AIAA SciTech
4-8 January 2016, San Diego, California, USA
AIAA Modeling and Simulation Technologies Conference

AIAA 2016-1186

Simulation framework for incorporating sensor systems in UAS conceptual design

Kshitij Jerath*
Washington State University, Pullman, WA 99164, USA

Jack W. Langelaan†
The Pennsylvania State University, University Park, PA 16802, USA

Sensor systems typically have not played a significant role in the aircraft conceptual design process. However, with the proliferation of unmanned aerial systems (UAS), the role of sensor systems on vehicle conceptual design can no longer be ignored. This issue is accentuated at smaller scales, where the sensing equipment may place significant constraints on the size, weight and power requirements of the aircraft. In this paper, we present a novel method that incorporates sensor systems into vehicle conceptual design and mission capability analysis. The presented method relies on a simulation framework in which the Blender Game Engine is used to generate complex cluttered flight environments. The MORSE simulator is used to fly vehicles with varying sensor configurations in these cluttered environments in order to assess mission capabilities. Results show that this approach has the potential to assess mission capabilities as a function of sensor configurations and specifications.

I. Introduction

Designing a new aircraft is a challenging and an inherently multi-disciplinary process. Arriving at an appropriate conceptual design marks the first step of this process. As a result, there a significant benefits to be achieved by identifying a good conceptual design at the beginning of this undertaking. Traditionally, conceptual design studies have focused on aerodynamics, structures, propulsion, stability, control, and manufacturing,[1] and approach the problem through multidisciplinary optimization techniques,[2,3] such as genetic algorithms.[4] While these studies have significant appeal, they are largely targeted towards traditional disciplines and do not take into account the potential role of sensor systems in conceptual design. Recent conceptual design studies have shown some interest in non-traditional factors, such as aircraft emissions[5] and aircraft cost models.[6] However, other significant factors, such as the size, weight, and power requirements of on-board sensing equipment, have thus far not been a fundamental part of the conceptual design stage. It is only very recently that researchers have begun to direct their focus towards the effects of sensors on conceptual design.[7,8]

The effects of sensors and the associated data processing algorithms on vehicle design and mission capabilities are quite significant for unmanned aerial system (UAS) applications. Specifically, there exists a trade-off between the equipment required for high levels of automation, and the performance of the vehicle as captured by its size, weight and power requirements. The industry is currently witnessing trends that push for smaller UAS in both civilian and military applications to enhance mobility and ease of deployment. As the scale of the vehicle decreases, the sensing equipment may scale disproportionately and actually begin to drive size and performance metrics. There exists a significant body of work on sensor systems required for autonomous operations in cluttered environments,[9] as well as on high-order modeling of vehicle dynamics.[10] However, conceptual design requires fast design cycles, so a method to incorporate sensor systems into vehicle conceptual design and mission performance is required.

*Assistant Professor, School of Mechanical and Materials Eng., Washington State University, kshitij.jerath@wsu.edu
†Associate Professor, Department of Aerospace Engineering, Associate Fellow, jlangelaan@psu.edu

In this paper, we present a methodology to take into account the effects of sensor systems and their specifications on mission performance at the conceptual design stage itself. The UAS mission is undertaken in a virtual 3D environment which is modeled and rendered using the Blender Game Engine (BGE). The simulation itself is performed using the Modular OpenRobots Simulation Engine (MORSE) simulator. Sensor systems with various specifications and configurations can be tested via flight simulations through a cluttered environment such as a forest. Mission performance metrics can then be compared for the various sensor system configurations to arrive at the appropriate conceptual design. The remainder of this paper is organized as follows. Section II discusses the Blender Game Engine and MORSE frameworks used to perform the included work. Section III discusses the simulation framework itself, including navigation and sensor perception algorithms. Section IV discusses the simulation results and mission performance metrics for a waypoint navigation mission Section V provides some concluding remarks and briefly discusses the potential of this approach for improving conceptual design of small-scale unmanned aerial systems.

