] in civil and military applications, e.g., aerial surveillance, reconnaissance, and inspection.[4,5] Compared to their manned counterparts, UAVs have the advantages of low cost, no risk of losing human pilot, and high confidence in mission success. In the next decade, it can be foreseen that the demand for UAVs will continue to grow. Currently, there is a trend of employing UAVs for civil applications,[6] though most of mutual UAV systems are still defense-related and the main functions are driven by future military applications. While both military forces and civil domains are embracing sophisticated and miniature UAV systems capable of vertical take-off and landing, beyond line-of-sight (LOS) observations, autonomous obstacle avoidance, as well as low-altitude flight in cluttered and complex environments, e.g., urban environment.[7–10] Such capability is extremely important for the commander to broaden battlefield situational awareness with less risk.[11] Of course, these capabilities will also provide researchers, rescuers, and other users a new and valuable tool in their working areas.[12–14]

To boost the progress in UAV development in such a direction, in year 2012, the Defense Advanced Research

Received 2 March 2013; Revised 17 July 2013; Accepted 18 July 2013; Published 9 September 2013. This paper was recommended for publication in its revised form by Guest Editor, Stephen D. Prior.
Email Addresses: *tsllinf@nus.edu.sg, †kevinang@nus.edu.sg, ‡wangfei@nus.edu.sg, §bmchen@nus.edu.sg, ¶eleleeth@nus.edu.sg

Projects Agency (DARPA) and Space and Naval Warfare Systems Center Atlantic (SSC Atlantic) collaboratively launched an initiative called the UAVForge competition to design, build and manufacture advanced micro unmanned air vehicle systems. To participate in the UAVForge competition, the NUS UAV research group started to design and develop an unmanned coaxial rotorcraft: GremLion, together with necessary avionics and a ground control station. GremLion is a new small-scale unmanned aerial vehicle (UAV) concept using two contra-rotating rotors and one cyclic swash-plate. It can fit within a rucksack and be carried easily by a single person.

In this paper, we propose a systematic design methodology to develop the GremLion system efficiently. The complete GremLion system consists of an onboard embedded hardware system, a real-time flight control software system, a vision processing software system, as well as mission-based flight control and vision algorithms. More specifically, the onboard embedded hardware system is designed to fulfill flight control, navigation, onboard real-time image processing and tracking control requirements by using commercial off-the-shelf (COTS) products, such as Gumstix embedded modules. The hardware construction of the system is optimized using a novel computer-aided technique. A comprehensive design methodology is proposed for the design of the hardware system.

Based on the custom developed hardware platform, a real-time flight control software is developed, which is running on the real-time operating system: QNX. As an embedded microkernal operating system, QNX requires less computation resources and can be tailored to suit most embedded systems. Under the QNX operating system, multiple-thread framework is implemented to coordinate multiple tasks, such as data acquisition, flight control, communication, and servo control.

Throughout the overall development of small-scale UAVs, deriving the linear and nonlinear dynamics models that could accurately capture the aerodynamics have continuously been a challenging issue due to their inherent instability and sensitive aerodynamics. The dynamics modeling work starts after the construction of the first prototype GremLion. The frequency-domain system identification toolkit CIFER, which is suitable for rotorcraft identification, has been used to obtain the linearized model in hover and near hover flight conditions.

With the identified dynamics models in hand, we have carried out the control law design and implementation using advanced nonlinear control techniques. Specifically, we have combined (1) H_∞ technique; (2) a robust and perfect tracking (RPT) approach; and (3) custom-defined flight scheduling to design a comprehensive nonlinear flight control law for GremLion and successfully realized the automatic control that includes take-off, hovering, and a variety of essential flight motions.

In order to detect and avoid obstacles during the flight of GremLion, the vision-based obstacle detection and avoidance approaches have been developed. The video captured by the onboard camera is used to extract the depth and angular information of obstacles. Such information is then used as a guidance law for obstacle avoidance of GremLion.

An advanced vision algorithm is also proposed and implemented to realize moving target following, which utilizes a robust feature detection and tracking scheme. As the target mentioned in the above statement is that of a vehicle traveling at about 15–30 MPH, it is necessary to first mark the target that is required to be tracked. This is achieved by using the mono-camera sensor to capture an image of the target vehicle and manually drawing a rectangular target box around the required target. The selected target box is extracted from the image automatically, and the target tracking in image sequence is implemented using the Continuously Adaptive Mean-SHIFT (CAMSHIFT) method. The UAV is then controlled to maintain the target in the center region of the image. The proposed target tracking algorithm has been implemented in a Gumstix-based embedded system, and the results obtained show that the proposed system is very robust and efficient.

The remainder of this paper is organized as follows: we give a brief introduction of the UAVForge competition first in Sec. 2. The design and implementation of hardware and software of GremLion is presented in Secs. 3–5. Whereas the modeling, control and navigation of GremLion are investigated in Secs. 6 and 7, respectively. A systematic design and implementation of vision-based obstacle detection and avoidance approaches are given in Sec. 8 with simulation results presented. The vision-based target tracking algorithms and test results are detailed in Sec. 9. Finally, we draw some conclusion in Sec. 10.

2. UAVForge Competition

In the UAVForge competition, the mission of each team is to outfit a fictional Task Force with an unmanned remotely operated micro air vehicle system. The entire air vehicle system must fit within a rucksack and a single person traveling by foot must be able to carry and operate the vehicle without assistance.

The job of the Task Force is to conduct observations of suspicious activities occurring within the vicinity of two nondescript buildings in an urban area. Due to the security in the region, all operations must be conducted beyond line of sight so as not to compromise their presence. If the UAV system is detected, the mission will be jeopardized. The total observation time required may be up to three hours of pictures and/or video to document the surveyed area. Once

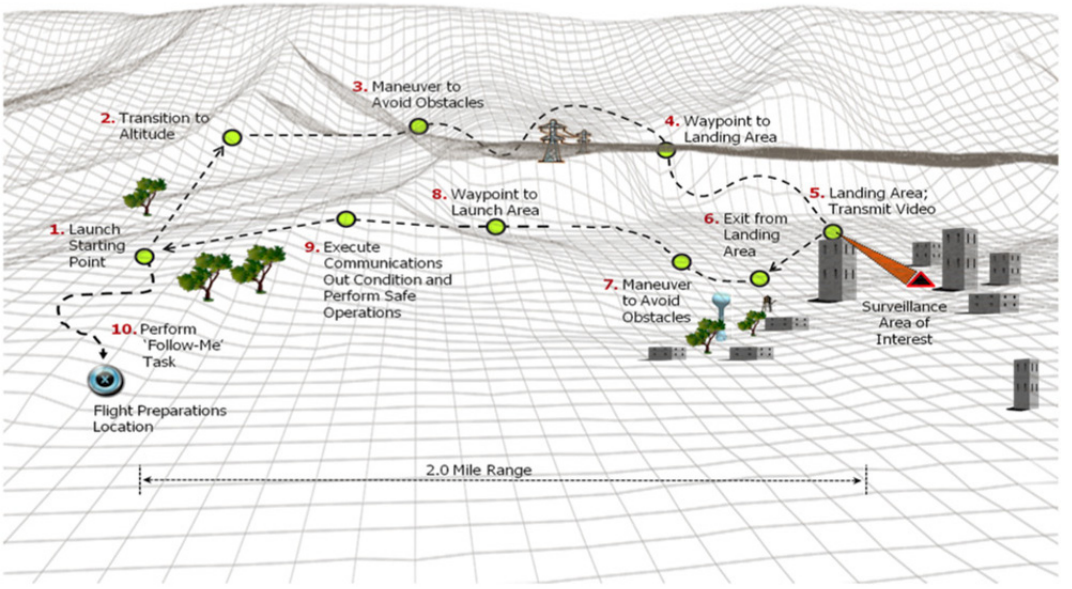

Fig. 1. UAVForge competition course.

key observations have been made, the team must quickly retreat to their designated rendezvous location.

These capabilities will be demonstrated and tested in a fly-off that is representative of the mission scenario. Figure 1 outlines the overall course of the competition, where there are six basic requirements need to be accomplished:

(i) ***Take-off(s)***. Take-offs will be from a common starting location, with headings dependent on weather. After take-off, the air vehicle needs to fly safely below 1000 feet.

(ii) ***Navigation***. Method of point-to-point navigation is to be determined by the designer(s) of the vehicle system. The air vehicle must stay within the defined flight corridors, operate within the assigned airspace, and avoid predefined no-fly zones. Therefore, the vehicle must operate in a safe, pre-defined altitude window from 5 feet to 1000 feet above the ground and must remain within 500 feet of the flight corridor.

(iii) ***Obstacle Avoidance***. During approach into the observation area, the vehicle will be required to maneuver to avoid obstacles. Typical obstacles include negotiating around stationary objects like buildings, water towers, and trees, though dynamic obstacles may be introduced. The landing area is similar to rooftop structures with HVAC equipment, communications gear, satellite dishes, and poles.

(iv) ***Total Mission Time***. Total mission time is defined by the declaration of mission start with permission to turn on the vehicle systems for flight until the vehicle has landed and the vehicle systems are shut off.

(v) ***Observation***. Once the vehicle has flown to the pre-defined search area, the vehicle needs to identify a vantage point from which to conduct observations. This task can be accomplished by any means such as landing, adhering, hanging, and/or hovering above or under a physical structure. The system must provide clear information based on real-time transmission of video.

(vi) ***Mission Completion***. Upon mission completion, all landings are required to be at a designated location different from the starting location.

2.1. *UAVForge Milestone 1: Concept video*

During the selection phase prior to the UAVForge Competition flyoff, competing teams were asked to submit videos (via YouTube) to advertize their skills, design sketches, vehicle components, algorithm behaviors, etc. A complete air vehicle system, structure or full vehicle system performance was not required at this stage. The top rated submissions defined by the crowd were encouraged to compete in the next phase. Team GremLion submitted a concept video which depicted a soldier deploying a UAV from his backpack, then it flew to the area of interest and perched on the roof to carry out surveillance. Figure 2 shows a snapshot of the concept video in YouTube. The video roused great interest in the general public and resulted in Team GremLion achieving the second highest vote score of 3.45. Table 1 summarizes the scoring for the first Milestone.

Fig. 2. GremLion UAV concept video.

Fig. 3. GremLion UAV proof of flight video.

Table 1. Concept video results.

Team	Score
MAAV	4.128
GremLion	3.450
Cooper UDT	3.339
IcarusLabs	3.259
Extractor X	3.182
VoRPaL	3.156
AeroQuad.com	3.137
Electric UAV	2.863
ATMOS	2.853
SlightlyNybbled	2.844

Table 2. Proof of flight results.

Team	Score
GremLion	3.148
DHAKSHA	2.944
AeroQuad.com	2.848
ATMOS	2.722
WIDrone	2.700
X-MUAS	2.688
EvaForge	2.667
HALO	2.611
Extractor X	2.550
SwiftSight	2.500

2.2. *UAVForge Milestone 2: Proof of flight video*

Competing teams were then asked to demonstrate the early flight behaviors and capabilities of their air vehicle system. We may also showcase other innovative and creative aspects of our air vehicle system design and performance through the uploading of another video through YouTube as shown in Fig. 3.

Team GremLion produced a video that showcased the capabilities of the GremLion UAV such as autonomous hovering, long ranged video transmission and target tracking. The video "Wow-ed" the public and thus resulted in Team GremLion obtaining the top score for the second Milestone as shown in Table 2.

2.3. *UAVForge Milestone 3: Live video demo*

Prior to the UAVForge competition, the competing teams were required to participate in a live video demonstration where DARPA and SSC Atlantic will ask teams to show their vehicles advanced capabilities. Real-time performance requests may include take-off from a set time/point as well as maneuvers. The videos will be recorded by DARPA and SSC Atlantic and made available for the crowd to view and vote on once all teams have flown. Additionally, teams must provide system design information for evaluation using a template provided by the advising manufacturer. A manufacturing assessment will be factored into the determination of the finalists invited to participate in the fly-off competition. This is considered an elimination round where only the top 10 entries will be invited to the competition fly-off.

During the "Live Demo", GremLion performed all the required tasks set by DARPA flawlessly and was chosen to be one of the top 10 entries to participate in the UAVForge Competition fly-off. Figure 4 shows the video of our live demo.

2.4. *UAVForge fly-off competition*

The UAVForge Competition was held at Fort Stewart, Georgia, USA around the middle of May 2012. The take-off

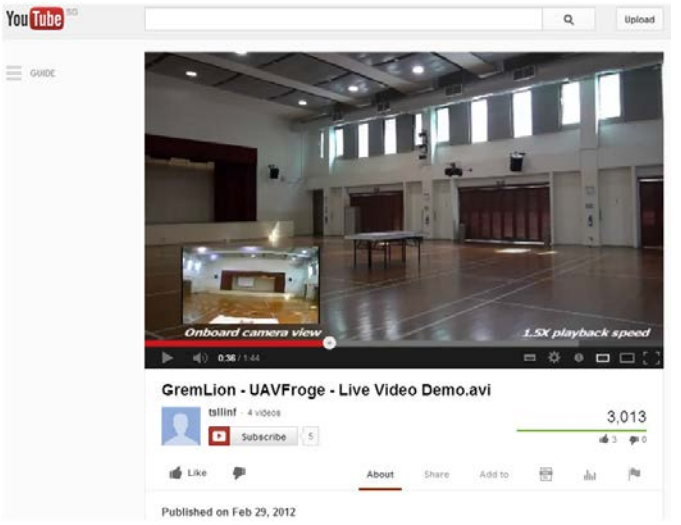

Fig. 4. GremLion UAV Live demo video.

area was an open grass patch near the size of a football field with very tall trees surrounding it. This required GremLion to first fly over these trees before embarking on its mission to fly to its first waypoint to accomplish its surveillance mission. Figure 5 shows a photo of our team preparing for the fly-off during the competition.

Vehicle performance during the competition fly-off will be based on a point scoring strategy, with points assigned for total system performance, completion of the competition course, and innovation in the vehicles design and manufacturability. A total of 200 points can be earned. Each baseline objective is pass-fail, and all of the baseline objectives must be completed in order to be eligible to earn points for advanced behaviors. If the team passes all baseline objectives on the first try, they will be awarded 30 points. The final fly-off results are shown in Table 3.

Fig. 5. UAVForge competition grounds.

Table 3. Fly-off results.

Team	Build score (30 pts)	Final score (60 pts)
AeroQuad	25	39.1
ATMOS	24	37.3
DHAKSHA	16	31.5
Extractor X	23	32.0
GremLion	14	19.2
HALO	27	47.7
NAVYEOD	25	36.5
Phase Analytic	25	30.5
SwiftSight	23	37.3

During the competition fly-off, GremLion managed to perform fully autonomous waypoint flight but due to a mechanical failure which resulted from the overheating of the ESC, GremLion crashed into the surrounding forest and was deemed unrepairable for the rest of the competition. All the teams were unable to fulfill the baseline requirements and were therefore not allowed to showcase their advanced behaviors on their UAVs. Also, there was a score of up to 30 points given to each team if they were to furnish their design information and parts list to the organizers. However, as the GremLion UAV is a proprietary platform for the National University of Singapore UAV Research Group, we were unable to give any detailed information to the UAVForge organizers as this would infringe on our Intellectual Property (IP) rights. This resulted in Team GremLion obtaining a low final score. However, GremLion is the only innovative and fully-customized platform in the competition, which has great potential in the future such as seamless indoor and outdoor operations and ground movement capabilities which the NUS UAV Research Group will continue to develop.

3. Hardware Configuration of the UAV System

A customized coaxial helicopter is utilized as the baseline of GremLion development to be upgraded upon in response to the UAVForge competition. The working principle of the GremLion system is shown in Fig. 6, composed of the following four parts: (1) a coaxial helicopter; (2) an avionic system; (3) a manual control system; and (4) a ground supporting system.

3.1. *The coaxial helicopter*

GremLion, shown in Fig. 7, features a coaxial design driven by two contra-rotating rotors that can compensate the torque due to aerodynamic drag. Such a design allows for a

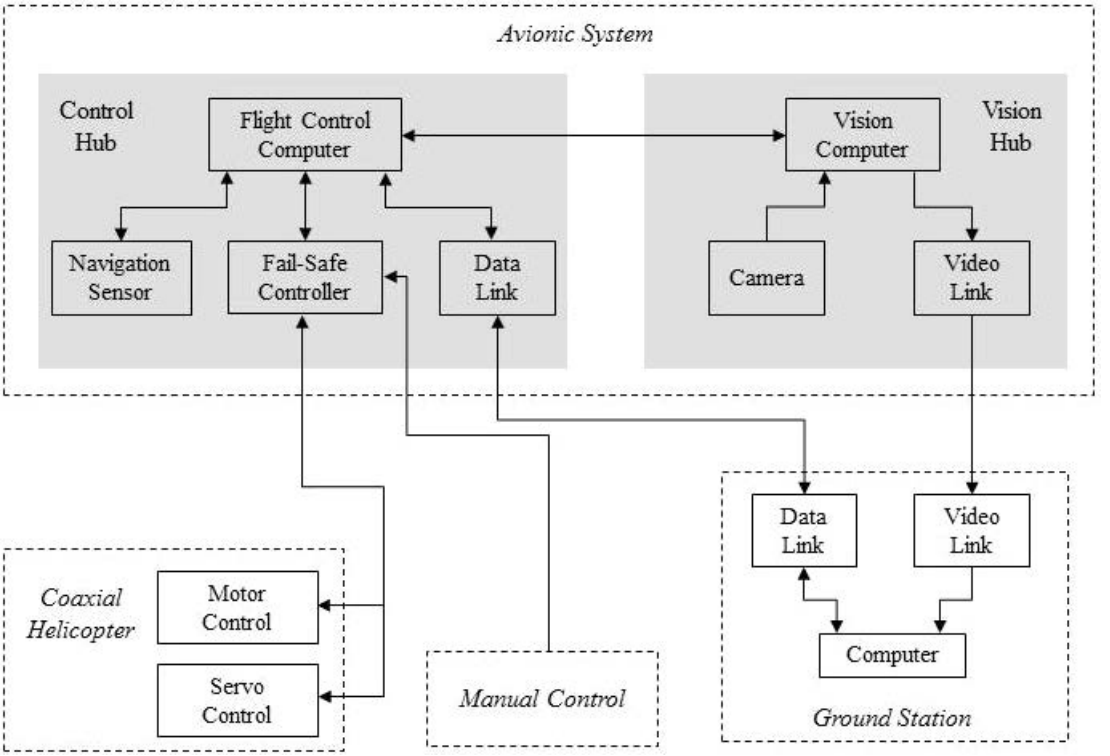

Fig. 6. Hardware configuration of GremLion.

more stable, more maneuverable, quieter and safer helicopter due to the inclusion of a coaxial main rotor and exclusion of a tail rotor which results in a smaller footprint. Coaxial helicopters also provide a better thrust-to-weight ratio than traditional helicopters, produce greater lift and are also much more efficient.[15] Therefore, this platform is suited for the size requirement of the competition, which is required to be kept in a rucksack (see Fig. 8). The key specifications of the platform are listed in Table 4.

To reduce complexity of the actuation system of conventional coaxial design, a novel actuation system has been

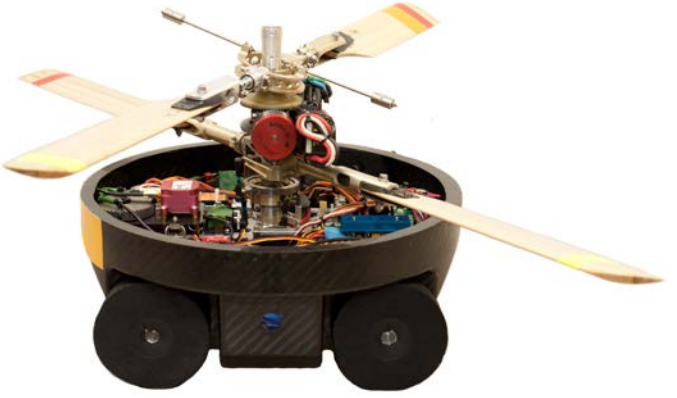

Fig. 7. The GremLion UAV.

