INTELLIGENT LANDING SYSTEM FOR LANDING UAVS AT UNSURVEYED AIRFIELDS

Paul Williams, Michael Crump
BAE Systems Australia
paul.williams6@baesystems.com;michael.crump@baesystems.com

Keywords: *UAVs, autonomous landing, computer vision*

Abstract

Autonomous UAV technology is currently limited in its ability to land safely at alternate airfields that have not been accurately surveyed and where differential GPS is unavailable. This paper presents flight-test results of a new system that is capable of autonomously recovering a UAV to a previously unvisited airfield. The system utilizes a priori location information of airfields (for example ERSA, which provides the location of the airfield to approximately 100 m) to decide which airfield is the most feasible for landing. When the aircraft reaches the vicinity of the airfield it uses an onboard gimbaled camera to provide a precise vision-aided landing.

1 Introduction

UNMANNED air vehicles (UAVs) are widely used in defense environments to carry out autonomous or remotely-piloted missions. Remotely-piloted vehicles require numerous personnel to operate and maintain – in some cases more than is required for a manned aircraft. Autonomy removes much of the low-level decision making and therefore relieves the operating personnel of their strenuous workload. In addition, the operating footprint can be significantly reduced for autonomous UAVs. UAVs are also expanding into the civilian domain for a variety of purposes such as border patrol, mine monitoring, and bush fire spotting.

Most UAVs typically rely on ground support infrastructure for asset recovery. Fixed-wing UAVs require pre-surveyed runways and a differential GPS correction source to enable accurate autonomous landing. This paper addresses the need for a self-contained autonomous recovery system that enables the UAV to locate and land on suitable runways without ground support equipment such as differential GPS or ILS. Such systems can be used to recover UAVs in the event of unplanned mission events such as component failure or deteriorating weather conditions. The system can also be used to recover to ad hoc landing sites, or to simply provide greater robustness to navigation errors during landing.

The system described in this paper was designed specifically for dealing with recovery of fixed-wing aircraft to runways. However, its applicability extends to wider areas such as rotary-wing UAVs landing on moving decks. Unlike recent work presented in [1][2][3] which uses LIDAR to generate detailed maps of landing zones, our system is capable of landing safely using only passive vision sensors. Additional sensing can be used to augment our solution to handle issues such as obstacles. However, the implementation presented here focuses on the central problem of landing the UAV in a robust manner.

The system presented in this paper combines several key technologies that enable a truly autonomous recovery system for UAVs: 1) Onboard auto-routing system for generating waypoints through complex no-fly regions [4], 2) Gimbaled camera control and associated image processing algorithms for runways [5], 3) Closed-loop image processing and flight control robust to occlusions.

This paper is organized as follows: First, a review of the Intelligent Landing concept is presented; next, an overview of the capability of the system is given; the implementation of our approach in simulation and on real-time hardware is presented; finally, simulation results and flight-test results are presented that demonstrate the effectiveness of the system when implemented with low-cost sensors.

2 Intelligent Landing System

UAV systems that are currently used operationally do not have cognitive abilities. Human operators are required to plan and undertake missions, with the UAV flight computer performing the low-level task of flying the aircraft. Some operational UAVs have a fully automatic takeoff and landing capability. However, the landing phase is usually carried out using a combination of pre-surveyed runway with known landing waypoints, an accurate height sensor for determining height above ground, and differential GPS. These requirements can severely limit the operational case of modern UAV technology. There are several examples of unplanned mission events that can lead to a UAV operator needing to land the aircraft as soon as possible, such as: engine performance problems (temperature/oil pressure); deteriorating weather conditions; bird strike or attack damage; flight control problems related to malfunctioning hardware (single lane on multi-lane flight control systems, actuator failure); and so on. In current systems, the above situations can easily lead to the complete loss of the aircraft. The operator must either attempt a recovery to a mission alternate runway, or in the worst case, undertake a controlled ditching. Most modern UAV control systems allow multiple alternate landing sites to be specified as part of the mission plan. However, the limit with these alternate landing sites is that they require the same level of a priori knowledge (i.e., accurate survey) and support infrastructure as the primary recovery site. Therefore, this generally limits the number and location of the alternate landing sites. This is due to the amount of time and manpower required to setup, maintain, and secure the sites. The combined cost/effort and low probability of use detracts from the willingness to establish alternates. As mission requirements become more complex, it may not be possible for the aircraft to reach one of its alternate landing runways, and controlled ditching may result in recovery of the aircraft by enemies, or could result in injury or third party damage. To enable truly autonomous UAV operations, a key feature of the flight control computer would be its ability to select and land at a suitable alternate not previously surveyed and without any existing landing aids. This is the core idea of the Intelligent Landing System.

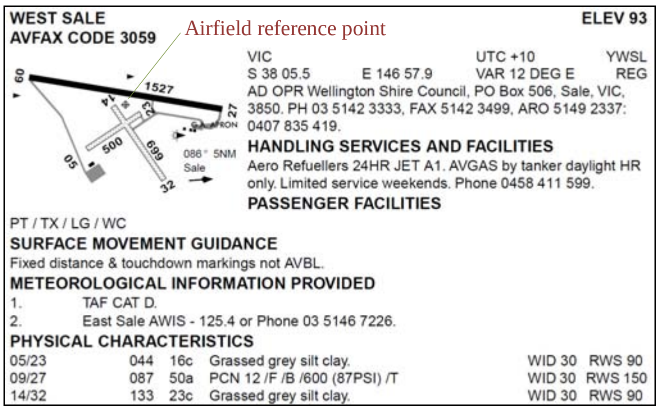
The system presented in this paper differs from similar concepts of using vision for aiding the landing of UAVs. Tang et al. [7] presented an algorithm for estimating the position and attitude of a UAV relative to a runway by projecting arbitrary points from the runway into the image frame. This assumes full knowledge of the runway position and orientation. Fitzgerald et al. [8] consider forced landing site selection using machine vision. However, their approach considers general landing sites such as paddocks and does not consider the more desirable case of landing on available runways. Their approach has not been used to land a UAV. Daquan and Hongyue [9] present simulation results of an algorithm that uses runway landing lights observed through an onboard camera to estimate a UAV's position and attitude relative to the runway. The results showed reasonably large errors. Their approach does not consider locating the runway initially, nor how to handle gaps in image data. Pan et al. [10] consider the estimation of approach angle and height of a UAV using a combination of monocular and stereo vision. The approach has not been used with imagery of real runways. The approach also requires continuous observation of the runway and does not consider initially locating the runway. Joo et al. [11] considered estimating the motion of a UAV during landing relative to a runway by using a ground-based camera system. However, this defeats the purpose of the Intelligent Landing System considered in this work. Pouya and

Saghafi [12] simulated the use of a fuzzy logic controller with runway relative position measurements for controlling the landing of a UAV. Their approach does not consider the actual image processing or its limitations. Meng et al. [13] considered the detection of runways within images based on template matching. They used actual images of runways in their work, but only considered the case where the aircraft is already on approach to the runway. Miller et al. [14] consider the use of image registration using a priori imagery of the runway to enable vision-aided landing. They demonstrated their approach via Microsoft flight simulator. To the best of our knowledge, all previous approaches to vision-aided landing of UAVs on runways have been based on a visual-servoing approach. These approaches are limited in nature and do not address the requirement for initially locating and planning an approach to the runway. The Intelligent Landing System described in this paper addresses all of these limitations.

3 Intelligent Landing Overview

3.1 Assumptions

In order to design a robust and reliable Intelligent Landing System capable of landing the plane as quickly as possible, it is necessary to make certain assumptions about the availability of runway locations. It is unrealistic to expect the UAV to perform a fast landing without any prior knowledge of nearby airfields. In fact, our system assumes a similar level of information is available to the flight control computer as is available to a manned aircraft in a similar situation. Our system utilizes an onboard database of airfields compatible with industry standard definitions such as Jeppesen NavData. The results presented in this paper were acquired using the En-Route Supplement Australia (ERSA) [15], which contains information about all airfields in Australia. An example of the key airfield data provided from ERSA is shown in Fig. 1. The important information used by the Intelligent Landing System is: 1) location of the airfield reference point (latitude = -38,05.5, longitude = 146,57.9), height above mean sea-level (93 feet), magnetic field offset (+12 deg), number of runways (3), runway surface characteristics (2 grass, 1 bitumen), runway lengths (1527, 699, 500 m) and width (30 m), and runway magnetic heading (44, 87, and 133 deg). Note that the airfield reference point gives the approximate location of the airfield to the nearest tenth of a minute in latitude and longitude (±0.0083 deg). This equates to an accuracy of approximately 100 m horizontally. Furthermore, the reference point in general does not lie on any of the runways and cannot be used by itself to land the aircraft. It is suitable as a reference point for pilots to obtain a visual of the airfield and land. The goal of the Intelligent Landing System is to perform a similar airfield and runway recognition and plan a landing/approach path. Note that the minimal amount of information required is the approximate location of the airfield. The remaining information is utilized, if available, but not required.


