
Proceedings Paper:
McAree, O., Aitken, J.M. and Veres, S.M. (2016) A model based design framework for safety verification of a semi-autonomous inspection drone. In: 2016 UKACC 11th International Conference on Control (CONTROL). UK Automatic Control Conference, 31 Aug - 02 Sep 2016, Belfast, UK. IEEE. ISBN: 978-1-4673-9891-6.

https://doi.org/10.1109/CONTROL.2016.7737551

A model based design framework for safety verification of a semi-autonomous inspection drone

Owen McAree, Jonathan M. Aitken, Sandor M. Veres
Department of Automatic Control and Systems Engineering
The University of Sheffield
Sheffield, S1 3JD
United Kingdom
Email: {o.mcaree, jonathan.aitken, s.veres}@sheffield.ac.uk

***Abstract*— In this paper, we present a model based design approach to the development of a semi-autonomous control system for an inspection drone. The system is tasked with maintaining a set distance from the target being inspected and a constant relative pose, allowing the operator to manoeuvre the drone around the target with ease. It is essential that the robustness of the autonomous behaviour be thoroughly verified prior to actual implementation, as this will involve the flight of a large multi-rotor drone in close proximity to a solid structure. By utilising the Robotic Operating System to communicate between the autonomous controller and the drone, the same Simulink model can be used for numerical coverage testing, high fidelity simulation, offboard execution and final executable deployment.**

***Index Terms*— Model based design, verification, autonomy, inspection, drone**

I. Introduction

The civilian use of Small Unmanned Aircraft Systems (SUAS) or drones for removing personnel from hazardous situations has grown significantly in recent years. One particular sector which has embraced the technology is that of structural inspection [1], [2], [3]. Previously, this type of activity required persons to work on scaffolding or via rope access, leading to 39 deaths in Great Britain during 2013/14 [4]. Using drones to collect imagery which can then by analysed by an expert represents a significant improvement in safety, whilst also reducing the cost of obtaining such data. This also allows inspections to be conducted on a more regular-cycle, this allows the condition of structures to be inspected more frequently, and allow maintenance to be target to components reaching the end of their life-cycle [5], for example on power-lines [6].

Despite these benefits, operation of a drone in close proximity to a structure can be a challenging task due to complex environmental conditions [7] and potentially poor situation awareness of the remote pilot [8]. To reduce the mental workload of the pilot in these situations it is beneficial to give the vehicle its own, artificial, situation awareness [9]. An inspection drone which is aware of its own proximity to a structure is able to perform the challenging task of distance keeping without pilot input. This frees up the pilot to concentrate on the task of data collection, ensuring the correct images are captured, without having to focus on the safety of the vehicle. Such a semi-autonomous drone allows inspection tasks to be carried out in more challenging conditions and at a greater distance from the pilot than is possible with current systems.

This paper discusses the development of a semi-autonomous inspection drone capable of maintaining a fixed distance and relative heading to a structure of interest. A Model Based Design (MBD) framework is introduced which enables any candidate control system to be thoroughly tested numerically and in high fidelity simulation prior to any real world flights. The focus of this testing is to ensure the real world flights can be conducted safely in close proximity to a structure, without undue risk to the vehicle.

The next section introduces the inspection scenario being considered and Section III details the hardware and software architecture of the vehicle. Section IV discusses numerical coverage testing, the first stage in the MBD process. Section V details the high fidelity simulation testing, with an example of how this can uncover unsafe controller performance. Section VI discusses two levels of real world testing, first executing the control system remotely and second executing a compiled binary onboard the vehicle. Finally, Section VII draws some conclusions and outlines further work.

II. Scenario

A number of commercial drone operators in the UK regularly undertake structural inspections with their vehicles. When flying in close proximity to a structure it is not possible to rely on Global Navigation Satellite System (GNSS) for positioning due to its limited accuracy and susceptibility to occlusion and multipath errors [10]. Additionally, it is unlikely the vehicle will posses a high fidelity geofence [11] (corresponding to the physical outline of the structure) required for a GNSS system to assure adequate separation. Pilots must instead position the vehicle manually using an attitude stabilisation mode [12], requiring significant skill.