Fig. 8. The GremLion in a rucksack.

Table 4. Specifications of the GremLion UAV.

Specifications	GremLion
Upper rotor span	798 mm
Lower rotor span	895 mm
Upper rotor speed	1900 rpm
Lower rotor speed	1700 rpm
No-load weight	2.4 kg
Maximum take-off weight	5.1 kg
Power source	LiPo battery
Flight endurance	15 min

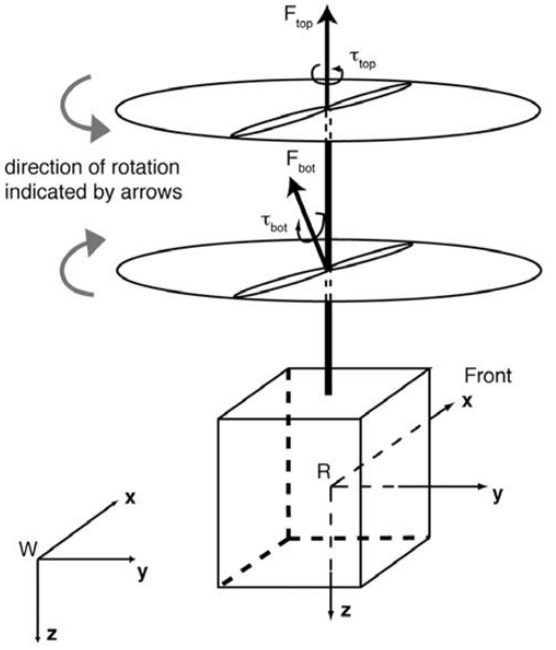

Fig. 9. The coaxial structure.

employed in GremLion, which is shown in Figs. 9 and 10. The operating principle of this actuation system is presented as follows:

(i) Unlike conventional single-rotor helicopters that utilize the collective pitch of their rotor blades to adjust the lift force, GremLion's rotor pitches are fixed and the thrust variation is accomplished by changing the rotor spinning speed simultaneously. Hence, the vertical motion is controlled by the pulse width modulation (PWM) signals fed to the motors attached to the top and bottom rotors. As illustrated in Fig. 10, the throttle input δ_{col} can adjust the speed of the upper rotor and the lower rotor simultaneously.

(ii) The helicopter yaw motion (head turning) is produced by the difference of spinning speed between the top and bottom rotors. When one rotor spins, other than the lifting force it creates, it also generates a rotational torque on the fuselage of the helicopter in the direction opposite to the rotor spinning direction. Note that the top and bottom rotors always spin in opposite directions so that their torques cancel each other. In order to make the heading of the helicopter stable, a hardware rate gyro is installed to finely adjust the spinning speed of the two rotors so that yaw dynamics becomes much more damped. As shown in Fig. 10, the rudder input δ_{ped} for control of the yaw of the vehicle differentiates the spinning speeds of the two rotors.

(iii) In order to have lateral and longitudinal motions, the bottom rotor cyclic pitch is actively controlled by three servos. This is done through a swash plate mechanism which acts as a link between the servos and the bottom rotor cyclic pitch. As shown in Fig. 10, the aileron input: δ_{lat} controls the leftward and rightward tilting motion of the swash plate. Such a movement changes the cyclic pitch angle of the lower rotor blades and results in both a rolling motion and lateral translation. The elevator input δ_{lon} is responsible for the forward and backward tilting motion of the swash plate. This

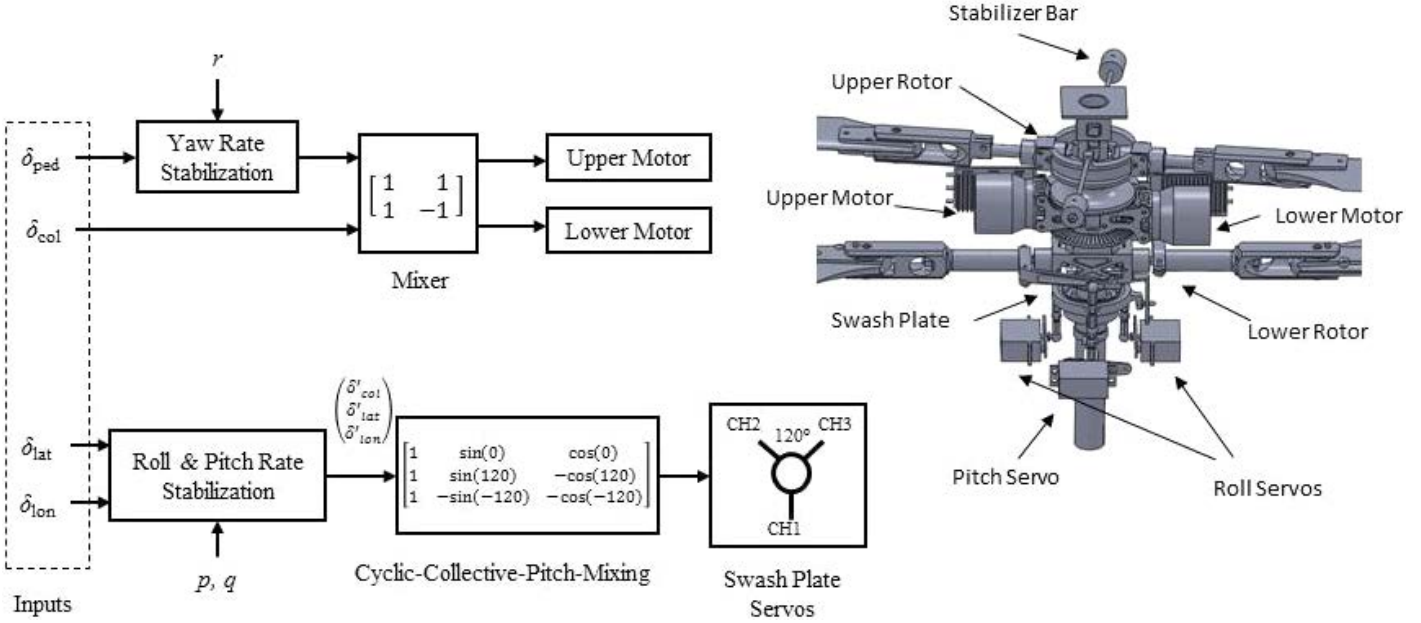

Fig. 10. Operating principle of GremLion.

tilting also changes the cyclic pitch angle of the lower rotor blades but results in pitching motion and longitudinal translation. The aileron and elevator inputs cooperate with the roll and pitch rate feedback controller to stabilize the angular rate of roll and pitch motions. Such a rate feedback controller is used to allow a human pilot to control the over sensitive dynamics of the bare platform.

(iv) The upper rotor is equipped with a stabilizer bar to further increase the stability of the vehicle. The top rotor is not actively linked to any servos, but it is passively controlled via a mechanical stabilizer bar. With the presence of this stabilizer bar, the top rotor always has a cyclic pitch (with respect to the body frame) countering the inclination of the fuselage at any single moment. This slows down the whole platform's response to the rapid changes in the cyclic pitch of the bottom rotor.[16] In this way, the helicopter stability is increased but the maneuverability is decreased.

3.2. *Avionic system*

To realize fully autonomous flight, an avionic system has been developed as illustrated in Fig. 11. All the components of the avionic system are the most suitable commercial off-the-shelf (COTS) products to date. The details and usage of each components are presented in the following parts.

3.2.1. *Navigation sensors*

IG-500 N (see Fig. 11) is one of the world's smallest GPS enhanced attitude and heading reference system (AHRS) embedded with an extended Kalman filter (EKF). It includes

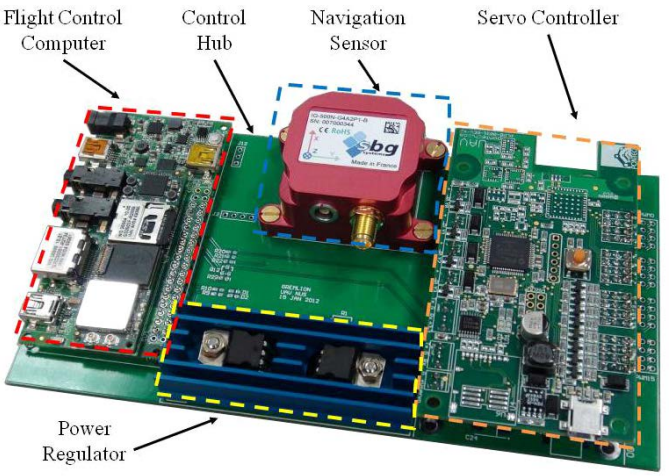

Fig. 11. Control Hub with all the modules.

Table 5. Main specifications of IG-500 N.

Specifications	IG-500 N
Attitude range	360° in three axes
Attitude accuracy	±0.5° (pitch, roll), ±1° (heading)
Accelerometer range	±5 g
Gyroscope range	±300°
Magnetometer range	±1.2 Gauss
GPS accuracy in CEP	2.5 m (horizontal), 5 m (vertical)
Output rate (Hz)	{1, 25, 50, 75, 100} selectable
Dimensions	36 × 49 × 22 mm
Weight	46 g (with aluminum enclosure)
Power consumption	550 mW @ 5.0 V

a MEMS-based inertial measurement unit (IMU), a GPS receiver and a pressure sensor. It is able to provide precise and drift-free 3D orientation and position even during aggressive maneuvers, updated at 100 Hz. Its key specifications are summarized in Table 5.

3.2.2. *Onboard computers*

The onboard processor is the brain of the whole avionic system. It collects measurement data from various sensors, performs sensor filtering and fusion, executes flight control law, and outputs control signals to carry out the desired control actions. In addition, it is also responsible for communicating with the ground control station (GCS) for real-time inspection and command issuing, as well as logging in-flight data for post-flight analysis. Hence, selecting suitable COTS processors is crucial to ensure successful implementation of the UAV system. We have chosen two Gumstix Overo Fire embedded computers for flight control and navigation purposes respectively (see Fig. 11). This embedded computer system has a main processor running at 720 MHz and a DSP coprocessor. The main processor is an OMAP3530 ARM processor from Texas Instruments and it is one of the fastest low-power embedded processor as of writing. Moreover, it has Wi-Fi functionality despite its tiny size and light weight. In order to improve its real-time performance, the original Linux operating system provided by the manufacturer is replaced by the QNX Neutrino real-time operating system (RTOS). Custom-built autopilot software developed by the NUS UAV Research Group is used to realize the desired autonomous flight control.

3.2.3. *Servo controller*

An 8-channel PWM servo controller: UAV100 (see Fig. 11), is adopted to allow servo outputs to be controlled by an onboard computer or control command from the radio

control (RC) receiver, depending on the state of a switching signal from the RC transmitter. While GremLion maneuvers autonomously in the air, it is desirable to have a failsafe feature to allow the ground pilot to take over control during emergencies. This servo controller provides the pilot with the option to take over the control of an UAV at the flick of a switch of the transmitter to prevent a catastrophic incident from a malfunction in the flight computer. This function gives us the ability to test flight control software early without fear of damaging a test vehicle.

3.2.4. *Communication*

The communication unit includes a pair of Microhard wireless data transceivers. This pair of transceivers establish communication between the onboard system and the ground station. They are configured to operate in a point-to-point mode and work in 2.400 to 2.4835 GHz. The transceiver used in the onboard system is set as a point-to-point slave, and connected to the flight control computer board. The transceiver in ground station is set as a point-to-point master, and connected to a laptop.

3.2.5. *Control hub*

Control Hub, shown in Fig. 11, is a mother board designed to host subsystems for control purposes. It has the following features:

(i) ***Module connection.*** Aforementioned modules, such as the Gumstix board, the IG-500 N, and the UAV100 servo control board, are installed on the slots on Control Hub and connected to the onboard power regulator and other essential components through Control Hub. Besides the mounting slots, extra mounting holes on Control Hub have been used to lock the installed modules to reduce the vibration and shock in flight and landing. Manual wire wrap has been minimized to improve reliability and quality of the system.

(ii) ***Level shifter.*** An onboard level shifter: MAX3232 has been built in Control Hub to convert the serial signal from RS-232 level to TTL level, which has been used to make the output of IG-500 N compatible with the Gumstix board.

(iii) ***Power regulation.*** To power up all the avionics, linear regulators are built in Control Hub to convert a power input from a 3-cell LiPo battery into a 5 V output with 10 A capacity and a 2–8 V adjustable output with 10 A capacity. The 5 V output powers the Gumstix board, the rate gyro and the electronic mixer. The adjustable output powers the servos.

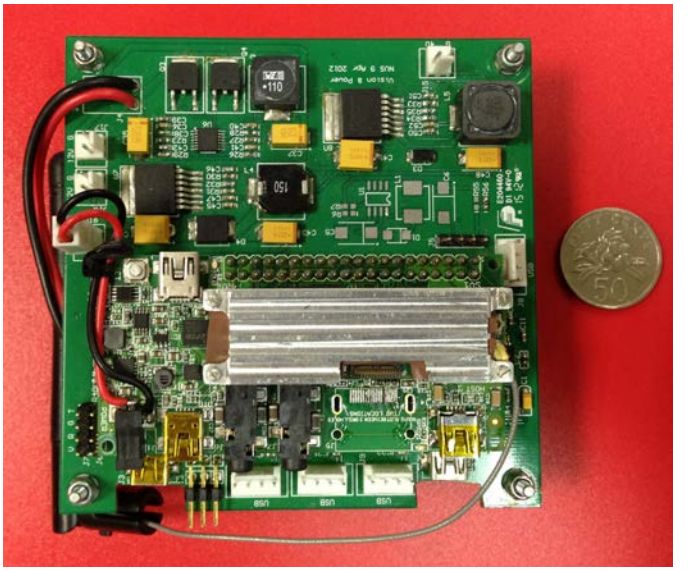

Fig. 12. The vision hub.

3.2.6. *Vision hub*

Vision Hub is another mother board designed for hosting vision subsystem and expending USB ports, which has the following features:

(i) ***Gumstix module plug.*** The plug is designed for a vision Gumstix board that is utilized for implementing mission algorithms. A heat sink is installed on the top of the vision Gumstix board to prevent overheating due to the intensive vision processing.

(ii) ***USB hub 1.*** Since the original Gumstix expansion board cannot provide sufficient interfaces for onboard sensors and inter-computer communication, we expand the USB port of the control Gumstix to four independent USB ports by using a GL850 chip. The expanded ports have been converted to universal asynchronous receiver/transmitter (UART) ports ($2 \times$ RS-232 level and $2 \times$ TTL level).

(iii) ***USB hub 2.*** The USB port of the vision Gumstix has been expanded to four independent USB ports too. One expanded port is connected to the onboard camera. The rest of the ports are reserved for future use.

3.3. *System integration*

The final integrated platform is shown in Fig. 7. Besides the essential mechanical parts, all the related avionics components have been assembled onto the system. This platform has been extensively used in test flights for model identification and verification.

3.3.1. *Layout design*

Layout design for onboard computer systems is a challenging issue for small-scale UAVs. In what follows, we propose a simple and unified layout design approach, which is independent of the hardware components used and can be easily adopted to construct any small-scale UAVs.

(i) ***Location of navigation sensor.*** The essential rule of this step is to mount the navigation sensor: IG-500 N with the Control Hub as close as possible to the center of gravity (CG) of the UAV to minimize the so-called lever effect, which can cause bias on the measured accelerations when the UAV performs rotatory motions. Based on the experience we gained from the construction of our earlier version UAVs,[17] we find that it is easier to control the UAV when the onboard system is mounted underneath the bare vehicle. For such a layout, the general guideline is to line up the CGs of the INS/GPS, the onboard computer system and the basic helicopter along the *z*-axis of the body frame. Since the CG location of the bare vehicle is fully known using the pendulum test introduced in,[18] the mounting location of the navigation sensor in *x*–*y* plane of body frame can be determined. The offset between the CG of the UAV and that of the navigation sensor is only in *z*-axis and unavoidable. However, it can be minimized by carefully considering the height of onboard system and adding necessary space between the bare helicopter and the onboard system for bumping avoidance. In addition, the GPS antenna of IG-500 N has been located in the top of the main shaft of GremLion to have a good view of the sky in order to obtain a stable signal lock.

(ii) ***CG balancing.*** The locations of the following four components, i.e., the Vision Hub, the onboard camera, the wireless modem, and the battery packs, have to be carefully selected. In general, the camera and the wireless modem are to be mounted at the front part for the convenience of observation and wireless communication. The Vision Hub is placed on the back to balance the overall CG of the onboard system. The battery packs are placed beneath the fuselage and

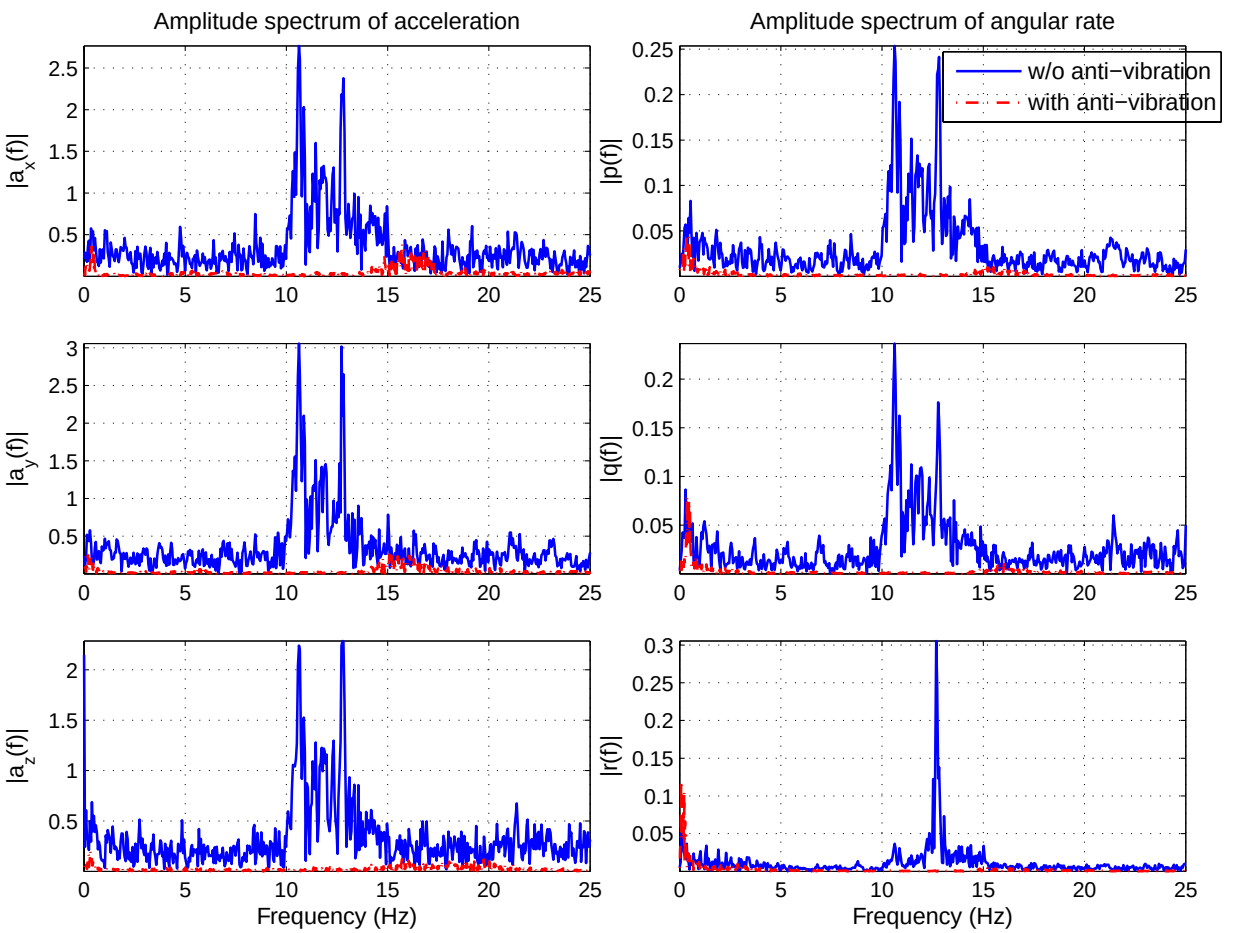

Fig. 13. Comparison of frequency response of the acceleration and angular rate with anti-vibration solutions.

along the *z*-axis of the body frame. Furthermore, we also guarantee that the CG of the onboard system coincides with the CG of the INS/GPS, and the onboard system is symmetrical in both longitudinal and lateral directions.

