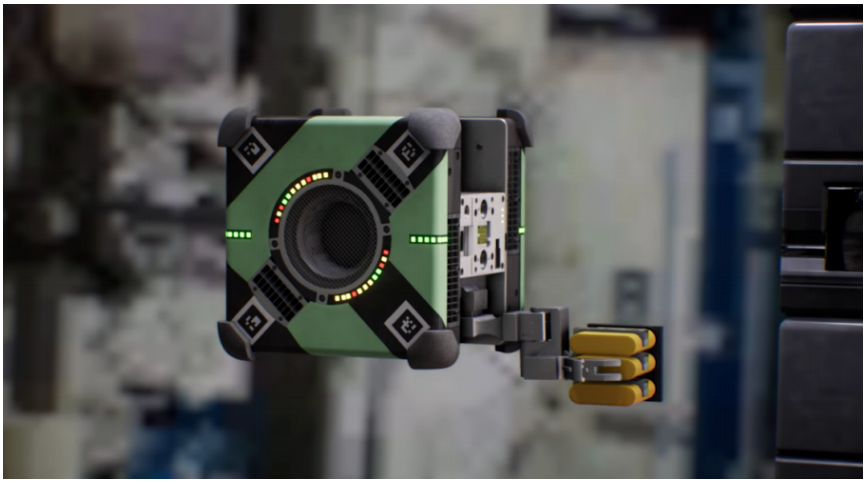
Multivariable Adaptive Robust Control for a Free-Flying Space Robot

Tammer H. Barkouki, Jason Dekarske, Abhinav G. Kamath



April 29, 2020

Contents

1 Introduction **2**

2 Literature Review **3**
2.1 Astrobee Design . . . 3
2.1.1 Design Overview . . . 3
2.1.2 Dynamics . . . 4
2.1.3 Control and Estimation . . . 4
2.1.4 Robotic Manipulator: Perching Arm . . . 5
2.2 Multibody Dynamics . . . 5
2.3 Multivariable Adaptive Robust Control . . . 5
2.4 Planning Algorithms . . . 6
2.4.1 Path Planning . . . 7
2.4.2 Trajectory Optimization . . . 8

3 Timeline **8**

4 Contributions **9**

5 References **9**

1 Introduction

Missions on the International Space Station require great attention to detail to ensure no small procedure is missed. Keeping track of tools and supplies, for example, have led to some creative ways to manage everything. Some astronauts may attach tools to their uniforms or to the walls with velcro. Some parts or tools may be left suspended in zero-g while the astronaut tends to a sub-task. Throughout a task, an astronaut may need verbal, visual, or physical assistance while cognitively fully-loaded. In this instance, in order to keep proper situational awareness, a robot assistant could provide an offload of their routine work [1]. Astrobee, the most recent robot to be added to the station, can autonomously navigate the modules and perform tasks with astronauts or alone. We propose the following scenario for astronauts where Astrobee can help.

> An astronaut is preparing for an extravehicular activity (EVA) inside the equipment airlock when they realize that they are required to use a wrench to resize a portion of the EVA mobility unit (EMU) that is in the Japanese Experiment Module (JEM). Astrobee is tasked to undock, locate and grab the tool and proceed to the astronaut's worksite for delivery. Astrobee then station-keeps at a short distance from the astronaut using a fiducial marker attached to the astronaut's sleeve, where it waits with the tool until the astronaut needs it. [1]

