Unmanned Systems, Vol. 3, No. 1 (2015) 63–87

DOI: 10.1142/S2301385015500053

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

Development of an Unmanned Helicopter for Vertical Replenishment

Fei Wang*, Peidong Liu†, Shiyu Zhao‡, Ben M. Chen§, Swee King Phang¶, Shupeng Lai‖, Tao Pang**, Biao Wang††, Chenxiao Cai‡‡, Tong H. Lee§§

Unmanned Systems Research Group, National University of Singapore, E4A-03-04, 3 Engineering Drive 3, Singapore 117582, Singapore

This paper presents an intelligent and robust guidance, navigation and control solution for a rotary-wing UAV to carry out an autonomous cargo transportation mission between two moving platforms. Different from the conventional GPS/INS-only navigation scheme, this solution also integrates sophisticated Lidar and vision systems capable of precisely locating cargo loading and unloading positions. Besides, another complementary GPS/INS system is set up on the moving platforms with communication to the unmanned helicopter so that the controlled UAV is able to follow the dynamic platforms with good tracking performance. The whole system has been successfully implemented, and with its superb performance the Unmanned Systems Research Group from the National University of Singapore won the first place in the final round of the rotary-wing category competition of the 2nd AVIC Cup — International UAV Innovation Grand Prix 2013.

Keywords: UAV vertical replenishment; guidance navigation and control; vision-based guidance.

US

1. Introduction

Rotorcrafts are often used for cargo transportation and vertical replenishment between seaborne vessels (see Fig. 1). The conventional way involves a manned helicopter, which relies on skills and experiences from a well-trained human pilot. The recent advancement of unmanned aerial vehicles (UAVs) however has opened the possibility of using unmanned rotorcrafts for this kind of cargo transportation tasks, which can reduce both risk and cost to a large extent.

One example of such UAV transportation application is AirMule UAV from UrbanAero. It is able to transport up to 500 kg of cargo to places as far as 50 km away and was used to transport cargo in Israel for military purposes [1]. In [2], an innovative control method has been proposed to solve the general UAV slung load problem. A group of researchers have also proposed the estimation of load position and velocity in such system [3]. The cargo transportation problem can also be solved by a rigid claw mechanism such as those appeared in [4, 5]. Besides, some researchers have also investigated the ability of load transporting with the collaboration of multiple UAVs [6, 7]. With this cooperative structure, the size and cost of each individual UAV can be reduced.

However, when solving this UAV cargo transportation problem, most of the existing works assume that the loading and unloading positions are accurately known. This assumption is reasonable in a few occasions where the environment is fully in control, but may not be valid for the more general cases. To expand the horizon of applications a small-scale UAV can do, an intelligent navigation and guidance system which can provide high-quality measurements and guidance information for UAV automatic flight control needs to be developed. One elegant solution is to integrate a computer vision sub-system for target searching

Received 14 May 2014; Revised 21 November 2014; Accepted 21 November 2014; Published 27 January 2015. This paper was recommended for publication in its revised form by editorial board member, Chang Chen.
Email Addresses: *wangfei@nus.edu.sg, †elelp@nus.edu.sg, ‡szhao@tx.technion.ac.il, §bmchen@nus.edu.sg, ¶king@nus.edu.sg, ‖shupenglai@nus.edu.sg, **tslpt@nus.edu.sg, ††billwang110@gmail.com, ‡‡ccx5281@vip.163.com, §§eleleeth@nus.edu.sg

Fig. 1. Rotorcraft vertical replenishment (U.S. Navy Photo: Use of released U.S. Navy imagery does not constitute product or organizational endorsement of any kind by the U.S. Navy).

Fig. 2. Cargo platform in UAVGP.

and tracking. In fact, vision-based target detection and localization have been investigated intensively. Some of them rely on visual targets with special shapes and features, such as [8] in which range estimation has been carried out based on specific geometric features including points, lines and curves. Others target on more general objects such as a helipad [9], a mobile ground vehicle [10, 11] or another UAV [12]. In addition, there is also a trend in integrating visual information in feedback control for mobile robot autonomous grasping and manipulation [13].

Although abundant solutions in solving the aforementioned individual problems have been proposed in literature, there is little research result on combining them together and implementing a fully functional cargo transportation UAV with all the useful characteristics and capabilities. In this paper, we propose a comprehensive UAV cargo transportation system which incorporates a small-size single-rotor helicopter with onboard sensors and processors, an innovative cargo grabbing mechanism, a set of UAV autonomous guidance, navigation and control (GNC) algorithms, and a cargo searching and localization vision sub-system.

The developed UAV system, named NUS^{2}T-Lion, has taken part in the 2nd AVIC Cup — International UAV Innovation Grand Prix (UAVGP), which was held in Beijing in September 2013. In this competition, the rotary-wing UAVs from various participating teams are required to automatically transport cargos between two parallel moving ships. The cargos are in the form of buckets with handles and they are initially placed within colored circles drawn on the surface of the first ship. Circles with a different color are drawn on the other ship, indicating the unloading positions. The ships are simulated by ground platforms moving on railways (see Fig. 2). Figure 3 shows a snap shot of NUS^{2}T-Lion carrying the cargo bucket in this Grand Prix.

The structure of this paper is as follows. Section 2 talks about design and integration of the UAV hardware system. Secs. 3–5 expand the UAV control, navigation and guidance algorithms respectively. Section 6 explains how to implement the GNC algorithm by the onboard software with the complete mission logics. Section 7 provides the flight test data to verify the fidelity and performance of the overall system. Concluding remarks are made in Sec. 8.

2. Hardware Configuration

The hardware configuration of NUS^{2}T-Lion follows the rotorcraft UAV structure proposed in [14]. As illustrated in Fig. 4 in which each block represents an individual hardware device, the whole system is constituted by four main parts, namely a bare rotorcraft platform, an onboard avionic system, a manual control system and a ground control system (GCS). While the manual control system and the GCS

Fig. 3. NUS^{2}T-Lion transporting a bucket cargo.

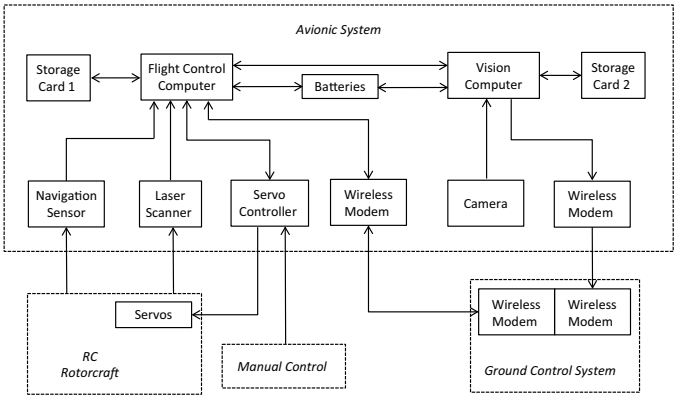

Fig. 4. Hardware configuration of NUS^{2}T-Lion rotorcraft system.

are quite standard for all kinds of UAV systems, the choices of the bare rotorcraft platform and its onboard avionic system are usually application dependent. For this case, they should be selected and integrated specifically for the UAV cargo transportation task. It is believed that by designing the hardware configuration effectively, difficulties for the later software algorithm development can be minimized.

2.1. *Bare rotorcraft platform*

The Thunder Tiger Raptor 90 SE Nitro radio-controlled (RC) helicopter is adopted as the bare rotorcraft platform in this work. It is a hobby-level single rotor helicopter originally designed for acrobatic flights. As compared with other commercial off-the-shelf (COTS) RC rotorcrafts such as Turbulence D3 and Observer Twin, Raptor 90 SE provides a reliable structural design and equivalent flight performance, at approximately half the price.

However, with the original Raptor 90's nitro engine and nitro fuel tank, the endurance of the UAV can barely reach 8 min with full load avionics. This is not sufficient for practical applications. To overcome this limitation, the original nitro engine is replaced by a gasoline counterpart, which is a product from Zenoah with model number G270RC. With the more efficient gasoline engine, a full-tank Raptor 90 can fly up to 30 min. This greatly widens the range of potential applications this UAV can do and it is especially beneficial to the cargo transportation task.

Unfortunately, this endurance improvement comes with two trade-offs. First, the vibration of the whole platform intensifies due to the gasoline engine. Second, the ignition magnet inside Zenoah G270RC is so large that its magnetic field can badly affect the onboard sensors. To overcome the vibration issue, wire rope isolators are used to protect the onboard avionics and filter out unwanted high frequency noises. The solution will be discussed in Sec. 2.3. For the problem of magnetic interference, the final solution is to replace the electro-magnetic ignition system inside the engine with a pure electric ignition system. With this modification, the onboard sensors, especially the magnetometer, all work in the way they originally should.

To cope with the cargo transportation task, there must be a loading mechanism integrated into the helicopter platform. By comparing the solution of a rigid claw-like grabbing mechanism and a long flexible rope hooking mechanism, the former is more precise in picking up the cargos, while the latter can avoid descending the UAV too low to the ship surface where the aerodynamic ground effect becomes significant.

In this work, an innovative design incorporating advantages from both sides has been proposed. The solution is a claw-like grabbing mechanism with very long arms (see Fig. 5). With this design, the UAV can keep a safe distance to the ship surface, and at the same time, grab and release the cargo in a precise and reliable way. Another highlight of this design is its omnidirectional feature, meaning no matter in which direction the cargo handle is oriented, it is not necessary for the UAV to adjust its heading to align accordingly. This saves time and minimizes unnecessary UAV maneuvers.

In addition, this design features a self-locking mechanism commonly used in landing gears of hobby-grade fixed-wing planes. The mechanism is enclosed in the rectangular boxes as shown in Fig. 5 with each box supports one arm and is powered by one servo motor. When the claw fully opens or closes, there is a slider inside the box to lock the position of the servo motor. In this way, the servo motors consume zero power while carrying a heavy cargo.

A load sensing mechanism which can differentiate a successful cargo loading from a failure is also installed. This mechanism acts as a safeguard in cases where the UAV makes a grasping action but the targeted cargo is not loaded successfully. By knowing that the cargo loading is unsuccessful, the UAV can descend and try grasping the cargo again. The detailed design is shown in Fig. 6, where four limit switches, which send out electrical signals when

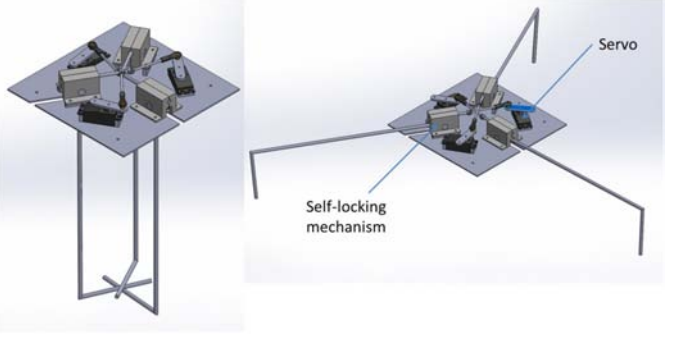

Fig. 5. Grabbing mechanism in closed and open configurations.

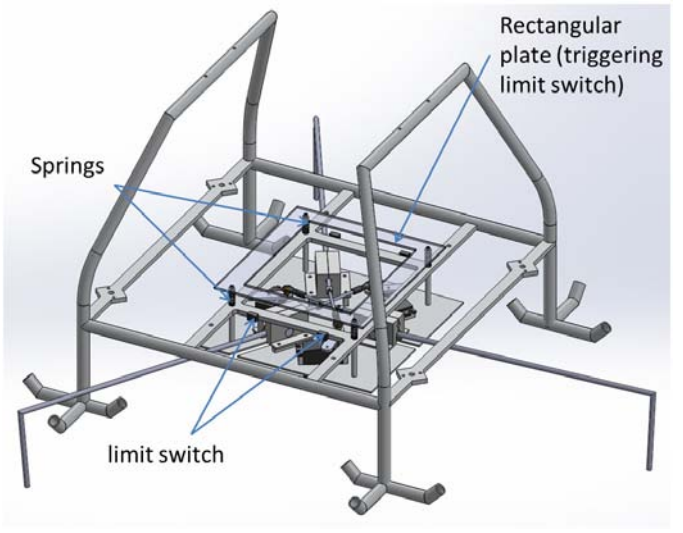

Fig. 6. Landing gear with bucket grabbing and load sensing functions.

pushed down, are installed on the customized landing skid. The baseplate of the claw is rigidly attached to a hollow rectangular plate on its top. The rectangular plate is then resting on the cross-over beams of the landing skid via four springs. When the claw is loaded, the rectangular plate compresses the spring and trigger one or more of the limit switches. When the claw is unloaded, the springs push up the rectangular plate to release the limit switches.

2.2. *Avionic system*

To realize fully autonomous flight, an onboard avionic system with sensors, processors and other electronic boards has to be designed. All components used on NUS^{2}T-Lion are the carefully chosen COTS products up to date. Figure 7 gives a complete view of the onboard system with the key components indicated. The details and usage of these components are explained as follows.

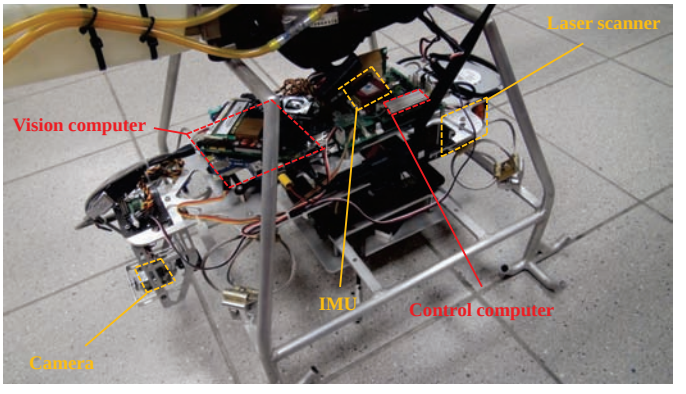

Fig. 7. Onboard avionic system of NUS^{2}T-Lion.

2.2.1. *Onboard sensors*

The IG-500N GPS/INS (GPS aided inertial navigation system) unit is chosen as the fundamental navigation sensor for NUS^{2}T-Lion. IG-500N is one of the world's smallest GPS enhanced attitude and heading reference system (AHRS) embedded with an extended Kalman filter (EKF). It includes a microelectromechanical systems (MEMS) based IMU, a GPS receiver and a barometer. It is able to provide precise and drift-free 3D orientation and position even during aggressive maneuvers, updated at 100 Hz. With its presence, the UAV's attitude, velocity and position can be consistently obtained, despite the fact that the position measurement from IG-500N alone is not accurate enough for the precise cargo loading and unloading task.

The second main sensor used onboard of NUS^{2}T-Lion is the mvBlueFOX camera from Matrix Vision. It is a compact industrial CMOS camera, compatible to any computers with USB ports. A superior image quality makes it suitable for both indoor and outdoor applications. In addition, it incorporates field-programmable gate array (FPGA), which reduces the computer load to minimum during image preprocessing. The standard Hi-Speed USB interface guarantees an easy integration without any additional interface board. In this specific cargo transportation application, it is the main guidance sensor for locating the cargos and their unloading points.