Airfield reference point
09
1527
14
23
27
APRON
500
699
05
32
086° 5NM
Sale

WEST SALE
AVFAX CODE 3059

ELEV 93

VIC		UTC +10	YWSL
S 38 05.5	E 146 57.9	VAR 12 DEG E	REG

AD OPR Wellington Shire Council, PO Box 506, Sale, VIC, 3850. PH 03 5142 3333, FAX 5142 3499, ARO 5149 2337: 0407 835 419.

HANDLING SERVICES AND FACILITIES

Aero Refuellers 24HR JET A1. AVGAS by tanker daylight HR only. Limited service weekends. Phone 0458 411 599.

PASSENGER FACILITIES

PT / TX / LG / WC

SURFACE MOVEMENT GUIDANCE

Fixed distance & touchdown markings not AVBL.

METEOROLOGICAL INFORMATION PROVIDED

1. TAF CAT D.
2. East Sale AWIS - 125.4 or Phone 03 5146 7226.

PHYSICAL CHARACTERISTICS

05/23	044	16c	Grassed grey silt clay.	WID 30	RWS 90
09/27	087	50a	PCN 12 /F /B /600 (87PSI) /T	WID 30	RWS 150
14/32	133	23c	Grassed grey silt clay.	WID 30	RWS 90

FIG. 1. EXAMPLE ERSA AIRFIELD DATA FOR WEST SALE AERODROME [15].

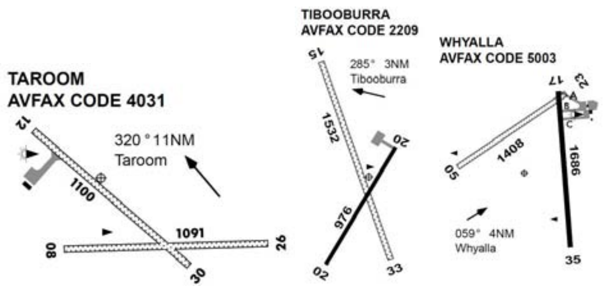


FIG. 2. EXAMPLE ERSA AIRFIELD REFERENCE POINTS (⊕).

In order for the UAV to identify and perform an autonomous landing on the desired runway, the navigation solution must be accurate. In low-cost UAVs, a GPS-aided INS system is the norm. In fact, GPS is heavily relied upon due to the poor performance of low-cost inertial measurement units. GPS has very good long term stability, but can drift in the short term due to variations in satellite constellation and ionospheric delays. The amount of drift is variable, but could be in the order of 20 m. This is one of the main reasons why differential GPS is a requirement for automatic landing of UAVs. Differential GPS allows accuracies of the navigation solution on the order of approximately 1-2 m. The Intelligent Landing System is assumed to have no differential GPS available.

The intended operation of the Intelligent Landing System is through the use of image processing to extract information about the airfield. The main reason for this is that virtually all UAVs are equipped with gimbaled cameras as part of their mission system. Therefore, in an emergency situation, the camera system can be re-tasked to enable a safe landing of the system. Other sensors such as LIDAR cannot always be assumed to be available. A core assumption is that additional sensors are not required to be installed on the platform to enable the Intelligent Landing System to work. Hence, only image processing can be used.

3.2 Intelligent Landing System Features

The Intelligent Landing System is designed as a plug-in mission system to existing UAV flight control computers. For the flight tests described in this paper, the flight control computer has been developed by BAE Systems Australia, and has been flown on the Mantis and Herti UAVs. It is a highly capable autonomous system with existing mission system interfaces. The Intelligent Landing System exploits many of the features of the flight control computer, such as routing and waypoint control.

The primary flight control computer is responsible for initiating an autonomous recovery via its internal health monitoring algorithms. A recovery can also be initiated via an operator command. Once a recovery has commenced, the Intelligent Landing System is fully responsible for commanding the UAV until it has completed the landing. If there is an active communications link, the Intelligent Landing System can be disabled. This is a feature that is required for flight testing in civilian airspace.

When the Intelligent Landing System is in control of the UAV, it finds the nearest suitable airfield using a tailored search algorithm. It must take into account runway surface characteristics, runway length, and flight time. Flight time is obviously affected by prevailing wind conditions, flyable regions, and current UAV position relative to neighboring airfields.

The Intelligent Landing System guides the UAV to the vicinity of the airfield, where onboard image processing algorithms locate the runway from background clutter. The system is able to distinguish the runway from other similar looking features such as roads and taxiways. A localization algorithm is able to convert image data into runway positioning information. Armed with this data, the Intelligent Landing System creates an alternate return-to-base (RTB) landing waypoint set, which includes inbound, and circuit waypoints.

In the final stage of the approach and landing, the onboard camera provides closed-loop information about the relative track error of the UAV from the desired landing path. The system accounts for the potential occlusion of a non-retractable undercarriage, as well as variations in satellite constellation that could result in relative errors between the aircraft and runway at touchdown. This results in the system typically landing close to the runway centerline. For the purposes of flight demonstration, the system aims to land longitudinally at roughly the runway designation marking, as shown in Figs. 3 and 4.

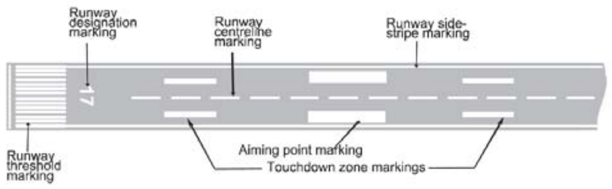
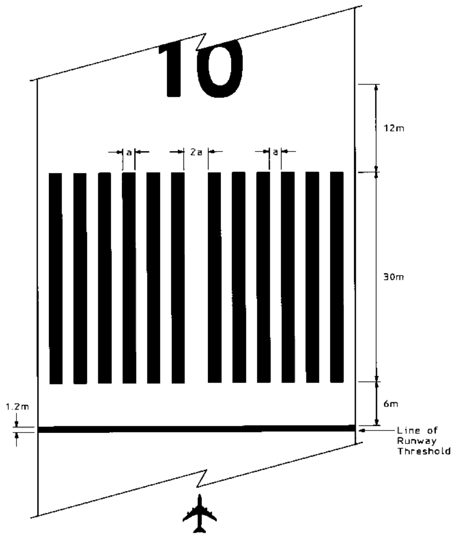

Runway designation marking
Runway centreline marking
Runway side-stripe marking
17
Runway threshold marking
Aiming point marking
Touchdown zone markings

FIG. 3. STANDARD RUNWAY MARKINGS [16].



RUNWAY WIDTH metres	NUMBER OF STRIPES	WIDTH OF STRIPE SPACE (a)metres
15,18	4	1.5
23	6	1.5
30	8	1.5
45	12	1.7
60	16	1.7

FIG. 4. RUNWAY THRESHOLD MARKING GEOMETRY [16].

4 Camera Control and Calibration

One of the key features of our system is its ability to work effectively even when low-cost sensor packages are used. The rationale for employing low-cost sensors is to demonstrate the robustness of the algorithms. It is reasonably straightforward to extend the system to use other sensing packages depending on the customer's requirements. It is, in general, much more difficult to go from using expensive sensors to using low-cost sensors due to the large difference in accuracy and reliability. A large amount of effort is required to mitigate the effects of sensor errors when using low-cost sensors.

The key requirement of the system is the presence of a gimbaled camera on the UAV. A gimbaled camera allows the Intelligent Landing System to control the direction and zoom level of the imagery it is analyzing. We have installed a turret manufactured by Rubicon with model number AHIR25D, shown in Fig. 5. This turret includes an electro-optical (EO) and infrared (IR) camera, and is capable of performing full 360 pan and -5 to 95 deg tilt. For the Intelligent Landing System, we have exclusively used the EO camera, which is a Sony FCB-EX408C. The camera uses the VISCA binary communication protocol, which is transmitted over RS-232. Turret control commands are transmitted to the Rubicon device using an ASCII protocol, also over RS-232.

Care must be taken when communicating with the Rubicon device to avoid packet collisions or sequential commands sent too quickly. Message rate handling is dealt with via algorithms implemented in Simulink. All turret and gimbal control is handled via the Simulink model, and translation of Simulink commands into a byte-stream occurs via a software codec. The Simulink implementation of the interfaces matches the interface used on the actual device, minimizing the need for hand written code.

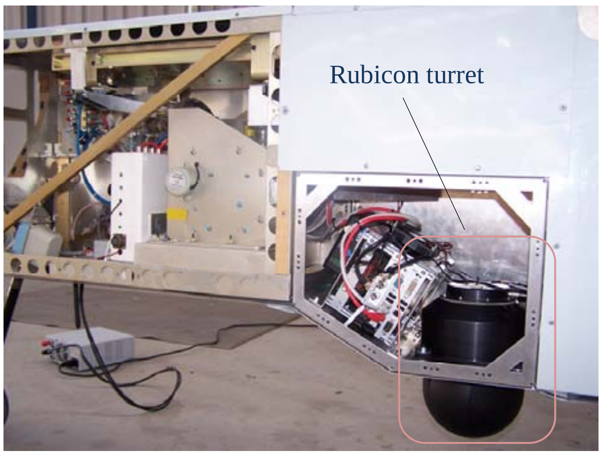


FIG. 5. RUBICON AHIR25D TURRET USED FOR INTELLIGENT LANDING TEST FLIGHTS.