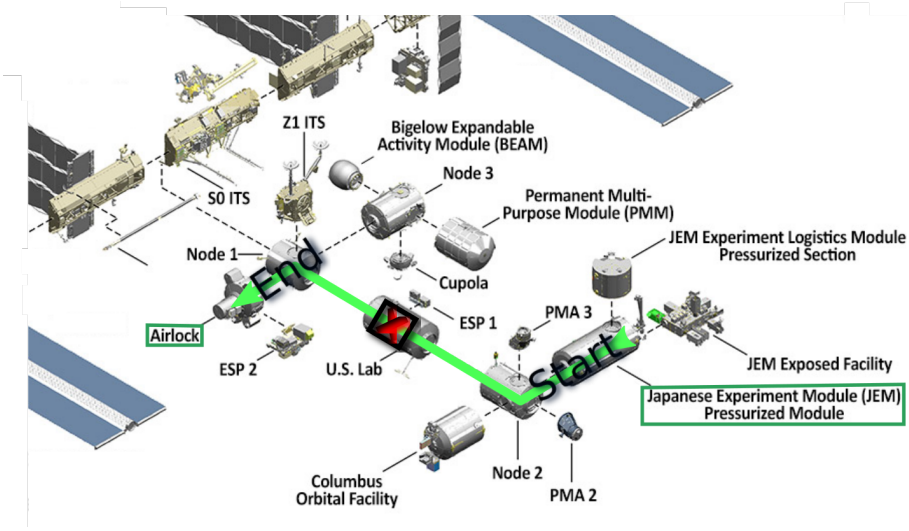


Figure 1: The proposed start and destination for the Astrobee navigation experiment [2].

With this use-case, we will demonstrate an improved adaptive, robust controller with path and trajectory planning for following a fiducial marker. All simulation will be done in ROS[2] using a public Astrobee repository[3]. Python will be used to model the multibody dynamics of the Astrobee system with the robotic manipulator unfurled and grasping a tool, as depicted in the title page figure. MATLAB/Simulink will be used to design the low-level control system.

[1] Scenario inspired by astronaut Steve Robinson's retelling of a common frustration aboard the ISS.
[2] https://www.ros.org/
[3] https://github.com/nasa/astrobee

2 Literature Review

2.1 Astrobee Design

2.1.1 Design Overview

Astrobee are a new generation of free-flying robots on the International Space Station (ISS) and is built upon previous free-flyers such as the Synchronized Position Hold, Engage, Reorient, Experimental Satellite (SPHERES) and Personal Satellite Assistant (PSA). Bualat et al. [3] provide a history of previous ISS free-flyers, their uses and interactions with crew, and some limitations and considerations required for operation in the conditions of the ISS. One main distinguishing feature of Astrobee is that it is self-contained using vision-based navigation and able to operate in any segment of the ISS without the need for external wall-mounted cameras or sensors, unlike SPHERES. Astrobee has two propulsion modules mounted on the sides. Each module contains an impeller that draws air in from the side to charge a plenum that feeds six louvred vent nozzles per module. Figure 4 depicts the locations of the nozzles. As there is no expendable propellant, Astrobee is able to move as long as it has charged batteries. Charging and high-speed data transfer are done via a docking station mounted on the wall of the ISS that Astrobee docks with autonomously. Astrobee robots are equipped with the following hardware:

- High-, mid-, and low-level processors
- Twelve exhaust nozzles arranged two on each face, providing holonomic motion
- A manipulator arm used for perching on ISS handrails
- Forward and aft flashlights
- Six external sensors used for navigation, localization, and recording video
- Touchscreen
- Status LEDS
- WiFi connectivity

Smith et al. [4] provide design details and specifications of Astrobee including sensors, pose estimation, navigation, and control. The primary sensors are the inertial measurement units (IMU), NavCam, DockCam, PerchCam (flash LIDAR depth sensor), and SpeedCam (sonar/optical flow sensor).

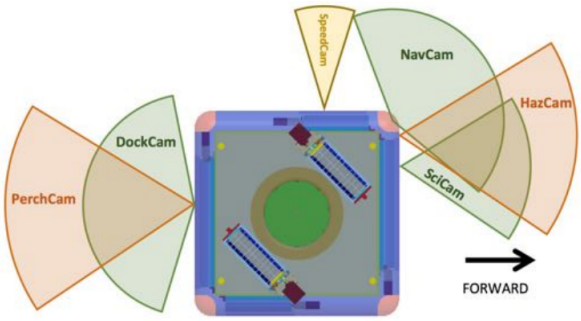


Figure 2: Location and FoV of Astrobee's six external sensors. [4]