II. System Setup

This section provides details regarding the system setup used to perform simulations with the goal to incorporate sensor specifications into the conceptual design process of small UAS. The scope of the simulations has been restricted to small UAS with gross take-off weight under 1,320 lbs., where such analyses are most likely to produce actionable results, but in principle the approach is applicable to larger UAS as well. The following subsections discuss the system requirements and simulation framework required to perform the included work.

A. Minimum system requirements

The minimum system requirements to load and run the simulation environment include an Intel i5 (or equivalent) processing unit, 4 GB RAM, and a graphics card that supports GLSL (OpenGL Shading Language) shading. The MORSE simulator is currently only supported for Linux operating systems. For the simulations included in this work, an 8-core Intel Core i7-4770 CPU, with 8 GB RAM and a GeForce GT 620/PCIe/SSE2 graphics card was used, with a 64-bit installation of Ubuntu 14.04 LTS.

B. Simulation framework

The simulation framework consists of three distinct components: the Blender Game Engine (BGE), the Modular OpenRobots Simulation Engine (MORSE) simulator, and the Robot Operating System (ROS) middleware. The *Blender Game Engine* (BGE) provides the tools to efficiently render the robot and environment features. BGE is part of an open-source 3D graphics tool called Blender that enjoys wide popularity and has a vibrant community of developers. Blender is available on the three major operating systems, Microsoft Windows, Mac OS X, and Linux. As will be seen later in this text, the Blender Game Engine enables the development of near photo-realistic environments for the UAS to fly through. Photo-realism is one of the key features of the presented simulation framework, and an important requirement for simulating unmanned aerial vehicles whose guidance and navigation systems may rely heavily on vision-based sensors. In addition to photo-realism, BGE provides features such as collision detection and physics-based simulations, as well as a Python-scripting API for customized control of the simulation environment. Specifically, Blender's Python API allows for programmatic generation of user-defined environments, a feature that is used in the presented work to generate cluttered forest environments. The physics simulation engine that is part of BGE is utilized by MORSE to run realistic simulations in virtual environments.

The *Modular OpenRobots Simulation Engine* (MORSE) has been in development since 2011, and has seen a recent increase in popularity with other developers, such as those of the Open Motion Planning Library (OMPL), providing integrations with MORSE.[11] The MORSE simulator can be controlled directly through the command line. MORSE contains in-built actuator, sensor, and robot modules that can be combined to simulate a robot in a virtual environment. Additionally, MORSE allows for limited simulation of multi-robot systems if vision systems such as cameras are not simulated. MORSE is open-source and exceptionally easy to customize. The robot model and flight actuator used in the included work are customized versions not available in the current MORSE repository. MORSE also supports four different middleware, viz. ROS, YARP, Picolibs, and MOOS, which enables researchers to re-use existing codebases for performing simulations.

In this work, the *Robot Operating System* (ROS) middleware was used, though any of the other middleware supported by MORSE could have been used as well. ROS is a mature platform whose first official release was in 2010, though it has been in development since 2007. ROS is currently supported by the Open Source Robotics Foundation (OSRF). The primary advantages of using ROS middleware in the current simulation framework is its increasing status as a de facto standard in the robotics community. ROS handles the communication between various sensing, actuation and computational modules, allowing developers to focus on algorithms rather than the intricacies of communication between them.

Figure 1 describes the relationship and dependencies between these three components of the simulation framework. Python scripts make use of the application programming interface (API) available in Blender to generate cluttered forest simulation environments. MORSE uses this simulation environment as well as BGE's in-built Bullet physics engine for flight simulation. Within this simulation environment, the ROS middleware provides a means for communication between the various robot sensors and actuators through ROS topics. The user may choose to define his or her own ROS topics or choose to use the topics provided as part of MORSE's library of sensors and actuators. The ROS middleware enables researchers to use existing Python codebase for functions such as sensor perception, guidance, navigation and control. The next section discusses how these three distinct components can be used together to develop a simulation framework for incorporating sensor systems into the conceptual design of unmanned aerial systems.