4.1 Camera Coordinates

A pinhole camera model is assumed which relates measurements in the sensor frame to the image frame (u,v), as shown in Fig. 6,

$$u = \frac{1}{f_u}(y_s / x_s) + u_0, v = \frac{1}{f_v}(z_s / x_s) + v_0 \quad (1)$$

where f_u and f_v are the camera focal lengths, and u_0 and v_0 are pixel offsets (principal point).

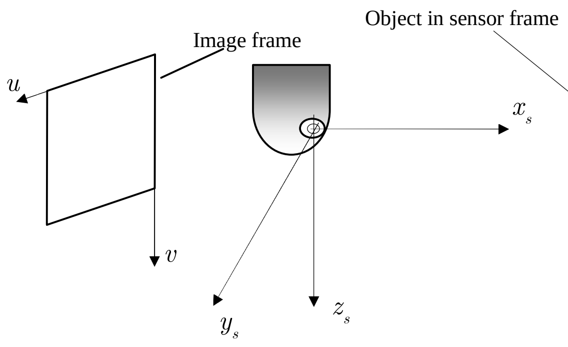


FIG. 6. PINHOLE CAMERA MODEL USED TO CONVERT PIXEL MEASUREMENTS INTO MEASUREMENTS IN MEASUREMENT FRAME.

The bearing and elevation measurements are derived from the pixel information according to

$$\varphi = \tan^{-1}\left[\left(u - u_0\right) / f_u\right] \qquad (2)$$

$$\vartheta = \tan^{-1}\left[\left(v - v_0\right)\cos\varphi / f_v\right] \qquad (3)$$

Note that distortion effects must be accounted for before using the raw pixel coordinates. This step has been omitted here. The uncertainty of a measurement is specified in the image plane, and must be converted into an equivalent uncertainty in bearing/elevation. The uncertainty in bearing/elevation takes into account the fact that the intrinsic camera parameters involved in the computation given in Eqs. (2) and (3) are not known precisely. The uncertainty is computed via

$$\sigma_{BE} = \left(\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}}\right)\sigma_x\left(\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}}\right)^T + \left(\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{p}}\right)\sigma_p\left(\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{p}}\right)^T \qquad (4)$$

where $\boldsymbol{p} = [f_u, f_v, u_0, v_0]^T$, $\boldsymbol{y} = \left[\varphi, \vartheta\right]^T$, $\boldsymbol{x} = [u, v]^T$, σ_x is the uncertainty in the pixel plane coordinates, and σ_p is the uncertainty in the camera parameters.

In order to compute the camera position in the Earth-Centered-Earth-Fixed (ECEF) frame, the coordinate frames used by the system must be first explained. Fig. 7 shows the coordinate frames used by the system. The ECEF frame (X, Y, Z) is the reference frame used for navigation. The local navigation frame (N, E, D) is an intermediate frame used for the definition of platform Euler angles and is common across external interfaces for navigation. The NED frame has its origin on the WGS84 ellipsoid. The IMU/body frame is aligned with the vehicle body axes and has its origin at the IMU. The installation frame (x_i, y_i, z_i) has its origin at a fixed point on the camera mount. This allows some of the camera extrinsics to be calibrated independently of the mounting on the airframe. The gimbal frame (x_g, y_g, z_g) has its origin at the center of the gimbal axes. Finally, the measurement frame has its origin at the focal point of the camera/sensor system.

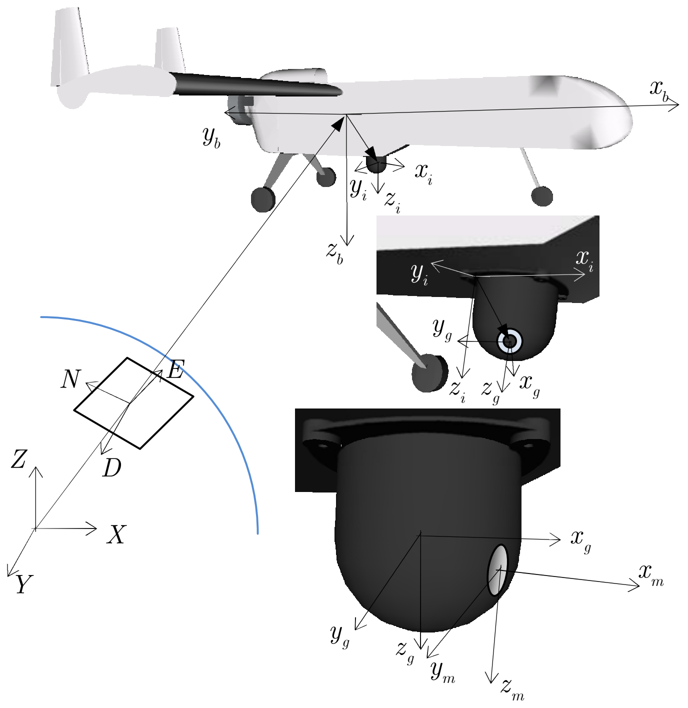


FIG. 7. COORDINATE FRAMES USED BY GIMBALED CAMERA.

The position of the camera in the ECEF frame is given by

$$\boldsymbol{p}_m^e = \boldsymbol{p}_b^e + \boldsymbol{C}_n^e \boldsymbol{C}_b^n (\boldsymbol{p}_i^b + \boldsymbol{C}_i^b (\boldsymbol{p}_g^i + \boldsymbol{C}_g^i \boldsymbol{p}_m^g)) \qquad (5)$$

where $\boldsymbol{p}_b^e$ is the position of the aircraft/IMU in the ECEF frame, $\boldsymbol{C}_n^e$ is the direction cosine matrix representing the rotation from the NED frame to the ECEF frame, $\boldsymbol{C}_b^n$ is the direction cosine matrix representing the rotation from the body to the NED frame, $\boldsymbol{p}_i^b$ is the position of the installation origin in the body frame, $\boldsymbol{C}_i^b$ is the direction cosine matrix representing the rotation from the installation from to the body frame, $\boldsymbol{p}_g^i$ is the position of the gimbal origin in the installation frame, $\boldsymbol{C}_g^i$ is the direction cosine matrix representing the rotation from the gimbal frame to the installation frame, and $\boldsymbol{p}_m^g$ is the origin of the measurement frame in the gimbal frame.

The direction cosine matrix representing the rotation from the measurement frame to the ECEF frame is given by

$$\boldsymbol{C}_m^e = \boldsymbol{C}_n^e \boldsymbol{C}_b^n \boldsymbol{C}_i^b \boldsymbol{C}_g^i \boldsymbol{C}_m^g \tag{6}$$

where $\boldsymbol{C}_m^g$ is the direction cosine matrix representing the rotation from the measurement frame to the gimbal frame.

In the case of a body-fixed camera, the matrix $\boldsymbol{C}_i^b \boldsymbol{C}_g^i \boldsymbol{C}_m^g$ is a fixed quantity and can be calibrated reasonably reliably. However, when a turret is used, the gimbal frame is constantly changing with respect to the installation frame. This makes calibration of the sensor more difficult, particularly where there are uncertainties in the timing, and is addressed in a later section.

4.2 Gimbal Control

The commands sent to the turret are in the form of rate commands about the pan and tilt axes. This is required if the stabilization function is to be used on the turret. The stabilization mode uses gyroscopes in the turret to mitigate the effects of turbulence on the pointing direction of the camera. A velocity control loop is executed on our turret control subsystem, which is responsible for control of the turret, camera system, and collecting and forwarding image data and associated meta-data to the Intelligent Landing System. The velocity control loop uses pan and tilt commands and closes the loop with measured pan and tilt values. Due to the delays in the turret communication protocol and the low rate of feedback sampling, the control loop employs a predictive mechanism to provide for fine angular control.

High-level pointing commands are received by the turret controller from the Intelligent Landing System. The Intelligent Landing System arbitrates to select from commands issued from a ground controller, and its own internally generated commands. In all cases, the ground controller has priority and can manually command the turret to move to a specified angle, angular rate, or point at a selected latitude/longitude/height. The operator must manually relinquish control for the autonomous system to be in control.

During intelligent landing, the turret is commanded to point in a variety of different modes controlled through a single interface. The turret can be made to "look at" selected points specified in different reference frames (camera frame, body frame, NED frame, ECEF frame). The most useful mode is a bounding box mode that adaptively changes the pointing position and zoom level to fit up to 8 points in the camera field-of-view. The turret control algorithm computes a line of sight and uses a Newton algorithm to iteratively calculate the required pan/tilt angles.

4.3 Zoom Control

Camera zoom is either controlled independently of the pointing command, or coupled to it. The zoom can be set via a direct setting command as a ratio or rate, or can be specified as an equivalent field-of-view measured by its projection onto the ground plane (i.e., units are in meters). This type of control maintains an area in the image quite well by adjusting the zoom level as a function of the navigation position relative to the pointing location. The combined zoom mode uses up to 8 points in the ECEF frame to select a zoom setting such that all 8 points lie within the image.