Figure 2 depicts the locations and approximate fields-of-view (FoV) of Astrobee's six sensors. In the *general-purpose* mode, the NavCam and SpeedCam are used to match images with a prior map of features to provide absolute position. In the *fiducial-relative* mode, Astrobee matches AR targets detected by NavCam and DockCam with a map of the target locations to provide more accurate and robust position information during docking, as depicted in Figure 3. Additional details on the localization approach can be found in Coltin et al. [5]. For navigation and control, Astrobee either

follows stored trajectories sent by ground controllers or is commanded by guest scientists. From the stored trajectory, repair trajectories are calculated, and a PID controller calculates the required forces and torques. From there, nozzle servo positions are calculated to provide the required thrust and are sent to a propulsion module controller (running in the *low-level processor*), which sends commands to the individual servos.

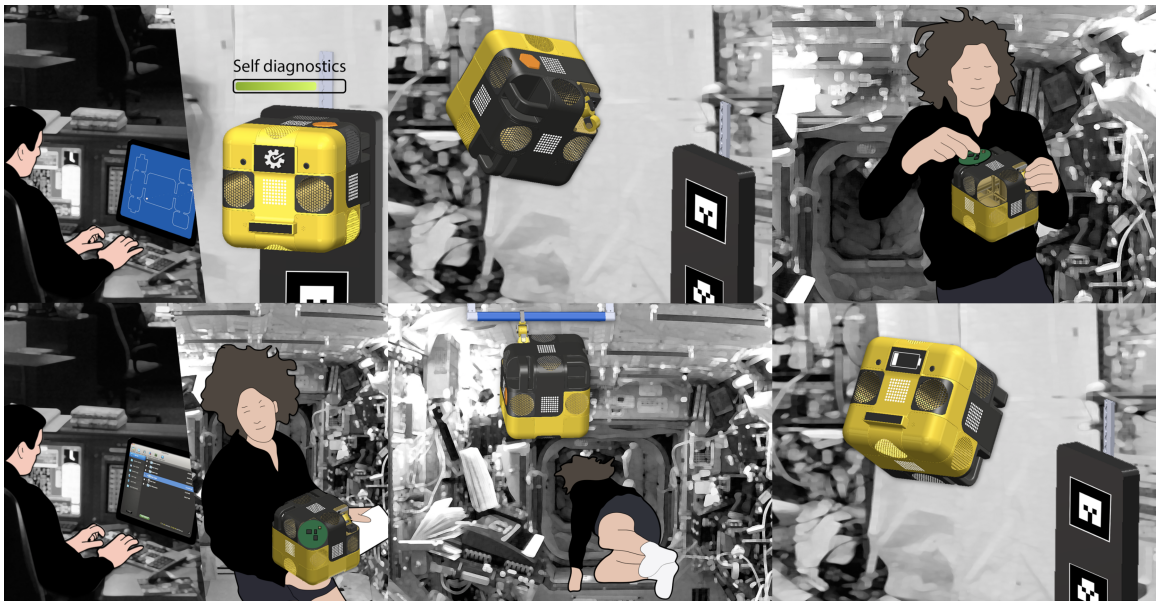


Figure 3: A scenario where Astrobee encounters fiducials while working with an astronaut [6].

2.1.2 Dynamics

Astrobee is a cubic robot with sides measuring about 30 cm across. Propulsion is provided by two fans located on the sides of the robot, and servo-actuated louvred vent nozzles, which allow for full six degree-of-freedom holonomic control. The twelve nozzles are arranged so that there are two on each face of the robot, and the air is directed to discharge perpendicular to the plane of the faces they are in, as depicted by Figure 4.

The dynamics of Astrobee can be determined by taking into account physical properties of the robot such as the location of the center of gravity, the impellers, vents, and the masses of any hosted payloads in the three payload bays of Astrobee. Watterson et al. [7] describe the system dynamics in detail, although the focus of their study was in developing a minimum-jerk smooth trajectory for obstacle avoidance while navigating through the ISS.