For cargo transportation applications, height measurement from GPS/INS or barometer may not be accurate enough for the UAV to pick up or drop the cargo appropriately. The UAV may even crash onto the surface of the cargo platform because of inaccurate height measurement, resulting in catastrophic consequences. While vision sensor or 1-D laser range finder may accomplish the task, the former can only be relied on when the visual target is within the field of view and the latter cannot handle ground surfaces with scattered obstacles. To make the height measurement accurate and consistent, a scanning laser range finder is the best choice. The laser scanner codenamed URG-30LX from Hokuyo is installed in the system. It has a maximum range of 30 m with fine resolution of 50 mm and it can scan its frontal 270° fan-shaped area with a resolution of 0.25°.

2.2.2. *Onboard computers*

There are two onboard computers in the avionic system; one for the implementation of guidance, navigation and control algorithms, and the other more powerful one

dedicated for vision processing. With this dual-computer structure, the vision algorithm can be implemented and tested separately at the development stage and it is very convenient to upgrade to a more powerful vision computer in future without modifying the control hardware and software system. It also improves the reliability of the overall system since this structure ensures control stability even when the vision computer malfunctions or encounters run-time errors. It happens more frequently on the vision computer compared to the control counterpart because the vision algorithm usually involves more sophisticated calculations and logics. If it ever happens, the UAV should still fly safely with the control computer alone and there will be enough time for human pilot to take over and land the UAV safely.

For the onboard control computer, 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 GCS as well as data logging. Being a light-weight yet powerful embedded computer for real-time tasks, the Gumstix Overo Fire embedded computer is selected for this purpose. It has a main processor running at 720 MHz and a DSP coprocessor. The main processor is an OMAP3530 ARM chip from Texas Instruments and it is one of the fastest low-power embedded processor as of writing. Moreover, it has built-in Wi-Fi module which saves the weight of an additional communication device.

For the onboard vision computer, it is mainly for implementing image processing algorithms, including color segmentation, object identification, object tracking and localization. Image processing tasks are usually computationally intensive and hence require powerful processors to run the algorithms in real time. We have chosen the Mastermind computer from Ascending Technologies. It has an Intel Core i7 processor but is still small and light enough to be carried by NUS^{2}T-Lion. It also has abundant communication ports to interact with peripheral devices like USB cameras and Wi-Fi devices. One UART port is used to communicate with the flight control computer.

2.2.3. *Servo controller*

An 8-channel pulse-width modulation (PWM) servo controller, UAV100 from Pontech, is used to enable servo control by either an onboard computer via serial port (automatic mode) or output from an RC receiver (manual mode). The switching between the two modes depends on the state of an auxiliary channel from the RC transmitter. While the UAV maneuvers autonomously in the air, it is desirable to have a failsafe feature to allow the ground pilot to take over control during emergencies. Besides, this servo controller has the function of outputting quantitative servo values. This makes collecting manual or autonomous control data possible and it is a necessary requirement for UAV dynamic modeling and system identification.

2.2.4. *Avionic hub*

A customized printed circuit board (PCB) called LionHub (see Fig. 8) is developed as an expansion board to host various hardware devices. It is an improved version of a similar board introduced in [15]. The aforementioned IG-500N navigation sensor, the Gumstix Overo Fire computer, and the UAV100 servo control board can be physically installed on the slots of this PCB hub and connected to the onboard power regulator and other essential components. Besides the mounting slots, extra mounting holes on LionHub are used to lock the installed modules to resist the vibration and shock generated in flight and landing. With the introduction of LionHub, manual wire wrap is minimized to improve the reliability and quality of the system. A serial RS-232 to TTL level voltage converter is included in LionHub to connect the output of IG-500N to the UART port of Gumstix. Furthermore, to power up all the avionics, linear regulators designed in the avionic hub to convert a power input from a 4-cell Lithium-Polymer (LiPo) battery to 12 V and 5 V outputs with sufficient current delivering. The 12 V output port powers the Mastermind computer and the Lidar sensor, while the 5 V output port powers the Gumstix computer and other electronic boards.

2.3. ***System integration***

After selecting and configuring the individual mechanical and avionic components, all these hardware parts need to

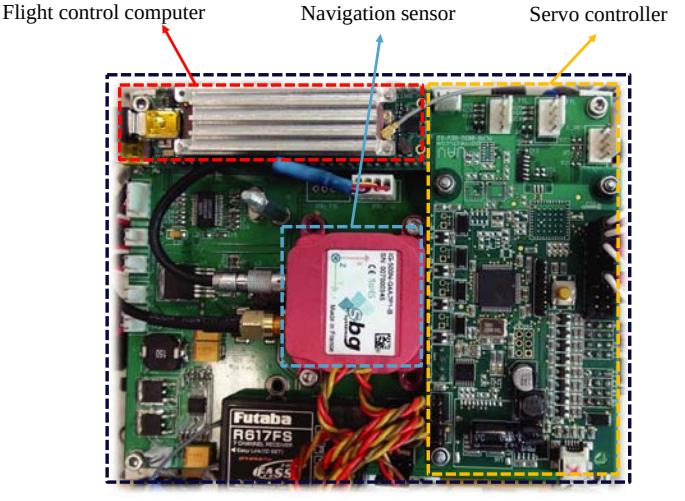

Fig. 8. Control hub with all hardware components attached.

be assembled to form a coherent UAV platform. To accomplish this task, special attention needs to be paid in the layout design of the overall onboard system and anti-vibration consideration.

2.3.1. *Layout design*

The first priority is to place the navigation sensor as close to the center of gravity (CG) of the whole UAV platform as possible to minimize the so-called lever effect, which causes bias to acceleration measurement when the UAV platform performs rotational motion. Note that all the other electronic boards on the LionHub will also be located near to the CG position because they are rigidly linked to the IMU. Usually there is no problem to align the IMU so that its planar *x*- and *y*-axis position coincide with the UAV CG. However, since a minimum space between the helicopter belly and the onboard system is needed for bumping avoidance, compromise needs to be made in the vertical *z*-axis and software compensation can be implemented to minimize the measurement error caused by this vertical offset. In order to have better signal reception, the GPS antenna is placed on the horizontal fin of the helicopter tail. Again, its 3D position offset to the IMU needs to be compensated.

The next priority goes to the camera sensor. By considering the fact that the UAV usually flies forward to search for targets and hovers right above the cargo for loading and unloading, the best position to place the camera is at the nose of the helicopter. In addition, a controlled pan-tilt gimbal (see Fig. 9) is designed to host the camera sensor so that it always looks vertically downwards despite the UAV rolling and pitching motions. Taking advantage of the camera's wide viewing angle, even when the UAV descends to the lowest altitude for cargo grabbing, the camera can still see the cargo which should be right under the UAV CG.

In order to retain CG balancing, the cargo loading mechanism needs to be installed precisely under the UAV CG. In this way, the UAV roll and pitch dynamics will not change too much after the cargo is loaded, thus the same set of robust control law can be used. This design also makes

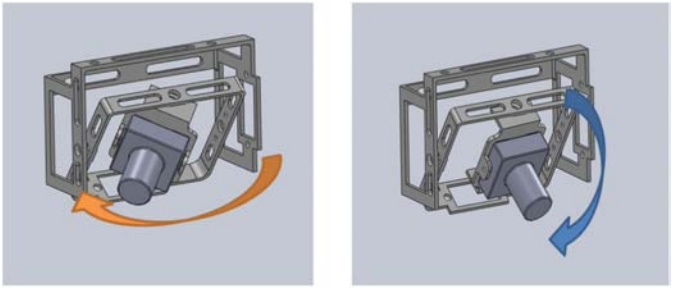

Fig. 9. Camera pan-tilt mechanism.

Fig. 10. Anti-vibration using wire rope isolators.

sure that controlling the UAV CG to the correct planar position is equivalent to controlling the cargo loading mechanism to the correct position so that a precise grabbing action can take place.

The placement of the remaining onboard components are less restricted. The overall CG balancing can be achieved by adjusting their mounting positions. For our case, the laser scanner is positioned at the back end of the onboard system, scanning downwards. The vision computer is put at the frontal part to counter-balance the laser scanner and to make wiring to the camera sensor shorter. The battery is slotted at a bottom middle position so that it adds on minimal moment of inertia to the whole UAV platform.

With the above layout design, the distribution of mass is balanced, the control challenge caused by the cargo loading is minimized, and all sensors are working properly. An aluminium plate is used to mount all the onboard components and it sits on four wire rope isolators (see Fig. 10) which helps to solve the mechanical vibration problem.

2.3.2. *Anti-vibration*

Anti-vibration for the onboard avionics is one of the most important considerations in hardware design. It can improve the overall performance of the UAV system significantly by reducing wear and tear of the mechanical and electrical connectors and attenuating unwanted noises at high frequencies. Indeed, the replacing of nitro engine with a gasoline engine amplifies the vibration issue. The main vibration sources on NUS^{2}T-Lion are from its main rotors and the engine. From a frequency analysis of the in-flight acceleration data logged while hovering (see Fig. 11), one can see that the most significant high-frequency vibration occurs at 22 Hz.

To attenuate noise at this specific frequency, the CR4-400 compact wire rope isolator from Enidine is used. According to the CR series manual provided by Enidine, the

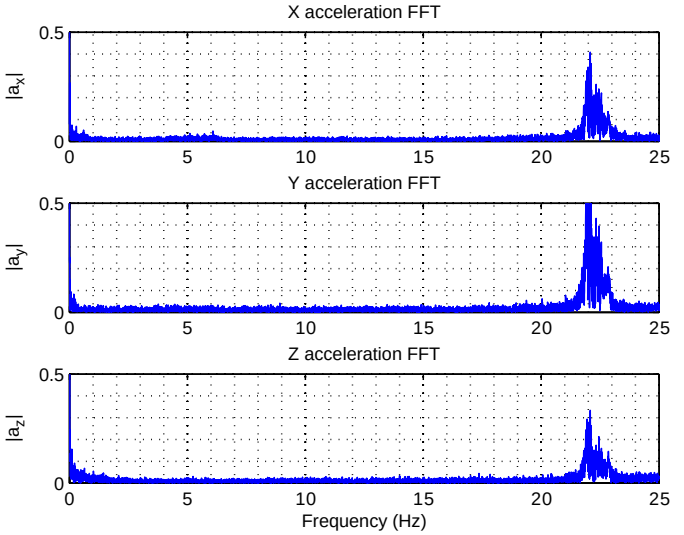

Fig. 11. Frequency analysis of acceleration without isolators.

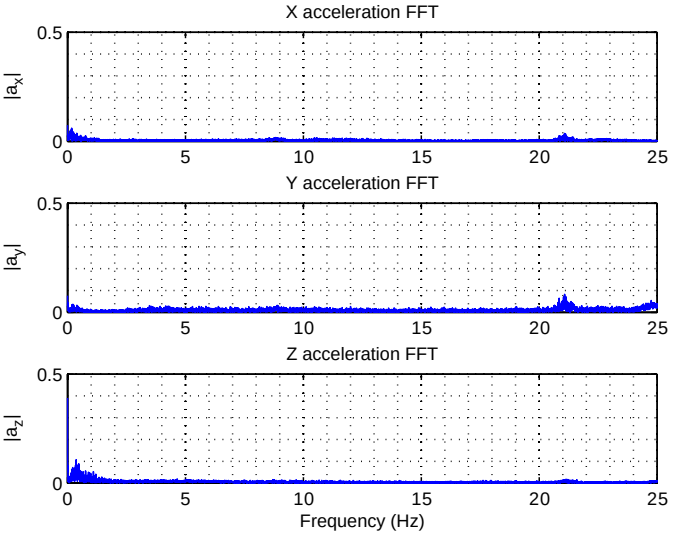

Fig. 12. Frequency analysis of acceleration with isolators.

best stiffness for the chosen isolator, K_v can be calculated as

$$K_v = W_s(2\pi f_i/3)^2/g, \tag{1}$$

where W_s is the static load on every isolator, f_i is the input excitation frequency needs to be attenuated, and g is the gravitational constant. For our case, about 2 kg of onboard load is shared by four isolators, which gives $W_s = 4.9$. By substituting also $f_i = 22$ and $g = 9.781$ into (1), K_v can be calculated as 1.06 kN/m which is best matched by the vibration stiffness value obtained by CR4-400 mounted in a "45° Compression/Roll" mode. There are also the "Pure Compression" and "Shear/Roll" mounting methods, but the "45° Compression/Roll" mode is the best for attenuating vibration in all three axes. After the installation of wire rope isolators, Fig. 12 shows the improved performance of acceleration measurement. As compared to the original graph, the higher frequency noises have been reduced by 10 times or more.

When the hardware platform is ready, guidance, navigation and control algorithms need to be developed to make the UAV autonomous and intelligent. Figure 13 illustrates the overview of the GNC structure implemented in this work. The *control* block makes sure the UAV attitude is stable and can robustly track way-point references. The *navigation* block provides all the necessary measurements by fusing raw data from UAV onboard GPS/INS unit, the ship GPS/INS unit and the UAV onboard laser scanner. Last but not least, by implementing a vision-based target detection and tracking sub-system, the *guidance* block generates smooth reference trajectories for the UAV to follow and do meaningful tasks. In the following three sections, these three blocks will be explained in the sequence of control, navigation and guidance. In fact, they form the most significant and innovative contributions from this paper.

3. Modeling and Control

For all UAV related applications, stability of the controlled platform is the most fundamental problem that needs to be solved first. Otherwise, there is no foundation for high-level navigation and guidance algorithms to be built upon. In this paper, we decompose the UAV control problem into two layers, namely the attitude stabilization layer and the position tracking layer. The former involves the design of an inner-loop control law which makes sure the UAV roll, pitch and yaw dynamics are robustly stable. The latter position tracking layer involves the design of an outer-loop control law which enables the UAV to track any smooth 3D trajectory references in a responsive and precise way. However, most advanced control design methods require an accurate dynamic model of the controlled UAV platform. Hence, the following context will start from NUS^{2}T-Lion's model identification, and then proceed to control structure formulation, inner-loop control law design, outer-loop control law design and the inner-loop command generator which connects the two layers in a reasonable way.