3.3.2. *Anti-vibration*

Anti-vibration for the platform is a key issue which affects performance of the system significantly. The main vibration sources in GremLion comes from the two main rotors with the frequency of 33.3 Hz. This frequency is calculated based on the designed main rotor speed at 2000 RPM, which was also verified using a handheld tachometer.

Several solutions have been employed for the anti-vibration purpose: (a) use four wire-rope isolators mounted symmetrically around the CG of the avionic system; (b) employ a re-designed landing skid that has better connections to the platform; (c) replace wooden blades with carbon blades, which have the same airfoil profile and size but with a smooth surface; and (d) configure the cut-off frequencies of the built-in low pass filters of the IMU at 10 Hz. These proposed anti-vibration solutions have been demonstrated in flight. The comparison of the vibrations before and after employing the anti-vibrations are shown in Fig. 13.

3.3.3. *Slanted rooftop landing mechanism*

As the GremLion is required to land on rooftops while doing surveillance, we are unaware of the conditions on the rooftop in general. Here we have designed a landing skid such that it is able to handle landing on rough surfaces, even on the slanted surface while not affecting its orientation stability. Motivated by the low center of gravity of the GremLion as its battery will be placed at the bottom of the UAV, an automatic adjusting landing skid was designed.

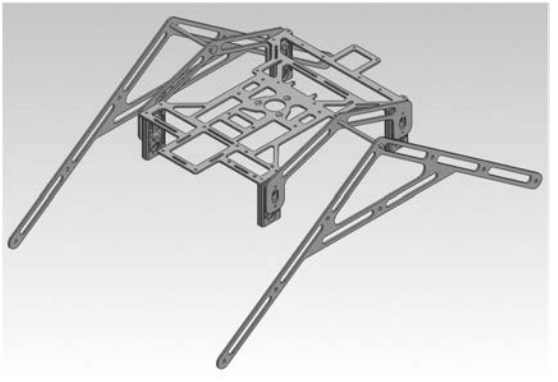

Fig. 14. Landing skid design.

The assembled landing skid together with the frame of GremLion is shown in Fig. 14. As shown in Fig. 14, upon contacting the ground or rooftop when landing, the legs of the landing skid, i.e., the second part in the diagram, will tilt to suit itself to the surface of the ground/rooftop. As the center of gravity of the GremLion is below the pivot point, the GremLion's body will stay upright as desired. Numerous experiments and flight tests have been undergone to verify the feasibility of such a design.

4. Onboard System Software

Based on the developed hardware system, a framework of a UAV software system is proposed, shown in Fig. 15. With this framework, all necessary UAV modules including onboard system and ground control system functions are clearly structured and presented. The logical data flows among different modules also facilitate the design and analysis of UAV systems.

The onboard system is the most critical component in the UAV systems. The main functions of the onboard system is to collect sensor data, process and fuse them, feed to the control law, send the servo driving signals to realize desired automatic and intelligent operations. Meanwhile, the UAV status data is transmitted back to the ground control station (GCS) and exchanged among other UAV team members.

4.1. *Sensing*

The sensing data may come from different sources, such as IMU, GPS, ultrasonic, laser scanner or vision system depending on the configuration of the avionics. The logical representation of sensing block is shown in Fig. 16. All the sensing data are fed into the data selection and the processing unit, which will be selected and combined for the flight control and other mission algorithms.

4.2. *Simulation model*

The simulation model block is to realize hardware-in-the-loop simulation, which can be represented as a black box. The black box has two inputs and one helicopter state output. In mathematical form, the UAV model can be formulated as a 14th-order ordinary differential equation (ODE) as below:

$$\dot{x} = f(t, u, v, x), \tag{1}$$

where x represents the UAV model output with 14 states, t is the current system time, u is the current control signal input with four input channels, and v is the wind disturbance in three directions. For the ODE implementation, the

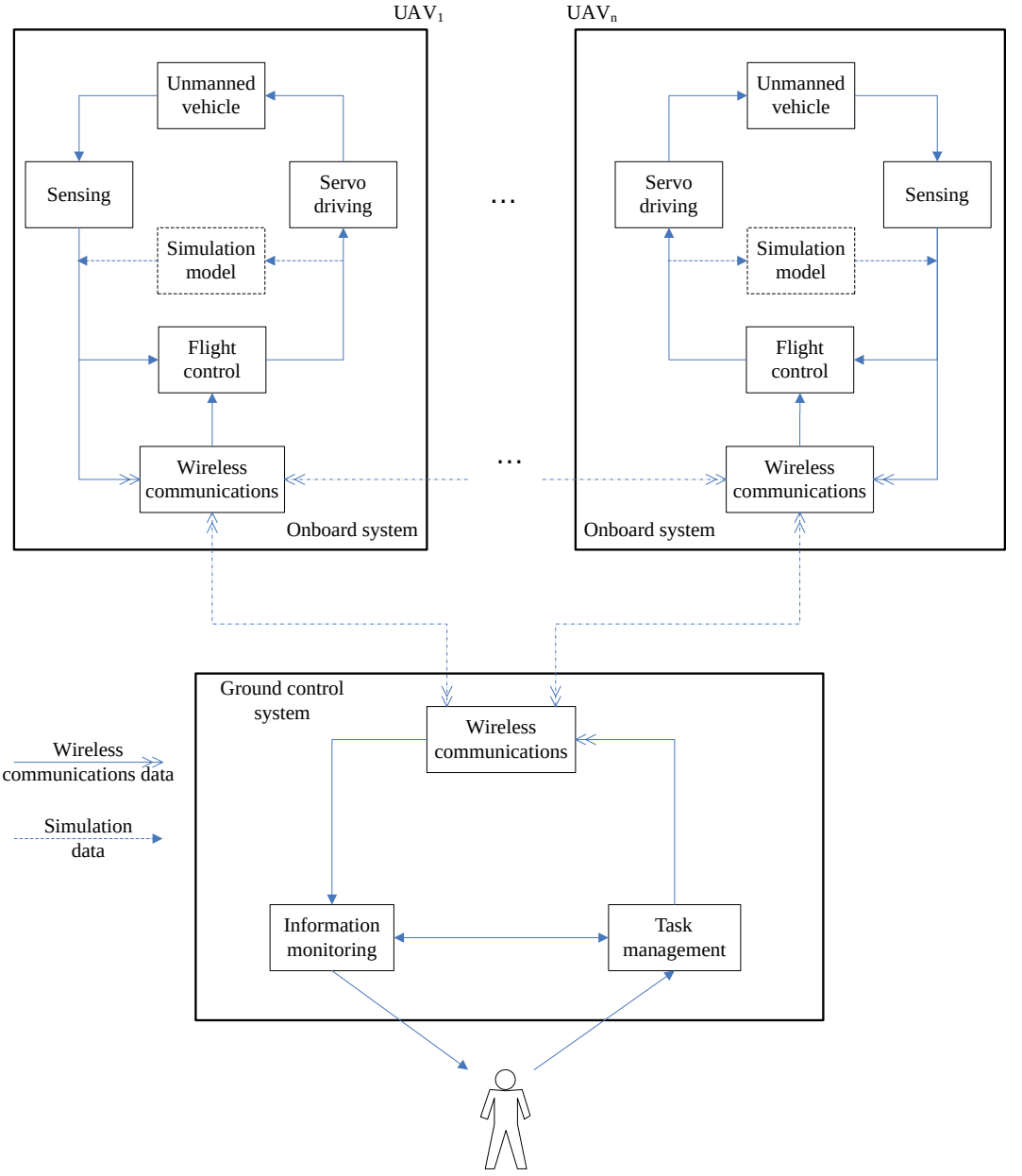

Fig. 15. Framework of UAV systems.

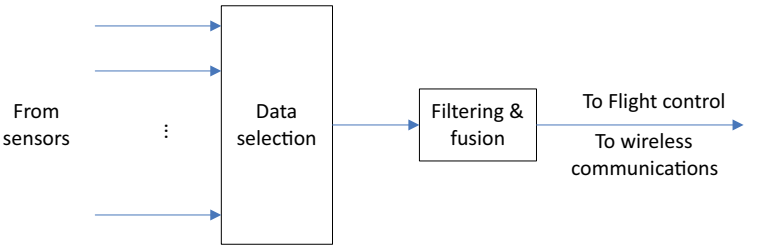

Fig. 16. Sensing description.

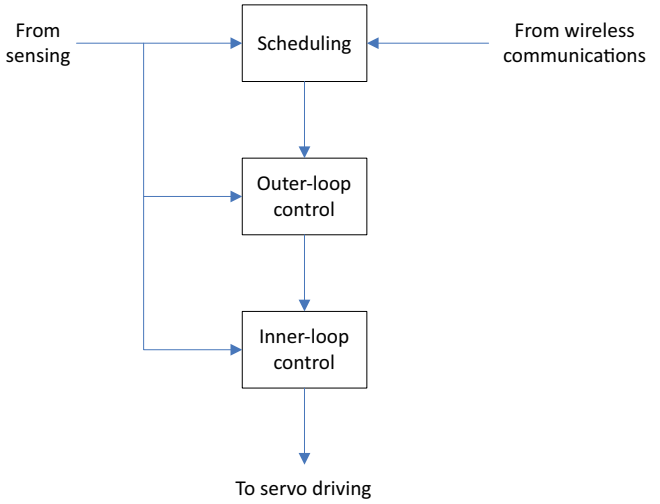

Fig. 17. Flight control description.

classical Runge–Kutta approximation method is applied in the software.

4.3. *Flight control system*

The description of flight control is shown in Fig. 17. The flight control module consists of mainly three units: task scheduling, outer-loop control and inner-loop control. The task scheduling is to generate the outer-loop references given current status data and user commands from wireless communications. Based on the reference signals, the outer-loop realizes position, velocity and heading control by generating references for the inner-loop. The inner-loop is to stabilize the UAV attitude. During the control procedure, the outputs are sent to the servo driving module to drive the actuators on the UAV.

4.3.1. *Control law implementation*

The control law implementation is realized via two hierarchical blocks, inner-loop and outer-loop as shown in Fig. 27. The outer-loop is to generate references for the inner-loop as the input. The flight scheduling module is to divide a whole flight mission into several specified tasks such as take-off, path tracking, and landing. The flight mission can be as simple as conducting an automatic hover operation, or can be as complicated as surveillance of a group of UAVs. With this hierarchical approach, various high level missions can be transferred into logical representation and practical implementations. Specifically, for different UAV platforms, the corresponding blocks of outer-loop and inner-loop are activated. In addition, given different control behaviors, such as landing and take-off, specific control blocks are also developed.

4.3.2. *Autonomous reference generation*

In some critical applications, the reference paths for a UAV must be generated online. This is commonly needed in dynamic environments such as in the cases of a lost link or waypoint updates from the ground pilot. The data flow diagram of autonomous path generation is shown in Fig. 18. There are basically two parts, one is the path creation given a certain task and the other is the outer-loop reference generation from the generated path.

In the path creation, the task needs to provide the destination waypoint, which when given current position and heading, the new path can be automatically generated with a specified tracking velocity. The destination waypoint can be uploaded from Google Maps with the GPS information, or by a user-defined relative distance to the launch position. The outer-loop reference generation will be detailed in Sec. 7.4, since the dynamics of the UAV needs to be considered.

4.4. *Servo driving*

The servo driving block is to control the deflections of actuators according to the control outputs. The servo driving block has two inputs. One is the manual control signal

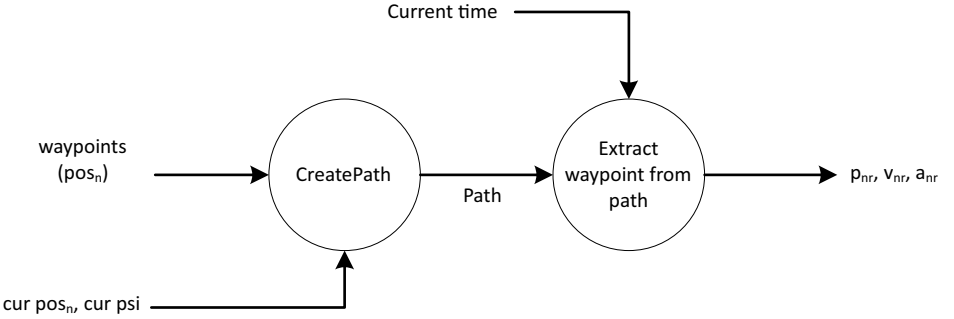

Fig. 18. Data flow diagram of autonomous reference generation.

from the ground pilot and another input is the automatic control signal generated by the flight control block. One of the two inputs can be selected to output in terms of the switch signal from the manual input. If the manual control is enabled, then the manual input signal will be translated and stored in the data store. Otherwise, the automatic signal will be translated, stored, and finally sent to the servos to realize desired deflection positions.

4.5. *Data logging*

The data logging block is to record the key avionics data in flights, including UAV status, manual and automatic servo signals, user commands, and so on. The recorded data can reflect the working status and property of the whole system, and are useful for off-line analysis. Considering the CPU load increase when writing data to the onboard CF card, the data logging thread is executed every 50 cycles, i.e., once per second.

4.6. *Emergency function*

The emergency function is learned from a couple of crash accidents of our UAV helicopters. There are many causes for UAV failures, e.g., drastic changes in environment, hardware failure, software failure, etc. To handle such emergency situations, the control task thread checks all sending data at every cycle before applying control action. Once an abnormality is detected, the emergency control function will be activated immediately to (i) send an alert signal to GCS to inform the pilot to take over the control authority; (ii) drive and maintain all the control outputs to their trimmed values in the hovering condition; and (iii) slow down the engine or motor speed if the control authority is still at the automatic side after a pre-defined alert time. The flight data will also be recorded, which is important for fault analysis.

4.7. *Task management*

The above discussed subsystem is represented as an individual task in the whole onboard software system. Based on the theory of systems and control, the above tasks should be executed in a reasonable order to fulfill the automatic flight control purpose. Considering the context-switching cost, the onboard avionics application is designed in a multi-thread fashion. Therefore, the above identified task can be accomplished within each working thread. The QNX provides the kernel level mechanism to support the message passing and synchronization of multi-thread software architecture.

To realize the predefined task execution and synchronization, a task management module is carefully designed which is shown in Fig. 19. Note that the tasks involved here are the active tasks which should be executed within every control loop. To achieve fair allocation of processor running time, the round robin scheduling policy is adopted in the multi-thread design. On the other hand, the background tasks such as communication receiving (UART based and TCP/IP based) are activated once data arrive and if there are still CPU running slices left and the activation mechanism is determined by the scheduling policy in QNX RTOS.

As shown in Fig. 19, the main program is responsible for activating the next task once the notification message is received from the current running task.

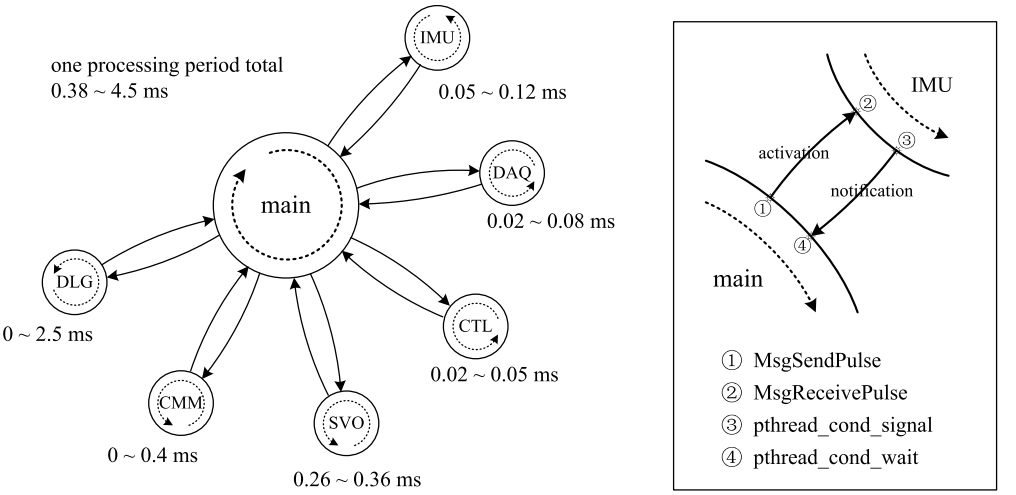

Fig. 19. Task management illustration diagram.

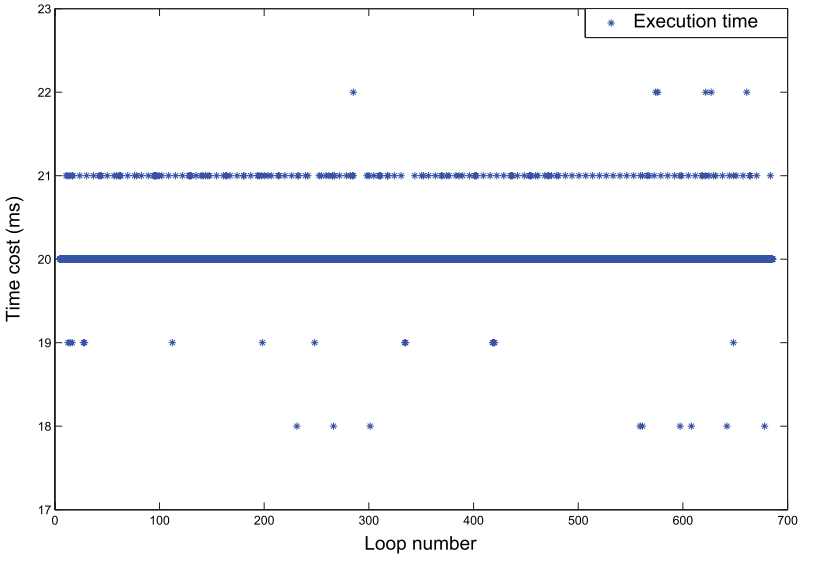

Fig. 20. Time intervals between each loop.

4.8. *Performance evaluation*

The time deadline is the most important property in real-time systems. As the main control period of the onboard system is set to 20 ms, the timing intervals between each running loop should be examined to test the robustness and efficiency of the avionics system framework. The total processing time for each loop is summarized in Fig. 20. It can be observed that the timing interval between each loop is separated by strictly following the 20 ms configuration which provides the fundamental support for correct and stable control law implementation.

4.9. *Onboard vision system*

Apart from the fundamental autonomous control functions, advanced intelligence is achieved via an onboard vision software system, which is deployed on another Gumstix board. Due to the rich hardware driver support, especially various camera driver support, the customized Linux operating system is adopted as the development environment.

The working principle of onboard vision system is illustrated in Fig. 21, including capturing raw images from the digital camera, receiving the UAV status data from onboard control system via UART, performing image processing including preprocessing and algorithms calculation, sending compressed image over 3G modem. However, both image capture and image processing are load intensive working threads, the main loop of vision onboard system is set as 10 Hz. As the vision algorithms is for high level guidance and control, the 10 Hz update rate is acceptable in most applications.

4.9.1. *Image processing*

The core function of the vision software system is the image processing. The functions of image processing contains the obstacle avoidance and ground target tracking which will be presented in the following sections. The output of image processing is the target information for the task scheduling module of onboard control computer. In addition, the processed images are down linked to the GCS for ground pilot monitoring.