This piloting method introduces a significant delay between the vehicle experiencing a disturbance and corrective action being taken by the pilot. This delay imposes strict limits on the operating conditions for the vehicle so as to

Research in part supported by the EPSRC, grant numbers EP/L024942/1 and EP/J011843/1

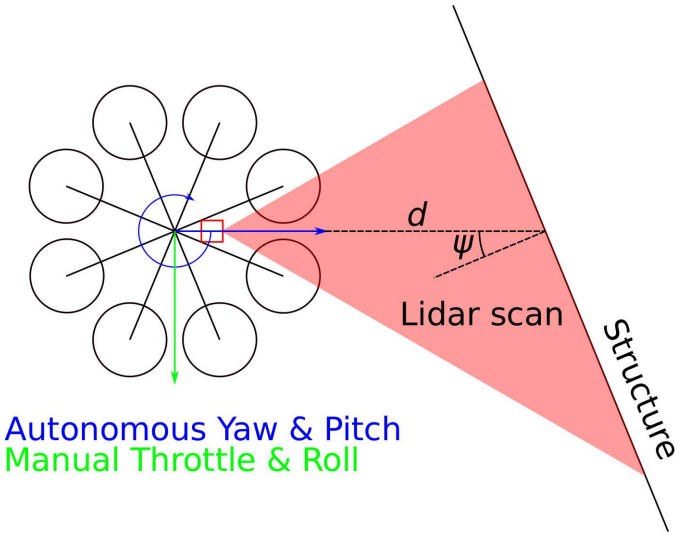

Fig. 1. Diagrammatic representation of the control task. d is controlled to a target value by altering the pitch angle of the vehicle. ϕ is controlled to zero by altering the yaw rate of the vehicle. The pilot retains control of throttle and roll angle

maintain safe separation from the structure at all times. The maximum wind gust which can be tolerated is primarily determined by the reaction time of the pilot which can be variable due to the high workload associated with the task. Additionally, the maximum distance from the pilot the vehicle can be operated is limited by the pilots ability to adequately judge the separation distance.

The semi-autonomous controller discussed in this paper is tasked with significantly reducing pilot workload during an inspection task by maintaining a fixed distance and relative heading to the structure at all times. The controller utilises a LiDAR scanner to detect the structure and takes control of the vehicles pitch and yaw axes away from the pilot, Fig. 1. Incorporation of such an assistive system may enable the operational limitations discussed previously to be relaxed, provided the performance of the system can be verified as safe.

III. System Architecture

The drone platform used in this study is an 8kg Octocopter airframe, equipped with a Pixhawk Flight Control System (FCS) running the ArduCopter firmware[1]. The autonomous control is deployed to an additional onboard computer, an Odroid XU3[2], running Linux and the Robot Operating System (ROS) [13]. Fig 2 illustrates a schematic view of the onboard systems, with the autonomous controller executing on the onboard processor. Fig. 3 shows the components on the actual vehicle.

The Radio Control (RC) input from the human pilot consists of eight channels corresponding to[3]