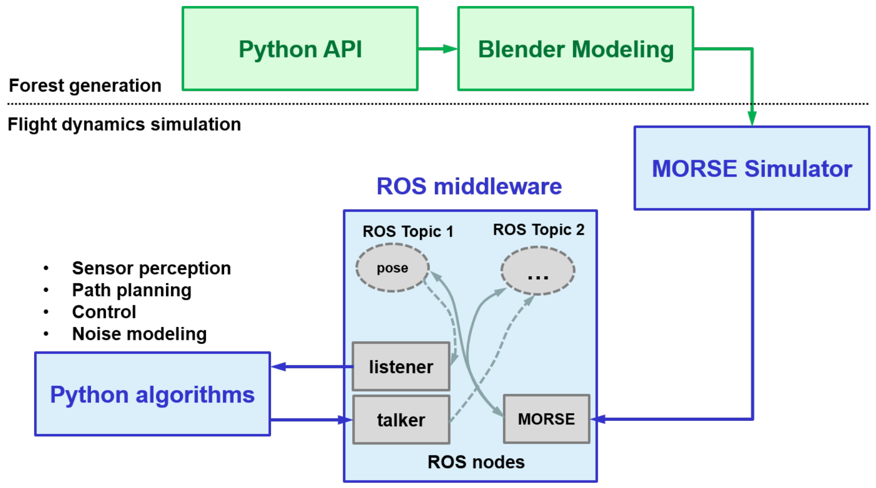

Figure 1. Relationship between Blender Game Engine, MORSE simulator, and ROS middleware.

III. Simulation Setup

This section discusses the method for generating the simulation environment using BGE, low-order kinematic models to simulate the UAS, waypoint navigation, and low-order power usage modeling for a given mission profile. The simulation setup establishes a framework for testing mission completion and success as a function of the capabilities of on-board sensor systems.

A. Generation of cluttered environments

In the included work, forests are taken as prototypical examples of cluttered environments. The simulation of autonomous flight through such cluttered environments requires virtual models that faithfully mimic the structure of real forests. Fortunately, the study of the distribution of trees in forests has been an area of significant research for several decades. Important works in this field, such as those of Hubbell[12] and Lieberman,[13] removed several misconceptions through accurate measurements of tree locations in forests. Other works, such as those by Keuls et al.,[14] have sought to establish a stochastic representation of the spatial distribution of individual elements. Together, these works have helped create a healthy body of literature that can be leveraged for creating virtual forests, as exhibited by the works of Moeur[15] and Karaman.[16]

According to prevalent methods, trees in a forest may be modeled as a Poisson process, which is essentially a counting process.[17] Thus, for dimension $d = 2$, let $N(A_i)$ be a random variable representing the number of trees in a region $A_i \subseteq \mathbb{R}^d$. Then the collection of these random variables $\{N(A_i) : A_i \subseteq \mathbb{R}^2, i = 1, 2, 3, ...\}$ is a Poisson process defined on $\mathbb{R}^2$ and a valid model for spatial distribution of trees in a forest. This Poisson process is characterized by a density parameter λ, which represents the density of trees in a forest. It is observed that typical values of tree density in a forest range from 0.00001 to 0.01 trees/m^2, depending on the tree species under consideration.[13] For the purposes of this work, the tree density is assumed to be 0.001 trees/m^2.

The simulated region A of the forest can be divided into disjoint cells of size $d_x \times d_y$, where d_x and d_y represent the cell dimensions in meters in the x and y directions, respectively. In order to simulate a Poisson forest, trees are placed in the region A according to the following rules:

- $N(A_i \to 0) = 0$, and
- $N(A_i)$ and $N(A_j)$ are uncorrelated when $A_i \bigcap A_j = \phi$, $\forall\, i \neq j$.