3.1. *Flight dynamics modeling*

The modeling of NUS^{2}T-Lion follows a similar procedure introduced in [14]. The model structure is shown in Fig. 14 where forces and torques ($\mathbf{F}_b$ and $\mathbf{M}_b$) generated by various mechanical parts of the helicopter are fed into the six

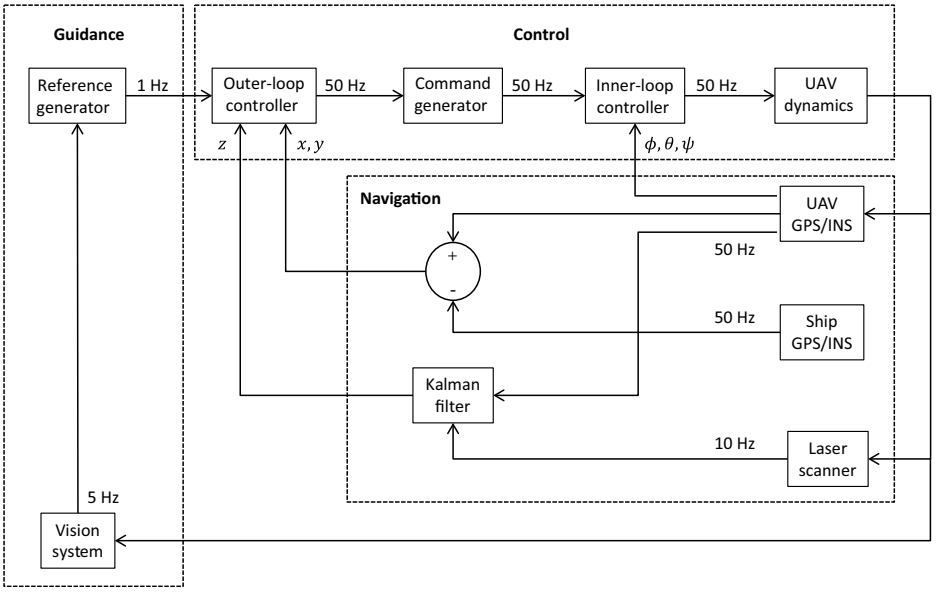

Fig. 13. Overall structure of guidance, navigation and control.

degree-of-freedoms (DOF) rigid-body dynamics followed by the kinematics. There are four inputs to the whole helicopter system, namely the collective pitch input δ_{col}, controlling the heave dynamics, the lateral input δ_{lat}, controlling the rolling dynamics, the longitudinal input δ_{lon}, controlling the pitching dynamics, and the pedal input δ_{ped}, controlling the yawing dynamics. Some cross-couplings exist among the four channels. The outputs of the nonlinear model can be found at the right side of Fig. 14, namely the UAV global frame position $\mathbf{P}_{\text{n}}$, body frame velocity $\mathbf{V}_{\text{b}}$, attitude angles ϕ

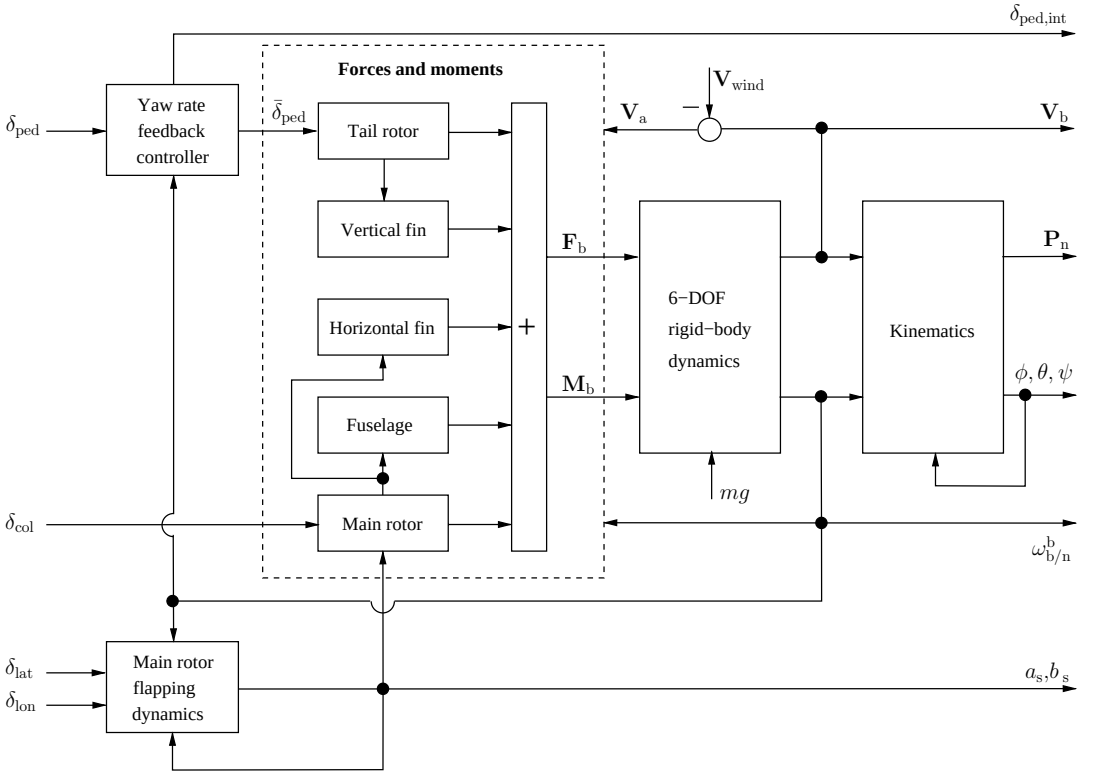

Fig. 14. Model structure of NUS^{2}T-Lion.

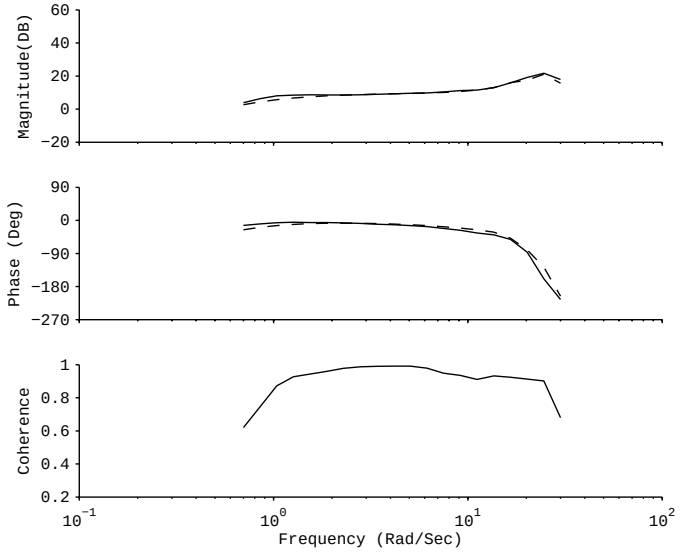

Fig. 15. Frequency-domain model fitting: δ_{lat} to p.

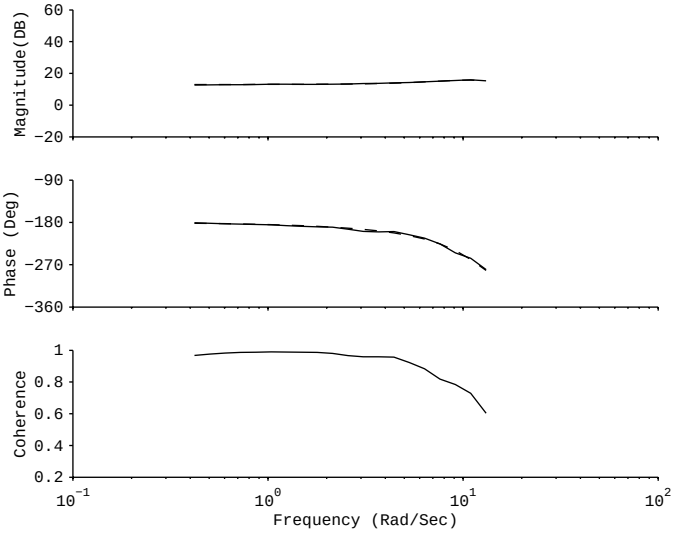

Fig. 17. Frequency-domain model fitting: δ_{rud} to r.

(roll), θ (pitch), ψ (yaw) and their corresponding angular rates p, q, r. Besides, a_s and b_s are the longitudinal and lateral flapping angles of the main rotor and $\delta_{\text{ped,int}}$ is an intermediate state variable in representing the yaw dynamics of the UAV. These three state variables are not measurable. V_{a} and V_{wind} represent air velocity and wind disturbance respectively.

The detailed nonlinear model formulation and parameter identification closely follow the procedures in [14], thus will not be repeated here. Some of the model parameters can be simply measured, such as the dimensions, mass, moment of inertia of the UAV platform, while the remainings need to be identified by carrying out test-bench experiments or actual flights. In identifying the model parameters of NUS^{2}T-Lion, a MATLAB-based software, Comprehensive Identification from FrEquency Response (CIFER) was used. CIFER is developed by the NASA Ames Research Center for military-based rotorcraft systems. It searches for optimum model parameters by comparing the frequency domain responses from the proposed model to the actual flight data. The most critical model-data fitting results showing the main channel responses are briefly shown here. In Figs. 15–17, the solid lines and the dashed lines show the frequency response of the in-flight data and the fitted model respectively. It can be seen that the model fits the flight test data very well, indicating a good fidelity of the derived parameters.

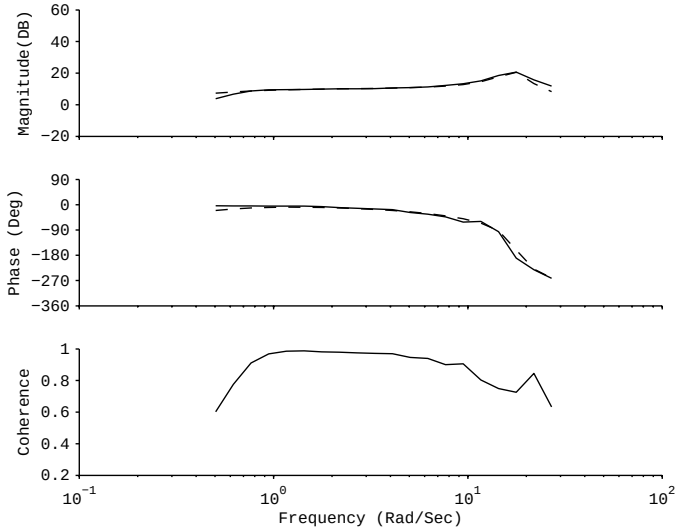

Fig. 16. Frequency-domain model fitting: δ_{lon} to q.

3.2. *Control structure*

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 decomposition of the full-order dynamic 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, which makes them severable in the frequency domain. Hence, the overall control system can be formulated in a dual-loop structure, so that the inner-loop and outer-loop controllers can be designed separately. Moreover, the linearized model of the single rotor helicopter system is found to be of non-minimum phase if the two motion dynamics are combined together. This non-minimum phase characteristics will highly complicate the control problem and needs to be avoided.

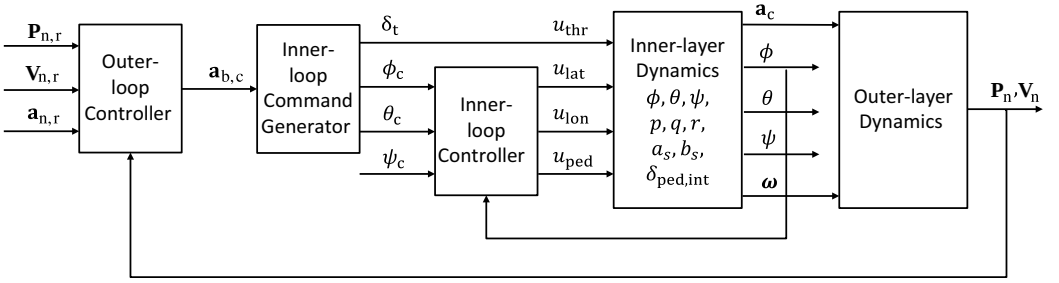

Fig. 18. Control structure of NUS^{2}T-Lion.

For the inner loop, the controlled object covers the rotational motion of the helicopter body, the flapping motion of rotor blades and the stabilizer bar, as well as the dynamics embedded within the head-lock gyro. The main task of the inner-loop controller is to stabilize the attitude and heading of the helicopter in all flight conditions. In our implementation, the H_∞ control method is used to minimize the disturbance from wind gusts. For the outer loop, the controlled object covers only the translational motion. The main task is to steer the helicopter flying with reference to a series of given locations. A robust and perfect tracking (RPT) approach is implemented to emphasize the time factor. Figure 18 gives an overview of the dual-loop control structure.

3.3. *Inner-loop control design*

Although the full model of NUS^{2}T-Lion is highly complicated and nonlinear, it is verified by simulation that its inner dynamics, after linearization, is more or less invariant under different non-acrobatic flight conditions. Hence, it is reasonable to design a feedback control law based on the linearized model of the inner-layer dynamics, while using the nonlinear model for verification purposes only. Besides, it is noted that NUS^{2}T-Lion falls into the category of small-scale UAV helicopter which is quite vulnerable to environmental disturbances such as wind gusts. Hence, the H_∞ control method, which is specifically developed to minimize output error caused by external disturbances, naturally becomes the best choice. The linearized inner-dynamics model of NUS^{2}T-Lion can be represented by a ninth-order state space form as shown below:

$$\begin{cases} \dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u} + E\mathbf{w} \\ \mathbf{y} = C_1\mathbf{x} + D_1\mathbf{w} \\ \mathbf{h} = C_2\mathbf{x} + D_2\mathbf{u} \end{cases}, \tag{2}$$

where $\mathbf{x}$ is the state, $\mathbf{y}$ is the measurement output, $\mathbf{h}$ is the controlled output, $\mathbf{u}$ is the input and $\mathbf{w}$ is the wind disturbance. More specifically,

$$\mathbf{x} = [\phi \quad \theta \quad \psi \quad p \quad q \quad r \quad a_s \quad b_s \quad \delta_{\text{ped,int}}]^{\mathrm{T}}, \tag{3}$$

$$\mathbf{u} = [\delta_{\text{lat}} \quad \delta_{\text{lon}} \quad \delta_{\text{ped}}]^{\mathrm{T}}, \tag{4}$$

$$\mathbf{w} = [u_{\text{wind}} \quad v_{\text{wind}} \quad w_{\text{wind}}]^{\mathrm{T}}, \tag{5}$$

$$A = [\mathbf{0}_{9\times 3} \,|\, \bar{A}], \tag{6}$$

where

$$\bar{A} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 620.5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 327.6 & 0 \\ 0 & 0 & -13.5 & 0 & 0 & 165.6 \\ -1 & 0 & 0 & -5.41 & 6.45 & 0 \\ 0 & -1 & 0 & -3.72 & -5.41 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 \end{bmatrix}, \tag{7}$$

$$B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -54.69 \\ 2.975 & -0.3 & 0 \\ 0.780 & 3.23 & 0 \\ 0 & 0 & -4.46 \end{bmatrix}, \tag{8}$$

$$E = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ -0.0001 & 0.1756 & -0.0395 \\ 0 & 0.0003 & 0.0338 \\ -0.0002 & -0.3396 & 0.6424 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{9}$$

As the onboard IMU can provide measurements of the first six state variables, C_1 can be formed accordingly and D_1 can be left as a zero matrix. C_2 and D_2 constitute weighting

parameters specifying the control objective, and usually need to be tuned for practical implementation. For this case, they are in the following form, which considers the first six state variables and the three control inputs

$$C_2 = \left[\begin{array}{ccccccccc} c_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & c_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & c_3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & c_4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & c_5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & c_6 & 0 & 0 & 0 \\ \hline \multicolumn{9}{c}{\mathbf{0}_{3\times 9}} \end{array}\right], \tag{10}$$

$$D_2 = \left[\begin{array}{ccc} \multicolumn{3}{c}{\mathbf{0}_{6\times 3}} \\ \hline d_1 & 0 & 0 \\ 0 & d_2 & 0 \\ 0 & 0 & d_3 \end{array}\right]. \tag{11}$$

The H_∞ control problem is to find an internally stabilizing proper measurement feedback control law,

$$\begin{cases} \dot{\mathbf{v}} = A_{\text{cmp}}\mathbf{v} + B_{\text{cmp}}\mathbf{y} \\ \mathbf{u} = C_{\text{cmp}}\mathbf{v} + D_{\text{cmp}}\mathbf{y} \end{cases}, \tag{12}$$

such that the H_∞-norm of the overall closed-loop transfer matrix function from $\mathbf{w}$ to $\mathbf{h}$ is minimized. According to [16], the minimum H_∞-norm, γ^*, can be exactly computed using some numerical algorithms. However, it is almost impossible to find a control law with finite gain to achieve this particular optimal performance. Usually, an H_∞ suboptimal controller is designed, resulting in a suboptimal H_∞-norm smaller than γ, where $\gamma > \gamma^*$. It is also proved in [16] that when the subsystem (A, E, C_1, D_1) is left invertible and of minimum phase, which is exactly the case for NUS^{2}T-Lion, the optimal achievable H_∞ control performance under the state feedback and the measurement feedback are identical. In other words, it is appropriate to design the state feedback control law and the observer separately for the inner loop of NUS^{2}T-Lion. Moreover, only a reduced-order observer is needed to estimate the three unmeasurable state variables, i.e., $a_s, b_s, \delta_{\text{ped,int}}$.