4.4 Image and Gimbal Timestamp

In addition to running the abstracted turret control loop, the turret control subsystem is responsible for capturing still images from the camera video feed and timestamping the data. The turret control subsystem obtains 100 Hz navigation data from the Intelligent Landing System, and zoom and gimbal measurements from the camera and turret, respectively. These measurements are buffered and interpolated upon receipt of an image timestamp (timestamps are in UTC Time). Euler angles are interpolated by using a rotation vector method, whereas other quantities can be interpolated in a straightforward manner.

Navigation data is timestamped according to the UTC Time of the IMU data packet used on the 100 Hz frame, which is kept in synchronization with GPS time via the PPS line from the GPS unit. Gimbal data is timestamped on transmission of a trigger pulse sent by the turret controller to the turret. The trigger is used by the turret to sample the gimbal angles, which is transmitted to the turret controller after it receives a request for the triggered data. Zoom data is timestamped by the turret controller on transmission of the zoom request message. Note that the camera does not provide a means for accurate timestamping of the zoom. We have assumed a constant delay time between writing the binary stream to the serial port and the sampling of the zoom position. This assumption is reasonable and is alleviated somewhat through the use of discrete zoom levels by the turret controller. Images are timestamped upon receipt of the first byte from the video capture card, and is intercepted on a device driver level. However, this does not provide the time that the image was actually captured by the camera. We have computed the assumed constant offset by placing an LED light in front of the camera in a dark room. A static map of pixel location versus pan position was obtained by manually moving the turret to various positions. The turret was then commanded to rotate at various angular rates while simultaneously capturing images. By extracting the position of the LED, and by using the inverse map of pan position, we were able to estimate the image capture time and compare it with the time of arrival of the first image byte. We found a constant offset of approximately 60 ms between the image capture time and the arrival of the first byte at the turret controller.

4.5 Intrinsic Calibration

Intrinsic calibration refers to estimating the properties of the camera lens system. The key parameters that must be calibrated are: 1) focal lengths, 2) principal point, 3) radial and tangential distortion. These parameters can be estimated using standard calibration techniques involving a checkerboard of known geometry. The lens parameters are determined by solving a least squares problem so as to minimize the error in pixel coordinates for the corners of the checkerboard tiles. This is done over a large number of images photographed at various orientations relative to the camera. The parameters are stored as a function of zoom level and interpolated by the turret control subsystem during operation. The interpolated intrinsic parameters are attached as meta-data, together with camera extrinsic parameters and navigation data, and passed to the Intelligent Landing System. This interface allows the turret control subsystem to be swapped for an alternate one without impacting the configuration of other subsystems.

We note that our choice of low-cost turret results in in-house development of camera and gimbal control algorithms. Much more expensive turrets can be used off-the-shelf.

4.6 Extrinsic Calibration

Extrinsic calibration refers to the estimation of the position and orientation of the camera relative to the navigation solution's reference axes on the aircraft. In our case, we are interested in estimating the installation position and installation angles, using the terminology given in Fig. 7. When the turret control subsystem is installed into the aircraft, the position and orientation is known coarsely, but not to the accuracy required for accurate localization of targets. In general, the installation angles dominate the projection error compared to the installation position. A dedicated flight is performed to obtain data to enable extrinsic calibration to be performed. The method used is described in this section.

Calibration is performed using an expectation maximization algorithm. A set of 5 white targets are placed in an accessible paddock near the airfield with the aid of a surveyor's GPS (accuracy ~2 cm), as shown in Fig. 8. The aircraft is flown in a manner to allow the turret to be pointed at the targets from a variety of positions and attitudes. The flight path is designed to incorporate a large number of alternating turns to give good observability of the true aircraft heading by the navigation system. During the flight, the turret is manually pointed at the targets with zoom settings covering the range from maximum to minimum.

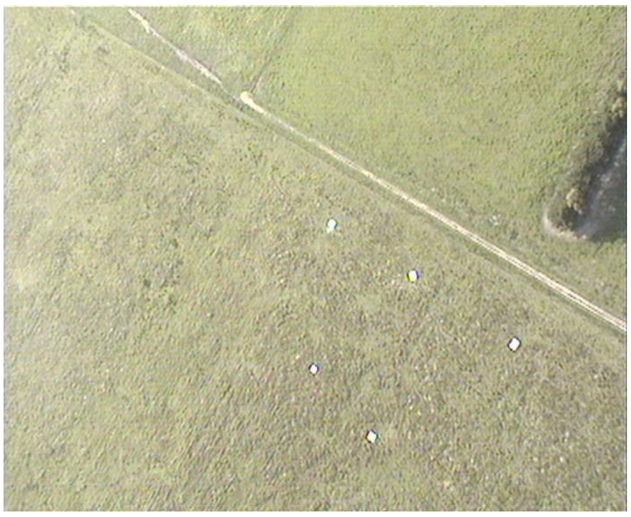


FIG. 8. EXAMPLE OF CAPTURED GEO IMAGE SHOWING CALIBRATION TARGETS.

The algorithm used to determine the extrinsic calibration parameters may be summarized as follows:

1) The current best estimate of the extrinsic parameters is used to approximately calculate if the targets are visible in any particular image.
2) If the targets are estimated to be visible, a blob detection algorithm is performed on the image to extract the pixel coordinates of the targets.
3) Using the known position of the targets, and the meta data associated with the image, together with the extrinsic estimates, the projected pixel coordinates of the targets are calculated.
4) A Mahalanobis gate is used to reject detected blobs that are too far away from the predicted pixel coordinates. All surviving coordinates are added to a list, together with the associated target index.
5) A sequential quadratic programming problem is solved to minimize the cost function of the pixel error squared divided by pixel uncertainty (uncertainty is the combined blob pixel uncertainty and the uncertainty in pixel coordinates arising from the use of the navigation solution in the calculation), summed over all images in the list.
6) The process is repeated starting at (1) until the estimated extrinsic parameters converge to within a tolerance.

We have found that the above iterative procedure converges quickly with an accuracy that depends on the combination of timestamp accuracy and navigation solution drift during the flight. We have also found that performing the calibration makes a large difference in the localization accuracy of the system.

5 Simulation Architecture

All components of the system are implemented in the Simulink environment (running MATLAB R2010b SP1). Simulink provides a powerful model-based design environment that can be linked to a set of functional requirements described in a set of IBM Rational DOORS modules. In fact, all system interfaces are described in DOORS modules, from which a complete set of interface code for C++ and MATLAB is autogenerated.

The complete flight vehicle is modeled using Simulink blocks (engine, undercarriage, aerodynamics, actuators), together with a representative model of the environment (wind, gust, turbulence, gravity, atmosphere, ground).

Flight physics are modeled in the Earth-Centered-Earth-Fixed (ECEF) coordinate frame. In additional to the physics model of the vehicle, a complete sensor simulation with interfaces and noise characteristics matching the real sensors is performed (Inertial Measurement Unit, GPS, air data, etc). The sensor data is fed into a Simulink model of the flight control computer to enable simulation of the closed-loop control system. Note that because the flight control computer code is generated from the Simulink models via Real-Time Workshop, the simulation is virtually an exact match to the implementation on the target hardware on the real vehicle. The high fidelity simulation environment allows the Intelligent Landing algorithms to be tested completely without ever having an aircraft in the sky.

Two levels of simulation based testing are performed. In the first level, the image processing algorithms are not in the loop (as they require a detailed synthetic environment to be generated). This enables the full Intelligent Landing System to be developed and tested without relying on a working image processing system. In the second level of tests, the model of the image processing algorithms is replaced with a Simulink S-function that communicates to an external vision processing module. The vision processing module runs on a dedicated PC that renders the camera view from the turret. The captured view from the turret is fed into the image processing algorithm code to exercise its logic and returns messages matching the software interface. During the rendering and image processing testing, the simulation in Simulink is held and waits for the vision processing module to return. This ensures that timing anomalies of the test system do not interfere with algorithmic development. Communication with the vision processing module is done over UDP.

6 Embedded Real-Time Implementation

The algorithms that are designed, implemented and tested in Simulink are converted into C code by the Real-Time Workshop Embedded Coder. As noted above, a set of software interfaces to the code is autogenerated from DOORS, minimizing the amount of hand-code that is required to run the software as an executable on the target hardware. The C code is wrapped in C++ with appropriate data and message handling code. It is scheduled using a real-time scheduler which runs in the main thread at 100 Hz. The linux operating system running a real-time kernel is used for the Intelligent Landing System. The Green Hills Integrity operating system is used for the actual flight control computer.

In addition to the core algorithms, an additional process is executed to log the input and output data of the algorithmic code. All the inputs and outputs are logged, which enables offline data replays to be conducted using a set of support tools that we have developed. This enables bugs to be found and removed in a rapid fashion.

7 Hardware Overview

The flight vehicle platform that we have used to conduct flight operations is the Kingfisher 2 vehicle, shown in Fig. 9. Kingfisher 2 was designed and built by BAE Systems Australia to allow rapid prototyping of payloads and is not a production system. The key characteristics of the platform are given in Table 1.

The flight control computer forms the core product of the vehicle management system, which consists of an actuation unit, two GPS units, an IMU, an air data system, weight on wheel sensors, an accurate height sensor, and a C2 communications system. The FCC uses a Radstone IMP2A as the hardware board. The FCC runs a variety of core processes in separate address spaces to maintain a high integrity system. The FCC enclosure with IMU is shown in Fig. 10. The FCC mounting in the airframe, together with the Intelligent Landing System computer and turret, is shown in Fig. 11.