Now, these requirements of the Poisson process are approximately adhered to by restricting the cell dimensions to $d_x = 0.5$ m and $d_y = 0.5$ m. Further, each cell is populated with trees independent of other cells. The probability of a tree being present in a specific cell of known area is given according to the following distribution:

$$P(N(|A|) = k) = \frac{e^{\lambda|A|}(\lambda|A|)^k}{k!}, \tag{1}$$

where $k \in \mathbb{Z}^+$ represents the number of trees in the cell. With the simulation environment in place, the next step for developing the simulation framework is to generate the low-order kinematic models for flight simulation, which is discussed in the next subsection.

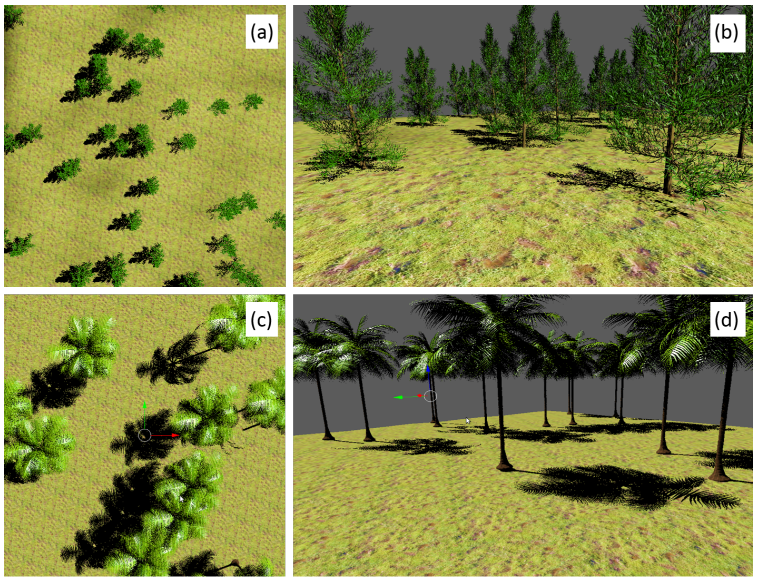

Figure 2. Simulation environments generated algorithmically using Blender's Python API. (a) Top view of Poisson forest with conifer trees, (b) First person view of Poisson forest with conifer trees, (c) Top view of Poisson forest with palm trees, (d) First person view of Poisson forest with palm trees

B. Low-order UAS system model and waypoint navigation

The unmanned aerial system is currently simulated using low-order kinematics, though higher-order dynamical models can easily be substituted into the plant description. The system state variables used to describe the vehicle kinematics include position coordinates (x, y, z) in the Earth-fixed reference frame, and the vehicle yaw (ψ) in body-fixed reference frame, whereas the bank angle (ϕ), the flight path angle (γ), and the true airspeed (v) serve as control inputs to the system. The low-order kinematic equations used in the simulations are:

$$\dot{x} = v\,\cos\psi\,\cos\gamma \tag{2}$$
$$\dot{y} = v\,\sin\psi\,\cos\gamma \tag{3}$$
$$\dot{z} = v\,\sin\gamma \tag{4}$$
$$\dot{\psi} = \frac{g\,\tan\phi}{v} \tag{5}$$

In addition, on-board energy (E) may be a another variable may be used to determine the state of the UAS as follows:

$$\dot{E} = -P \tag{6}$$

where P denotes the on-board power usage profile. Since, the eventual goal of the study is to create a simulation framework to assist in the conceptual design of small UAS, it is essential to develop models for studying the power consumption in these systems as a function of sensor capabilities. Moreover, varying sensor suites that have different power requirements may impact UAS mission success. Low-order power usage models are developed using thrust and sensor suite power consumption as follows:

$$P = \quad v\,T + P_{sensor} \tag{7}$$

where,

$$T = \quad D + mg\,\sin\gamma \tag{8}$$
$$D = \quad \frac{1}{2}\rho v^2 S C_D \tag{9}$$
$$C_D = \quad C_{D0} + kC_L^2, \tag{10}$$
$$C_L = \quad \frac{2mg}{\rho v^2 S\cos\phi}, \quad \text{and } k = \frac{1}{\pi A R e_0} \tag{11}$$

where, T represents the thrust, D represents drag, mg denotes weight of the UAS, ρ represents atmospheric density, v denotes airspeed, S denotes wing span area, C_D denotes drag coefficient, C_{D0} represents parasitic drag, C_L denotes lift coefficient, AR denotes wing aspect ratio, and e_0 denotes Oswald efficiency. P_{sensor} denotes the power consumption by the sensor suite and is a function of the individual sensors on-board the UAS.