4.9.2. *3G communication*

The data transmission mechanism is based on the 3G network that possesses continental distance (around tens of km) and high data bandwidth (100 $\sim$ 200 kpbs). To facilitate the image transmission, the images are compressed via a JPEG compression method provided by OpenCV libraries. The size of the compressed image size is about 3 K Bytes that can be reasonably accommodated by the 3G network

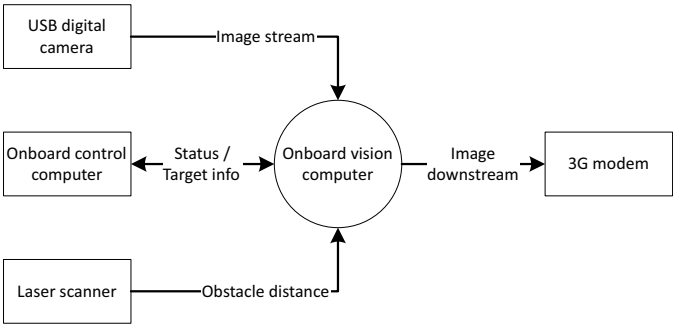

Fig. 21. UAV onboard vision software system.

bandwidth. The update frequency is achieved as 2 Hz. The 3G module adopted is the UC864-E from Telit company, which is well compatible with the Gumstix board. The communication protocols for image transmission is selected as TCP that can guarantee reliable image transmission though some delays can be expected for the handshaking and image retransmission.

5. Ground Control Station

The GCS is composed of background tasks and foreground tasks. The background layer has mainly two tasks, receiving flight status from and sending commands to UAVs, both of which interact with the UAV onboard CMM task module. The receiving thread accepts all the data from the fleet of UAVs, and identify each status data via the telegraph packet header. Consequently, the corresponding multiple display is executed, and the cooperative waypoints of the paths are demonstrated. Similarly, the upload link can broadcast the commands to all UAVs, or alternatively send commands to a specific UAV, both via the sending task. The global status data from UAVs are dynamically updated from the background layer. The foreground task composes of information monitoring and task management, where the information monitoring module consists of various user-friendly views. A document class implementation in MFC[19] is deployed to realize the communication between the background tasks and foreground tasks. The document class performs the flight data store (up to 2000 updates), data processing (rotation computation in 3D view), command interpreting and packaging, etc.

Five kinds of views are developed on GCS, including the map view, the curve view, the text view, the command view and 3D view, which are all shown in Fig. 22. To facilitate navigation and better demonstrate the flight trajectories of UAVs, we downloaded the map from the Google map server and used the map tiles off-line on the flight field conveniently without bothering the Internet service.[20] In the flight test, the GPS data from the onboard system will be updated on the global shared data, and the flight trajectories are drawn correspondingly on the Google map view.

6. Dynamics Modeling

To accomplish the required tasks of the UAVForge competition, the flying vehicle needs to be attitude-wise stable in the first place. This requires a robust inner-loop feedback control law which works well even in windy conditions. Next, to achieve GPS-based waypoint flight, there should also be a stable yet responsive outer-loop control law and an elegant reference generation mechanism which can provide smooth tracking references with regard to position, velocity and heading. All of these, if high performance is needed, requires an accurate flight dynamics model of the controlled platform. This leads to the following context which will detail GremLion's model formulation and parameter identification.

In order to systematically design a set of flight control laws for the GremLion platform with good performance, an accurate mathematical model reflecting the flight dynamics of the flying platform needs to be derived. To obtain such a model, two approaches can be considered. One is the first-principles modeling approach which focuses on direct mathematical formulation of the system based on the law of physics and aerodynamics, while the other one is the system identification approach which numerically estimates the parameters of a 'black box' system with sufficient in-flight data. Although both approaches have shown their

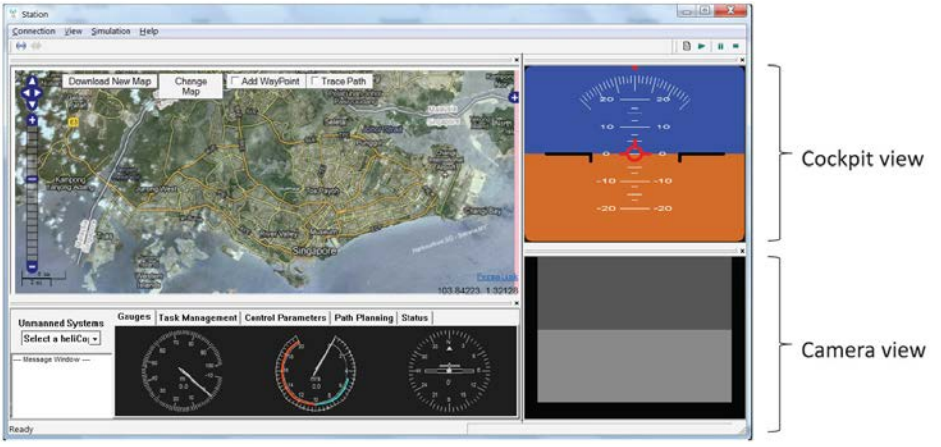

Fig. 22. Snapshot of GCS layout.

individual successes in literature,[21–26] using either of them alone may not be reliable enough to generate a model with good fidelity for a comprehensive envelope of UAV flight. Theoretically, the first-principles nonlinear model should cover all operation points in a full flight envelope. However, it is tedious and impractical in the sense that the model needs to be tuned iteratively for different ground and flight conditions (e.g., landing or taking off, with wind disturbance or without). On the other hand, the second method is suitable to deliver a simplified linear model for certain operating point or small flight regime. However, if a large flight envelope needs to be covered, the identification procedures have to be repeated for all conditions and gain scheduling is usually needed when designing the control law. The efficiency and convenience of obtaining the model using the second method generally decrease in situations associated with higher vehicular speed or more aggressive motions because in these situations, data collection is extremely difficult or even impossible to be carried out. Hence, for the modeling of GremLion, the above two methods will be used in a complementary way. In the following content, model structure of the UAV in flight will be explained first. Then the parameter identification results derived from several sets of manual flight data will be shown accordingly.

6.1. *Modeling assumptions and notations*

Due to the nature of the UAVForge competition, nonaggressive waypoint flight is more than necessary to complete all the competition tasks. Hence, the controlled platform will always be hovering, moving or turning at very low linear speed or angular rate. This leads to the near-hover assumption and the model complexity can be drastically reduced as compared to full envelope flight. First of all, linear acceleration, linear velocity, angular rates, and roll pitch angles are all near zero when the UAV motion is near hovering, thus terms involving the second order of these variables can be neglected when expanding mathematical expressions. Second, we assume very fast response of servos and motors (i.e., the response time from control input to the change of servo positions or change of speed of motors is much faster than the UAV dynamics).

Although the coaxial helicopter flies with the top and bottom rotors pitching cyclically at different angles (refer to a, b, c, d in Table 6), it is reasonable to look at the two rotors as a whole system and model them as a single imaginary rotor with the so-called resultant longitudinal and lateral flapping angles expressed as a_s and b_s. With this assumption, the model can be simplified further, yet maintaining good fidelity provided that the UAV does not do rapid maneuvering.

Table 6. Definition of symbols

Symbols	Definition
m	Mass of the UAV
g	Earth gravity
$(F_x, F_y, F_z)^{\mathrm{T}}$	Resultant force acting on the body axis x-, y-, z-directions
$(T_x, T_y, T_z)^{\mathrm{T}}$	Resultant torque acting on the body axis x-, y-, z-directions
$(u, v, w)^{\mathrm{T}}$	Body-axis velocity in x-, y- and z-directions
$(p, q, r)^{\mathrm{T}}$	Angular velocity in x-, y- and z-axes
$(a, b)^{\mathrm{T}}$	Longitudinal and lateral flapping angles of the bottom rotor
$(c, d)^{\mathrm{T}}$	Longitudinal and lateral flapping angles of the top rotor
$(a_s, b_s)^{\mathrm{T}}$	Equivalent longitudinal and lateral flapping angles of the top and bottom rotors together
$(\omega_{\mathrm{top}}, \omega_{\mathrm{bot}})^{\mathrm{T}}$	Spinning speed of the top and bottom rotors
$(\delta_{\mathrm{lat}}, \delta_{\mathrm{lon}}, \delta_{\mathrm{thr}}, \delta_{\mathrm{ped}})^{\mathrm{T}}$	Inputs to the system (aileron, elevator, throttle, rudder)

6.2. *Lateral and longitudinal fuselage dynamics*

By applying the well-known Newton–Euler rigid body motion equation to the UAV fuselage motion in the lateral and longitudinal directions, we have

$$\dot{u} = \frac{F_x}{m} - wq + vr, \tag{2}$$

$$\dot{v} = \frac{F_y}{m} - ur + wp, \tag{3}$$

$$\dot{p} = \frac{T_x}{I_{xx}} - qr\frac{I_{yy} - I_{zz}}{I_{xx}}, \tag{4}$$

$$\dot{q} = \frac{T_y}{I_{yy}} - pr\frac{I_{zz} - I_{xx}}{I_{yy}}, \tag{5}$$

where the following assumptions can be made at a near-hover condition,

$$\begin{aligned} &u, v, w, p, q, r \approx 0,\\ &\sin a_s \approx a_s, \quad \sin b_s \approx b_s,\\ &\sin \phi \approx \phi, \quad \sin \theta \approx \theta,\\ &\cos a_s \approx 1, \quad \cos b_s \approx 1,\\ &\cos \phi \approx 1, \quad \cos \theta \approx 1. \end{aligned}$$

Hence, we can simplify the equations and obtain the following linear expressions:

$$\dot{u} = -g\theta - ga_s - X_u u, \tag{6}$$

$$\dot{v} = g\phi + gb_s - X_v v, \tag{7}$$

$$\dot{p} = Y_b b_s, \tag{8}$$

$$\dot{q} = Y_a a_s, \tag{9}$$

where X_u, X_v, Y_a, Y_b are parameters that need to be determined in the later system identification step.

6.3. *Lumped rotor flapping dynamics*

An intuitive way to represent the helicopter rotor flapping dynamics is to see the whole system as a by-default horizontal rigid disc which can tilt about its rotational axis in the longitudinal and lateral directions. This motion corresponds to the first harmonic approximation in the Fourier Series description of the complicated rotor flapping equations. Moreover, to avoid the complicated analysis of force interaction between GremLion's coaxial dual-layer rotors, a single resultant rotor with longitudinal and lateral flapping motion is virtually created to represent the two rotor mechanism in the coaxial structure. Another fact to note is that, in order to ease manual control for the later in-flight data collection for parameter identification, there is a hardware rate feedback gyro pre-installed in between the control signals (aileron, elevator) and the swashplate servos. This results in a nonunity gain before the terms p and q in the following first-order rotor flapping dynamics equations:

$$\dot{a}_s = -K_q q - \frac{a_s}{\tau} + K_a \delta_{\text{lon}}, \tag{10}$$

$$\dot{b}_s = -K_p p - \frac{b_s}{\tau} + K_b \delta_{\text{lat}}. \tag{11}$$

These two equations share the same time constant τ because the flapping of the disc is symmetric both longitudinally and laterally.

6.4. *Heave dynamics*

Similar to the Newton–Euler motion equations used for the longitudinal and lateral dynamics, the heave dynamics can be represented as:

$$\dot{w} = \frac{F_z}{m} - vp + uq, \tag{12}$$

where $u, v, p, q \approx 0$ at hovering condition and F_z is a combination of rotor thrust, UAV weight, and air dragging force. After linearization,we can obtain

$$\dot{w} = -\frac{w}{\tau_w} + K_w \delta_{\text{thr}}. \tag{13}$$

6.5. *Yaw dynamics*

Yaw motion control was once a very challenging issue in the RC helicopter community because the motion in this channel is extremely sensitive and hard to be controlled by a human pilot. To overcome this problem, most RC helicopters nowadays are equipped with a yaw gyro, which consists of an angular rate sensor and an electronic feedback control circuit to damp the helicopter heading dynamics and facilitate manual control. Ideally, this explicit hardware component could be removed in UAV systems. However, they are commonly reserved for the purpose of manual control backup. The GremLion platform retains this configuration and its yaw gyro is set as a clean proportional angular rate feedback controller. Thus, the identified linear dynamics in the yaw channel should be inherently stable and can be expressed as a first-order equation:

$$\dot{r} = N_r r + N_{\text{ped}} \delta_{\text{ped}}. \tag{14}$$

6.6. *Data collection and model identification*

From the above analysis, a full linear model of the GremLion flying at the near-hover condition can be derived, which is an ideal model that does not consider any coupling effects among the four channels. In reality, couplings in the roll-pitch and yaw-heave dynamics are very significant for the case of co-axial helicopters. Given this concern, a cross-coupled model of GremLion flying at the near-hover condition can be represented in the following state-space form (oval boxes indicate the coupling terms):

$$\dot{\mathbf{x}} = A_{\text{id}}\mathbf{x} + B_{\text{id}}\mathbf{u}, \tag{15}$$

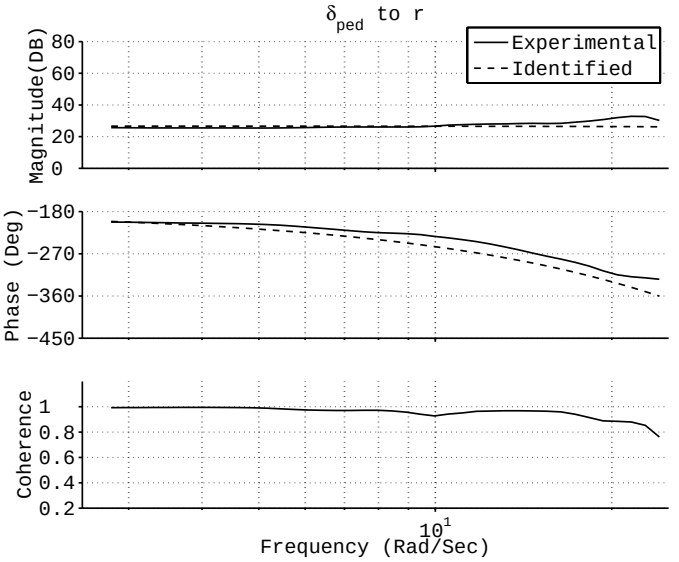

Fig. 23. System identification from δ_{rud} to r.

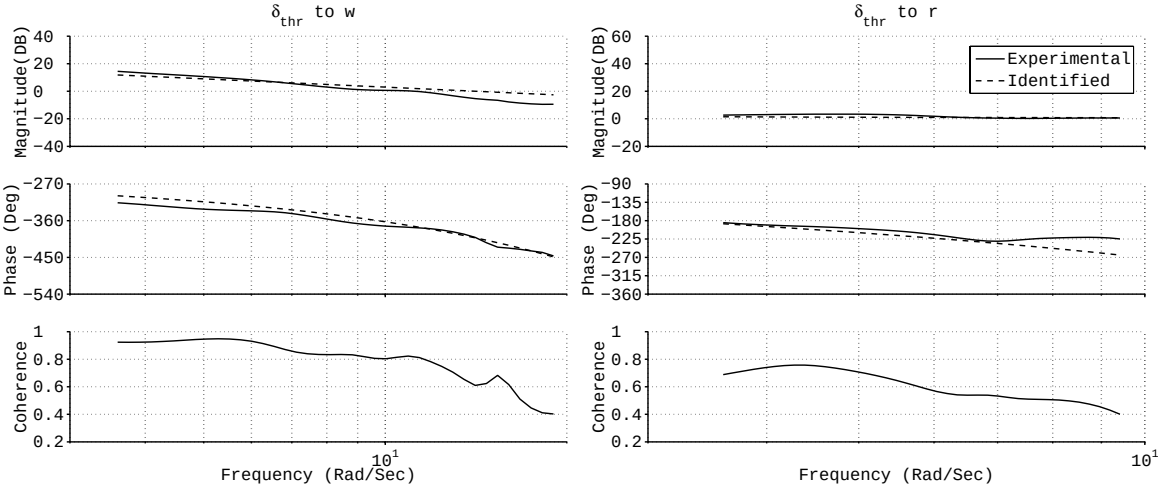

Fig. 24. System identification from δ_{thr} to w and r.

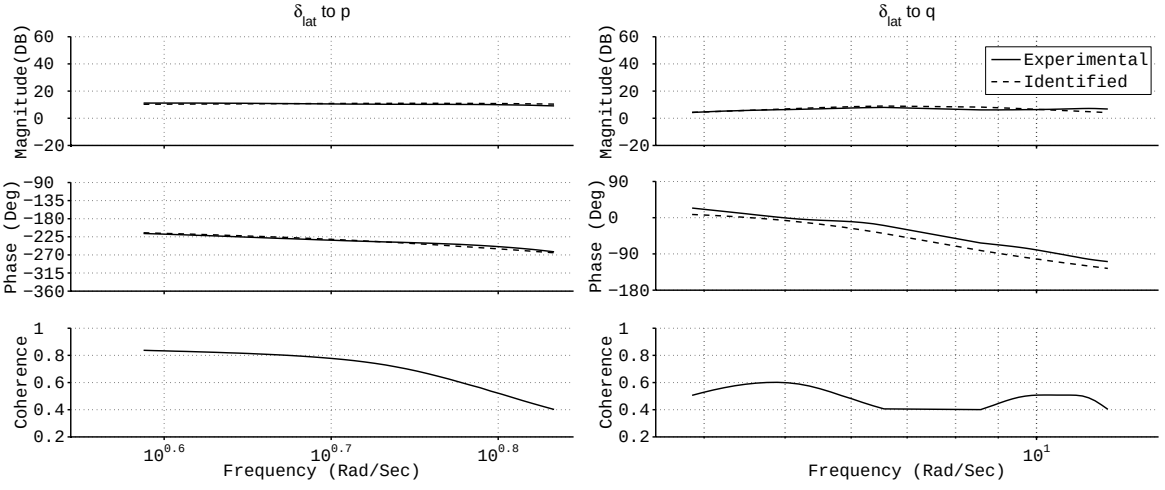

Fig. 25. System identification from δ_{lat} to p and q.