To design an H_∞ reduced-order output feedback control law for the inner-dynamics of NUS^{2}T-Lion, the original system (2) can be rewritten as follows:

$$\begin{cases} \begin{pmatrix} \dot{\mathbf{x}}_1 \\ \dot{\mathbf{x}}_2 \end{pmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{pmatrix} + \begin{bmatrix} B_1 \\ B_2 \end{bmatrix} \mathbf{u} + \begin{bmatrix} E_1 \\ E_2 \end{bmatrix} \mathbf{w} \\ \begin{pmatrix} \mathbf{y}_0 \\ \mathbf{y}_1 \end{pmatrix} = \begin{bmatrix} 0 & C_{1,02} \\ I & 0 \end{bmatrix} \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{pmatrix} + \begin{bmatrix} D_{1,0} \\ 0 \end{bmatrix} \mathbf{w} \\ \mathbf{h} = [C_{2,1} \quad C_{2,2}] \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{pmatrix} + D_2\mathbf{u} \end{cases}. \tag{13}$$

In this form, the original state $\mathbf{x}$ is partitioned into a measurable state $\mathbf{x}_1$ and an unmeasurable state $\mathbf{x}_2$; $\mathbf{y}$ is partitioned into $\mathbf{y}_0$ and $\mathbf{y}_1$ with $\mathbf{y}_1 \equiv \mathbf{x}_1$. If we define an auxiliary subsystem characterized by a matrix quadruple (A_R, E_R, C_R, D_R), where

$$(A_R, E_R, C_R, D_R) = \left(A_{22}, E_2, \begin{bmatrix} C_{1,02} \\ A_{12} \end{bmatrix}, \begin{bmatrix} D_{1,0} \\ E_1 \end{bmatrix}\right), \tag{14}$$

then the following procedures can be followed to design the reduced-order output feedback H_∞ controller:

Step 1: *Construction of State Feedback Gain Matrix*

Define an auxiliary system

$$\begin{cases} \dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u} + E\mathbf{w} \\ \mathbf{y} = \mathbf{x} \\ \mathbf{h} = C_2\mathbf{x} + D_2\mathbf{u} \end{cases}. \tag{15}$$

Select $\gamma > \gamma^*$, compute its corresponding H_∞ γ-suboptimal state feedback gain matrix F.

Step 2: *Construction of Observer Gain Matrix*

Define another auxiliary system

$$\begin{cases} \dot{\mathbf{x}} = A_R^{\mathrm{T}}\mathbf{x} + C_R^{\mathrm{T}}\mathbf{u} + C_{2,2}^{\mathrm{T}}\mathbf{w} \\ \mathbf{y} = \mathbf{x} \\ \mathbf{h} = E_R^{\mathrm{T}}\mathbf{x} + D_R^{\mathrm{T}}\mathbf{u} \end{cases}. \tag{16}$$

Select a sufficiently small $\gamma > 0$, compute its corresponding H_∞ γ-suboptimal state feedback gain matrix F_R and then let $K_R = F_R^{\mathrm{T}}$.

Step 3: *Construction of Output Feedback Controller*

Partition F and K_R as

$$F = [F_1 \quad F_2], \quad K_R = [K_{R0} \quad K_{R1}], \tag{17}$$

in conformity with the partitioning of $\mathbf{x}$ and $\mathbf{y}$, respectively. Now define

$$G_R = [-K_{R0}, A_{21} + K_{R1}A_{11} - (A_R + K_RC_R)K_{R1}],$$

then the reduced-order output feedback controller is given by (12), where

$$\begin{aligned} A_{\text{cmp}} &= A_R + B_2F_2 + K_RC_R + K_{R1}B_1F_2, \\ B_{\text{cmp}} &= G_R + (B_2 + K_{R1}B_1)[0, F_1 - F_2K_{R1}], \\ C_{\text{cmp}} &= F_2, \\ D_{\text{cmp}} &= [0, F_1 - F_2K_{R1}]. \end{aligned}$$

Based on the above procedures while choosing an appropriate set of C_2, D_2, a H_∞ reduced-order output feedback controller can be determined. After several rounds of tunings, the final values for the weighting parameters in C_2 and D_2 are chosen to be

$$c_1 = 13, \quad c_2 = 12, \quad c_3 = 1, \quad c_4 = 1, \quad c_5 = 1, \quad c_6 = 6 \tag{18}$$

and

$$d_1 = 13, \quad d_2 = 12, \quad d_3 = 30. \tag{19}$$

The corresponding $\gamma^* = 0.2057$ and we choose $\gamma = 0.21$, which results in the following γ-suboptimal state feedback gain matrix,

$$F = \begin{bmatrix} -0.9952 & -0.1177 & 0.0017 & -0.0271 \\ 0.1386 & -0.9927 & -0.0005 & -0.0056 \\ -0.0186 & 0.0096 & 0.0526 & -0.0006 \end{bmatrix}$$

$$\begin{bmatrix} 0.0098 & 0.0143 & -1.8795 & -0.5324 & 0.0457 \\ -0.0467 & -0.0043 & 0.0253 & -1.8175 & -0.0503 \\ 0.0026 & 0.2379 & -0.0925 & -0.0216 & 1.3287 \end{bmatrix}. \tag{20}$$

The corresponding feed-forward matrix is calculated as

$$G = \begin{bmatrix} 0.9952 & 0.1177 & -0.0017 \\ -0.1386 & 0.9927 & 0.0005 \\ 0.0186 & -0.0096 & -0.0526 \end{bmatrix}.$$

The last three unmeasurable state variables, denoted by $\hat{\mathbf{x}}$, can be estimated by an observer as follows:

$$\dot{\hat{\mathbf{x}}} = \bar{F}\hat{\mathbf{x}} + \bar{G}\mathbf{y} + \bar{H}\mathbf{u} \tag{21}$$

where

$$\bar{F} = \begin{bmatrix} -0.9952 & -0.1177 & 0 \\ 0.1386 & -0.9927 & 0 \\ -0.0186 & 0 & -28 \end{bmatrix}, \tag{22}$$

$$\bar{G} = \begin{bmatrix} 0 & 0 & 0 & -9.309 & 0.2404 & 0 \\ 0 & 0 & 0 & -1.225 & -5.106 & 0 \\ 0 & 0 & 0 & 0 & 0 & -3.452 \end{bmatrix}, \tag{23}$$

$$\bar{H} = \begin{bmatrix} 2.975 & -0.3 & 0 \\ 0.780 & 3.23 & 0 \\ 0 & 0 & 4.78 \end{bmatrix}. \tag{24}$$

With this set of gain matrices, the inner-loop system is stable with a bandwidth of 2.95 rad/s for the roll angle dynamics, 2.8 rad/s for the pitch angle dynamics and 2.17 rad/s for the yaw angle dynamics.

3.4. *Outer-loop control design*

For the outer loop, an RPT controller is designed to let the UAV track any 3D trajectories precisely. The controller structure and design techniques are adopted from [17]. By perfect tracking, it means the ability of the controlled system to track a given reference with arbitrarily fast settling time subjected to disturbances and initial conditions. Considering the following linear time invariant system

$$\Sigma : \begin{cases} \dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u} + E\mathbf{w} \\ \mathbf{y} = C_1\mathbf{x} + D_1\mathbf{w} \\ \mathbf{h} = C_2\mathbf{x} + D_2\mathbf{u} + D_{22}\mathbf{w} \end{cases}, \tag{25}$$

with $\mathbf{x}, \mathbf{u}, \mathbf{w}, \mathbf{y}, \mathbf{h}$ being the state, control input, disturbance, measurement and controlled output respectively, the task of an RPT controller is to formulate a dynamic measurement control law in the form of

$$\begin{aligned} \dot{\mathbf{v}} &= A_c(\varepsilon)\mathbf{v} + B_c(\varepsilon)\mathbf{y} + G_0(\varepsilon)r + \cdots + G_{\kappa-1}(\varepsilon)r^{\kappa-1}, \\ \mathbf{u} &= C_c(\varepsilon)\mathbf{v} + D_c(\varepsilon)\mathbf{y} + H_0(\varepsilon)r + \cdots + H_{\kappa-1}(\varepsilon)r^{\kappa-1}, \end{aligned}$$

so that when a proper $\varepsilon^* > 0$ is chosen,

(1) The resulted closed-loop system is asymptotically stable subjected to zero reference.
(2) If $e(t, \varepsilon)$ is the tracking error, then for any initial condition $\mathbf{x}_0$, there exists:

$$\|e\|_p = \left(\int_0^\infty |e(t)^p| dt \right)^{1/p} \to 0, \quad \text{as } \varepsilon \to 0.$$

For nonzero references, their derivatives are used to generate additional control inputs. Thus, any reference of the form of $r(t) = p_1 t^k + p_2 t^{k-1} + \cdots + p_{k+1}$ are covered in the RPT formulation. Furthermore, any references that have a Taylor series expansion at $t = 0$ can also be tracked using the RPT controller.

Similar to the case introduced in [18], the outer dynamics of NUS^{2}T-Lion is differentially flat. That means all its state variables and inputs can be expressed in terms of algebraic functions of flat outputs and their derivatives. A proper choice of flat outputs could be

$$\sigma = [x \quad y \quad z \quad \psi]^{\mathrm{T}}. \tag{26}$$

It can also be observed that the first three outputs, x, y and z, are totally independent. In other words, we can consider the UAV as a mass point with constrained velocity, acceleration, jerk, and so on in the individual axes of the 3D global frame when designing its outer-loop control law and generating the position references. Hence, a stand-alone RPT controller based on a double integrator model in each axis can be designed to track the corresponding reference in that particular axis. For each axis, the nominal system can be written as

$$\begin{cases} \dot{\mathbf{x}}_{\mathrm{n}} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \mathbf{x}_{\mathrm{n}} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \mathbf{u}_{\mathrm{n}} \\ \mathbf{y}_{\mathrm{n}} = \mathbf{x}_{\mathrm{n}} \end{cases}. \tag{27}$$

To achieve better tracking performance, it is common to include an integrator to ensure zero steady state error

subjected to step inputs. Thus, the RPT controller proposed here is with integral action. This requires an augmented system to be formulated as

$$\begin{cases} \dot{\mathbf{x}}_{\mathrm{o}} = \begin{bmatrix} 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \mathbf{x}_{\mathrm{o}} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \mathbf{u}_{\mathrm{o}} \\ \mathbf{y}_{\mathrm{o}} = \mathbf{x}_{\mathrm{o}} \\ \mathbf{h}_{\mathrm{o}} = [1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]\mathbf{x}_{\mathrm{o}} \end{cases}, \tag{28}$$

where $x_{\mathrm{o}} = [\int(p_e)\ p_r\ v_r\ a_r\ p\ v]^{\mathrm{T}}$ with p_r, v_r, a_r as the position, velocity and acceleration references, p, v as the actual position and velocity and $p_e = r_p - p$ as the tracking error of position. By following the procedures in [16], a linear feedback control law of the form below can be acquired,

$$u_{\mathrm{o}} = F_{\mathrm{o}} x_{\mathrm{o}}, \tag{29}$$

where

$$F_{\mathrm{o}} = \begin{bmatrix} \dfrac{k_i \omega_n^2}{\varepsilon^3} & \dfrac{\omega_n^2 + 2\zeta\omega_n k_i}{\varepsilon^2} & \dfrac{2\zeta\omega_n + k_i}{\varepsilon} \\ 1 & -\dfrac{\omega_n^2 + 2\zeta\omega_n k_i}{\varepsilon^2} & -\dfrac{2\zeta\omega_n + k_i}{\varepsilon} \end{bmatrix}. \tag{30}$$

ε is a design parameter to adjust the settling time of the closed-loop system. ω_n, ζ, k_i are the parameters that determines the desired pole locations of the infinite zero structure of (28) through

$$p_i(s) = (s + k_{\mathrm{i}})(s^2 + 2\zeta\omega_n s + \omega_n^2). \tag{31}$$

Theoretically, when the design parameter ε is small enough, the RPT controller gives arbitrarily fast response. However, in real life, due to the constraints of the UAV physical dynamics and its inner-loop bandwidth it is safer to limit the bandwidth of the outer loop to be one-fifth to one-third of the controlled inner-loop system. For the case of NUS^{2}T-Lion case, the following design parameters are used for different axes:

$$x, y: \begin{cases} \varepsilon = 1 \\ \omega_n = 0.707 \\ \zeta = 0.707 \\ k_i = 0.25 \end{cases} \quad z: \begin{cases} \varepsilon = 1 \\ \omega_n = 0.99 \\ \zeta = 0.707 \\ k_i = 0.29 \end{cases}.$$

3.5. *Inner-loop command generator*

We have designed the inner-loop and the outer-loop controllers separately to avoid the non-minimum phase problem and to relieve task complexity. As the inner-loop dynamics is designed much faster than that of the outer loop, it can be treated as a non-dynamic static gain matrix when viewed from outside. However, the output from the outer-loop controller in physical meaning is the desired accelerations in the global frame, $\mathbf{a}_{\mathrm{c}}$, while the inner-loop controller is looking for attitude references (ϕ_{c}, θ_{c}, ψ_{c}). Obviously, a global-to-body rotation followed by a command conversion is needed. Moreover, the body-axis acceleration $\mathbf{a}_{\mathrm{b}}$ does not mean anything to the heading direction reference ψ_{c}. Therefore, unlike the other two attitude angle references (ϕ_{c}, θ_{c}), ψ_{c} is not involved in this conversion, but generated independently. In addition, the acceleration reference in the UAV body z-axis directly links to the needed collective control input δ_{col}, which is not manipulated by the inner loop at all. Based on the above ideas, if $\mathbf{G}_{\mathrm{a}}$ is the steady-state gain matrix from the inputs (δ_{col}, ϕ_{c}, θ_{c}) to the UAV body-frame accelerations, then we can get an approximated conversion matrix G_c as its inverse. So,

$$(\delta_{\mathrm{col}} \quad \phi_{\mathrm{c}} \quad \theta_{\mathrm{c}})^{\mathrm{T}} = \mathbf{G}_{\mathrm{c}} \mathbf{a}_{\mathrm{b,c}} = \mathbf{G}_{\mathrm{a}}^{-1} \mathbf{a}_{\mathrm{b,c}}. \tag{32}$$

Note that $\mathbf{G}_{\mathrm{a}}$ must be non-singular. Otherwise, it means $\mathbf{a}_{\mathrm{b}}$ cannot be manipulated by the control inputs u_{col}, u_{lat}, u_{lon}. For the case of NUS^{2}T-Lion,

$$\mathbf{G}_{\mathrm{c}} = \begin{bmatrix} 0 & 0 & 0.0523 \\ 0 & 0.1022 & 0 \\ -0.1022 & 0 & 0 \end{bmatrix}. \tag{33}$$

4. Navigation

The previous section has solved the control problem of NUS^{2}T-Lion. However, it assumes that all the measurements needed by the control law are available and reliable, which is too ideal and rarely happens in practical situations. Although the onboard GPS/INS sensor can provide rich information including the UAV's global position, velocity and attitude angles, there is still missing information or unacceptable amount of measurement noises for the UAV to do precise loading and unloading of cargoes on the moving platforms. As the moving platforms are actually set up to simulate cargo ships, they will be called 'ships' in the following context to avoid ambiguity from the term 'platform' which is used for both the UAV platform and the cargo platform.