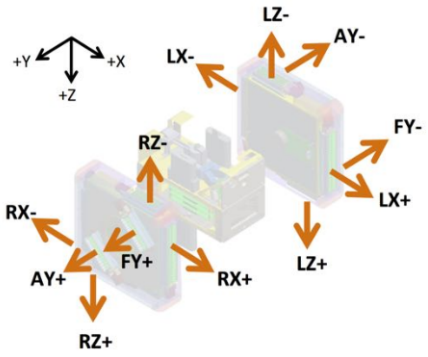


Figure 4: Exploded view of Astrobee showing names and locations of the nozzles. [4]

2.1.3 Control and Estimation

Astrobee uses a PID controller to compute the force and torque commands to correct the error between its current state (pose and velocity) and the output of the command shaper. The command shaper

computes a short-term repair trajectory that smoothly transitions from the current state to the target state, while taking deviations from the target trajectory into account and satisfying operating constraints in the process.

Astrobee uses an augmented-state Extended Kalman Filter (EKF) for state estimation [4]. The filter fuses information from Astrobee's cameras depending on the navigation modes mentioned earlier. This includes image matching, point clouds, IMU, or optical flow measurements.

2.1.4 Robotic Manipulator: Perching Arm

Park et al. [8] describe the development of the 3-DoF perching arm for Astrobee. The arm includes a 1-DoF gripper, and a proximal and distal joint that allows Astrobee to pan and tilt while the gripper "perches" Astrobee to a handrail. While the use of the perching arm has been documented as possibly supporting manipulator research on the ISS [3, 4], the authors of this proposal could find no literature on the use of the Astrobee perching arm for carrying equipment.

2.2 Multibody Dynamics

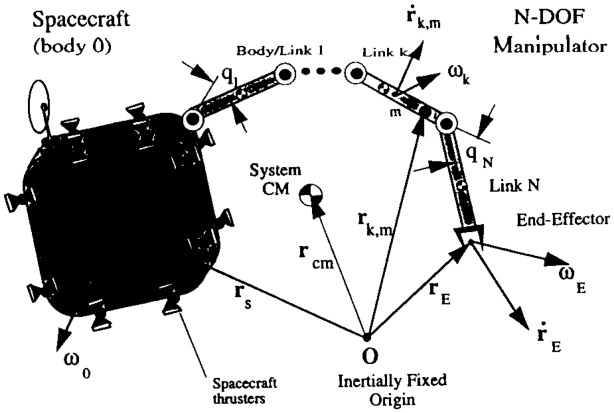


Figure 5: The generalized coordinates of a multibody free-flying space manipulator system [9]

The modeling of dynamical systems and robotic manipulators has been studied extensively in the literature. Classical methods such as the Newton-Euler method, Lagrangian formulations and Hamiltonian mechanics are popular, but can be cumbersome and computationally intensive for analytical dynamics modeling. Kane's method [10], on the other hand, is a highly systematic approach that becomes especially desirable in the multibody dynamics modeling of complex systems, such as the one depicted in Figure 5.

We will model Astrobee as a combination of cuboids for the main rigid body and simple linkages for the arm. The unknown tool will be modeled as a cylinder with dimensions and mass within reasonable bounds.

2.3 Multivariable Adaptive Robust Control

PID controllers have been widely studied and are very popular, given the ease of design and implementation. However, these controllers usually suffer from the lack of robustness to system parameter uncertainty and external disturbances. This can pose a significant danger in safety-critical environments such as the ISS (especially in the Cupola and Airlock) [6]. Astrobee's control must come with guaranteed performance in the name of safety and reliability, and hence, must rely

on more robust solutions for tasks involving grasping drifting objects with its perching arm, given the uncertainty in not only the properties of the object to be carried (mass, etc), but also the *way* in which the object is grasped. For example, a partial grasp of a tool could lead to uncertainty in the inertial properties of the combined system (moment of inertia, etc).