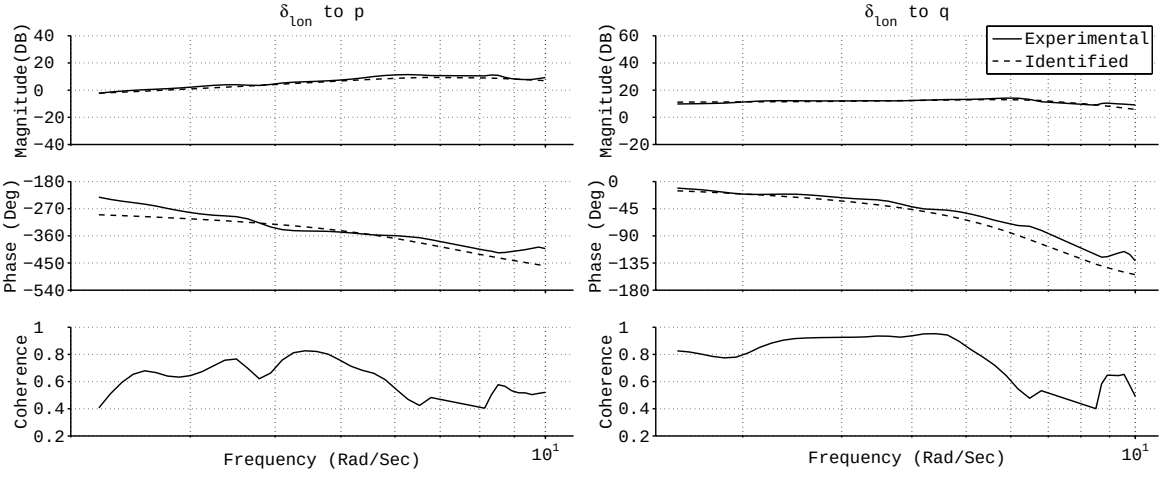

Fig. 26. System identification from δ_{lon} to p and q.

where $\mathbf{x} = \mathbf{x}_{\text{act}} - \mathbf{x}_{\text{trim}}$ is the difference between the actual state variables and their trimmed values, and similarly, $\mathbf{u} = \mathbf{u}_{\text{act}} - \mathbf{u}_{\text{trim}}$, which are respectively given as

$$\mathbf{x} = (u, v, p, q, \phi, \theta, a_s, b_s, w, r)^{\mathrm{T}},$$
$$\mathbf{u} = (\delta_{\text{lat}}, \delta_{\text{lon}}, \delta_{\text{thr}}, \delta_{\text{ped}})^{\mathrm{T}},$$

and A_{id} and B_{id} matrices are given by

$$A_{\text{id}} = \begin{bmatrix} -X_u & 0 & 0 & 0 & 0 & -g & -g & 0 & 0 & 0 \\ 0 & -X_v & 0 & 0 & g & 0 & 0 & g & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \boxed{C_1} & Y_b & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & Y_a & \boxed{C_2} & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -K_q & 0 & 0 & -\frac{1}{\tau} & \boxed{C_3} & 0 & 0 \\ 0 & 0 & -K_p & 0 & 0 & 0 & \boxed{C_4} & -\frac{1}{\tau} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{\tau_w} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \boxed{C_5} & N_r \end{bmatrix}.$$

A series of flight tests have been conducted to collect sufficient high-quality input and output data for model identification purposes. Frequency sweeping signals (sinusoidal signals with various frequencies) has been adopted due to its suitability in rotorcraft system identification. A well-established model identification toolkit, named CIFER (Comprehensive Identification from Frequency Responses) and developed by NASA Ames Research Centre, has been utilized for obtaining the specified values of the unknown parameters. The parameter identification process is performed in frequency domain, and the frequency responses derived from the manual flight data and the respective coherence metrics (describing how linear the input and output relationship is) are depicted in Figs. 23–26. From the plots, we can see that there is large coupling between the aileron and elevator channels and also between the throttle and rudder channels. The identified parameters of the final model after sufficient number of refining iterations by the tool kit is shown in Table 7. With the presence of those coupling terms between the channels: $C_i, i = 1, \ldots, 8$, the final model describes the GremLion dynamic system more accurately as compared to an ideal diagonal (no coupling is considered) one.

Table 7. Values of the parameters.

Parameters	Values
X_u	0.057
X_v	0.132
Y_a	83.264
Y_b	151.777
K_p	1.146
K_q	1.146
τ	0.060
τ_w	3.240
N_r	−78.401
C_1	48.621
C_2	−51.837
C_3	−9.672
C_4	5.469
C_5	−11.596
C_6	0.853
C_7	−0.110
C_8	−85.159

7. Navigation and Control

In control engineering, the 'Divide-and-Conquer' strategy is usually used when a relatively complex system needs to be handled. In flight control engineering, a natural stratification of the full-order dynamics model of a helicopter is based on motion types, i.e., rotational motion and translational motion. In general, the dynamics of rotational motion is much faster than that of the translational motion. Thus, the dynamics of the controlled object can be divided into two parts and the overall control system can be formulated into a dual-loop structure. In this way, inner-loop and outer-loop controllers can be designed separately. Moreover, we find that the linearized model of GremLion is of nonminimum phase if the two motion dynamics are combined together. This will highly complicate the control problem and degrade control performance. Hence, we prefer the dual-loop approach for the design of control laws.

For the inner loop, the controlled object covers the rotational motion of the helicopter body, flapping motion of rotor blades with the stabilizer bar, and dynamics embedded within the head lock gyro. The main task of the inner-loop controller is to stabilize the attitude and heading of GremLion in all flight conditions. H_∞ technique is preferred for robust stability. For the outer loop, the controlled object covers only the translational motion. The main task is to steer the UAV to fly with reference to a series of given locations. A robust and perfect tracking (RPT) approach is implemented for the outer-loop since time factor is important. It should be noted that both control laws are designed using the asymptotic time-scale and eigenstructure assignment (ATEA) method, which is fully developed for

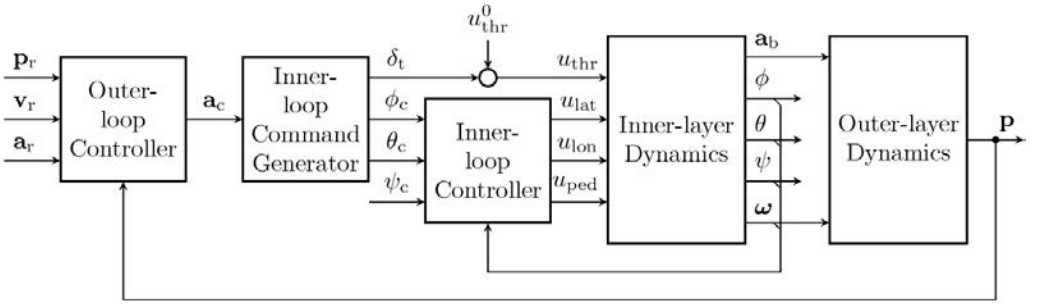

Fig. 27. Dual-loop structure of flight control system.

MIMO LTI systems by Chen.[27] It makes the design process very systematic and effective. To give an overall view, the dual-loop control structure is shown in Fig. 27.

7.1. *Inner-loop control design*

The inner-layer dynamics is a 8th-order MIMO system with three control inputs, namely u_{lat}, u_{lon}, and u_{ped}. The uninvolved fourth input, u_{thr}, is reserved for control of vertical motion and needs be set at its trimming value (denoted as u_{thr}^0) at this stage. For the measurement part, IMU gives ϕ, θ, ψ, p, q, and r. The other two state variables (i.e., the flapping angles b_s, a_s) have to be estimated by an observer. Therefore, the linearized inner-layer controlled object can be formulated as

$$\begin{cases} \dot{\mathbf{x}} = A_{\text{in}}\mathbf{x} + B_{\text{in}}\mathbf{u} + E_{\text{in}}\mathbf{w} \\ \mathbf{y} = C_{\text{in},1}\mathbf{x} + D_{\text{in},11}\mathbf{u} + D_{\text{in},1}\mathbf{w}, \\ \mathbf{z} = C_{\text{in},2}\mathbf{x} + D_{\text{in},2}\mathbf{u} + D_{\text{in},22}\mathbf{w} \end{cases} \tag{16}$$

with

$$\begin{aligned} \mathbf{x} &= (\phi \quad \theta \quad \psi \quad p \quad q \quad r \quad b_s \quad a_s)^{\text{T}}, \\ \mathbf{y} &= (\phi \quad \theta \quad \psi \quad p \quad q \quad r)^{\text{T}}, \\ \mathbf{z} &= (\phi \quad \theta \quad \psi)^{\text{T}}, \\ \mathbf{u} &= (\delta_{\text{lat}} \quad \delta_{\text{lon}} \quad \delta_{\text{ped}})^{\text{T}}, \end{aligned}$$

and

$$A_{\text{in}} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 151.78 & 48.62 \\ 0 & 0 & 0 & 0 & 0 & 0 & -51.84 & 83.26 \\ 0 & 0 & 0 & 0 & 0 & -78.40 & 0 & 0 \\ 0 & 0 & 0 & -1.15 & 0 & 0 & -16.64 & 5.47 \\ 0 & 0 & 0 & 0 & -1.15 & 0 & -9.67 & -16.64 \end{bmatrix},$$

$$B_{\text{in}} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -1680.1 \\ -3.25 & -0.11 & 0 \\ 0.85 & 4.05 & 0 \end{bmatrix},$$

where $\mathbf{y}$ is the measured output vector, $\mathbf{z}$ is the controlled output vector, and all variables are the deviations from their trimming values. Note that the direct feedthrough matrices $D_{\text{in},11}$ and $D_{\text{in},2}$ are both zero. No external disturbance is considered for this part of the model at the current stage, so the disturbance input matrix E_{in} and the feedthrough matrices $D_{\text{in},1}$ and $D_{\text{in},22}$ are all empty. They are reserved in the expression for integrity so that external disturbances such as wind gusts can be considered in future. The controlled subsystem characterized by quadruple (A_{in}, B_{in}, $C_{\text{in},2}$, $D_{\text{in},2}$) is both observable and controllable. By transforming the quadruple into the special coordinate basis (SCB) form,[27] we find that the subsystem is invertible and of minimum phase. Hence, we can design an H_∞ controller via the ATEA method using state feedback to obtain robust stability. After that, an observer-based controller can be designed utilizing measurement feedback also via the same method. Usually, the control law designed via the ATEA method is parameterized by a number $\gamma > \gamma^*$, where γ^* is the infimum of the H_∞-norm of the closed-loop transfer matrix from disturbance $\mathbf{w}$ to controlled output $\mathbf{z}$. We will find that our design result does not depend on the number γ because the controlled subsystem (A_{in}, B_{in}, $C_{\text{in},2}$, $D_{\text{in},2}$) is right invertible and of minimum phase, and the measurable subsystem (A_{in}, E_{in}, $C_{\text{in},1}$, $D_{\text{in},1}$) is left invertible and of minimum phase. This simplifies the design process significantly.

The L-shape dotted box in Fig. 28 covers the inner-loop control structure of GremLion. Matrix F_{i} is the state feedback gain, Matrix G_{i} is the corresponding reference feed

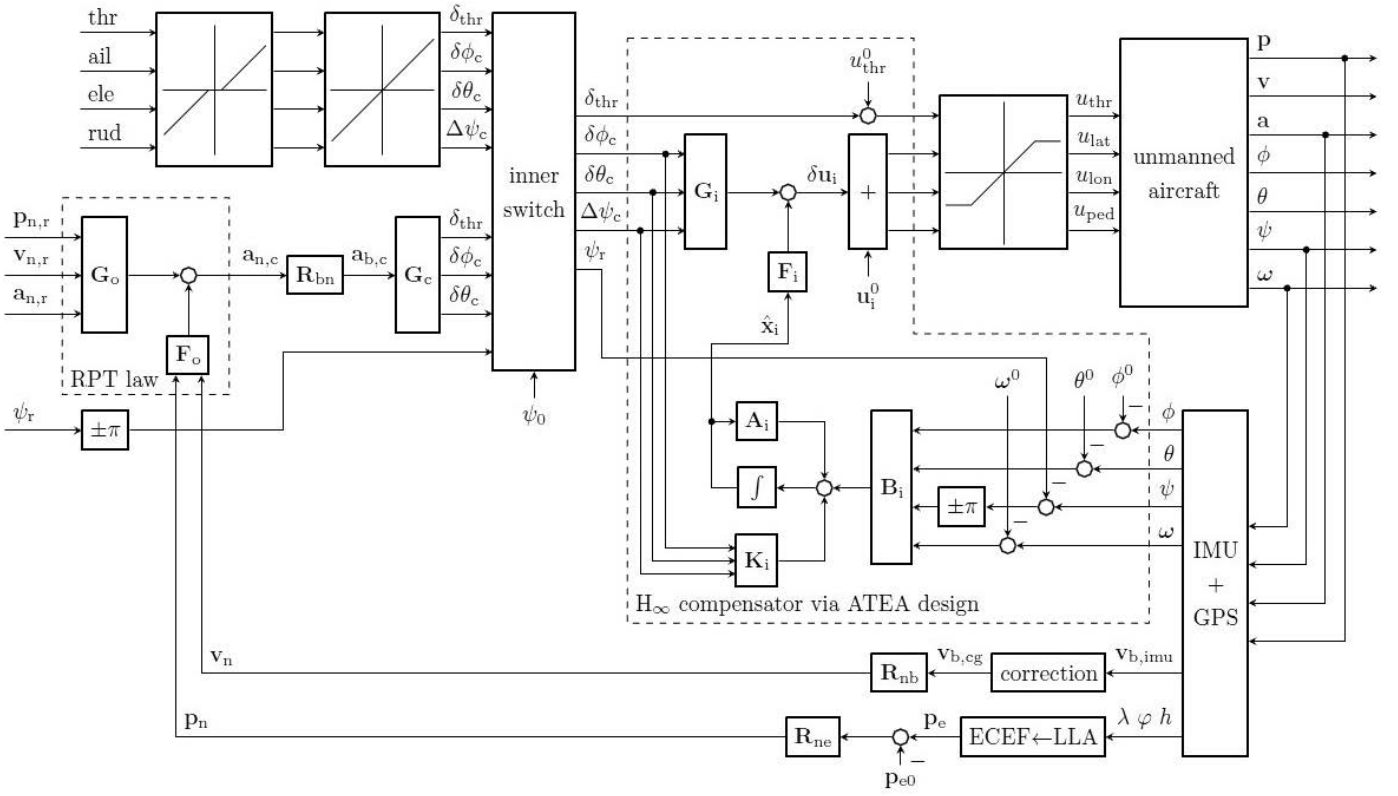

Fig. 28. Outer- and inner-loop control laws based on RPT and H_∞ approach via the ATEA design method.

forward gain to make sure the ratio between output and reference is unity. Matrices A_i, B_i and K_i constitute the observer. Their values are designed as follows:

$$F_i = \begin{bmatrix} 0.346 & -0.133 & 0 & 0.051 & -0.218 & 0 & 0.727 & 2.603 \\ -0.242 & -0.332 & 0 & -0.382 & 0.073 & 0 & 0.761 & 0.743 \\ 0 & 0 & 0.071 & 0 & 0 & 0.002 & 0 & 0 \end{bmatrix},$$

$$G_i = \begin{bmatrix} -0.3463 & 0.1325 & 0 \\ 0.2416 & 0.3316 & 0 \\ 0 & 0 & -0.0714 \end{bmatrix},$$

$$A_i = \begin{bmatrix} -75.017 & 0 & 0 & 0 & 0.2208 & 0 & 0 & 0 \\ 0 & -25 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -30 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -28.613 & -9.8191 & 0 & 151.7766 & 48.6206 \\ 0.416 & 0 & 0 & 5.393 & -83.0858 & 0 & -51.8372 & 83.2644 \\ 0 & 0 & -120 & 0 & 0 & -33.0994 & 0 & 0 \\ -1.085 & 0.467 & 0 & -0.145 & -2.7899 & 0 & -19.0879 & -3.0714 \\ -0.642 & -1.457 & 0 & -0.045 & -3.6149 & 0 & -5.9690 & -11.4121 \end{bmatrix},$$

$$B_{\mathrm{i}} = \begin{bmatrix} 75.017 & 0 & 0 & 1 & -0.221 & 0 \\ 0 & 25 & 0 & 0 & 1 & 0 \\ 0 & 0 & 30 & 0 & 0 & 1 \\ 0 & 0 & 0 & 28.613 & 9.819 & 0 \\ -0.416 & 0 & 0 & -5.393 & 83.086 & 0 \\ 0 & 0 & 0 & 0 & 0 & -48.401 \\ -0.014 & 0 & 0 & -1.064 & 3.140 & 0 \\ -0.042 & 0 & 0 & -0.707 & 2.524 & 0 \end{bmatrix},$$

$$K_{\mathrm{i}} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 120 \\ 1.099 & -0.467 & 0 \\ 0.684 & 1.457 & 0 \end{bmatrix}.$$

7.2. *Outer-loop control design*

As mentioned above, the whole dynamics model of GremLion has been partitioned into two parts and we have finished the inner-loop design which stabilizes attitude and heading. The outer-loop control can then be designed separately and based on GremLion's translational motion only, provided that the outer loop is slow enough as compared to the inner loop. Furthermore, the outer-loop control signals are all defined in the North-East-Down (NED) frame and for all three directions, the dynamics are approximately formulated as double integrators. So,

$$\begin{cases} \dot{\mathbf{x}} = A_{\mathrm{out}}\mathbf{x} + B_{\mathrm{out}}\mathbf{u} + E_{\mathrm{out}}\mathbf{w} \\ \mathbf{y} = C_{\mathrm{out},1}\mathbf{x} \\ \mathbf{z} = C_{\mathrm{out},2}\mathbf{x} \end{cases}, \tag{17}$$

with

$$\begin{aligned} \mathbf{x} &= (x \quad y \quad z \quad u \quad v \quad w)^{\mathrm{T}}, \\ \mathbf{y} &= (x \quad y \quad z \quad u \quad v \quad w)^{\mathrm{T}}, \\ \mathbf{z} &= (x \quad y \quad z)^{\mathrm{T}}, \\ \mathbf{u} &= (a_x \quad a_y \quad a_z)^{\mathrm{T}}, \end{aligned}$$

and

$$A_{\mathrm{out}} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad B_{\mathrm{out}} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Since the translational motion in these three directions are largely decoupled (inner-loop should have decoupled them if designed correctly), the RPT control laws for these three channels can be designed separately. Since they are all standard 2nd-order systems, by choosing an appropriate natural frequency and damping ratio, they should be controlled without any problems. Of course, minor tuning is needed after trial flight tests have been carried out. The final chosen gains (see F_{o} and G_{o} in Fig. 28) are:

$$F_{\mathrm{o}} = \begin{bmatrix} -0.72 & 0 & 0 & -1.2 & 0 & 0 \\ 0 & -1.125 & 0 & 0 & -1.5 & 0 \\ 0 & 0 & -0.5 & 0 & 0 & -1 \end{bmatrix},$$

$$G_{\mathrm{o}} = \begin{bmatrix} 0.72 & 0 & 0 & 1.2 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1.125 & 0 & 0 & 1.5 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0.5 & 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}.$$

Note that there are nine reference signals (for each direction, there are position reference, velocity reference, and acceleration reference).

7.3. *Inner-loop command generator*

We have designed the inner-loop and the outer-loop controllers separately to avoid the nonminimum phase problem and to relieve task complexity. To preserve the overall system stability, the closed outer loop should be slower than the closed inner loop. In this case, the closed inner loop can be seen as a static gain when combining with the outer loop. In physical meaning, the output of the outer-loop controller is the commanded acceleration described on the body-axis coordinate system, so denoted with $\mathbf{a}_{\mathrm{c}}$ in Fig. 27. However, the inner-loop controller requires the attitude deflection commands (ϕ_{c}, θ_{c}, ψ_{c}) as control inputs. Obviously, a command conversion is needed. Furthermore, the body-axis acceleration $\mathbf{a}_{\mathrm{b}}$ does not interact with heading direction ψ, which is relatively independent of linear motion for helicopters. The throttle control input u_{thr} is not manipulated by the inner-loop controller since it is not the direct dominator of attitude. It should also be transferred from the outer-loop controller because it dominates heave acceleration. Based on this idea, let G_{a} be the steady-state gain matrix from the inner-loop inputs (δ_{t}, ϕ_{c}, θ_{c}) to the acceleration output $\mathrm{a_b}$. We obtain an approximated inner-loop command generator from the outer-loop controller output $\mathbf{a}_{\mathrm{c}}$ as,

$$(\delta_{\mathrm{t}} \quad \phi_{\mathrm{c}} \quad \theta_{\mathrm{c}})^{\mathrm{T}} = G_{\mathrm{c}}\mathbf{a}_{\mathrm{c}} = G_{\mathrm{a}}^{-1}\mathbf{a}_{\mathrm{c}}. \tag{18}$$

Notice that G_{a} must be nonsingular. Otherwise, $\mathbf{a}_{\mathrm{b}}$ cannot be manipulated by the control inputs u_{thr}, u_{lat}, u_{lon}. Flight tests show that this inner-loop command generator G_{c} is feasible,

which is given by.

$$G_c = \begin{bmatrix} 0 & 0 & -0.071 \\ 0 & 0.102 & 0 \\ -0.102 & 0 & 0 \end{bmatrix}.$$

7.4. *Outer-loop reference generation*

Considering the dynamics and constraints of the UAV platform, a smooth path generation mechanism is proposed to realize both velocity and position references. As shown in Fig. 27, the generated velocity $\mathbf{v}_{n,r} = [v_x, v_y, v_z]^T$ and position references $\mathbf{p}_{n,r} = [x, y, z]^T$ are fed into outer-loop controller module. Given the current state and destination waypoints, the maximum acceleration and maximum cruise speed between each two waypoints are specified by the GCS operator. As shown in Fig. 29, given the cruise speed, the flight duration from one waypoint to another waypoint can be easily calculated as T. Then by transforming the rectangle to a trapezoid, the area covered from t_{start} to t_{end} in the trapezoid is the same as the one covered by the

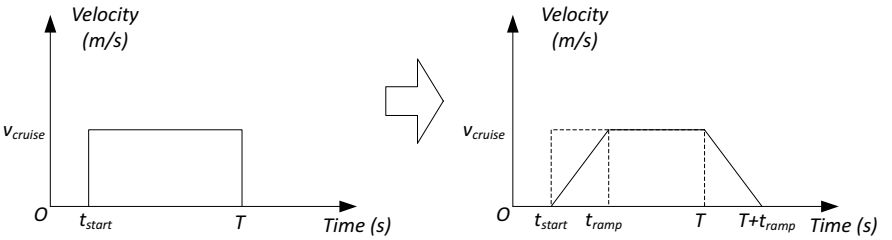

Fig. 29. Illustration of the velocity reference generation.