In order to synchronize with the motion of the ships, the controlled UAV needs to know at every moment how the ships are moving. A simple and reliable solution is to install another GPS/INS sensor on the ship and send its information to the UAV onboard system. By doing so, the UAV can be controlled in a ship-referenced frame instead of the global frame. In this ship-referenced frame, or called ship frame for simplicity, a zero steady-state position and

velocity tracking error means the UAV is controlled right above the ship with the same velocity as the ship.

However, it should be noted that this zero error is judged by the measurement difference of the two GPS/INS sensors. By considering their respective circular error probable, the measurement error sometimes goes beyond 6 m in all three axes. Obviously, it is not accurate enough for the cargo transportation task. In this section, the UAV z-axis measurement will be first complemented by fusing the information from a scanning laser range finder. As the height information extracted from the laser scanner is available and reliable for all time, it can be consistently fused in to improve the navigation accuracy in the z-axis. On the other hand, the x- and y-axis measurement errors are more difficult to be reduced in a navigation sense because there is no other sensor which can provide a consistent and accurate measurement in these two axes. However, after realizing that the best tracking performance is not really needed throughout the mission, but only required at the loading and unloading instances, they can be compensated in a guidance sense when the target enters the view angle of the onboard camera. This vision-based guidance will be discussed later in Sec. 5. Here, we only explain the ship-frame navigation, height calculation via laser scanner, and height measurement fusion.

4.1. *Navigation in ship frame*

As measurements provided by GPS/INS sensors on the UAV and on the ship are defined in the same global frame and the motion of the ship involves no rotation, it is adequate to convert all position, velocity and acceleration measurements into the ship frame by simple subtraction. So,

$$\begin{cases} \mathbf{p} = \mathbf{R}_{s/g}(\mathbf{p}_{uav} - \mathbf{p}_{ship}) \\ \mathbf{v} = \mathbf{R}_{s/g}(\mathbf{v}_{uav} - \mathbf{v}_{ship}) \\ \mathbf{a} = \mathbf{R}_{s/g}(\mathbf{a}_{uav} - \mathbf{a}_{ship}) \end{cases}. \tag{34}$$

If we refer back to Fig. 18, the outer-loop measurements and references are now both represented in the ship frame instead of the NED frame. Following this convention, it is also straight forward to convert the UAV heading angle to the ship frame as well. Hence,

$$\psi = \psi_{uav} - \psi_{ship}. \tag{35}$$

By redefining ψ this way, the original rotational matrix $\mathbf{R}_{b/n}$ (rotation from the NED frame to the UAV body frame) can be substituted by $\mathbf{R}_{b/s}$, which is the rotation from the ship frame to the UAV body frame. Note that ϕ and θ in the rotational matrix are still the UAV roll and pitch angles as we assume that the ship has almost zero roll and pitch angles.

4.2. *Height calculation via laser scanner*

As mentioned previously, a very accurate height measurement is needed for the cargo loading and unloading tasks. In fact, it is also the most helpful information for the UAV to carry out autonomous taking-off and landing. Motivated by this, a high-end scanning laser range finder is installed onboard of the UAV platform. The corresponding algorithm to calculate the UAV height via its range measurements is explained below.

For each frame of scanning, the laser sensor will output 1081 integer numbers representing the measured distances in millimeter from its starting point on the right to the end point on the left sequentially. Each distance data is associated with its own angle direction, thus the data can be seen as in polar coordinates. A simple transformation can be applied to the raw measurement data to convert it from polar coordinates (r_i, θ_i) to Cartesian coordinates (x_i, y_i) by

$$\begin{cases} x_i = r_i \cos\theta_i \\ y_i = r_i \sin\theta_i \end{cases}, \tag{36}$$

where $i \in \{1, 2, 3, \ldots, 1081\}$ is the index of the laser scanner measurements. Then, the *split-and-merge* algorithm [19] is applied to this array of 2D points so that they can be divided into clusters, with each cluster of points belonging to an individual line segment. The main steps of the *split-and-merge* algorithm is summarized below with Fig. 19 giving a graphical illustration.

(a) Connect the first point A and the last point B of the input data by a straight line.
(b) Find point C among all data points that has the longest perpendicular distance to line AB.
(c) If this longest distance is within a threshold, then a cluster is created containing points in between A and B.
(d) Else, the input points will be split into two subgroups, A-C and C-B. For each group, the *split-and-merge* algorithm will be applied recursively.

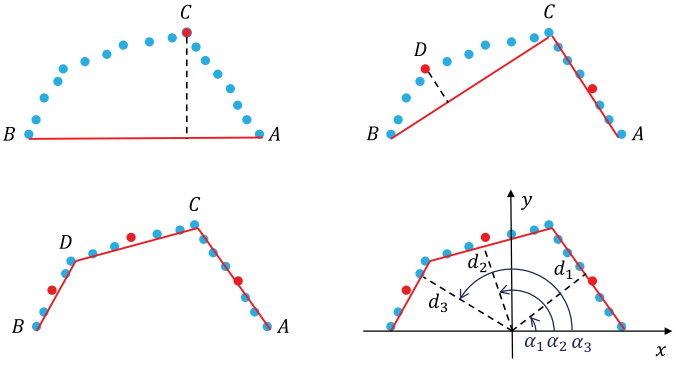

Fig. 19. The *split-and-merge* algorithm for line extraction.

Clusters of points will be created thereafter. Least square line fitting algorithm can then be applied to points in each cluster to obtain the individual lines. Each line can be represented by two parameters, namely the line's normal direction α_k and its perpendicular distance to the center of laser scanner d_k. In the last sub-figure of Fig. 19, *xy* axes represent the laser scanner frame. Normal direction of the line is defined as the angle from the *x*-axis to the line normal, counterclockwise as positive. The next step is to filter out lines with dissimilar gradient as the ground plane. Since the obtained lines are still expressed in the laser scanner frame, their directions α_k should be compensated by the UAV roll angle ϕ and then compared with the normal line of the ground plane which is at $\pi/2$. Let

$$\Delta\alpha_k = \alpha_k - \phi - \pi/2. \tag{37}$$

If $|\Delta\alpha_i|$ is greater than a threshold, say 5°, then the corresponding line is filtered out. The remaining lines are sorted by their perpendicular distances to the laser scanner and the furthest ones are kept. Among them, the longest line is believed to be the true ground. Finally, the perpendicular distance of this line to the laser scanner center is compensated with the UAV pitch angle θ and the offset between the laser scanner and the UAV CG, Δh, leaving the final height estimation to be

$$h = r\cos(\theta) - \Delta h. \tag{38}$$

Figure 20 has shown the flow chart of the laser scanner based height calculation algorithm. By using this method, an accurate height measurement can be obtained as long as the laser scanner projects a portion of its laser beams onto the true ground. Hence, it even works for the case when the UAV flies over protruding objects on the ground or on the ship surface.

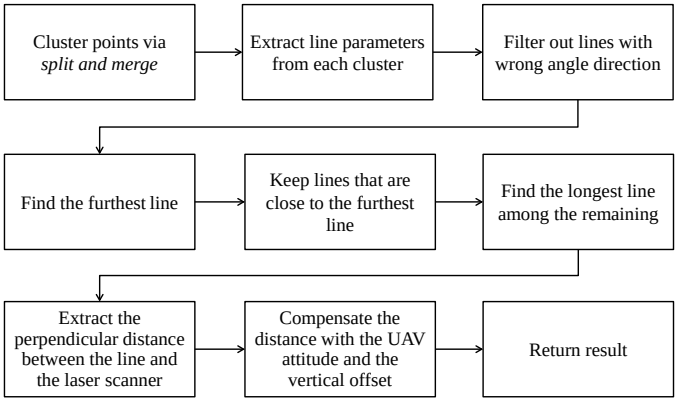

Fig. 20. Steps to compute height via laser scanner measurement.

4.3. *Height measurement fusion*

Since there are two sources of height measurements, one from GPS/INS and the other from laser scanner, it is best to combine them so that the most reliable and accurate UAV state variables in the *z*-axis, i.e., $\mathbf{x}_\mathrm{h} = [z\ w_g\ a_{z,g}\ \delta_z]^\mathrm{T}$, can be obtained. Here, z is the UAV vertical height with respect to the ground surface, w_g and $a_{z,g}$ are the corresponding velocity and acceleration in this axis and δ_z is the offset between the GPS/INS height and the laser counterpart. This offset has to be considered because the two sensory systems have different zero references and it also accounts for the time-varying position bias of the GPS/INS sensor. Here, we also formulate the estimator by considering the physical dynamics of a single-axis mass point system as:

$$\begin{cases} \dot{\mathbf{x}}_\mathrm{h} = A_\mathrm{h}\mathbf{x}_\mathrm{h} + E_\mathrm{h}\mathbf{w}_\mathrm{h} \\ \mathbf{y}_\mathrm{h} = C_\mathrm{h}\mathbf{x}_\mathrm{h} + \mathbf{v}_\mathrm{h} \end{cases}, \tag{39}$$

where

$$A_\mathrm{h} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad E_\mathrm{h} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad C_\mathrm{h} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{40}$$

and $\mathbf{w}_\mathrm{h}$, $\mathbf{v}_\mathrm{h}$ are Gaussian noises with covariance matrices Q_h and R_h respectively. Q_h and R_h can be chosen by analyzing signal noise levels logged in UAV hovering flight test with the assumption that all measurements are Gaussian and independent of each other. In our implementation, they are set as:

$$Q_\mathrm{h} = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.01 \end{bmatrix}^2, \quad R_\mathrm{h} = \begin{bmatrix} 0.05 & 0 & 0 & 0 \\ 0 & 0.8 & 0 & 0 \\ 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0.3 \end{bmatrix}^2. \tag{41}$$

By discretizing the system at a sampling rate of 50 Hz and applying Kalman filter, a reliable estimation of UAV height can be obtained. In implementing this filter, an additional technique is utilized to discard occasional outliers in the height measurement from the laser scanner. The idea is to check whether the received measurement is within a threshold multiply of the current process noise. If the discrepancy is too large, then the measurement at this particular instance is ignored. In [20], a similar technique is

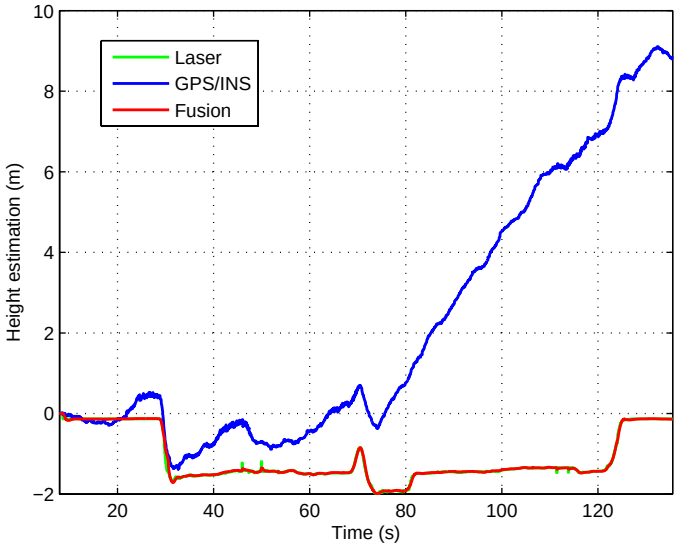

Fig. 21. Result of height estimation by data fusion.

introduced and it is called the *innovation filter*. Figures 21 and 22 show the height estimation result via data collected in one of the flight tests. It can be seen that the fused result has higher quality than the original height information from GPS/INS or laser scanner alone. The problem of slow drifting of GPS/INS (see Fig. 21) and a few small outliers from laser height measurement (see Fig. 22) are not affecting the fused result too much. At the same time, the estimated values of UAV vertical velocity and acceleration are also less noisy than their respective raw measurements from GPS/INS (see Figs. 23 and 24).

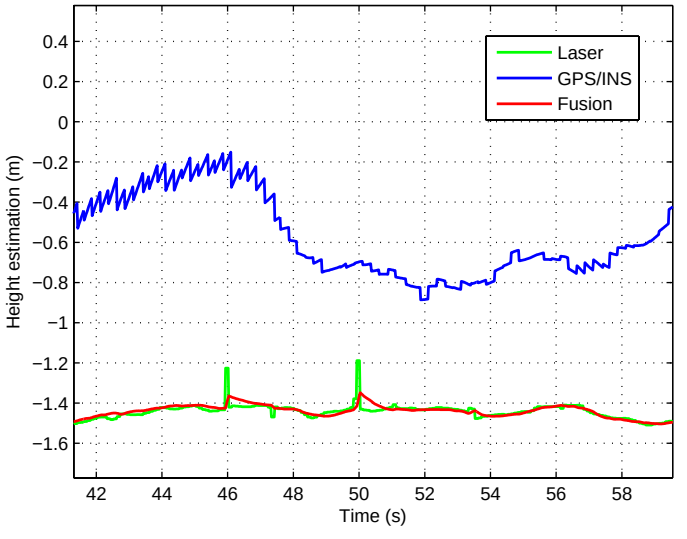

Fig. 22. Result of height estimation by data fusion (zoomed in).