The low-order aircraft models can be used in the navigation, guidance and control operations of the aircraft. Specifically, the navigation and guidance functions may be implemented through path planning algorithms, with waypoint navigation implemented at a global scale and obstacle avoidance implemented at a local scale. Waypoint navigation is currently implemented with a PID controller with saturation of the roll angle (ϕ) actuation input. More sophisticated guidance, navigation and control algorithms may be used, but they are not necessary to demonstrate the novel contributions of this work.

C. Advantages and limitations of the simulation framework

One of the key advantages of the presented simulation framework is the tight integration between the game engine and the environment renderer, with both being part of the Blender open-source development effort. Together with the Python API that allows researchers to write scripts for algorithmically generating simulation environments, these aspects of the Blender Game Engine tool provide a powerful method for creating simulation environments. In addition, while Blender has a significant learning curve, it also has an excellent developer community that researchers can turn to for technical support.

Due to its relatively recent appearance on the robotics simulation development scene, one of the drawbacks of the MORSE is its relatively limited documentation as compared to other platforms such as ROS. However, MORSE appears to have a fast update cycle with updates for the simulator appearing approximately every six months, and the documentation is also being updated frequently. In addition, the relative ease with which new robots and actuators can be added to the existing MORSE codebase makes it an attractive platform for robotics development and testing of unmanned aerial systems.

IV. Simulation results

The previous sections describe the development of the simulation framework using Blender to generate cluttered simulation environments, MORSE for running the simulation, and ROS for the communication framework between individual sensors, actuators and other simulation entities. In this section, we discuss some of the potential uses of the simulation framework in assessing the conceptual design of a UAS which may have a wide variety of on-board sensors. For example, Figure 3 shows that custom aircraft can be designed to fly user-defined missions in virtual scenarios representative of real-world environments. The virtual environment includes real-world physics, such as gravity, enabled by the Bullet physics engine. This framework can also be advanced to include wind forces, but this has not yet been activated in our simulation framework.

Figure 4 shows the waypoint navigation capabilities developed as part of the simulation framework. One can notice that the UAS is able to navigate to all waypoints and return to the origin, while mapping its environment. The blue dots in Figure 4 indicate the raw data observed by an on-board LIDAR sensor, representing the positions of the trees in the forest. The map generated by the UAS will change with variations in the capabilities of the on-board sensor. Specifically, LIDARs with lower specifications (e.g. lower angular resolution) will produce world maps with lower information content. This is evident in the two maps generated by sensor suites differing in their specifications. The map shown in Figure 4(a) has been generated by an on-board LIDAR with an angular resolution of 1 degree, whereas the map in Figure 4(b) has been generated by a LIDAR with an angular resolution of 5 degrees. It is evident that the map on the left contains more information and has the potential to be more useful for obstacle avoidance. The ability to virtually install sensor suites with varying capabilities has implications in the conceptual design of new aircraft, especially small-sized UAS. For example, Figure 5 shows the power usage of the waypoint navigation mission depicted in Figure 4(b). If the sensor suite had inferior specifications resulting in an uncertain world view, it may be expected that the UAS will execute additional maneuvers to avoid obstacles that its assumes are present in its world view. These additional maneuvers could lead to increased power demands which may limit the aircraft's ability to complete its mission. Alternatively, a UAS equipped with a high-performance, power-hungry sensor suite may utilize its energy reserves too quickly and may also be unable to complete its mission. These results can help guide the conceptual design of UAS and also guide UAS reconfiguration to better fit the desired mission profile.