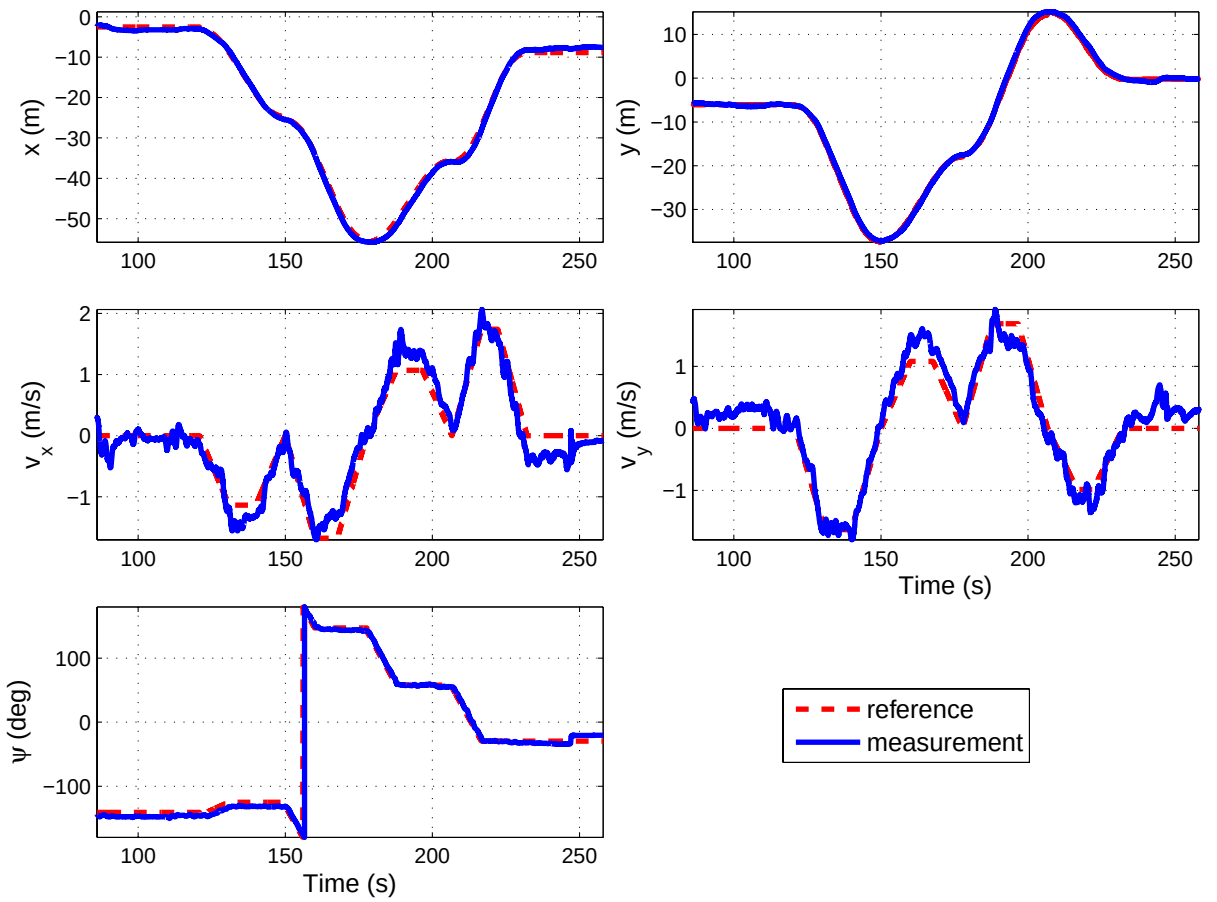

Fig. 30. Experimental results of the reference generation in flight.

rectangle from time t_{start} to T. It is clear that in the trapezoid, the velocity gradually accelerates from 0 to v_{cruise} from time t_{start} to t_{ramp} with a slope of the maximum acceleration value. The UAV will then transit into the cruise flight status with the speed of v_{cruise} with a time duration of $T - t_{\text{ramp}}$. In the de-acceleration stage, the speed reference decreases from v_{cruise} to 0 smoothly from time T to $T + t_{\text{ramp}}$. With this approach, both the velocity and position reference gradually change during the acceleration stage and de-acceleration stage with the prolonged flight duration time of $T + t_{\text{ramp}}$. Regarding the heading control, the heading reference is generated together with the velocity reference during the acceleration stage. With the online generated references, the corresponding reference of velocity and position given a specific time are provided for the outer-loop module in each control loop.

Figure 30 shows clearly the reference generation onboard with a total of four trapezoids which represent the four waypoints uploaded by the GCS operator on the Google Maps. Between every two waypoints, the longitudinal and lateral velocity reference signals increase and decrease smoothly and the corresponding position reference is also generated correctly.

7.5. *Semi-autonomous mode switch*

We have also implemented a so-called semi-autonomous flight mode of GremLion in case of emergency or more complicated situations. When GremLion is switched to this mode, the joystick signals on the RC transmitter directly represent the inner-loop references (see upper left corner of Fig. 28). This mode shares the same inner-loop control law with the fully autonomous mode. So the closed-loop system remains stable, but human intelligence will decide where the UAV should go in the next moment.

7.6. *Performance evaluation*

Here, to verify the robustness of the proposed flight control law, we conduct the simulation of the closed-loop system with the inclusion of wind gust. The wind gust input and the corresponding output responses are shown in Fig. 31. In the simulation process, the 20-second-long "$1 - \cos(\cdot)$"-type wind gust disturbance has been sequentially included to the *X*-, *Y*- and *Z*-direction of the body frame, with the peaking amplitude of 5 m/s, 5 m/s and 2 m/s, respectively. It can be clearly observed that the wind gust effect has been

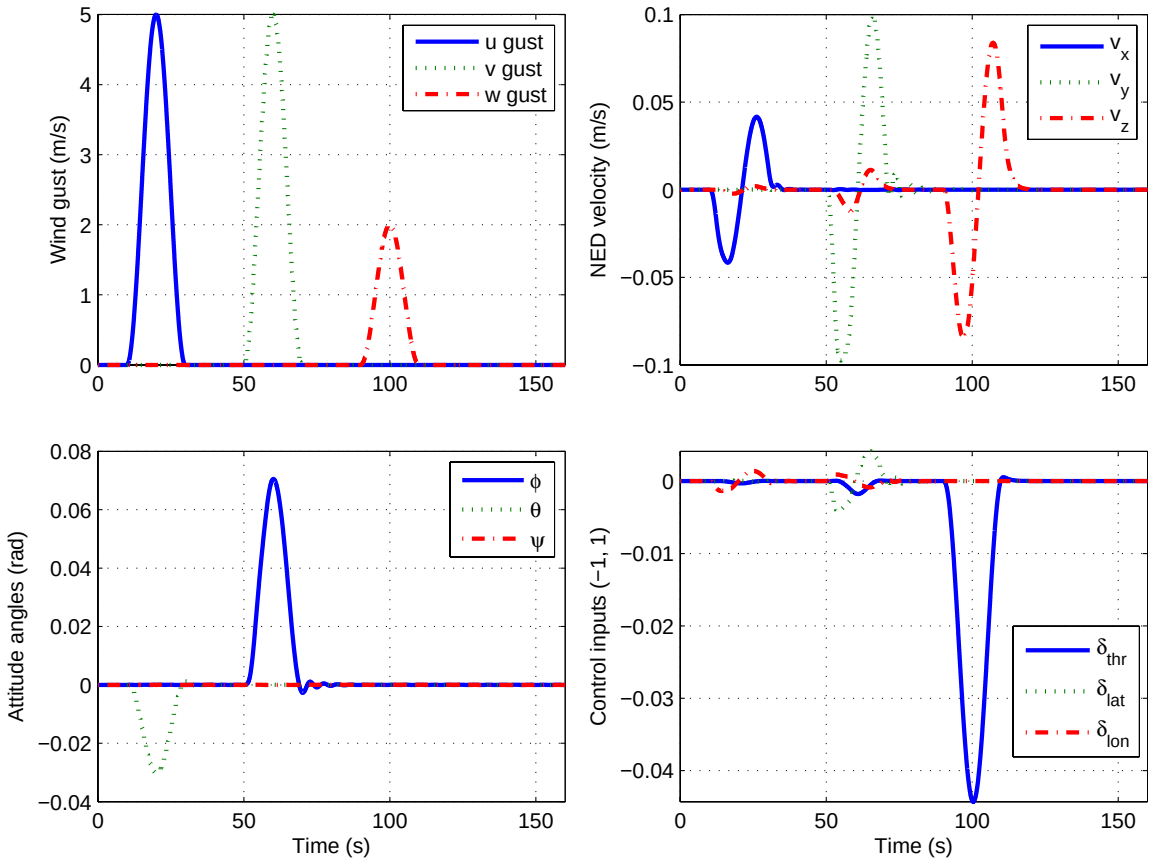

Fig. 31. Simulation results of the wind gust attenuation of the closed-loop system.

significantly attenuated by the proposed flight control law. The largest deviations for the Euler angles and velocity are only 0.0703 rad and 0.1 m/s, respectively. It is noted that the effect of the wind gust to the rotors has been ignored in the simulation since such effect is normally trivial in the near hover condition and other low-speed steady flight conditions.

7.7. *Flight results*

On the actual UAVForge competition day, UAVs from all teams were required to take-off at a place surrounded by trees with approximately 50 m height and fly all the way to the destination 2 miles away and perform landing at any suitable location for the next task: 3 h stationary observation. As the GPS location of the destination point was precisely provided by the competition organizer, and we chose to fly GremLion at a safe altitude (about 50 m high) to avoid crashing into the trees in the flight path, the whole path reference can be reasonably determined. The detailed path generation is as follows:

(i) Take the current x, y position as reference and ascend in the z-direction for 50 m at a speed of 1 m/s.
(ii) Hover for 10 s.
(iii) Adjust heading towards destination.
(iv) Hover for 10 s.
(v) Increase forward speed gradually (0.5 m/s^2) until 4.5 m/s.
(vi) Keep a forward speed of 4.5 m/s and fly towards the destination.
(vii) Before reaching the destination, decrease forward speed gradually (-0.5 m/s^2) back to 0 m/s.
(viii) Hover for 10 s.
(ix) Take the current x, y position as reference and descend in the z-direction at a speed of 0.5 m/s until 10 m above the ground.
(x) Adjust descending speed to 0.2 m/s and keep until touching the ground.
(xi) Motors slow down until fully shut down.

The above procedures had been tested before the competition day, but with lower altitude and shorter destination distance. The control law was sufficiently stable and the whole strategy was capable to finish the tasks. However, we failed to estimate the burden on the motors and ESCs while the platform kept ascending for 50 m on the actual

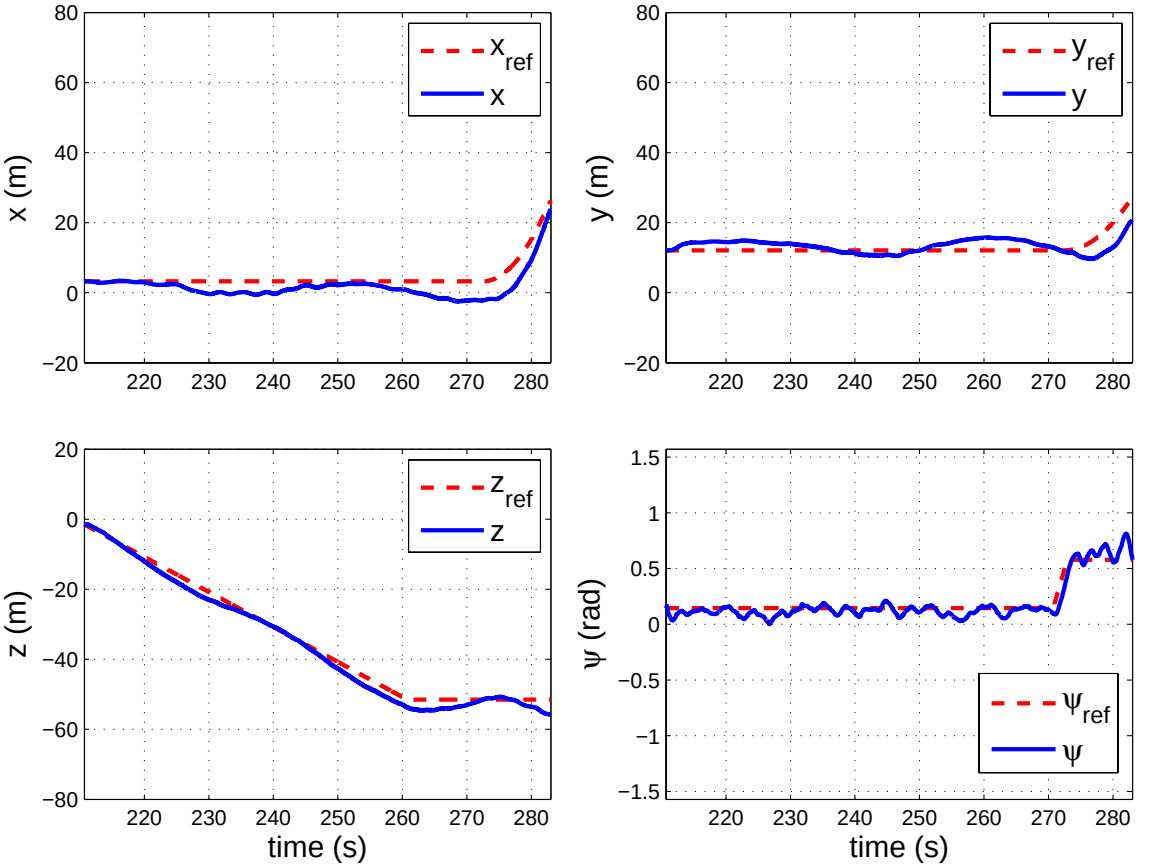

Fig. 32. Flight result: trajectory tracking — position and heading.

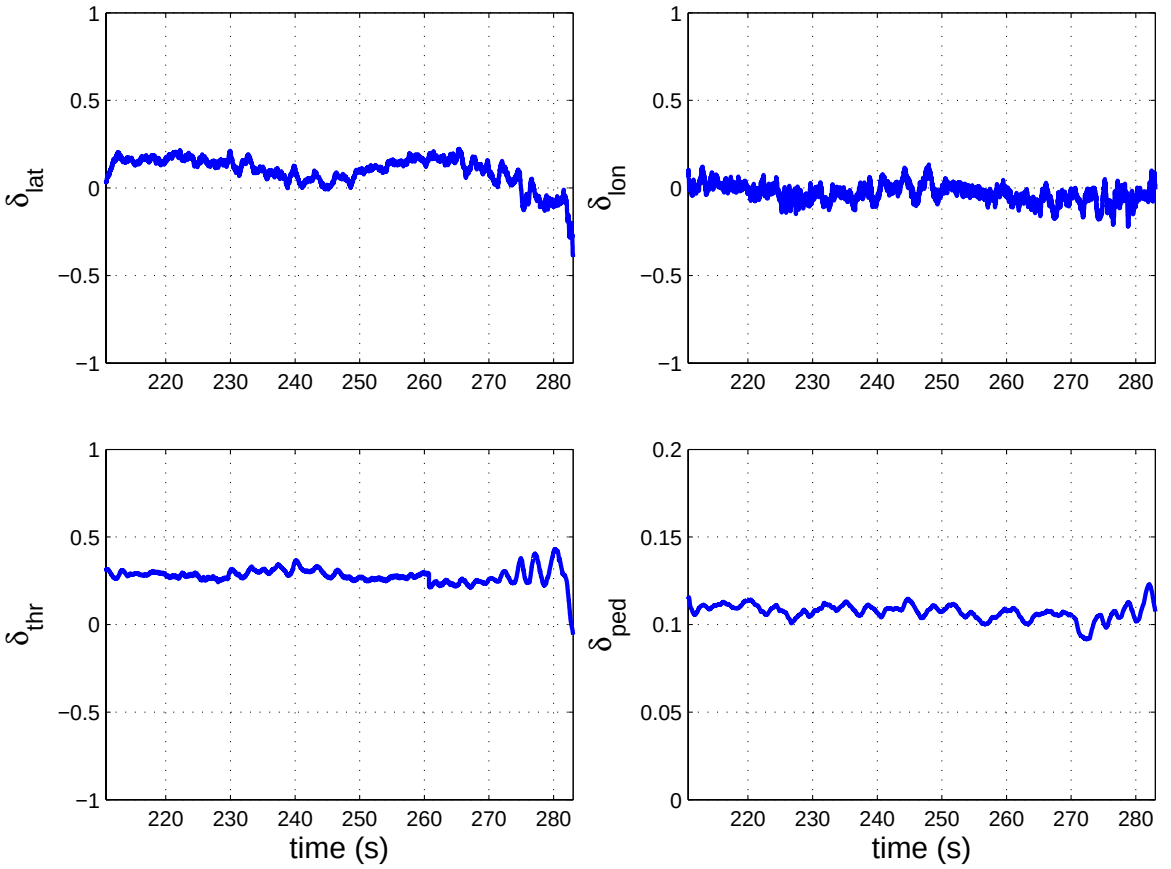

Fig. 33. Flight result: trajectory tracking — control effort.

competition day. The motor ESCs had been overheated when GremLion began to fly forward. This resulted in the ESCs cutting off temporarily and the consequence was disastrous as the platform could not stand such strong impulsive disturbance. The top and bottom rotors struck each other followed by a loss of lift. GremLion crashed into the trees and our attempt was suspended. Nonetheless, the on-board recorded data had been saved and transferred to PC and Figs. 32 and 33 show various signals just before the crash. GremLion's position, velocity and heading were following their respective references quite well until the vehicle crashed at $t = 285$ s.

8. Vision-Based Obstacle Detection and Avoidance

Besides the aforementioned autonomous flight control capability, GremLion was equipped with an onboard vision sensor to realize the obstacle detection and avoidance function, though it was not tested in the competition.

A variety of sensors can be used to detect obstacles, such as radar, sonar, laser scanner[28] and vision sensors.[29,30] We chose to use vision sensors to detect obstacles. That is because a vision sensor such as a monocular camera is light weight and low priced and can provide rich information of the environment.

If the obstacle is known in advance, it is possible to use pattern recognition to recognize the obstacle during flight. In order to do that, the knowledge of the obstacle, such as color or structure, should be input into the system and recognition algorithms should be well trained in advance. But for our case, the knowledge of obstacles is not available beforehand. We need to detect obstacles online.

There are several vision techniques which can be applied to realize obstacle detection in unknown environments, such as edge detection, image segmentation and 3D vision techniques. Our obstacle detection method is based on 3D vision techniques, more specifically, optical flow.[31–33] Optical flow can provide the velocity of the feature on the images. Given the state of the UAV, we can recover the depth of the features. Thus the depth of the scene in front of the UAV can be estimated. If the depth of certain parts of the scene is less than a prescribed threshold, then that part will be classified as an obstacle and obstacle avoidance procedures will be activated.