TABLE 1. KINGFISHER PLATFORM ATTRIBUTES.

Attribute	Value
Mass (including payload)	125 kg
Wing span	4.13 m
Wing area	2.67 m^2
Max. airspeed	100 kts
Max. cross-wind	15 kts
Max. tail-wind	10 kts

FIG. 9. KINGFISHER 2 VEHICLE ON APPROACH TO WEST SALE.

A ground station that communicates over a C2 link is used to control and monitor the vehicle. It uses the same hardware as the FCC for reliability, but additionally utilizes a Windows-based graphical user interface.

A manual handset is used by an external pilot to take over control in the case of failure of a flight critical component (such as IMU or air data). The pilot's commands are sent to the vehicle via the ground station and bypass the core autonomous processes (the inputs are scaled and limited to prevent the pilot from overstressing the airframe). In the flight tests reported in this paper, the pilot was never required to take control of the vehicle.

FIG. 10. BAE SYSTEMS AUSTRALIA'S FLIGHT CONTROL COMPUTER (FCC) USED FOR VEHICLE MANAGEMENT OF KINGFISHER 2 PLATFORM.

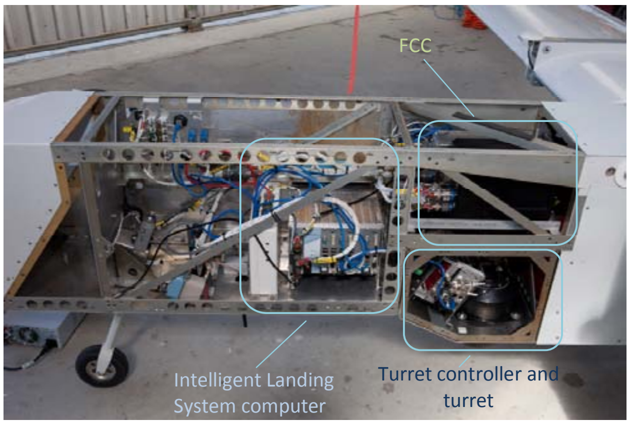

FCC
Intelligent Landing System computer
Turret controller and turret

FIG. 11. KINGFISHER 2 SHOWING CORE HARDWARE COMPONENTS FOR AR SYSTEM.

The Intelligent Landing System algorithms are housed on a Kontron CP308 board, which features a Core-2 Duo. One core is dedicated to running the All-Source Navigation system described in [6], and the second core is dedicated to running the Intelligent Landing System and path generation [4] algorithms. In addition, inputs and outputs from all processes are logged on a solid state hard drive. The turret control subsystem runs on a Kontron CP307, and is responsible for managing the turret control and logging all raw imagery. A 50 minute flight generates roughly 40 GB of image data.

8 West Sale Aerodrome Operations

The Kingfisher 2 vehicle is operated and maintained at a facility situated near the West Sale Aerodrome, see Figs. 12 and 13. The ground station is housed in the hangar, and communication antennae as well as a differential GPS antenna are mounted on the hangar roof.

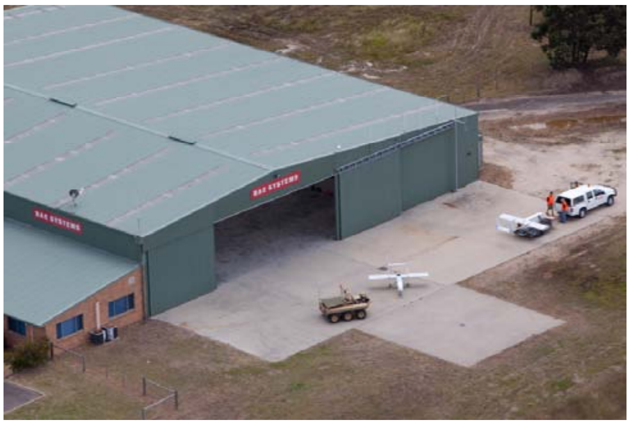


FIG. 12. VIEW OF BAE SYSTEMS AUSTRALIA'S HANGAR FACILITY.

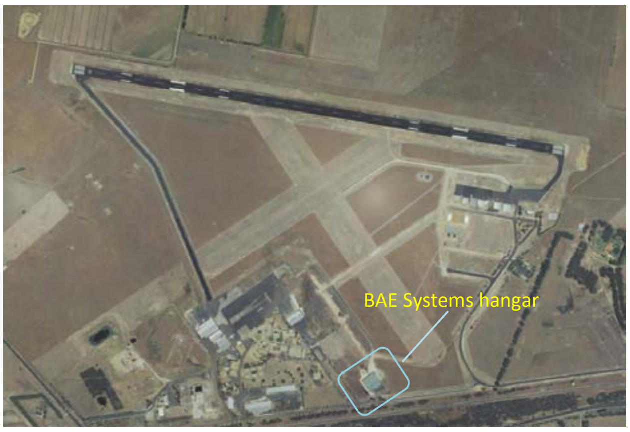


FIG. 13. AERIAL PHOTO OF WEST SALE AERODROME SHOWING BAE SYSTEMS AUSTRALIA'S HANGAR FACILITY.

The vehicle is first prepared for flight at the hangar, and navigation alignment and pre-flight tests are conducted outside of the hangar. The aircraft is connected to an external power source to prevent the internal batteries from running low. Once standard operation of the system is confirmed, it is towed on the back of a trailer to one of the ends of the runway (see Fig. 13). The takeoff direction is specified either via air traffic control (when inside ATC hours), or by the wind direction (when outside ATC hours). After the appropriate set of radio calls are made to ATC, the vehicle is deployed to the runway and becomes airborne under autonomous control.

During UAV operations, other aircraft frequently takeoff and land at the Aerodrome, and the operations crew must ensure separation is maintained (this is achieved via the autonomous system using features such as loiter, heading hold, altitude hold, etc.). Aircraft deconfliction can sometimes interrupt a flight test. The Intelligent Landing System is designed to perform its recovery uninterrupted. If the flight vehicle is routed away from the runway area, the Intelligent Landing process is manually restarted.

9 Intelligent Landing Results

9.1 Simulation Results

Simulations of the complete system were conducted using the MATLAB/Simulink test environment described earlier. Simulation results are presented in this section using two sets of flight extents: a relaxed set that allows the UAV to perform the way it would in normal operations if the Intelligent Landing System were to be used, and the actual extents used for the flight tests. The actual flight extents restrict the UAV to fly North of the airfield at all times. Fig. 14 shows an expanded set of flight extents that allows the vehicle to perform orbits of the runway. This should be contrasted with the actual flight extents used during flight tests, as shown in Fig. 15.

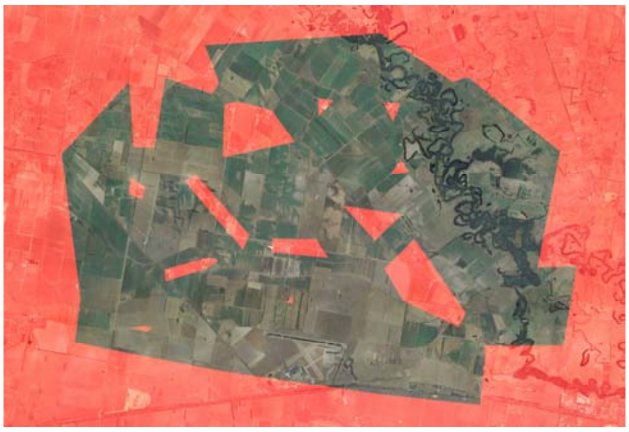


FIG. 14. FLIGHT AREA WITH EXPANDED FLIGHT EXTENTS AROUND AIRFIELD.

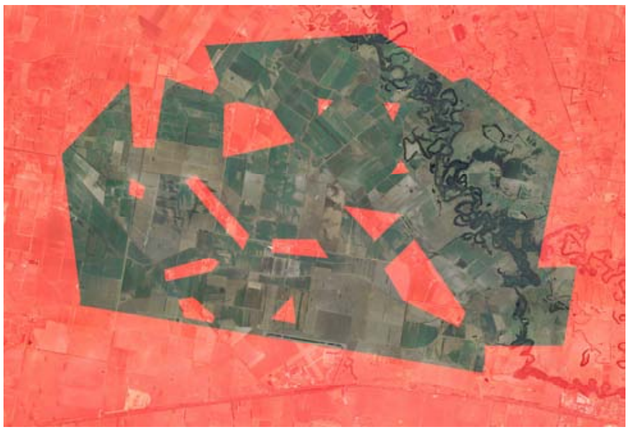


FIG. 15. FLIGHT AREA WITH RESTRICTED FLIGHT EXTENTS AROUND AIRFIELD.