[1]http://copter.ardupilot.com/wiki/common-pixhawk-overview/

[2]http://www.hardkernel.com/main/products/prdt_info.php?g_code=G140448267127

[3]Items in italics are not relevant to this discussion

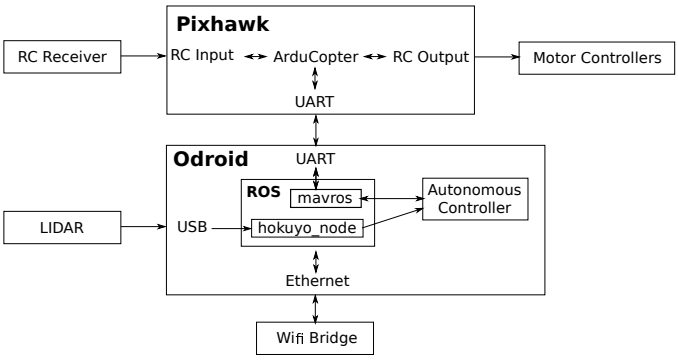

Fig. 2. Schematic view of the onboard systems architecture of the drone

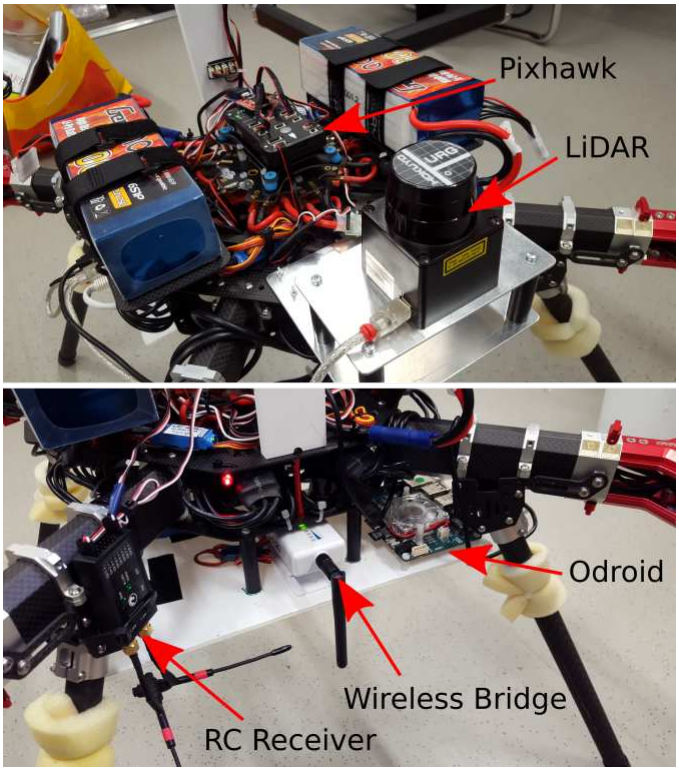

Fig. 3. Major system components onboard octocopter vehicle

1) Roll angle command
2) Pitch angle command
3) Throttle command
4) Yaw rate command
5) *Flight mode*
6) Autonomous control enable
7) *Auxiliary 1*
8) *Auxiliary 2*

Under manual control, the ArduCopter firmware uses the first four of these channels as commands for its attitude stabilisation system. The autonomous controller is developed in Simulink and utilises the Robotics System Toolbox to communicate with the autopilot via a combination of Universal Asynchronous Receiver/Transmitter (UART) connection

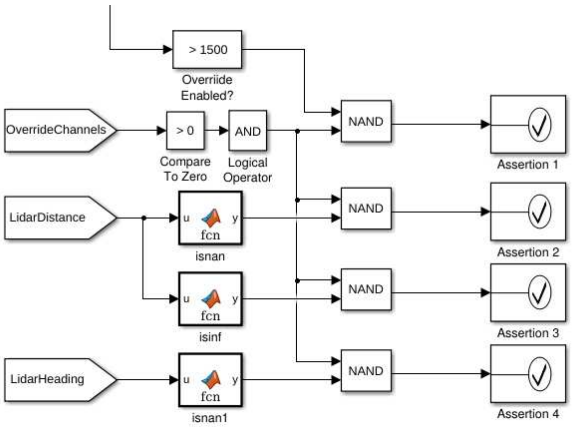

Fig. 4. Logical assertions ensuring controller only enabled when requested and able

and *mavros*[4] interface. When RC channel 6 indicates that autonomous control should be enabled, the controller then overrides RC input channels 2 and 4 with its control signals.

Due to the safety critical nature of the autonomous controller it is not desirable to progress straight to real world testing, as a failure of the system could quickly destroy the vehicle. Instead, a MBD approach is used to assure the performance of the system within a software based test environment. The following sections detail the increasing level of fidelity of test environments, up to a deployed executable running onboard the drone.

IV. Coverage Testing

Preliminary testing is focused on the discrete logic of the controller, ensuring that the control system is only active when requested and when it is receiving data of sufficient quality. The following assertions are included in the model

- Controller only enabled when RC channel 6 is high ($>$ 1500)
- Controller only enabled when structure is in range (LiDAR distance is not infinite)
- Controller only enabled when LiDAR distance is valid
- Controller only enabled when LiDAR heading is valid

These assertions can be seen within the Simulink model in Fig. 4.