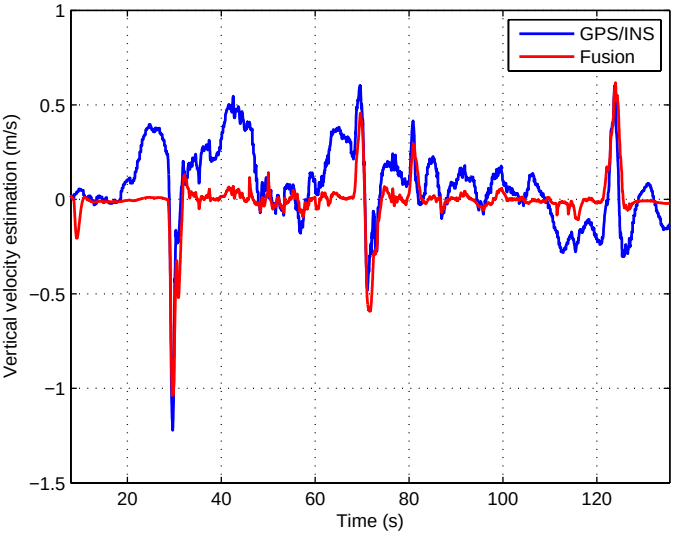

Fig. 23. Result of vertical velocity estimation by data fusion.

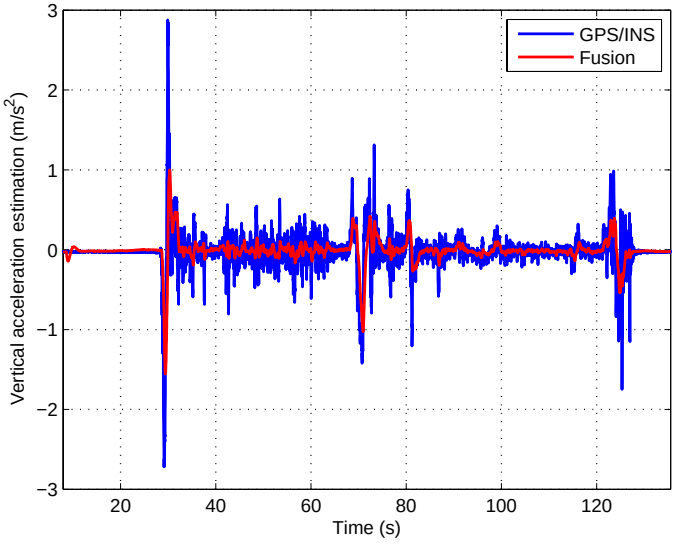

Fig. 24. Result of vertical acceleration estimation by data fusion.

5. Guidance

After the control and navigation problems have been solved, the next task is to guide the UAV to do meaningful movements by generating mission oriented reference trajectories. There are two major problems here. One is how to use vision as a guidance sensor to locate the target positions. In this problem, the target localization result must be correct and accurate enough for the UAV to carry out precise loading and unloading actions. The other problem is how to generate a continuous trajectory linking the UAV current status to the final destination by considering the physical limitations of the UAV dynamics. This is to make sure that

the generated reference is smooth enough, while satisfying the UAV outer-loop bandwidth requirements. In the following content, the solutions to these two problems will be presented accordingly.

5.1. *Vision-based target localization*

Vision-based target detection and localization are usually mission dependent. In the context of the UAVGP requirements, the cargoes to be transported are in the form of plastic buckets located inside circular patterns drawn on the ship surfaces. The first idea naturally comes into mind is to use image processing to detect circles and then do circle-based pose estimation. However, it should be noted that after camera projection, circles become ellipses in an image. Hence the main task of the vision sub-system here is to first select the correct target ellipse in the captured image and then estimate the 3D pose of the actual circle on the ship surface. The flow chart of the vision system is given in Fig. 25.

Before zooming into the detailed steps, three key algorithms in this vision system need to be highlighted. They are *ellipse detection*, *ellipse tracking* and *single-circle-based pose estimation*. These three algorithms are not restricted to the specific UAVGP competition tasks, but also applicable to many other UAV guidance tasks such as circle-based target following and circle-based landing.

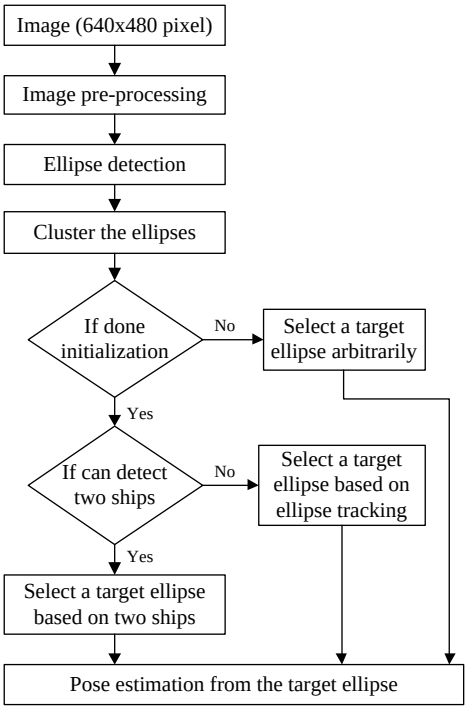

Fig. 25. Flow chart of the vision system.

(a) *Ellipse detection* has been investigated extensively in literature. Ellipse fitting, introduced in [21, 22]] is chosen as the core ellipse detection algorithm in this work, because it is very efficient compared to hough transform based ellipse detection algorithms proposed in [23, 24]. Unfortunately, ellipse fitting only fits the best ellipse for a given contour without questioning whether the contour is suitable to be seen as an ellipse in the first place. To complement its shortage, a three-step procedure, consisting of pre-processing, ellipse fitting and post-processing, is proposed and implemented for real-time and robust ellipse detection. The pre-processing is based on affine moment invariants (AMIs) [25], while the post-processing is based on the algebraic error between the contour and the fitted ellipse. The three-step procedure is not only robust against non-elliptical contours, but also can handle partial occlusion cases.

(b) *Ellipse tracking* is to continuously track a single ellipse after its detection has been initialized. In practical applications, multiple ellipses may be detected in an image but only one of them is to be targeted. There are two main challenges for ellipse tracking. First, the areas enclosed by the ellipses (the interested one and the others) are similar to each other in both shape and color. Thus, template matching based on color, shape or feature points may not be suitable for this task. Second, when implementing vision-based tracking algorithms on a flying UAV, the fast dynamical motion of the UAV may cause large displacement of the target ellipse between two consecutive images. In order to track the target ellipse smoothly, the frame rate of the image sequence must be high, which requires a very efficient implementation of the vision algorithm. To best solve these problems, an efficient image tracking method CAMShift [26] is chosen as the core of the tracking algorithm in this work. This algorithm can robustly track the target ellipse even when the scale, shape or color of the ellipse area are dynamically changing.

(c) *Single-circle-based pose estimation* is to calculate the 3D position of the target circle after its projected ellipse on the 2D image has been identified and tracked. Circle-based camera calibration and pose estimation have been studied in [27–31]. However, these existing work mainly focused on the cases of concentric circles [29–31], but our aim is to do pose estimation via only one circle. Theoretically, it is impossible to estimate the pose of a single circle purely from its perspective projection [28]. But from a practical point of view, the single-circle-based pose estimation problem can be solved by adopting a reasonable assumption that the image plane of the camera is parallel to the plane that contains the circle. This assumption is satisfied in our work because the onboard camera is installed on a pan-tilt mechanism which can ensure the image plane to be always parallel to the ground plane. By exploiting this assumption,

the 3D position of the targeted circle can be estimated from its projection as a single ellipse in the image.

More detailed explanations and discussions about these vision algorithms are documented in another paper due to its own research significance in the vision society [32]. We next explain the steps shown in Fig. 25.

(a) *Image* — Color images are captured at 5 Hz by the onboard camera. The image resolution is 640×480 pixels.

(b) *Image pre-processing* — The purpose of this block is to detect all the possible contours that may be formed by the projected circles. It consists of four sub-steps, namely image un-distortion, RGB to HSV conversion, color thresholding and contour detection. Since the color information of the circles has been given in the competition, the contours of the circles can be efficiently detected based on color thresholding in the HSV color space. It is also possible to detect the contours using other methods such as edge detection. However, they will consume more computational power.

(c) *Ellipse detection* — The perspective projections of the circles are ellipses. Based on the contours given in the previous step, the ones that correspond to ellipses should be detected.

(d) *Ellipse clustering* — The main aim for ellipse clustering is to decide whether the two ships are in the camera's field of view. If they are, there should be two clusters of ellipses. Since the four ellipses on each ship are of the same size and they are distributed evenly in a line, the size and position information can be used for ellipse clustering. If two clusters are obtained, then we can conclude that the two ships are in the field of view.

(e) *Initialization* — The UAV takes off at a location far from the ships and is guided to the two ships based on GPS. Once the vision system can detect two ships, an initialization procedure will be triggered. The purpose of the initialization procedure is to select a proper target circle according to the UAV's current task. The tracking algorithm is also initialized in this step.

(f) *Select a target ellipse arbitrarily* — This is a failsafe mechanism. After the UAV takes off, it will be guided to the ship area by GPS. However, the position measurement from GPS is not very precise. Hence, the UAV may not be guided to the exact position at which both ships are in the onboard camera's field of view. To handle this kind of situation, the vision system will return any detected ellipse until the *initialization* condition has been detected.

(g) *Select a target ellipse based on two ships* — If two ships are in the field of view, we will select the target ellipse based on the knowledge of the initial placements of the cargoes and the current task status. Suppose the buckets are initially placed on the right ship and they are required to be taken to the left one. If the current mission for the UAV is to grab a bucket, the vision system will select one circle that contains a bucket from the right ship. If the current mission for the UAV is to drop a bucket, the vision system will select one circle that does not contain a bucket from the left ship.

(h) *Select a target ellipse based on ellipse tracking* — In many of the cases, the two ships may not be in the field of view simultaneously. Then ellipse tracking algorithm is used to track a target ellipse over the image sequence.

(i) *Pose estimation from the target ellipse* — Once the target ellipse is selected in any of the ways mentioned above, the ellipse will be used to estimate the position of the circle center relative to the camera. The detailed method is explained in [32].

Figure 26 shows a number of consecutive images taken by the onboard camera. All the detected ellipses have been

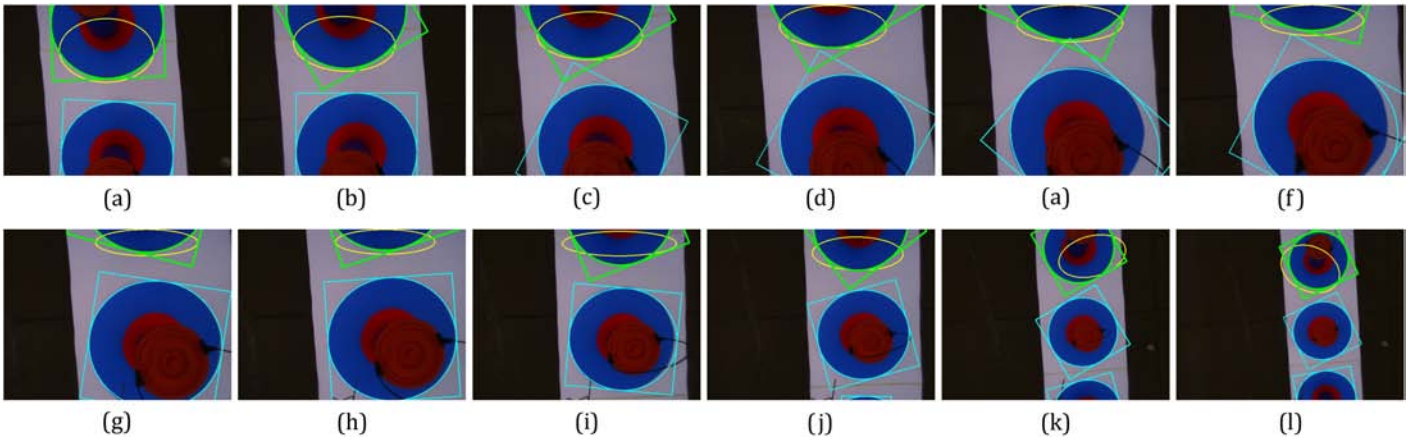

Fig. 26. Onboard images with the ellipse detection and tracking result.

drawn on each image. The green ellipse is the target ellipse tracked by the vision algorithm. The yellow ellipse is the area of interest returned by the CAMShift algorithm. It can be seen that all the ellipses have been successfully detected. The target ellipse is also tracked steadily even when its scale and shape keep varying.

5.2. *Trajectory planning*

When the target location has been precisely obtained in the camera frame, say $\mathbf{T}_c$, it needs to be transformed to the ship orientation, and the position offset between the camera and the UAV CG needs to be compensated. This leads to the ship-frame target location relative to the UAV expressed as:

$$\mathbf{T}_s = \mathbf{R}_{s/c}\mathbf{T}_c - \mathbf{R}_{s/b}\tilde{\mathbf{T}}_b, \tag{42}$$

where $\tilde{\mathbf{T}}_b$ is the camera's position in the UAV body frame. With the first two elements in $\mathbf{T}_s$ known, say t_x and t_y, the UAV x- and y-axis trajectory planning can be carried out independently. For each axis, the trajectory planning algorithm needs to know the current UAV velocity v_0, the maximum allowed velocity v_{max}, the maximum allowed acceleration a_{max} and the displacement from the current position to the final position S. In this specific case, $S = t_x$ or $S = t_y$. Figure 27 illustrates the fundamental idea of the proposed trajectory planning algorithm with the following characteristics:

(a) The resulting position reference is continuous and smooth.
(b) The resulting velocity reference is continuous.
(c) The resulting acceleration reference is non-continuous and only have three discrete possibilities, a_{max}, $-a_{max}$ and 0.
(d) The trajectory can start from a nonzero velocity but always ends at zero velocity.
(e) The area under the velocity profile from time 0 to T integrates to the total displacement.

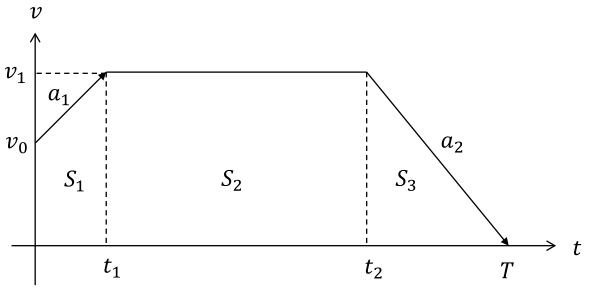

Fig. 27. Trajectory planning with continuous velocity.