The main steps in our algorithm are feature extraction, feature matching, and depth estimation. The objective of

feature extraction and matching is to obtain the feature position on the image, and to calculate the feature velocity on the image. The details of the depth estimation will be given in the next section. The idea behind our depth estimation is based on structure from motion. More precisely, if the state (i.e., position, velocity and attitude) of the UAV can be measured, e.g., by GPS and inertial sensors, and the image of a 3D point can be matched between two images, then the 3D position of the 3D point can be determined. In our work, since we are more interested in the distance between the obstacle and the UAV, we focus on how to estimate the depth of the feature points. We can measure the state of the UAV using GPS and inertial sensors.

8.1. *Obstacle detection and depth estimation*

The coordinate frames involved in our depth estimation algorithm are given in Fig. 34. The notations used in this paper are given below.

- $P_c = [x_c, y_c, z_c]^T$: the coordinates of a 3D point P expressed in the camera frame. This is the unknown variable we are going to estimate.
- $\omega_c = [\omega_{x,c}, \omega_{y,c}, \omega_{z,c}]^T$: the angular velocity of the camera frame relative to the NED frame, with coordinates expressed in the camera frame.
- $v = [v_{x,c}, v_{y,c}, v_{z,c}]^T$: the linear velocity of the camera frame relative to the NED frame, with coordinates expressed in the camera frame.
- $p = [x, y]^T$: the image feature point of the 3D point on the ideal image plane (i.e., the depth of the points on this plane is one).

The angular rate ω_b and velocity v_b of the UAV resolved in the body frame can be obtained from GPS and inertial sensors. Then $\omega_c = R_{c/b}\omega_b$, $v_c = R_{c/b}v_b$, where $R_{c/b}$ is the rotation transformation from the body frame to the camera frame. The feature point position p can be obtained from feature detection algorithms. The feature velocity on the image $\dot{p}$ can be computed from optical flow.

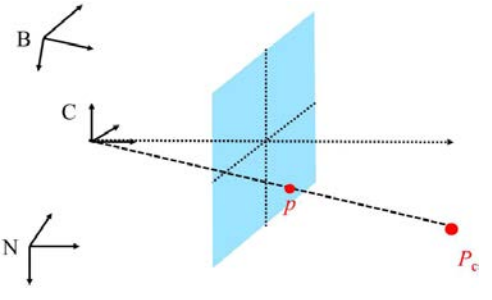

Fig. 34. Reference frames involved in depth estimation. N: NED frame, B: body frame, C: camera frame.

The following depth estimation algorithm is valid only for static scene, i.e., $\dot{P}_n = 0$. From

$$P_c = R_{c/n}(P_n - P_{c_0,n}),$$

and

$$\dot{R}_{c/n} = -[\omega_c]_\times R_{c/n},$$

we have

$$\begin{aligned}\dot{P}_c &= \dot{R}_{c/n}P_n - \dot{R}_{c/n}P_{c_0,n} - R_{c,n}\dot{P}_{c_0,n}\\ &= -[\omega_c]_\times R_{c/n}P_n + [\omega_c]_\times R_{c/n}P_{c_0,n} - v_c\\ &= -[\omega_c]_\times P_c - v_c.\end{aligned}$$

The operator $[*]_\times$ indicates a skew-symmetric matrix. Expanding the above equation into components gives

$$\begin{cases}\dot{x}_c = -v_{x,c} - \omega_{y,c}z_c + \omega_{z,c}y_c\\ \dot{y}_c = -v_{y,c} - \omega_{z,c}x_c + \omega_{x,c}z_c\\ \dot{z}_c = -v_{z,c} - \omega_{x,c}y_c + \omega_{y,c}x_c\end{cases}. \tag{19}$$

Substituting

$$x = \frac{x_c}{z_c}, \quad y = \frac{x_c}{z_c},$$

into Eq. (19) yields

$$\dot{z}_c = -v_{z,c} - \omega_{x,c}yz_c + \omega_{y,c}xz_c,$$

which is equivalent to

$$\dot{z}_c = (\omega_{y,c}x - \omega_{x,c}y)z_c - v_{z,c}. \tag{20}$$

On the other hand, we have

$$\dot{x} = \left(\frac{x_c}{z_c}\right)', \quad \dot{y} = \left(\frac{y_c}{z_c}\right)',$$

where $'$ denotes the derivative with respect to time. Substituting Eq. (19) into the above equation gives

$$\begin{pmatrix}\dot{x}\\ \dot{y}\end{pmatrix} = \frac{1}{z_c}\begin{bmatrix}-1 & 0 & x\\ 0 & -1 & y\end{bmatrix}\begin{pmatrix}v_{x,c}\\ v_{y,c}\\ v_{z,c}\end{pmatrix} + \begin{bmatrix}xy & -(1+x^2) & y\\ 1+y^2 & -xy & -x\end{bmatrix}\begin{pmatrix}\omega_{x,c}\\ \omega_{y,c}\\ \omega_{z,c}\end{pmatrix}.$$

The above equation can be rewritten as

$$b = A\frac{1}{z_c},$$

where

$$b = \begin{pmatrix}\dot{x}\\ \dot{y}\end{pmatrix} - \begin{bmatrix}xy & -(1+x^2) & y\\ 1+y^2 & -xy & -x\end{bmatrix}\begin{pmatrix}\omega_{x,c}\\ \omega_{y,c}\\ \omega_{z,c}\end{pmatrix},$$

$$A = \begin{bmatrix} -1 & 0 & x \\ 0 & -1 & y \end{bmatrix} \begin{pmatrix} v_{x,c} \\ v_{y,c} \\ v_{z,c} \end{pmatrix}.$$

So the depth z_c can be computed as

$$z_{c,\,\text{measure}} = \frac{1}{(A^T A)^{-1} A^T b}.$$

To summarize, we have

$$\dot{z}_c = (\omega_{y,c} x - \omega_{x,c} y) z_c - v_{z,c} + \varepsilon_1,$$
$$z_{c,\text{measure}} = z_c + \varepsilon_2.$$

The above first equation is the process model of the depth z_c. Note ω_c and v_c can be measured by GPS and inertial sensors; x and y can be obtained from feature extraction. The above second equation is the measurement model of the depth z_c. This measurement actually is a closed-form estimate of the depth. However, since there are measurement noises in x, y, $\dot{x}$, $\dot{y}$, ω_c and v_c, the closed-form estimate is also corrupted by noises. Therefore, a better solution is to combine the process model and measurement model together then apply an EKF.

8.2. *Simulation results of obstacle detection*

Next we present simulation results to verify the proposed depth estimation approach. The simulation scenario is as follows: suppose the UAV is equipped with a monocular forward-looking camera, which can take images of the obstacles in front of the UAV. The UAV flies towards a forest, where the trees are the obstacles to be detected. We have developed an image generation program. First, we have a large image (2048 × 1336 pixels) of the forest. According to the position and attitude of the UAV relative to the forest image, we then generate small images (320 × 240 pixels) to simulate the images taken by the forward-looking onboard camera. The depth of the obstacles in these generated images are detected by using the proposed approach.

The simulation results are shown in Fig. 35. The depth of every feature detected in the image is estimated using the proposed approach. Then we use grids to divide the image into 12 areas. The average value of all the depth of the features in each area is treated as the depth of that area.

For example, in the first image in Fig. 35, the depth of the areas in the first row is very large. Then these areas can be classified as collision-free. On the other hand, the areas in the three rows below have relatively small depth. Hence obstacles exist in these areas. When the depth of the obstacles is smaller than a user-defined threshold, the UAV may take actions such as flying upward in order to avoid the obstacles.

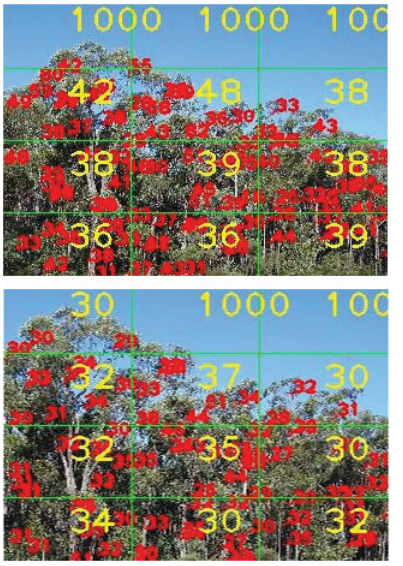

Fig. 35. Depth estimation results.

8.3. *Obstacle avoidance*

The obstacle avoidance strategy introduced here is to control the UAV to climb upwards high enough in the presence of obstacles and fly over the obstacles to the point above the waypoint to be visited, and then descend down to the waypoint. Considering a scenario illustrated in Fig. 36, the task of the UAV is to visit the next waypoint E starting from current point A. The UAV has a camera mounted on the front of the UAV to detect the obstacles on the flight path. As can be seen, there is a tree in the path from A to E. Since it is too far from the UAV current position A, at first, the UAV flies directly to waypoint E. When the UAV comes to point B on the path of AE, it detects that the tree is near enough and begins to ascend vertically to point C which is high above the tree. It then passes over the tree from above to point D which is the point above point E. It then descends vertically to point E.

Since the obstacle avoidance strategy is based on vision sensing, to describe the strategy more specifically, the geometry for the camera detection is given in Fig. 37. In Fig. 37, the current position of the UAV at point A in NED frame is represented as x_t, y_t, z_t, the next waypoint E for the

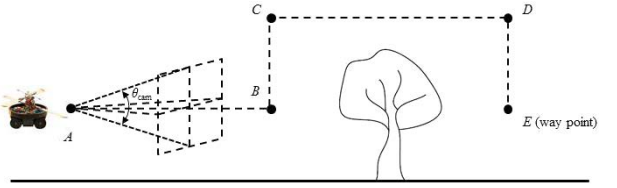

Fig. 36. Obstacle avoidance strategy.

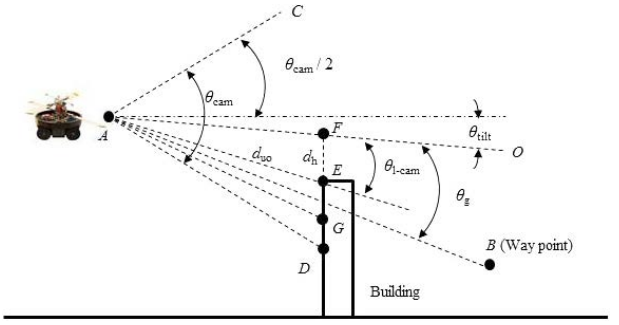

Fig. 37. Geometry for obstacle avoidance.

UAV to visit is represented as x_g, y_g, z_g. The vertical view angle $\angle DAC$ of the camera is θ_{cam}. AO is the optical axis of the camera. The UAV cannot move directly from point A to point B, because there is an obstacle, e.g., a building, standing in the way. So the camera can detect features of the building and the onboard vision computer can calculate the depth and the angle of the detected features. For example, Fig. 37 shows the UAV obtains the information of three features E, G and D, in which E is the highest feature point. The distance and angle to the optical axis of camera are denoted as d_{uo} and θ_{l-cam}, respectively. So the fan shaped area formed by $\angle EAC$ is clear for the UAV to fly. Considering energy saving, we do not expect the UAV to fly too high. On the other hand, the UAV can not fly too low relative to the top of the obstacle, otherwise it might collide with it if there is a fluctuation in the height direction. Taking disturbance into consideration, we need to let the UAV keep a minimum safe distance from the top of obstacle, here it is defined as d_h to fly over the obstacle to avoid collision with the obstacle. It should be noted that the UAV does not perform obstacle avoidance upon detecting an obstacle. We assume that only when the obstacle enters within a range, denoted as d_{min}, to the UAV, does it begin to avoid the obstacle.

The strategy can be described as follows:

(i) ***Flying to goal.*** In this mode, there is no obstacle in the path or detected obstacles will not bring any effect on the UAV flight. The UAV flies directly to the next waypoint. The conditions for this mode is

$$\theta_{l-cam} + \theta_{tilt} - \theta_g < 0, \tag{21}$$

and

$$d_{uo}|\sin(\theta_{ob})| \geq d_h, \tag{22}$$

where θ_{tilt} is the onboard tilt angle, θ_g is the angle of LOS to horizontal line. Equation (21) indicates that there is no obstacle on the LOS from the UAV to the waypoint. Equation (22) indicates that the highest point of the object is low enough, and the UAV can fly safely along LOS. When these two conditions are satisfied, the UAV enters the flying to goal mode. Commands in this mode are

$$\begin{cases} v_{x,n} = v\cos(\theta_g)\cos(\psi) \\ v_{y,n} = v\cos(\theta_g)\sin(\psi) \\ v_{z,n} = -v\sin(\theta_g) \end{cases}, \tag{23}$$

where v is the desired velocity of the UAV. $v_{x,n}$, $v_{y,n}$, $v_{z,n}$ are the referential velocity of the UAV along x-, y- and z-axes under the NED frame, respectively. ψ is the azimuth angle of the waypoint to the UAV. In the formula above, the negative sign of $v_{z,n}$ is caused by the direction of z-axis,

(ii) ***Ascending and flying over obstacle.*** When at least one of conditions in Eqs. (21) and (22) is not satisfied, the UAV decelerates to hover, and then ascends high above the obstacle for the UAV to fly over the obstacle. This condition can be expressed as

$$d_{uo}\sin(\theta_{l-cam} + \theta_{tilt}) \geq -d_h. \tag{24}$$

This indicates when Eq. (24) is satisfied, the UAV has reached the required height relative to the obstacle. The commands for ascending process is

$$\begin{cases} v_{x,n} = 0 \\ v_{y,n} = 0 \\ v_{z,n} = -v_{const} \end{cases}. \tag{25}$$

Here v_{const} is a given ascending velocity. Once the UAV ascends to the required height, it stops ascending and hovers. It then flies to a virtual goal point. The virtual goal has the same x, y coordinates with the next waypoint of the UAV and has the same altitude with the current z-coordinate of the UAV. It can be expressed as

$$\begin{cases} x_{vtu} = x_g \\ y_{vtu} = y_g \\ z_{vtu} = -z_t \end{cases}. \tag{26}$$

This means that when Eq. (24) is satisfied, the UAV will fly directly to the virtual point. The command for this process is

$$\begin{cases} v_{x,n} = v\cos(\psi) \\ v_{y,n} = v\sin(\psi) \\ v_{z,n} = 0 \end{cases}. \tag{27}$$

(iii) ***Descending.*** When the UAV arrives at the virtual point, it begins to descend vertically to the goal waypoint. The command is

$$\begin{cases} v_{x,n} = 0 \\ v_{y,n} = 0 \\ v_{z,n} = v_{const} \end{cases}. \tag{28}$$

Once the UAV reaches the goal waypoint, it can begin to visit the next waypoint in the same way described above.

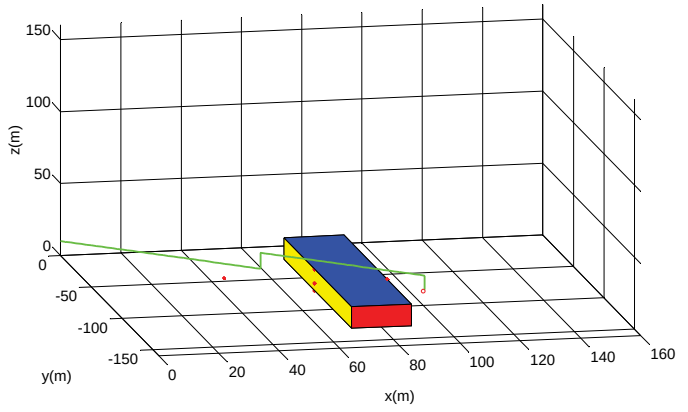

Fig. 38. (Color online) Simulation result for obstacle avoidance.

8.4. *Simulation results of obstacle avoidance*

Suppose that the start point of the UAV is $x_0 = 0\,\text{m}, y_0 = 0\,\text{m}$, $z_0 = -10\,\text{m}$, the goal or next waypoint is $x_g = 100\,\text{m}$, $y_g = -100\,\text{m}$, $z_g = -10\,\text{m}$, $\theta_{\text{cam}} = 50^\circ$, $d_h = 5\,\text{m}$, $d_{\min} = 15\,\text{m}$. There is a building standing in the way. Some feature points come from ground and the building are as follows:

$$\begin{aligned}
x_1 &= 45, \quad y_1 = -45, \quad z_1 = 0;\\
x_2 &= 70, \quad y_2 = -70, \quad z_2 = 0;\\
x_3 &= 70, \quad y_3 = -70, \quad z_3 = -5;\\
x_4 &= 70, \quad y_4 = -70, \quad z_4 = -10;\\
x_5 &= 70, \quad y_5 = -70, \quad z_4 = -15;\\
x_6 &= 90, \quad y_6 = -90, \quad z_2 = -15.
\end{aligned}$$

The simulation result is shown in Fig. 38. In Fig. 38, the red circle represents the goal or the waypoint for the UAV to visit. The red stars represent the detected features. The green solid green line is the flight path which shows the UAV flies over the building. The simulation demonstrates that the strategy is feasible.

9. Vision-Based Target Following

The UAVForge competition requires UAVs to perform a series of advanced behaviors, such as "Follow Me task". This task requires the UAV to autonomously follow a ground vehicle from the starting point back to the designated flight preparation area and maintain a safe altitude. The ground vehicle travels at about 15–30 mph. We adopt the onboard camera to achieve the vision-based target tracking. The initialization of the target location in the first frame is achieved by manually selecting a rectangular target box around the specified target. The selected target box is then extracted from the image, and the target tracking is implemented using the CAMSHIFT algorithm.[34] Based on the detected vision information, GremLion is then controlled to maintain the target in the center of the image and also follow the motion of the target.

9.1. *Target tracking in image*

The flow chart of the CAMSHIFT algorithm adopted is shown in Fig. 39. For each frame, the raw image is converted to a color probability distribution image via a color histogram model of the color being tracked, e.g., flesh color in the case of face tracking. The center and size of the color object are found via the CAMSHIFT algorithm operating on the color probability image. The current size and location of the tracked target are calculated and used to set the size and location of the search window in the next frame. The process is repeated for continuous tracking. This algorithm is a generalization of the mean shift algorithm, which is described as follows:

(i) Set the calculation region of the probability distribution from the whole image.

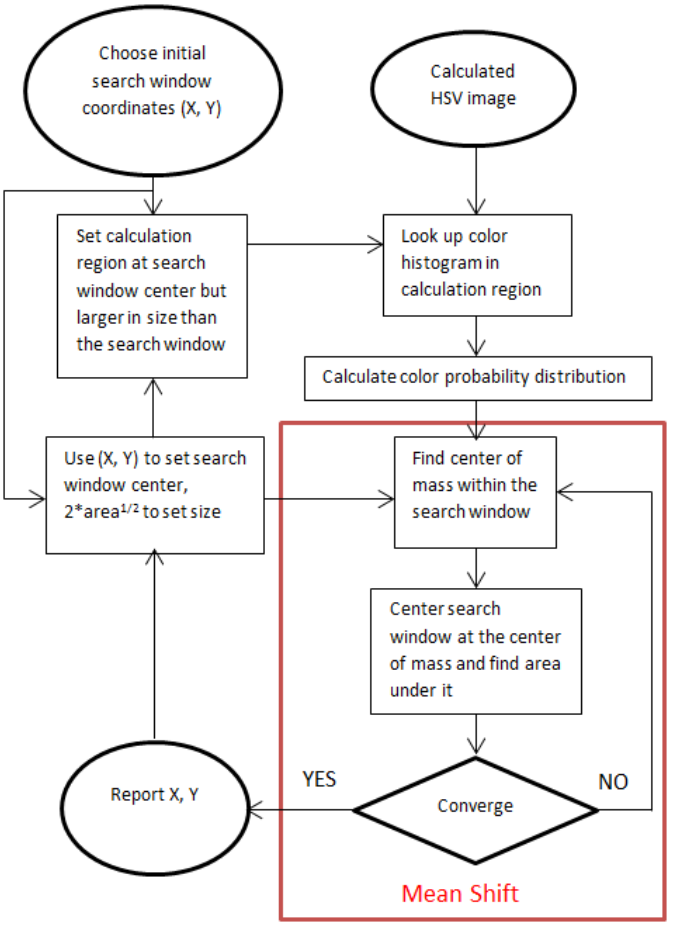

Fig. 39. The flowchart of the CAMSHIFT algorithm.