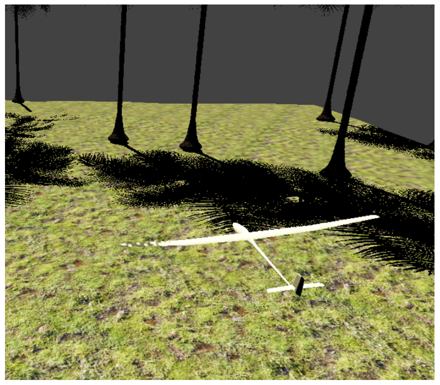

Figure 3. Custom aircraft model navigating a Poisson forest environment.

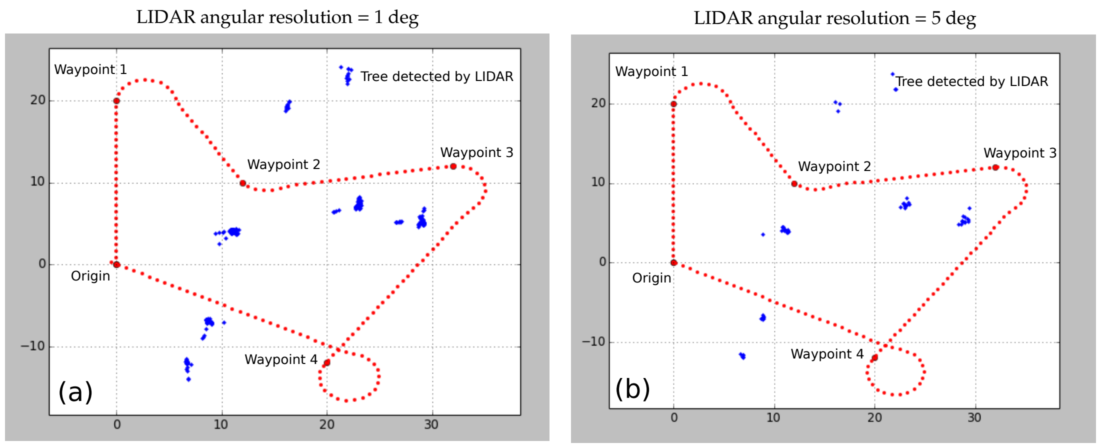

Figure 4. Waypoint navigation using PID controller with roll angle actuator saturation. Blue dots on the map indicate raw LIDAR sensor data depicting obstacles (trees) in the simulation environment. Maps generated with virtual on-board LIDAR with angular resolution of (a) 1 degree, (b) 5 degrees. Map dimensions are in meters.

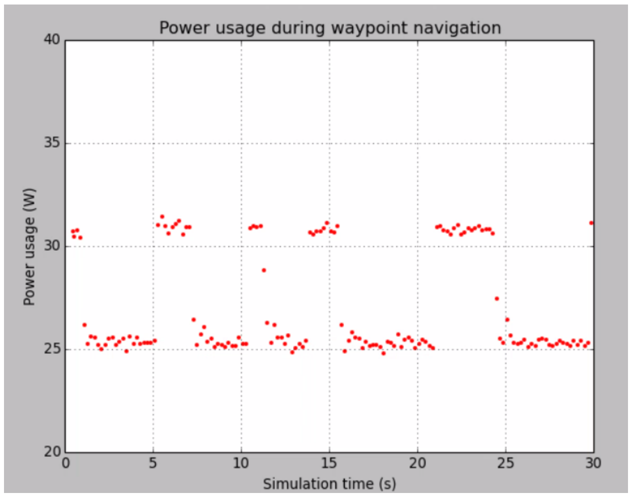

Figure 5. Power consumption profile for waypoint navigation mission shown in Figure 4. Power consumption peaks during banking maneuvers. Sensor power consumption is based on representative commercially available sensors including inertial sensors, LIDARs and cameras.