The first set of simulation results is for the case where the flight extents are more generous around the airfield than is actually the case. Fig. 16 shows the major simulation results for this case, including the aircraft trajectory and RTB set. The Intelligent Landing System is initiated with the vehicle directly north of the airfield flying in a northwest direction. The Intelligent Landing System generates the magenta route in Fig. 16, which takes the vehicle from its original position, through the flight extents and into the route that passes close to the runway. The route forces the vehicle to climb to 700 m ellipsoid height. After locating the runway, a full set of inbound waypoints are generated by the Intelligent Landing System (Fig. 16). The inbound waypoints form a tree that the flight control computer uses to determine the shortest path into circuit from the location that a landing command is received. The inbound waypoints are guaranteed to allow the aircraft to reach the runway from within any flight extent, and results in a complex tree. When the vehicle enters into inbound state in this example, it chooses the first inbound point connected to the terminal inbound points, which allows it enough time to descend from altitude and join circuit at the correct height. The set of circuit waypoints, together with the circuit trajectory are also shown in Fig. 17. It can be seen that the system utilizes the available flight area to its fullest to enable the longest possible approach.

Fig. 18 shows the height profile during the landing phase. The aircraft enters landing state at 70 m ellipsoid height. The runway is simulated at 28 m height above ellipsoid, which is consistent with the approximate height of the navigation solution after touchdown. Fig. 18 shows that the sink rate is maintained fairly constant until flare, which arrests the down velocity.

Fig. 19 shows the 09 runway end together with the runway position uncertainty converted into uncertainty in the runway corners. It can be seen that the real runway corners (selected for illustration as the piano key corners) lie within their corresponding error ellipses. In fact, the error is on the order of a piano key width (1.5 m). Similar results can be seen for the 27 end of the runway in Fig. 20. Fig. 21 shows the complete runway solution, which illustrates the accuracy of the runway position. In the simulation, the aircraft is able to land on the runway centerline without any major issues.

Simulation results for the case of normal extents (flight extents to be used for actual flights) are shown in Fig. 22. In this example, the aircraft commences the Intelligent Landing process to the west of the airfield flying in a westerly direction. The aircraft immediately performs a left-hand turn and returns to the vicinity of the airfield. The aircraft is forced to fly further North of the airfield due to the flight extents boundary.

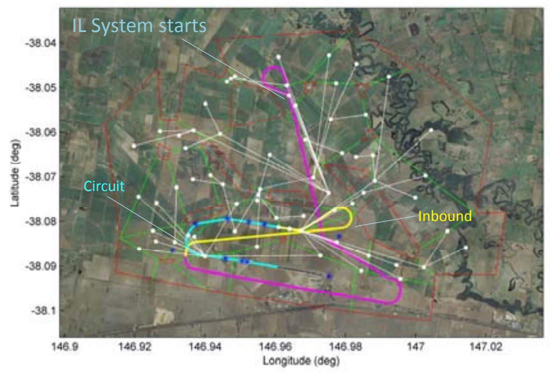


FIG. 16. SIMULATED LANDING AT WEST SALE AERODROME WITH EXPANDED FLIGHT EXTENTS.

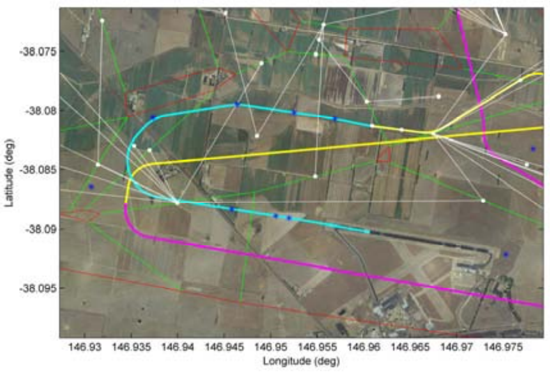


FIG. 17. CLOSEUP OF CIRCUIT FOR SIMULATED LANDING AT WEST SALE AERODROME WITH EXPANDED FLIGHT EXTENTS.

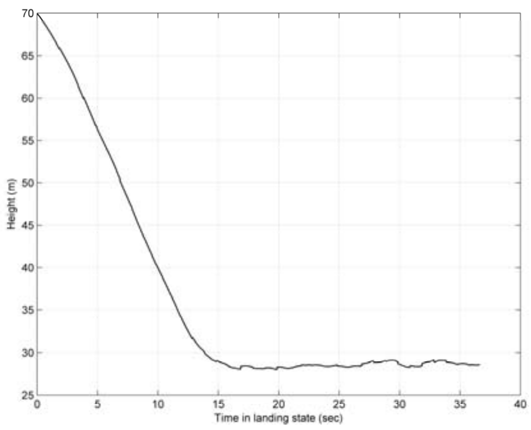


FIG. 18. VEHICLE HEIGHT ABOVE WGS84 ELLIPSOID DURING LANDING PHASE FOR EXPANDED EXTENTS CASE.

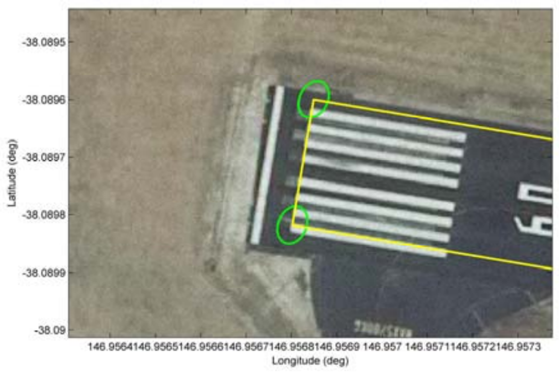


FIG. 19. RUNWAY POSITIONING SOLUTION FOR SIMULATED EXPANDED FLIGHT EXTENTS (09 RUNWAY END).

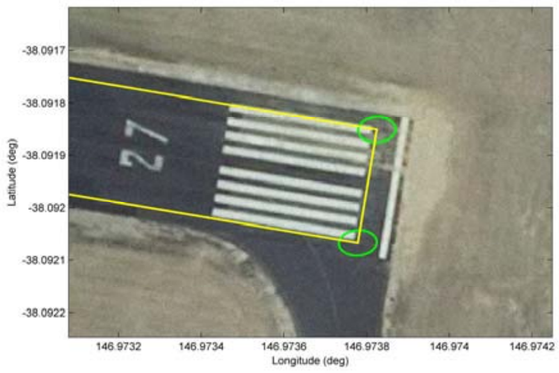


FIG. 20. RUNWAY POSITIONING SOLUTION FOR SIMULATED EXPANDED FLIGHT EXTENTS (27 RUNWAY END).

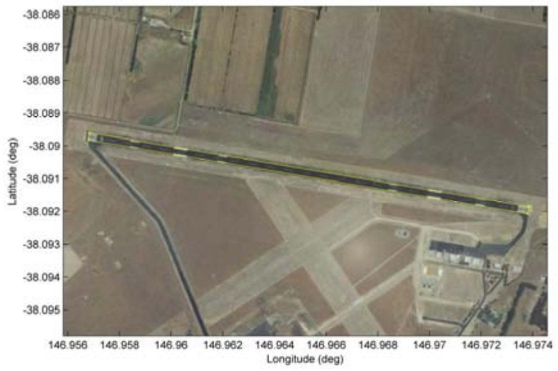


FIG. 21. RUNWAY POSITIONING SOLUTION FOR SIMULATED EXPANDED FLIGHT EXTENTS (COMPLETE RUNWAY).

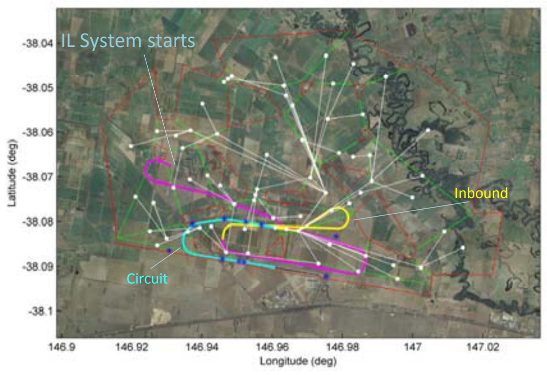


FIG. 22. SIMULATED LANDING AT WEST SALE AERODROME WITH NORMAL FLIGHT EXTENTS.

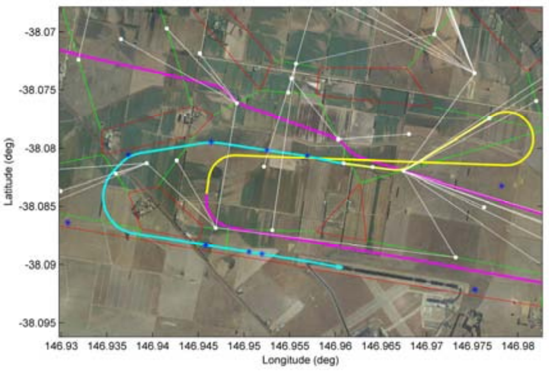


FIG. 23. CLOSEUP OF CIRCUIT FOR SIMULATED LANDING AT WEST SALE AERODROME WITH NORMAL FLIGHT EXTENTS.