(ii) Choose the initial location of the 2D mean shift search window.
(iii) Calculate the color probability distribution in the 2D region centered at the search window location but slightly larger than the mean shift window size.
(iv) Run a mean shift algorithm to find the search window center. Store the moment (area or size) and center location.
(v) For the next video frame, center the search window at the mean location obtained in Step 4 and set the window size to a function of the moment found there. Go to Step 3.

CAMSHIFT operates on a 2D color probability distribution image produced from the histogram back projection.[35] The core part of the CAMSHIFT algorithm is the mean shift algorithm[36] that is a nonparametric feature-space analysis technique used frequently for clustering in computer vision applications. The mean shift algorithm is a procedure used for locating the maxima of a density function. It is an iterative method that is useful for detecting the modes of the density. In our application, it is used to detect the centroid of the selected target based on the color information.

However, unlike the mean shift algorithm that is designed for static distributions, CAMSHIFT is designed for dynamically changing distributions. In other words, CAMSHIFT is the use of mean shift calculation with an adaptive window size that finds the optimal location of the selected target. This occurs when objects in video sequences are being tracked and the object moves so that the size and location of the probability distribution changes in time. The CAMSHIFT algorithm adjusts the search window size in the course of its operation. The initial window size can be set at any reasonable value. Instead of using a predefined or externally adapted window size, CAMSHIFT relies on the moment information, extracted as part of the internal workings of the algorithm, to continuously adapt its window size within or over each video frame.

9.2. *Target localization in the image*

For discrete 2D image probability distributions, the mean location (the centroid) within the search window is found using the zeroth moment and the first moments[37] for x and y,

$$M_{00} = \sum_x \sum_y I(x,y), \tag{29}$$

$$M_{10} = \sum_x \sum_y xI(x,y), \tag{30}$$

$$M_{01} = \sum_x \sum_y yI(x,y), \tag{31}$$

where I is the pixel (probability) value in the position (x,y) in the image and x and y range over the search window. The mean location is then obtained

$$x_c = \frac{M_{10}}{M_{00}}, \tag{32}$$

$$y_c = \frac{M_{01}}{M_{00}}. \tag{33}$$

The 2D orientation of the probability distribution can be obtained easily by using the second moments in CAMSHIFT where the point (x,y) ranges over the search window. Second moments calculations are given by

$$M_{20} = \sum_x \sum_y x^2 I(x,y), \tag{34}$$

$$M_{02} = \sum_x \sum_y y^2 I(x,y). \tag{35}$$

The object orientation or the direction of the major axis is

$$\Theta = \frac{1}{2} \arctan \left[\frac{2\left(\frac{M_{11}}{M_{00}} - x_c y_c\right)}{\left(\frac{M_{20}}{M_{00}} - x_c^2\right) - \left(\frac{M_{02}}{M_{00}} - y_c^2\right)} \right]. \tag{36}$$

The first two eigenvalues which is length and width of the probability distribution of the blob can be calculated in a closed form as follows:

$$l = \sqrt{\frac{(a+c) + \sqrt{b^2 + (a-c)^2}}{2}}, \tag{37}$$

$$w = \sqrt{\frac{(a+c) - \sqrt{b^2 + (a-c)^2}}{2}}, \tag{38}$$

where

$$a = \frac{M_{20}}{M_{00}} - x_c^2, \tag{39}$$

$$b = 2\left(\frac{M_{11}}{M_{00}} - x_c y_c\right), \tag{40}$$

$$c = \frac{M_{02}}{M_{00}} - y_c^2. \tag{41}$$

With the calculated orientation, length and width, the CAMSHIFT algorithm adjusts the search window and track window based on these values.

9.3. *Simulation results*

The proposed target tracking algorithm has been verified by using a video of several moving cars. The simulation is

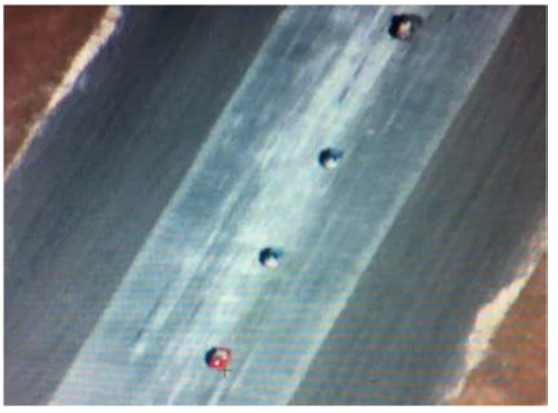

Fig. 40. Select the target in a frame.

Fig. 41. Track the target in subsequent frames.

carried out on the GCS with the following steps:

(1) The target is selected by clicking the four corners around the intended target (see Fig. 40). The coordinates of the selection are then sent to the onboard computer for real-time image tracking.
(2) The tracked target and the search area are shown on the GCS. The search area can be configured to be adaptive or static.
(3) The target is then tracked in real time within the captured frame, shown in Fig. 41.

The tracking error between the measured and actual target centroid in the video has been shown in Fig. 42. The simulation results show that the algorithm is able to track the target successfully, if the motion of the target can be approximated using a nonmaneuvering motion model. The tracking error is mainly caused by the target shadow that is detected as a part of the target. The proposed tracking algorithm can deal with the partial occlusion of the target.

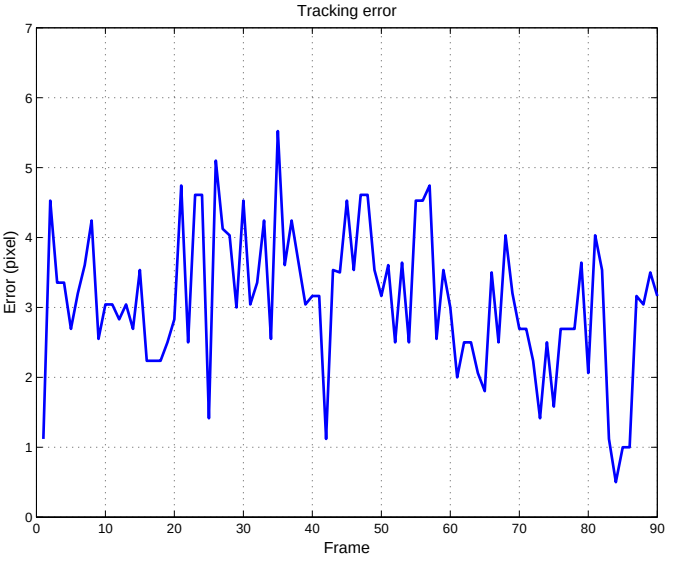

Fig. 42. The target tracking error in the video.

9.4. *Pan/tilt servo control*

Assume that P_s is the target in the servo-based coordinate system, which is given by

$$P_s = \begin{pmatrix} x_s \\ y_s \\ z_s \end{pmatrix} = \lambda R_{s/c}(v) \begin{pmatrix} \frac{x_i}{f_x} \\ \frac{y_i}{f_y} \\ 1 \end{pmatrix}, \tag{42}$$

where

$$R_{s/c} = \begin{bmatrix} \cos(v_\theta) & 0 & \sin(v_\theta) \\ 0 & 1 & 0 \\ -\sin(v_\theta) & 0 & \cos(v_\theta) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(v_\phi) & -\sin(v_\phi) \\ 0 & \sin(v_\phi) & \cos(v_\phi) \end{bmatrix} = \begin{bmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ r_7 & r_8 & r_9 \end{bmatrix}, \tag{43}$$

and

$$\lambda = z_c = \frac{h}{r_7 \frac{x_i}{f_x} + r_8 \frac{y_i}{f_y} + r_9}, \tag{44}$$

where x_i, y_i is the projection of target P in the image plane. $R_{s/c}$ is the rotation matrix from the camera frame to the servo-based frame. λ is scaling factor, which is the depth of the target P in the camera coordinate system. f_x and f_y are vertical and horizontal focal length measured in pixels. The

azimuth and elevation of the camera is calculated by

$$P_{\mathrm{e}} = \begin{pmatrix} p_\phi \\ p_\theta \end{pmatrix} = N(P_{\mathrm{i}}, v) = \begin{pmatrix} \sin^{-1}\left(\dfrac{y_{\mathrm{s}}}{r_{\mathrm{sp}}}\right) \\ \tan^{-1}\left(\dfrac{x_{\mathrm{s}}}{z_{\mathrm{s}}}\right) \end{pmatrix}, \tag{45}$$

where

$$r_{\mathrm{sp}} = \sqrt{x_{\mathrm{s}}^2 + y_{\mathrm{s}}^2 + z_{\mathrm{s}}^2}. \tag{46}$$

Thus, the input to the pan/tilt servo system is:

$$e(k) = \begin{pmatrix} e_\phi \\ e_\theta \end{pmatrix} = P_{\mathrm{e}} - P_{\mathrm{e}}^*, \tag{47}$$

where P_e^* is the current orientation of the camera which is set at the center of image. The coordinate of the target in the body frame is

$$\begin{pmatrix} x_{\mathrm{tg}} \\ y_{\mathrm{tg}} \\ z_{\mathrm{tg}} \end{pmatrix}_{\mathrm{b}} = R_{\mathrm{s/c}}(v) \begin{pmatrix} \dfrac{x_{\mathrm{i}}}{f_{\mathrm{x}}} \\ \dfrac{y_{\mathrm{i}}}{f_{\mathrm{y}}} \\ 1 \end{pmatrix} z_{\mathrm{c}}. \tag{48}$$

Define the expected distance for target tracking in the body system is $c_{\mathrm{x}} = d$, $c_{\mathrm{y}} = 0$, and the flight altitude of the UAV is h_0. ψ_0 is the initial heading of the UAV pointing to the target. The tracking position in the NED frame is given by

$$\begin{pmatrix} x_{\mathrm{uav}} \\ y_{\mathrm{uav}} \\ z_{\mathrm{uav}} \end{pmatrix}_{\mathrm{ref,n}} = \begin{pmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} R_{\mathrm{n/b}} \begin{pmatrix} x_{\mathrm{tg}} \\ y_{\mathrm{tg}} \\ 0 \end{pmatrix} - \begin{pmatrix} c_{\mathrm{x}} \\ 0 \\ 0 \end{pmatrix} \\ h_0 \end{pmatrix} + \begin{pmatrix} x_{\mathrm{uav,c}} \\ y_{\mathrm{uav,c}} \\ 0 \end{pmatrix}, \tag{49}$$

where $x_{\mathrm{uav,\,c}}$, $y_{\mathrm{uav,\,c}}$ is the current position of the UAV. The velocity reference in the NED frame is given by

$$\begin{pmatrix} \Delta\dot{x}_{\mathrm{x,n}} \\ \Delta\dot{y}_{\mathrm{y,n}} \\ \Delta\dot{z}_{\mathrm{z,n}} \\ \dot{\psi} \end{pmatrix} = \begin{pmatrix} [1 \quad 0 \quad 0] R_{\mathrm{n/b}} \Delta V_{\mathrm{b}} \\ 0 \\ (h_0 - h)/\mathrm{T_s} \\ k\dot{\theta} \end{pmatrix}, \tag{50}$$

where $\theta = \tan^{-1}(\frac{y_{\mathrm{uav}}}{x_{\mathrm{uav}}})$, $k \in [0, 1]$ is a weighting factor.

10. Conclusion and Future Work

In this paper, the comprehensive design and implementation of GremLion has been presented, including hardware construction, software development, navigation and control, as well as vision related mission algorithms. The flight test results of GremLion in the UAVForge competition have been presented too. Although the development of GremLion system is generally challenging, and time consuming, but we believe the developed avionics, GCS, and mission algorithms can be extended to other unmanned vehicles or robots smoothly in future. The flight control law has been verified in the real flight tests, including hovering, climbing, turning, and forward flight. In addition, the semi-auto flight scheme and the GPS waypoint navigation have been realized too. The semi-auto flight schemes will be useful for human operators to control the UAV from the ground control station. We have implemented proposed mission algorithms on the onboard vision computer, including obstacle avoidance and target tracking. These algorithms are definitely useful for us in future, though we did not test them in the competition.

Considering the requirements on various practical implementations, extensive contributions could be achieved by extending the GremLion research in the following directions. Further experiment and research work is required to obtain a high-fidelity dynamics model of GremLion in the different operation conditions. That is also important for the automatic control law design. A 3G video and data transmission system is required to realize the non-LOS communication that is required in cluttered areas, such as urban environments. It is necessary to further improve the vision-based obstacle avoidance and tracking algorithms. Tests of these algorithms in real flight will be conducted in future.

Acknowledgment

This research is funded by DSO National Laboratories, Singapore.

References

[1] G. Cai, B. M. Chen and T. H. Lee, *Unmanned Rotorcraft Systems* (Springer, New York, 2011).
[2] G. W. Cai, F. Lin, B. M. Chen and T. H. Lee, Development of fully functional miniature unmanned rotorcraft systems, in *Proc. 29th Chinese Control Conf.*, Beijing, China (2010), pp. 1–6.
[3] F. Kendoul, Survey of advances in guidance, navigation, and control of unmanned rotorcraft systems, *J. Field Robot.* **29** (2012) 315–378.
[4] J. Keller, D. Thakur, V. Dobrokhodov, K. Jones, M. Pivtoraiko, J. Gallier, I. Kaminer and V. Kumar, A computationally efficient approach to trajectory management for coordinated aerial surveillance, *Unmanned Syst.* **1** (2013) 59–74.
[5] A. N. Kopeikin, S. S. Ponda, L. B. Johnson and J. P. How, Dynamic mission planning for communication control in multiple unmanned aircraft teams, *Unmanned Syst.* **1** (2013) 41–58.
[6] Visiongain, The unmanned aerial vehicles (uav) market 2012–2022, Tech. Rep., Visiongain, London, UK (2012).

[7] Z. Sarris and S. Atlas, Survey of UAV applicatins in civil markets, in *Proc. 9th Mediterranean Conf. Control and Automation*, Vol. WA2-A, Dubrovnik, Croatia (2001), pp. 1–11.

[8] B. Enderle, Commercial applications of UAV's in Japanese agriculture, in *Proc. AIAA 1st UAV Conf.*, No. AIAA-2002-3400, Portsmouth, Virginia, 20–23 May 2002.

[9] B. Ludington, E. Johnson and G. Vachtsevanos, Augmenting UAV autonomy, *IEEE Robot. Autom. Mag.* **13**(3) (2006) 63–71.

[10] M. E. Campbell and W. W. Whitacre, Cooperative tracking using vision measurements on seascan UAVs, *IEEE Trans. Control Syst. Technol.* **15**(4) (2007) 613–626.

[11] U. S. Army, Unmanned aircraft systems roadmap 2010–2035, Tech. Rep., Office of the Secretary of Defense (2010).

[12] J. Hu, J. Xu and L. Xie, Cooperative search and exploration in robotic networks, *Unmanned Syst.* **1** (2013) 121–142.

[13] P. Doherty, F. Heintz and J. Kvarnstrom, High-level mission specification and planning for collaborative unmanned aircraft systems using delegation, *Unmanned Syst.* **1** (2013) 75–119.

[14] T. Wongpiromsarn, U. Topcu and R. M. Murray, Synthesis of control protocols for autonomous systems, *Unmanned Syst.* **1** (2013) 75–119.

[15] A. Koehl, H. Rafaralahy, M. Boutayeb and B. Martinez, Aerodynamic modelling and experimental identification of a coaxial-rotor uav, *J. Intell. Robot. Syst.* **68** (2012) 53–68.

[16] B. Mettler, M. Tischler and T. Kanade, System identification of a model-scale helicopter, Tech. Rep. CMU-RI-TR-00-03, Robotics Institute, Pittsburgh, PA, January 2000.

[17] G. Cai, F. Lin, B. M. Chen and T. H. Lee, Systematic design methodology and construction of UAV helicopters, *Mechatronics* **18**(10) (2008) 545–558.

[18] C. Harris, *Shock and Vibration Handbook* (McGraw-Hill, New York, 1996).

[19] D. J. Kruglinski, *Inside Visual C++*, 4th edn. (Microsoft Press, Washington, 1995).

[20] P. Liu, X. Dong, B. M. Chen and T. H. Lee, Development of an enhanced ground control system for unmanned aerial vehicles, in *Proc. IASTED Int. Conf. Engineering and Applied Science*, Colombo, Sri Lanka, December 2013, pp. 136–143.

[21] M. L. Civita, W. Messner and T. Kanade, Modeling of small-scale helicopters with integrated first principles and integrted system identification techniques, *58th Forum of American Helicopter Society*, Montreal, Canada (2002).

[22] V. Gavrilets, B. Mettler and E. Feron, Nonlinear model for a small-size acrobatic helicopter, *AIAA Guidance Navigation and Control Conf.*, Montreal, Canada (2001).

[23] G. D. Padfield, *Helicopter Flight Dynamics: The Theory and Application of Flying Qualities and Simulation Modeling* (AIAA Press, Reston, VA, 1996).

[24] R. P. Cheng, M. B. Tischler and G. J. Schulein, Rmax helicopter state-space model identification for hover and forward-flight, *J. Am. Helicopter Soc.* **51** (2006) 202–210.

[25] S. K. Kim and D. M. Tilbury, Mathematical modeling and experimental identification of an unmanned helicopter robot with flybar dynamics, *J. Robot. Syst.* **21** (2004) 95–116.

[26] B. M. Mettler, M. B. Tischler and T. Kanade, System identification modeling of a small-scale unmanned rotorcraft for control design, *J. Am. Helicopter Soc.* **47** (2002) 50–63.

[27] B. M. Chen, *Robust and H_∞ Control* (Springer-Verlag, London, 2000).

[28] K. Kang and J. V. R. Prasad, Development and flight test evaluations of an autonomous obstacle avoidance system for a rotary-wing uav, *Unmanned Syst.* **1** (2013) 3–19.

[29] D. W. Yoo, D. Y. Won and M. J. Tahk, Optical flow based collision avoidance of multi-rotor uavs in urban environments, *Int. J. Aeronaut. Space Sci.* **12** (2011) 252–259.

[30] H. Lang, M. T. Khan, K. K. Tang and C. W. de Silva, Developments in visual servoing for mobile manipulation, *Unmanned Syst.* **1** (2013) 143–162.

[31] H. C. Longuet-Higgins and K. Prazdny, The interpretation of a moving retinal image, *Proc. R. Soc. Lond. B* **208** (1980) 385–397.

[32] A. M. Waxman, B. Kamgar-Parsi and M. Subbarao, Closed-form solutions to image flow equations for 3D structure and motion, *Int. J. Comput. Vis.* **1** (1988) 239–258.

[33] L. Matthies, T. Kanade and R. Szeliski, Kalman filter-based algorithms for estimating depth from image sequences, *Int. J. Comput. Vis.* **3** (1989) 209–238.

[34] J. G. Allen, R. Y. Xu and J. S. Jin, Object tracking using camshift algorithm and multiple quantized feature spaces, in *Proc. Pan-Sydney Area Workshop on Visual Information Processing* (Australian Computer Society, Inc., 2004), pp. 3–7.

[35] M. J. Swain and D. H. Ballard, Indexing via color histograms, in *Computer Vision, 1990. Proc., Third Int. Conf.* (IEEE, 1990), pp. 390–393.

[36] D. Comaniciu and P. Meer, Mean shift: A robust approach toward feature space analysis, *IEEE Trans. Pattern Anal. Mach. Intell.* **24**(5) (2002) 603–619.

[37] C. W. Niblack, R. Barber, W. Equitz, M. D. Flickner, E. H. Glasman, D. Petkovic, P. Yanker, C. Faloutsos and G. Taubin, Qbic project: Querying images by content, using color, texture, and shape, *IS&T/SPIE's Symp. Electronic Imaging: Science and Technology* (International Society for Optics and Photonics, 1993), pp. 173–187.