V. Concluding Remarks

The primary objective of this work was to demonstrate the use of Blender Game Engine and MORSE simulator, along with ROS middleware, to successfully act as a means for conceptual design of unmanned aerial systems while incorporating sensor specifications. The novel contribution of this work is in terms of assessing the impact of sensor system capabilities on waypoint navigation mission performance via virtual flight in near photo-realistic simulation environments. Current work is directed towards running Monte Carlo simulations to identify the effect of various sensor configurations and mission profiles on mission endurance. Results included in this work show that the BGE-MORSE-ROS framework could be used to aid conceptual design of unmanned aerial systems.

Acknowledgements

This project was sponsored through the Vertical Lift Research Center of Excellence (VLRCoE) at The Pennsylvania State University.

References

[1]Fielding, J., *Introduction to Aircraft Design*, Cambridge Aerospace Series, Cambridge University Press, 1999.

[2]Perez, R. E., Liu, H. H., and Behdinan, K., "Evaluation of multidisciplinary optimization approaches for aircraft conceptual design," *AIAA/ISSMO multidisciplinary analysis and optimization conference, Albany, NY*, 2004.

[3]Neufeld, D., Chung, J., and Behdinan, K., "An Approach to Multi-Objective Aircraft Design," *Future Application and Middleware Technology on e-Science*, edited by O.-H. Byeon, J. H. Kwon, T. Dunning, K. W. Cho, and A. Savoy-Navarro, Springer US, 2010, pp. 103–112.

[4]Ng, T. and Leng, G., "Application of genetic algorithms to conceptual design of a micro-air vehicle," *Engineering Applications of Artificial Intelligence*, Vol. 15, No. 5, 2002, pp. 439 – 445.

[5]Antoine, N. E. and Kroo, I. M., "Framework for aircraft conceptual design and environmental performance studies," *AIAA journal*, Vol. 43, No. 10, 2005, pp. 2100–2109.

[6]Curran, R., Price, M., Raghunathan, S., Benard, E., Crosby, S., Castagne, S., and Mawhinney, P., "Integrating aircraft cost modeling into conceptual design," *Concurrent Engineering*, Vol. 13, No. 4, 2005, pp. 321–330.

[7]Goerzen, C. and Whalley, M., "Sensor requirements for autonomous flight," *Proceedings of the 2012 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2012.

[8]Dittrich, J. S., Adolf, F.-M., and Schopferer, S., "Impact of Obstacle Sensor Choice on Preliminary Unmanned Helicopter Design and Mission Performance in Obstacle Fields," *AHS International 71st Annual Forum Proceedings*, May 2015.

[9]Thrun, S., Burgard, W., and Fox, D., *Probabilistic Robotics*, Intelligent robotics and autonomous agents, MIT Press, 2005.

[10]Etkin, B. and Reid, L., *Dynamics of Flight: Stability and Control*, Wiley, 1995.

[11]Şucan, I. A., Moll, M., and Kavraki, L. E., "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, Vol. 19, No. 4, December 2012, pp. 72–82.

[12]Hubbell, S. P., "Tree dispersion, abundance, and diversity in a tropical dry forest." *Science (New York, N.Y.)*, Vol. 203, No. 4387, Mar 1979, pp. 1299–1309.

[13]Lieberman, M. and Lieberman, D., "Patterns of density and dispersion of forest trees," *La Selva: ecology and natural history of a neotropical rain forest. University of Chicago Press, Chicago, Illinois, USA*, 1994, pp. 106–119.

[14]Keuls, M., Over, H. J., and Wit, C. T., "The distance method for estimating densities," *Statistica Neerlandica*, Vol. 17, No. 1, Mar 1963, pp. 71–91.

[15]Moeur, M., "Characterizing Spatial Patterns of Trees Using Stem-Mapped Data," *Forest Science*, Vol. 39, No. 4, Nov 1993, pp. 756–775.

[16]Karaman, S. and Frazzoli, E., "High-speed flight in an ergodic forest," *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, May 2012, pp. 2899–2906.

[17]Papoulis, A. and Pillai, S., *Probability, random variables, and stochastic processes*, McGraw-Hill electrical and electronic engineering series, McGraw-Hill, 2002.