The model is executed within a test harness which generates feasible, but otherwise random, input signals to replace those normally received from the LiDAR and FCS. These random signals are fed to the model, which is running as a desktop simulation, via a locally executed ROS environment. The use of ROS in this way means there are no modifications required to the model to run in this coverage test compared with running fully deployed on the real vehicle.

Should any of the above assertion fail during the test, the model will be terminated allowing the designer to inspect the inputs which lead to the failure and modify the system appropriately.

[4]http://wiki.ros.org/mavros

Model Hierarchy/Complexity		Test 1			
		D1		Execution	
1. Controller	13	100%	▬	100%	▬
2. . . . Distance Control	6	100%	▬	100%	▬
3. . . . . Check NaN	2	100%	▬	NA	
4. . . . . Compare To Constant		NA		100%	▬
5. . . . . Controller	4	100%	▬	100%	▬
6. . . . . Scaled to PWM		NA		100%	▬
7. . . . Relative Heading Control	6	100%	▬	100%	▬
8. . . . . Check NaN	2	100%	▬	NA	
9. . . . . Controller	4	100%	▬	100%	▬
10. . . . . Scaled to PWM		NA		100%	▬
11. . . . . PWM to Boolean		NA		100%	▬

Fig. 5. Discrete decision coverage during testing

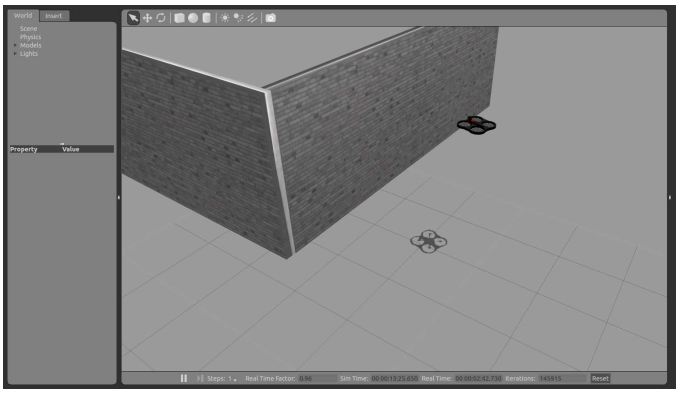

Fig. 6. AR.Drone 2.0 equipped with a laser scanner hovering near a wall in the gazebo simulator.

During coverage testing it is desirable to ensure that all possible discrete states of the model have been explored. For complex models this may not always be possible, but for this relatively simple example 100% coverage was achieved, as confirmed by a Simulink coverage report, Fig. 5. None of the assertions failed during this test, therefore it is safe to move on to a higher testing fidelity.

V. Simulation-Based Testing

Once the logical performance of the controller has been assured by coverage testing, an analysis of its physical performance is required. To facilitate this testing, an appropriate model of a multirotor has been selected, using the Technical University of Munich Simulator package [14]. A simulated AR.Drone 2.0 equipped with a laser scanner is shown in Fig. 6 hovering near a structure. This provides an adequate software in the loop simulator to begin the exploration of controllers before they can be deployed on real platforms.

When assessing the physical performance of a control system, many possible time response criteria may be used, such as rise time and overshoot [15]. In order to perform

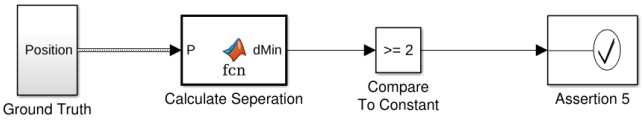

Fig. 7. Safety performance assertion ensuring separation to structure

such analysis, however, the system model must match the real system very closely. Due to the difficulties in accurately modelling the flight of a multirotor close to a structure, such as rotor wake interactions [16], [17], this approach was not suitable. Instead, a high level, safety based, performance assessment was conducted to determine whether the controller was safe to deploy to the real vehicle. The criteria for this assessment was captured in an additional runtime assertion

- Multirotor remains $> 2m$ from structure at all times

This assertion can be seen in Simulink in Fig. 7, utilising the ground truth position information available from the simulation environment.