Viewing Fig. 27 in a geometrical way, one can get the following two relationships:

$$\begin{aligned} S &= S_1 + S_2 + S_3 \\ &= \frac{1}{2}(v_0 + v_0 + a_1 t_1)t_1 + (v_0 + a_1 t_1)(t_2 - t_1) \\ &\quad + \frac{1}{2}(v_0 + a_1 t_1)(T - t_2), \end{aligned} \tag{43}$$

$$v_1 = v_0 + a_1 t_1 = -a_2(T - t_2). \tag{44}$$

If v_0, a_1, a_2 and T are known, t_1 can be solved in a quadratic equation form:

$$A_t t_1^2 + B_t t_1 + C_t = 0, \tag{45}$$

where

$$\begin{cases} A_t = a_1^2 - a_1 a_2, \\ B_t = 2v_0 a_1 + 2a_1 a_2 T, \\ C_t = v_0^2 + 2a_2 v_0 T - 2a_2 S. \end{cases} \tag{46}$$

Define $D_t = B_t^2 - 4A_t C_t$, then

$$\begin{cases} t_1 = -C_t/B_t & \text{if } A_t = 0; \\ t_1 = \dfrac{-B_t - \sqrt{D_t}}{2A_t} & \text{if } A_t C_t > 0 \ \&\ A_t B_t < 0 \ \&\ D_t > 0; \\ t_1 = \dfrac{-B_t + \sqrt{D_t}}{2A_t} & \text{if } A_t C_t < 0 \ \&\ D_t > 0; \\ t_1 = -1 & \text{if otherwise,} \end{cases} \tag{47}$$

and correspondingly,

$$t_2 = T + \frac{v_0 + a_1 t_1}{a_2}. \tag{48}$$

However, a_1, a_2 and T are not known exactly. To solve this problem, a recursive algorithm by listing all four cases of a_1–a_2 combination and continuously increasing T by 1-second step is proposed in Fig. 28. The iteration stops until a feasible solution occurs.

In actual implementations, this trajectory planning algorithm runs at 1 Hz only because it consumes high computational power with respect to an embedded computer. In addition, instead of inputting v_0 as the current velocity measurement, we use the current velocity reference, and instead of accumulating on the current position measurement for future position reference, we accumulate on the current position reference. This is to make sure that the velocity and position reference signals are always continuous. In fact, it is quite reasonable because at a slow rate of trajectory planning with strict velocity and acceleration constraints, the UAV's actual position, velocity and acceleration should be settled to more or less the same value of their corresponding reference signals at every instance the trajectory planning algorithm is called.

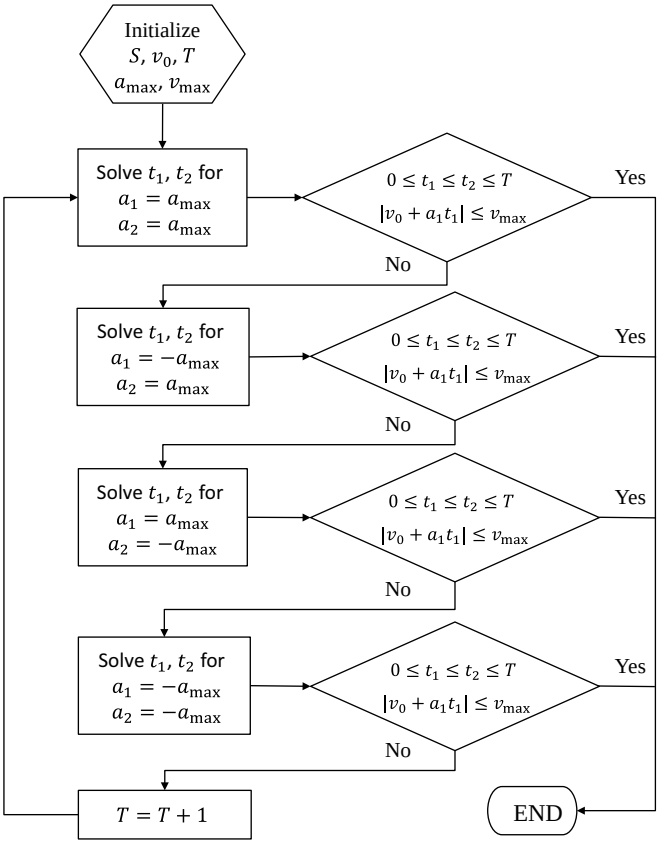

Fig. 28. Flowchart of the trajectory planning algorithm.

6. Software Realization

In order to implement the aforementioned GNC algorithms and to solve the mission logics in completing the UAVGP tasks, a real-time software system was developed. The followings will show the key features about NUS^{2}T-Lion's software system.

6.1. *Software structure*

The software structure of NUS^{2}T-Lion is illustrated in Fig. 29. It consists of three separate software programs, namely the onboard control software, the onboard vision software and the GCS software. For the onboard control and vision software, they both utilize the multiple-thread framework, so that time resource can be precisely distributed among different functional blocks (threads). It is also a more reliable way of implementing real-time UAV software so that the malfunction of individual thread will not halt the executing of others.

The onboard control software is developed using a real-time operating system (RTOS), namely QNX Neutrino, which provides reliable support for high precision timer and synchronization operations. Multiple tasks (threads), including operating with hardware devices like the navigation sensor, the Lidar and the servo control board, implementing the automatic control algorithms, logging in-flight data and communicating with GCS and the vision computer, are

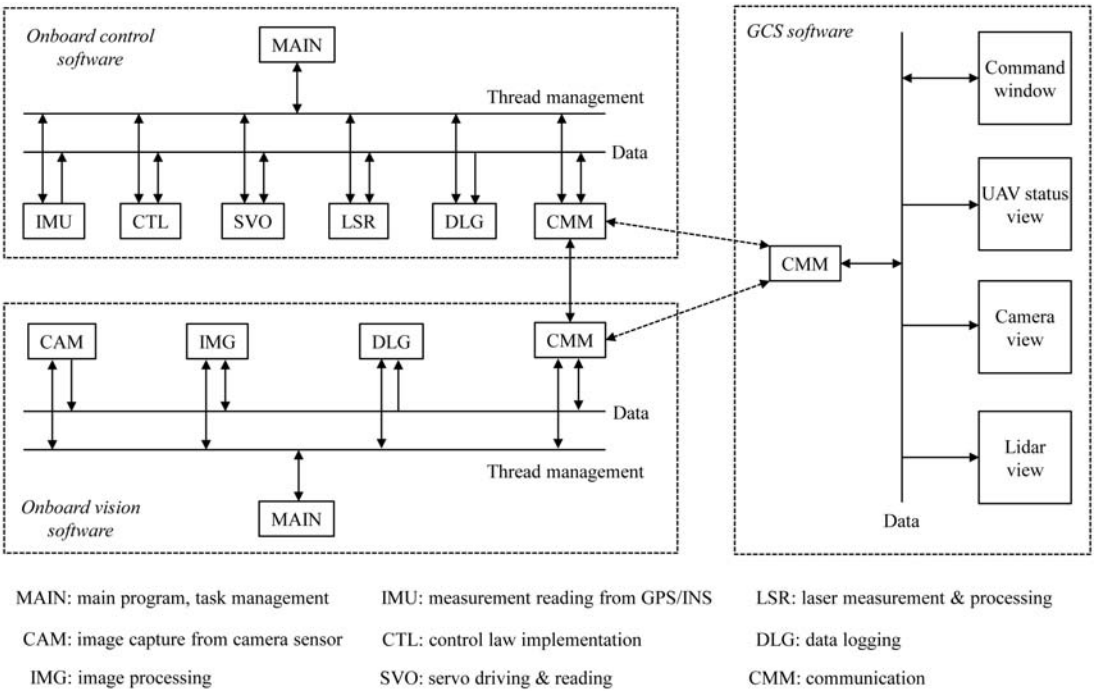

Fig. 29. Software structure of NUS^{2}T-Lion.

managed by the MAIN function. With the multiple-thread framework, it is also easy to run different threads with different frequencies. More details about this onboard control software can be found in [33].

Similarly, the onboard vision software is also divided by multiple tasks, namely image capturing from camera sensor, image processing, data logging and communication with GCS and the control computer. The operating system utilized on the vision computer is the very popular Ubuntu Linux. It supports abundant hardware drivers such as USB cameras and WiFi adapters and software libraries such as OpenCV, which are very convenient for the development of complicated vision algorithms.

For the GCS software, it runs on a laptop with Windows 7 system. Such a commercial operating system provides strong support for the development of user interfaces. Visual C++ is employed to develop and debug the GCS software. By utilizing the MFC (Microsoft Foundation Class) library, the global shared data are hosted in a document class, in which a variety of functions for data operating and visiting are integrated. While this document class is the kernel of the software program, there are also the communication thread and other threads to control the multiple views at the foreground user interface. The communication thread receive and send data to the UAV onboard system through WiFi link and the multiple displaying views periodically visit the contents of the document and update their respective displaying of new data received.

6.2. *Mission logics*

As the UAV cargo transportation application is mission oriented, the sequence and logics of the whole task operation are rather critical. It is implemented in the CTL thread of the onboard control software. An overview of the logic flow is illustrated in Fig. 30. There are six sequential tasks, namely taking off, navigating to ship, vision initialization, transporting cargos, returning home and landing. Since the mission is time constrained, a timer interrupt is also implemented. It triggers the return home task once a predefined maximum mission duration runs out. The details of each task is discussed as follows.

(1) The *Take Off* task will be triggered once the onboard software program receives the 'Mission Start Command' from the GCS. In this stage, the program let the helicopter warms up its engine by issuing a small and constant value to the throttle channel. After a while, the throttle channel control signal will be increased gradually until the engine enters the governor mode (main blades will now be controlled at a predefined rotational speed of 1750 rpm). After that, the program will slowly increase the control signal of the collective pitch channel so that the lift force increases. Once the collective pitch signal hits its trimming value for the hovering condition, the program will ask the reference generation function to issue a 'going up' trajectory. At the end of the trajectory, the programs throws a 'Take-off Event End' signal.

(2) The software program now enters the *Navigate to Ship* mode. In this stage, the program collects the position and velocity information from the GPS/INS system on the ship. A relative path to the ship with continuous relative position, relative velocity and relative acceleration references will be generated. The flight controller will continuously asks the helicopter to track this path so that the helicopter can catch up with the ship and have the same position and velocity profiles as the ship at steady state. Once catching up with the ship, the software will throw a 'Navigation Event End' signal. Note that this decision is made based on GPS/INS information. Physically the UAV may not be hovering so precisely above the center of the two ships.

(3) In the *Vision Initialization*, the vision system will first check whether it can detect two ships. If only one ship or part of a ship has been detected, the vision system will guide the helicopter to move towards one of the detected circles. In this way, there will be very high chance to see the other ship by taking advantage of the onboard camera's wide viewing angle. Once both ships are successfully detected, the software will be scheduled to the *Transporting cargo mode*.

(4) The *Transporting Cargo* task is the most sophisticated part of the mission. In this stage, the UAV will choose one of the cargos and fly to a position right above it. When the UAV horizontal position to the target cargo enters a small threshold, its height reference will be gradually adjusted down to an appropriate value so that the mechanical claw can grasp the bucket handle. Once the mechanical claw is closed, the UAV will be commanded to fly upwards quickly so that the limit switch sensors mounted under the claw platform can sense whether the cargo has been successfully loaded. If it senses that cargo has not been grasped successfully, the UAV will be commanded to go down again for another try. The above procedure will be repeated until the limit switch system detects a successful cargo loading. After that, the helicopter will be commanded to move to the unloading point. For the unloading task, the UAV has a similar procedure to check whether the cargo has been successfully released. If the detection is false, the UAV will quickly close and open its claw to try another release. For failsafe, when the vision system loses the cargo target for more than 10 s during the 'grasping' stage, the software will issue a 'going up' command so that the vision system can have a wider view, which leads to higher chance to retrieve the target. Once the vision system retrieves the target, the UAV will be commanded to go down and try grasping the cargo again. There is a counter to record how many cargos

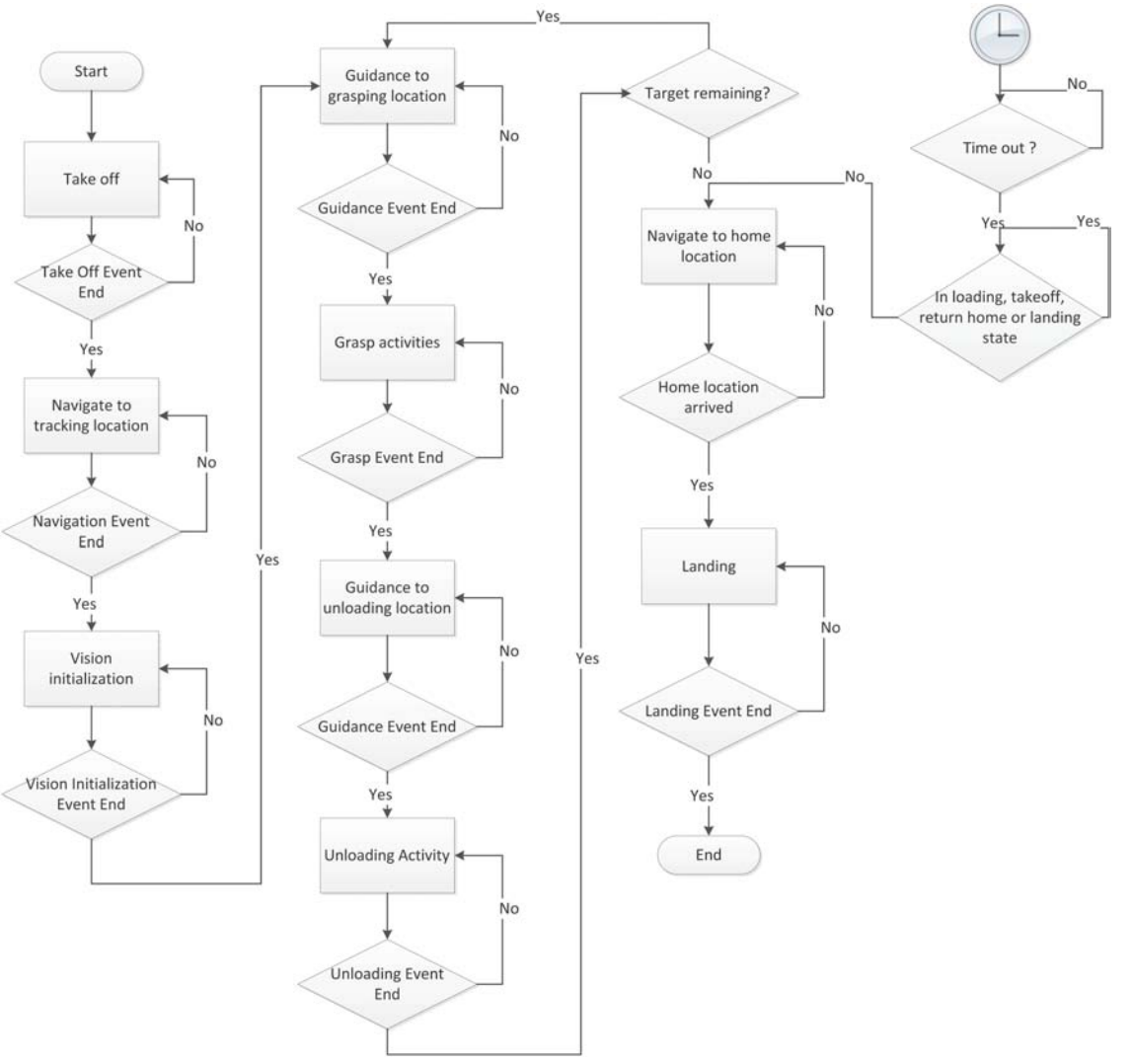

Fig. 30. Mission logics.