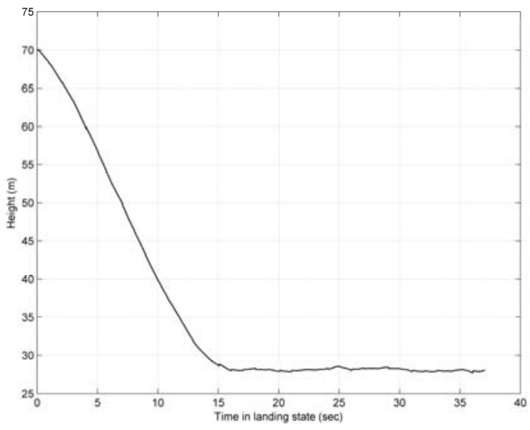


FIG. 24. VEHICLE HEIGHT ABOVE WGS84 ELLIPSOID DURING LANDING PHASE FOR NORMAL EXTENTS CASE.

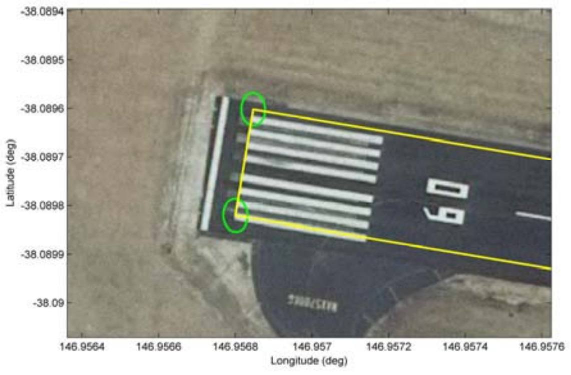


FIG. 25. RUNWAY POSITIONING SOLUTION FOR SIMULATED NORMAL FLIGHT EXTENTS (09 RUNWAY END).

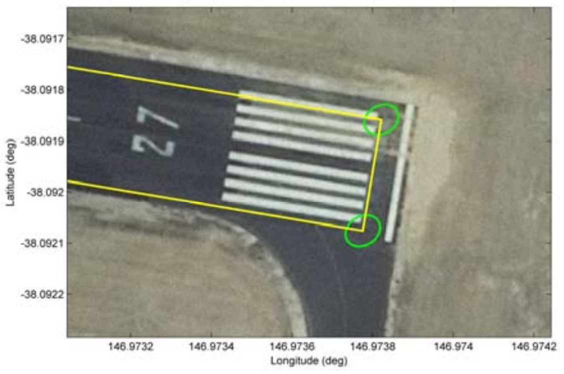


FIG. 26. RUNWAY POSITIONING SOLUTION FOR SIMULATED NORMAL FLIGHT EXTENTS (27 RUNWAY END).

Fig. 23 shows a close-up of the generated circuit and landing phase of the recovery. The circuit is restricted to fly through the available gap in the flight extents following the turn onto final. The results show that the algorithms successfully generate a circuit satisfying all requirements.

Fig. 24 shows the height profile of the vehicle during the landing phase. Comparing Figs. 18 and 24 illustrates the similarity of the two landings performed by the system under vastly different conditions. The height tracking is virtually identical.

Fig. 25 shows the 09 runway positioning solution, which is consistent with the results of the first simulation. In fact, the accuracy is improved compared to the first simulation, although this is variable depending on actual simulation parameters used. Similar results are obtained for the 27 runway end (Fig. 26). The true runway corners lie within the error bounds of the estimate.

9.2 Flight Results

Flight trials of the Intelligent Landing System were conducted in November 2011. The flight trials were undertaken to achieve a variety of objectives, one of which was to flight test the Intelligent Landing algorithms. A total of 22 flights were undertaken, but only 14 had active payload control during landing. All autonomous landings were carried out successfully.

The flight trials were segmented into stages. In the first stage, flights were conducted with no active image processing. In the second stage, image processing was used to close-the-loop in the Intelligent Landing System. Examples of images captured by the system are shown in Figs. 27 and 28. Fig. 29 shows imagery obtained during landing using closed-loop image processing. This illustrates the accuracy of the approach and landing using the system.

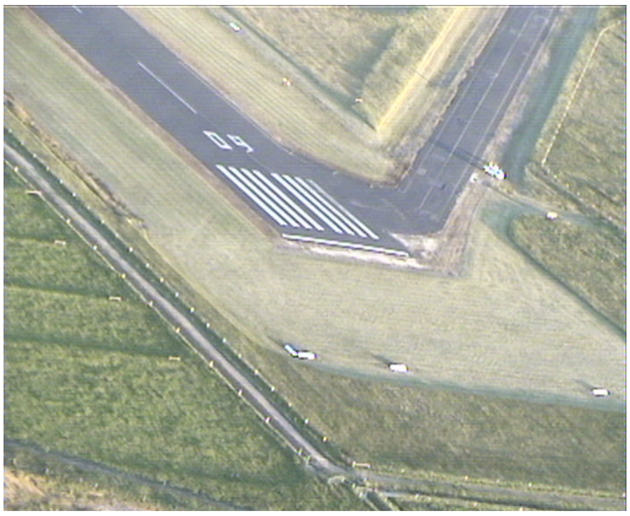


FIG. 27. EXAMPLE RUNWAY IMAGE CAPTURED DURING FLIGHT TEST FOR 09 RUNWAY END.

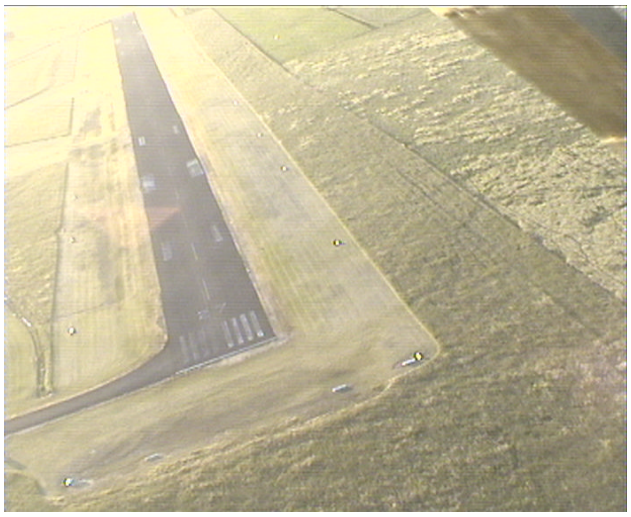


FIG. 28. EXAMPLE RUNWAY IMAGE CAPTURED DURING FLIGHT TEST FOR 27 RUNWAY END.

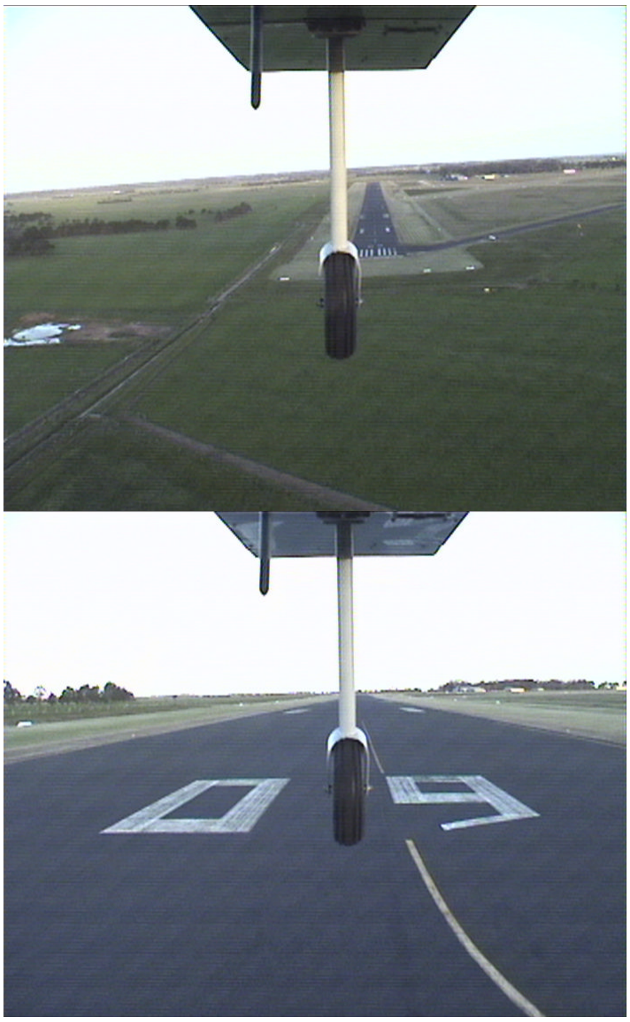


FIG. 29. EXAMPLE OF RUNWAY IMAGERY DURING CLOSED-LOOP APPROACH TO RUNWAY.

During the flight trials of the Intelligent Landing System, the aircraft was restricted from flying significant distances from the airfield due to the eyes-on-vehicle flying rules. Although this does not invalidate the results of the algorithms in any way, it renders the flight scenario somewhat less than ideal for demonstrating the full capability of the system. An example of the Intelligent Landing System flight results is shown in Fig. 30. The Intelligent Landing System is enabled directly north of the airfield. The aircraft performs a right-hand turn followed by left-hand turn to return to the airfield. Fig. 31 shows that the real flight system behavior during inbound, circuit and landing follows the results obtained in simulation very well. The ground track remains within the flight extents at all times. Fig. 32 shows the height profile of the vehicle during the landing phase. Comparison of the height profile with the simulation results shows that the vehicle is at a slightly higher height (~2 m higher) than in simulation when landing commences. The amount of time spent in the landing state is slightly longer due to the head wind present in the real flight. The navigation height at touchdown is approximately 25 m, which differs by 3 m from the simulated runway height.