During this test, the simulation environment was used to produce the ROS signals to fed the controller, once again requiring no modification from the final code. Fig. 8 shows the layout of the ROS Graph for the simulated AR.DRone 2.0 in Gazebo. The simulated quadrotor uses a standard setup which is platform agnostic, this means that the same controllers can be quickly ported to real aircraft once behaviour has been verified in simulation. The "octo1" and "InspectionControl" nodes are launched from within Simulink. A bespoke "FlyOnMavRos" node translates the standard RC controls to Twist commands within ROS, this can then be passed onto the AR.Drone 2.0 and flown in exactly the same manner as a physical platform. In order to maintain complete transparency between simulation and real-world the "FlyOnMavRos" node echoes back the commands it is sending to the AR.Drone 2.0 via a standard MavRos topic "/octo1/mavros/rc/in", with appropriate offsets to provide all information that would be available on the real platform.

Fig. 9 illustrates the results of a drone tasked with performing an exterior inspection task in simulation. It can be seen that at no time does the drone enter the safety zone, therefore the safety assertion never fails.

Fig. 10 illustrates the done conducting an interior inspection task. In this situation it can be seen that the drone does infringe the safety zone, triggering the assertion and terminating the simulation. The cause of this failure can be determined by inspecting the sensor system details in Fig. 1, which shows the LiDAR region of interest to be located in front of the drone, facing towards the object being inspected. Whilst traversing and inspecting the lower wall in Fig. 10, the drone is not actively sensing in its direction of movement, leading to the possibility of collision with the leftmost wall.

The failure of the control system in this situation illustrates the utility of conducting this degree of simulation testing prior to real world deployment. Identifying this failure mode of the system during real world testing would likely result

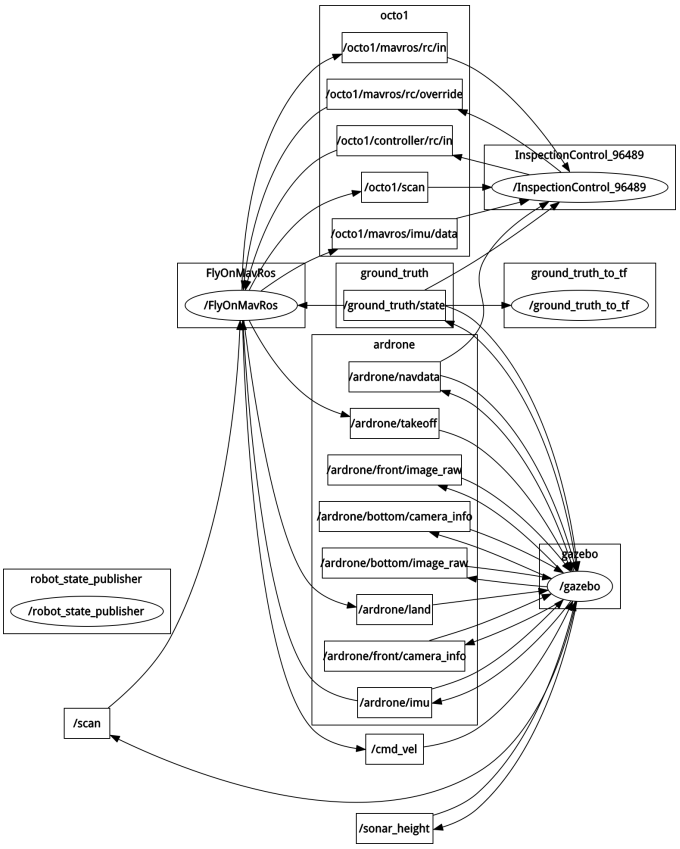

Fig. 8. ROS Graph showing node layout and topic connection for simulated scanning in the gazebo simulator.