Fig. 31. NUS^{2}T-Lion in the International UAV Innovation Grand Prix.

remain to be transported. Once the counter hits zero, the program will jump out the current mode and enter the *Return Home* mode.

(5) When the helicopter has finished all its transportation tasks or the maximum mission time runs out, the *Return Home* task will be triggered. The software will generate a reference trajectory ending at a predefined height with the UAV's final planar position set to be the initial take-off point. The UAV will then follow this trajectory back to home location.

(6) *Landing* will be triggered as the helicopter flies right above its home location. The procedure for the *Landing* task is similar to the *Take Off* task. The software asks the flight controller to regulate the helicopter moving downwards with a constant speed at 0.5 m/s (if height is greater than 5 m) or 0.2 m/s (if height is less than 5 m). Once the UAV

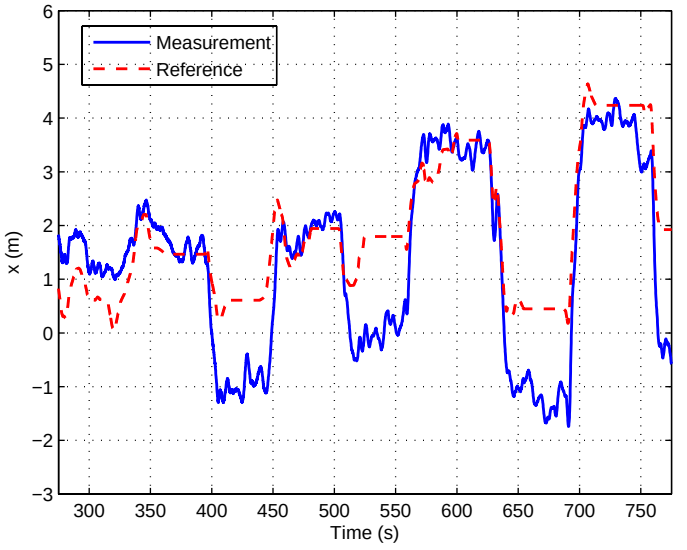

Fig. 32. UAV position response in the ship-frame *x*-axis.

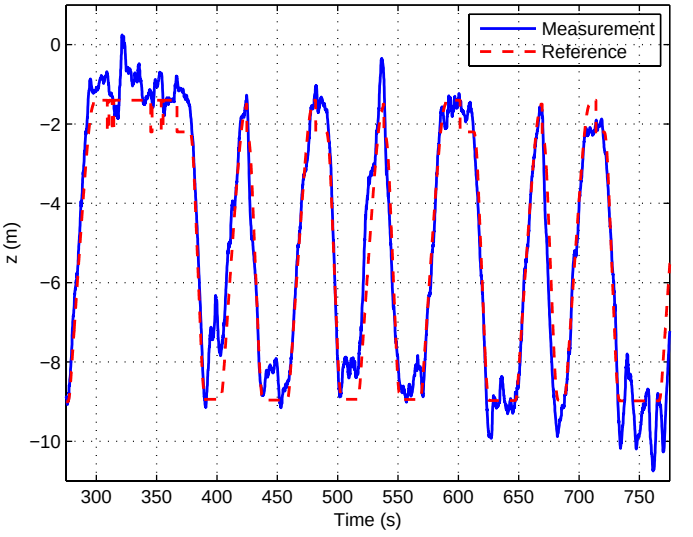

Fig. 34. UAV position response in the NED-frame *z*-axis.

landing gear approaches the ground (within 8 cm), the control signal to the throttle channel will jumps to a minimum value so that the engine shuts down completely.

7. Experimental and Competition Results

In preparation for the UAVGP competition, numerous flight tests have been carried out to verify the overall solution and to tune for the optimal performance. Figure 31 shows a snap shot of NUS^{2}T-Lion going to grab the second bucket in this competition. Figures 32–34 show the position data logged in one of the flight tests. As the raw data is obtained by GPS/INS and then converted to the ship frame, it may not be the ground truth. However, it still shows the control performance in a general sense and indicates whether the UAV is doing the correct movement. In Fig. 32, the *x* position signal becomes larger progressively because the UAV is moving from the first bucket to the fourth bucket. It always comes back to a position around zero because the reference path is purposed defined in a way that the onboard camera has the best view of the two ships before every loading or unloading dive. In Fig. 33, the *y* position signal goes back and forth, indicating alternative movements between the two ships. In Fig. 34, it is clear to see all the diving motions of the UAV. The UAV will stay at a very low altitude with a variable time duration depends on how many loading or unloading trials have been performed until the final success one.

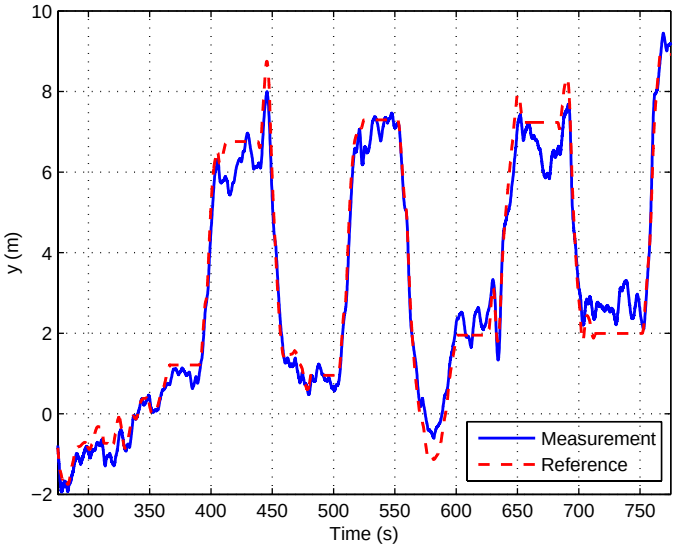

Fig. 33. UAV position response in the ship-frame *y*-axis.

With this kind of performance, NUS^{2}T-Lion has successfully accomplished the competition tasks in the UAVGP rotary-wing category. A final score of 1127.56 with 472.44 from the preliminary contest and 655.13 from the final has made the team second position in the overall Grand Prix. In fact, 655.13 is the highest score in the final round of the competition. It should be highlighted that unlike the preliminary contest, the final round of the competition requires the UAV to carry out the cargo transportation task with the 'ships' moving. This demands for better robustness and higher intelligence from the participants' UAV systems, and it is indeed the strongest point of the GNC solution proposed in this paper. The full process has been video-recorded and

uploaded to [34] and [35] for the English and Chinese versions, respectively.

8. Conclusions

In conclusion, this paper has presented a comprehensive design and implementation methodology in solving the rotorcraft UAV cargo transportation problem. Despite the hardware innovation, the main contribution of this paper is about the UAV guidance, navigation and control algorithms in solving this practical problem. Different from the conventional GPS/INS based UAV navigation scheme, this solution also integrates a sophisticated vision system capable of locating the cargo loading and unloading positions. By setting up a second GPS/INS unit on the moving platform with communication to the UAV onboard system, the controlled UAV is able to follow the dynamic platform with good performance. The GNC algorithms are also backed up by a multi-layer multi-thread software system, thus implemented successfully. With this set of technologies, the application of UAV cargo transportation or good delivery is more than realizable.

References

[1] A. Bujak, M. Smolarek and A. Gebczyńska, Applying military telematic solutions for logistics purposes, in *11th Int. Conf. Transport Systems Telematics* (2011) 248–256.
[2] M. Bernard and K. Kondak, Generic slung load transportation system using small size helicopters, in *IEEE International Conference on Robotics and Automation* (2009), pp. 3258–3264.
[3] M. Bisgaard, A. la Cour-Harbo and J. D. Bendtsen, Adaptive control system for autonomous helicopter slung load operations, *Control Eng. Pract.* **18**(7) (2010) 800–811.
[4] M. Orsag, C. Korpela and P. Oh, Modeling and control of MM-UAV: Mobile manipulating unmanned aerial vehicle, *J. Intell. Robotic Sys.* **69**(1–4) (2013) 227–240.
[5] J. Thomas, J. Polin, K. Sreenath and V. Kumar, Avian-inspired grasping for quadrotor micro UAVs, in *IDETC/CIE*, ASME, 2013.
[6] I. Maza, K. Kondak, M. Bernard and A. Ollero, Multi-UAV cooperation and control for load transportation and deployment, *J. Intell. Robotic Syst.* **57**(1–4) (2010) 417–449.
[7] K. Kosuge and M. Sato, Transportation of a single object by multiple decentralized-controlled nonholonomic mobile robots, in *IEEE Int. Conf. Intelligent Robots and Systems* (1999), pp. 1681–1686.
[8] B. K. G. Mrdjan Jankovic, Visually guided ranging from observations of points, lines, and curves via identifier based nonlinear observer, *Syst. Control Lett.* **25**(1) (1995) 63–73.
[9] S. Saripalli, J. F. Montgomery and G. Sukhatme, Vision-based autonomous landing of an unmanned aerial vehicle, in *IEEE Int. Conf. Robotics and Automation* (2002).
[10] F. Lin, X. Dong, B. M. Chen, K. Y. Lum and T. H. Lee, A robust real-time embedded vision system on an unmanned rotorcraft for ground target following, *IEEE Trans. Indust. Electron.* **59** (2012) 1038–1049.
[11] V. N. Dobrokhodov, I. I. Kaminer, K. D. Jones and R. Ghabcheloo, Vision-based tracking and motion estimation for moving targets using small UAVs, in *American Control Conference* (2006).
[12] P. Vela, A. Bester, J. Malcolm and A. Tannenbaum, Vision-based range regulation of a leader follwer formation, *IEEE Trans. Control Syst. Technol.* **17**(2) (2009) 442–448.
[13] H. Lang, M. T. Khan, K.-K. Tan and C. W. de Silva, Developments in visual servoing for mobile manipulation, *Unmanned Syst.* **1**(1) (2013) 143–162.
[14] G. Cai, B. Chen and T. Lee, *Unmanned Rotorcraft Systems* (Springer, New York, 2011).
[15] F. Lin, K. Z. Y. Ang, F. Wang, B. M. Chen, T. H. Lee, B. Yang, M. Dong, X. Dong, J. Cui, S. K. Phang, B. Wang, D. Luo, K. Peng, G. Cai, S. Zhao, M. Yin and K. Li, Development of an unmanned coaxial rotorcraft for the DARPA UAVForge Challenge, *Unmanned Syst.* **1**(2) (2013) 211–258.
[16] B. M. Chen, *Robust and H_∞ control*, in *Communications and Control Engineering Series* (Springer, 2000).
[17] B. M. Chen, T. H. Lee and V. Venkataramanan, Hard disk drive servo systems, in *Advances in Industrial Control Series* (Springer, New York, 2002).
[18] D. Mellinger and V. Kumar, Minimum snap trajectory generation and control for quadrotors, in *IEEE Int. Conf. Robotics and Automation* (2011), pp. 2520–2525.
[19] G. A. Borges and M. J. Aldon, A split-and-merge segmentation algorithm for line extraction in 2d range images, in *15th Int. Conf. Pattern Recognition* (2000), pp. 441–444.
[20] S. Zhao, F. Lin, K. Peng, B. M. Chen and T. H. Lee, Homography-based vision-aided inertial navigation of UAVs in unknown environments, in *Proc. 2012 AIAA Guidance, Navigation, and Control Conference* (2012).
[21] A. Fitzgibbon, M. Pilu and R. B. Fisher, Direct least square fitting of ellipses, *IEEE Trans. Pattern Anal. Mach. Intell.* **21** (1999) 476–480.
[22] S. J. Ahn, W. Rauh and H.-J. Warnecke, Least-squares orthogonal distances fitting of circle, sphere, ellipse, hyperbola, and parabola, *Pattern Recogn.* **34** (2001) 2283–2303.
[23] D. H. Ballard, Generalizing the hough transform to detect arbitrary shapes, *Pattern Recogn.* **13**(2) (1981) 111–122.
[24] R. A. McLaughlin, Randomized hough transform: Improved ellipse detection with comparison, *Pattern Recogn. Lett.* **19** (1998) 299–305.
[25] J. Flusser and T. Suk, Pattern recognition by affine moment invariants, *Pattern Recogn.* **26** (1993) 167–174.
[26] J. G. Allen, R. Y. D. 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*, Darlinghurst, Australia (2004), pp. 3–7.
[27] J. Heikkila and O. Silven, A four-step camera calibration procedure with implicit image correction, in *Proc. 1997 IEEE Conf. Computer Vision and Pattern Recognition* San Juan, Puerto Rico (1997), pp. 1106–1112.
[28] G. Wang, J. Wu and Z. Ji, Single view based pose estimation from circle or parallel lines, *Pattern Recogn. Lett.* **29** (2008) 977–985.
[29] J.-S. Kim, P. Gurdjos and I.-S. Kweon, Geometric and algebraic constraints of projected concentric circles and their applications to camera calibration, *IEEE Trans. Pattern Anal. Mach. Intell.* **27**(4) (2005) 637–642.
[30] G. Jiang and L. Quan, Detection of concentric circles for camera calibration, in *Proc. 10th IEEE Int. Conf. Computer Vision*, Beijing, China, (2005), pp. 333–340.
[31] D. Eberli, D. Scaramuzza, S. Weiss and R. Siegwart, Vision based position control for MAVs using one single circular landmark, *J. Intell. Robotic Syst.* **61** (2011) 495–512.

[32] S. Zhao, Z. Hu, M. Yin, K. Ang, P. Liu, F. Wang, X. Dong, F. Lin, B. Chen and T. Lee, A robust real-time vision system for autonomous cargo transfer by an unmanned helicopter, *Industrial Electronics, IEEE Transactions*, **PP**(99) (2014) 1–1.

[33] M. Dong, B. M. Chen, G. Cai and K. Peng, Development of a real-time onboard and ground station software system for UAV helicopter, *AIAA J. Aerospace Comp. Info. Comm.* **4** (2007) 933–955.

[34] Unmanned Aircraft Systems Group, National University of Singapore, "The video of the final round competition of UAVGP (English version)." http://www.youtube.com/watch?v=Ex0jgtX2ZrY (2013).

[35] Unmanned Aircraft Systems Group, National University of Singapore, "The video of the final round competition of UAVGP (Chinese version)." http://v.youku.com/v_show/id_XNjI5MTM2MDI0.html (2013).