An example of the runway positioning for the flight test is given in Figs. 33 and 34. Results for the 09 runway end clearly show an almost exact match to the true corner positions.

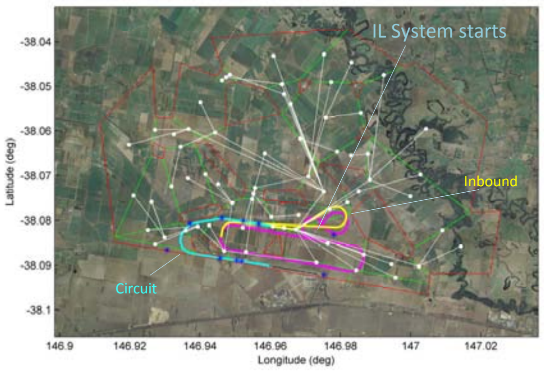


FIG. 30. EXAMPLE OF FLIGHT TEST RESULTS FOR INTELLIGENT LANDING SYSTEM.

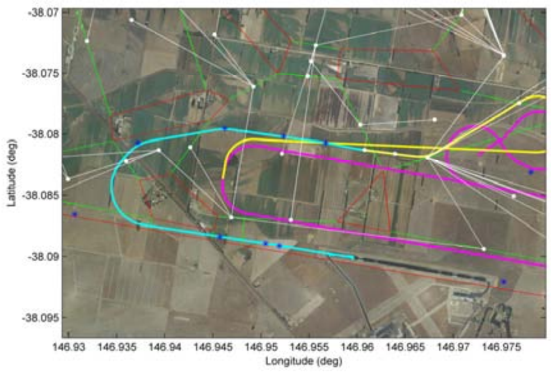


FIG. 31. CLOSEUP OF CIRCUIT FOR EXAMPLE FLIGHT TEST RESULTS OF AUTORECOVERY SYSTEM.

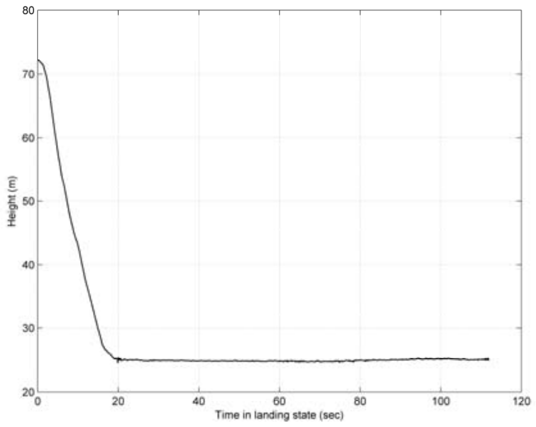


FIG. 32. VEHICLE HEIGHT ABOVE WGS84 ELLIPSOID DURING LANDING PHASE FOR EXAMPLE FLIGHT TEST RESULTS.

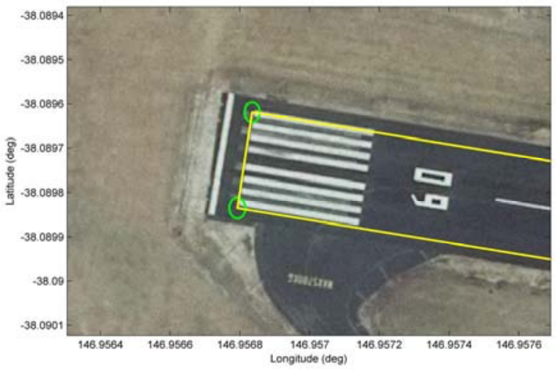


FIG. 33. RUNWAY POSITION SOLUTION DURING FLIGHT TEST (09 RUNWAY END).

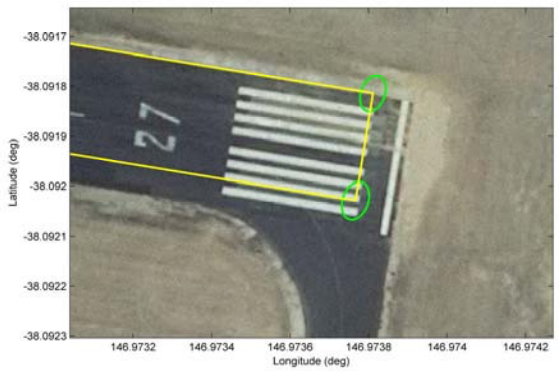


FIG. 34. RUNWAY POSITION SOLUTION DURING FLIGHT TEST (27 RUNWAY END).

10 Conclusions

The Intelligent Landing System is an important element to be incorporated into future UAVs. It allows the UAV to recover to previously unknown airfields and execute precise autonomous landings. The Intelligent Landing System combines several key technologies, including all-source navigation, autonomous waypoint generation, and image processing. Simulation and flight test results demonstrate that the Intelligent Landing System is able to negotiate difficult flight areas, locate the runway, and land without any human intervention. The system has been proven to work using low cost sensors and without an active differential GPS correction source. The Intelligent Landing System has further applications to other types of UAV systems, such as rotary wing UAVs. Future plans include extending the system to deal with non-static landing areas such as a moving deck.

Acknowledgements

The authors would like to thank Ramon Wilkinson, Kynan Graves, Andrew Fairmaid, Travis Pereira, Nelson Evans, Jason Bult, and Paul Harbison for their support in making the system described here a reality. All the technology described in this paper was funded internally be BAE Systems Australia.

References

[1] Scherer, S., *Low-Altitude Operation of Unmanned Rotorcraft*. PhD Thesis, The Robotics Institute, Carnegie Mellon University, Pittsburgh, 2011.

[2] Chamberlain, L., Scherer, S., and Singh, S., Self-aware helicopters: full-scale automated landing and obstacle avoidance in unmapped environments. *Proceedings of AHS Forum 67*, Virginia Beach, May 2011.

[3] Scherer, S., Chamberlain, L., and Singh, S., Online assessment of landing sites. *AIAA Infotech@Aerospace*, Atlanta, April 2010.

[4] Williams, P., and Crump, M., Auto-routing system for UAVs in complex flight areas. *Proceedings of the 28th International Congress of the Aeronautical Sciences*, Brisbane, September 2012.

[5] Graves, K., Visual detection and classification of runways in aerial imagery. *Proceedings of the 28th International Congress of the Aeronautical Sciences*, Brisbane, September 2012.

[6] Williams, P., and Crump, M., All-source navigation for enhancing UAV operations in GPS-denied environments. *Proceedings of the 28th International Congress of the Aeronautical Sciences*, Brisbane, September 2012.

[7] Tang, D., Li, F., Shen, N., and Guo, S., UAV attitude and position estimation for vision-based landing. *Proceedings of the 2011 International Conference on Electronic and Mechanical Engineering and Information Technology*, August 2011, pp.4446-4450.

[8] Fitzgerald, D., Walker, R., and Campbell, D., A vision based forced landing site selection system for an autonomous UAV. *Proceedings of the 2005 International Conference on Intelligent Sensors, Sensor Networks and Information Processing*, December 2005, pp.397-402.

[9] Daquan, T., and Hongyue, Z., Vision based navigation algorithm for autonomic landing of UAV without heading & attitude sesnors. *Proceedings of the Third International IEEE Conference on Signal-Image Technologies and Internet-Based System*, December 2007, pp.972-978.

[10] Pan, X., Ma, D.-Q., Jin, L.-L., and Jiang, Z.-S., Vision-based approach angle and height estimation for UAV landing, *Proceedings of the Congress on Image and Signal Processing*, May 2008, pp.801-805.

[11] Joo, S., Ippolito, C., Al-Ali, K., and Yeh, Y.-H., Vision aided inertial navigation with measurement delay for fixed-wing unmanned aerial vehicle landing. *Proceedings of the 2008 IEEE Aerospace Conference*, March 2008, pp.1-9.

[12] Pouya, S., and Saghafi, F., Autonomous runway alignment of fixed-wing unmanned aerial vehicles in landing phase. *Proceedings of the Fifth International Conference on Autonomic and Autonomous Systems*, April 2009, pp.208-213.

[13] Meng, D., Yun-Feng, C., and Lin, G., A method to recognize and track runway in the image sequences based on template matching. *Proceedings of the 1st International Symposium on Systems and Control in Aerospace and Astronautics*, January 2006, pp.1221-1224.

[14] Miller, A., Shah, M., and Harper, D., Landing a UAV on a runway using image registration. *Proceedings of the 2008 IEEE International Conference on Robotics and Automation*, Pasadena, May 2008, pp.182-187.

[15] Anon., *En-Route Supplement Australia (ERSA)*, Air Services Australia, March 2012.

[16] Anon., *Manual of Standards Part 139 – Aerodromes*, Civil Aviation Safety Authority, Australian Government, February 2012.

Copyright Statement