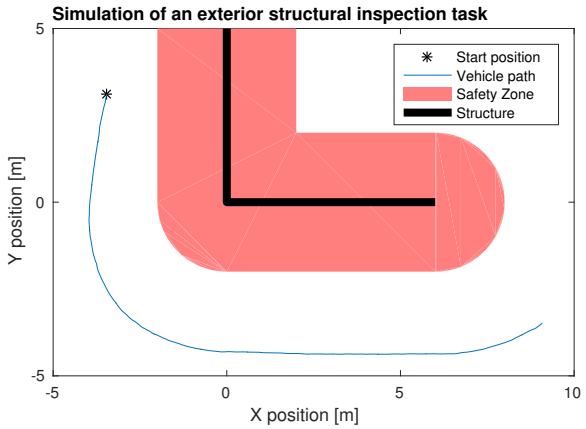

Fig. 9. Simulation of drone performing an exterior inspection task without infringing the safety zone

in a collision between the drone and a structure, causing significant damage to both and delaying further development. Finding such a failure in simulation, however, is a relatively quick process and the system can then either be redesigned to cope with this condition, or operational limitations placed on it to prevent this condition being encountered.

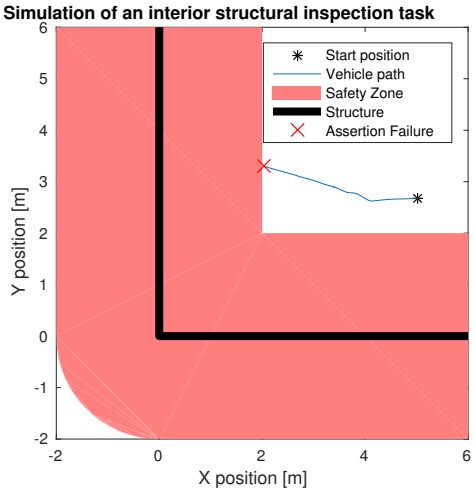

Fig. 10. Simulation of drone performing an interior inspection task and infringing the safety zone

Fig. 11. 8kg Octocopter performing semi-autonomous indoor inspection task

VI. Real world testing

A. Preliminary offboard testing

Once simulation based verification has been completed, fully defining the safe operating conditions for the system, it can be progressed to real world testing. By utilising the distributed nature of ROS, it is possible to first execute system on a ground based computer, transferring data to and from the drone via the wireless bridge shown in Fig. 2. Executing the system in this way allows for straight forward monitoring and parameter tuning at runtime, providing an increased level of confidence in system performance and shorter development times.

The only modification made to the system to support real world testing is to disable model termination on the failure of an assertion, in favour of a runtime warning. This ensures the pilot is always able to disable the RC override before terminating the model. The separation assertion discussed in the previous section is left in the model to minimise the modifications needed, however it is not active because no ground truth measurement is available during real world testing.

Fig. 11 illustrates the aircraft detailed in Section III conducting preliminary flight trials. By operating within the safety limitations identified during simulation based testing, the distance and heading controllers can now be tuned to achieve the desired time response performance.

B. Deployment in ROS

After tuning the control systems with the model executing on an offboard computer, the final step is to deploy it as an executable running on the vehicles onboard computer. In previous Simulink versions this was a time consuming process, often requiring custom C code to be written to interface with hardware peripherals. The Robotics System Toolbox streamlines this entire process by using a conventional ROS build environment on the target computer to compile the model.

Once deployed as an executable, the final testing of the system can be conducted to ensure that the desired performance achieved during offboard testing is still achieved. By completing the various stages of simulation testing prior to deployment, there are no possible cases in which the control system can fail to operate safely.

VII. Conclusions

This paper has introduced a MBD framework for the development of a semi-autonomous control system for a structural inspection drone. The primary aim of this approach was to verify that the control system would perform safely during real world testing, by providing as much assurance as possible in simulation. The utility of MBD has been illustrated by its ability to detect a fundamental design flaw which (if encountered during real world testing) could have significantly damaged the vehicle. Work is ongoing to tune the real world control system within the bounds identified in simulation.

Further development of the control system discussed in this paper, to include higher degrees of autonomy, will place a greater burden on MBD testing. For example, for a complex decision making system it may not be possible to achieve 100% decision coverage during early testing. Additionally, application of MBD to autonomous drones conducting more complex tasks than simple structural inspection will require a significantly larger suite of simulation based test environments. Further work is needed to develop this framework to support the safety verification of these more complex systems.

It was noted in this paper that time response performance could not be adequately assessed during simulation due to the complexities of modelling multirotor vehicles in close proximity to structures. As drones are increasingly developed to operate in these environments it may become desirable to conduct more significant controller tuning in simulation as this provides a faster turnaround time than real world testing. Further work is required in the area of modelling and simulation of rotorcraft in close proximity to structures in order to support this development.

REFERENCES

[1] N. Hallermann and G. Morgenthal, "Visual inspection strategies for large bridges using unmanned aerial vehicles (uav)," in *7th International Conference on Bridge Maintenance, Safety and Management, IABMAS 2014, July 7, 2014-July 11*, 2014.

[2] F. Bonnin-Pascual, E. Garcia-Fidalgo, and A. Ortiz, "Semi-autonomous visual inspection of vessels assisted by an unmanned micro aerial vehicle," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 3955–3961.

[3] A. Ellenberg, A. Kontsos, F. Moon, and I. Bartoli, "Bridge related damage quantification using unmanned aerial vehicle imagery," *Structural Control and Health Monitoring*, 2016.

[4] "Slips & trips and falls from height in Great Britain, 2014," http://www.hse.gov.uk/statistics/causinj/slips-trips-and-falls.pdf, Health and Safety Executive, Tech. Rep., 2014, online; accessed 10-March-2016.

[5] D. Hughes, G. Dennis, J. Walker, and C. Williamson, "Condition based risk management (cbrm)enabling asset condition information to be central to corporate decision making," in *Engineering Asset Management*. Springer, 2006, pp. 1212–1217.

[6] J. Toth and A. Gilpin-Jackson, "Smart view for a smart grid - Unmanned aerial vehicles for transmission lines," in *Applied Robotics for the Power Industry (CARPI), 2010 1st International Conference on*. IEEE, 2010, pp. 1–6.

[7] C. A. Marinho, C. de Souza, T. Motomura, and A. G. da Silva, "In-service flare inspection by unmanned aerial vehicles (uavs)," in *18th World Conference on Nondestructive Testing. Durban, South Africa*, 2012.

[8] R. R. Murphy, K. S. Pratt, and J. L. Burke, "Crew roles and operational protocols for rotary-wing micro-uavs in close urban environments," in *Proceedings of the 3rd ACM/IEEE international conference on Human robot interaction*. ACM, 2008, pp. 73–80.

[9] O. McAree and W.-H. Chen, "Artificial situation awareness for increased autonomy of unmanned aerial systems in the terminal area," *Journal of Intelligent & Robotic Systems*, vol. 70, no. 1, pp. 545–555, 2012. [Online]. Available: http://dx.doi.org/10.1007/s10846-012-9738-x

[10] E. Kaplan and C. Hegarty, *Understanding GPS: principles and applications*. Artech house, 2005.

[11] J. T. Luxhoj, "System safety modeling of alternative geofencing configurations for small uas," *International Journal of Aviation, Aeronautics, and Aerospace*, vol. 3, no. 1, p. 2, 2016.

[12] A. Sámano, R. Castro, R. Lozano, and S. Salazar, "Modeling and stabilization of a multi-rotor helicopter," *Journal of Intelligent & Robotic Systems*, vol. 69, no. 1-4, pp. 161–169, 2013.

[13] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2, 2009, p. 5.

[14] J. Engel, J. Sturm, and D. Cremers, "Scale-aware navigation of a low-cost quadrocopter with a monocular camera," *Robotics and Autonomous Systems (RAS)*, vol. 62, no. 11, pp. 1646–1656, 2014.

[15] B. L. Stevens, F. L. Lewis, and E. N. Johnson, *Aircraft Control and Simulation: Dynamics, Controls Design, and Autonomous Systems*. John Wiley & Sons, 2015.

[16] C. Powers, D. Mellinger, A. Kushleyev, B. Kothmann, and V. Kumar, "Influence of aerodynamics and proximity effects in quadrotor flight," in *Experimental Robotics*. Springer, 2013, pp. 289–302.

[17] D. Lentink, A. F. Haselsteiner, and R. Ingersoll, "In vivo recording of aerodynamic force with an aerodynamic force platform: from drones to birds," *Journal of The Royal Society Interface*, vol. 12, no. 104, p. 20141283, 2015.