Many optimal control techniques such as the Linear Quadratic Regulator (LQR) and the Linear Quadratic Gaussian (LQG) have also been widely investigated in the literature for both single-input-single-output (SISO) and multiple-input-multiple-output (MIMO) systems. Although these techniques can be used for the control of multivariable processes, methods such as LQG have been proven to be non-optimal, while also lacking robustness to system parameter uncertainty [11]. This fact was exemplified by the failures of LQG controller implementations on two separate occasions in 1975: the controller on a Trident submarine caused it unexpectedly surface in a rough sea simulation. Although heuristic methods such as Loop Transfer Recovery (LTR) can increase the robustness of LQG controllers, they come at the price of severely degrading the original LQG cost-function, and thus lead to non-optimal solutions [12]. Methods such as nonlinear Model Predictive Control (MPC) are capable of handling complex nonlinear system dynamics, but they come at the cost of being computational intensive. This is certainly true for online implementations, given that the algorithms require a solution to an optimization problem at every sampling instant to obtain the necessary control commands [13].

Adaptive control is concerned with marked changes in a system plant. While knowledge about a system may not be exact, knowledge about how the system *changes* may be available. For example, an aircraft may perform well with a controller designed around its starting mass, but as it loses fuel, a new controller may be necessary to reach the required performance specifications. These controllers may be be fuzzy in that the controller changes depending on the system configuration [14]. In the UAV context, [15] has shown fuzzy logic for a path following controller. Alternatively, a range of linear controllers may be used to cover various operating points of a non-linear or time-varying plant. The linear controller is decided based on the system state and its closest operating point.

Given that there are strict safety-driven constraints on the achievable velocities and accelerations for free-flying robots inside the ISS, and that such robots usually travel at very low speeds without any sudden motion, linear control system design becomes increasingly desirable. Linear, constant-gain, multivariable robust controllers can be designed offline and deployed for real-time control of systems, with high efficiency and low computational intensity. These designs can be extended to adapt to changing system parameters, thus making them adaptive and robust in nature. Methods such as the well-known H_∞-Optimization and the less-explored Youla Parameterization techniques provide a procedural framework for the design of efficient and reliable multivariable control systems, with guaranteed performance and stability robustness characteristics. Youla Parameterization-based control design has been investigated and successfully implemented in the automotive industry, especially for robust observer and estimation design [16, 17]. H_∞-Optimization-based control has been successfully deployed on the Ariane 5 Evolution launch vehicle for the atmospheric flight phase, replacing the previously used LQG controller. In telecommunication satellites, H_∞-Optimization-based control has been shown to reduce the propellant mass consumption by 10% during station-keeping maneuvers [18]. Multivariable adaptive robust control design techniques can ensure safe and robust robotic operations on the ISS, by adapting to different system configurations (eg. perching arm folded vs. deployed), while remaining robust to system parameter uncertainty and external disturbances in every possible configuration.

2.4 Planning Algorithms

An autonomous vehicle requires a set of instructions on where to go and when. The areas of path and trajectory planning have gained considerable traction upon the proliferation of UAVs and commercial robotics in general. Path planning considers the line or space along which a robot is

required to travel while a trajectory includes information about the time in which it does so [19]. A higher-level controller can place constraints on the nature of how a robot moves along a path or trajectory. These position or timing constraints can be dependent on environmental geometry, such as obstacles or danger zones. In order to generate a given motion, the controller must provide the current state and the target state. Advanced trajectory-tracking controllers may also require knowledge about the vehicle dynamics to generate feasible plans.

2.4.1 Path Planning

In general, path planning is an easier task because there is no time requirement. Path-only convergence is accomplished with smaller control effort and less transient error [20]. With these advantages, obstacle avoidance may be more robust and effective, leading to a safer robot. Two simpler algorithms for path planning that are used on open-source flight controllers are the carrot-chasing algorithm [15] and the navigation vector field method [21]. These algorithms have the advantages of fast computation and broad applicability.

The carrot chasing algorithm uses a simple virtual path, sometimes constructed with waypoints, which is followed by a UAV by adjusting its heading toward a virtual point on the virtual path. The virtual point is a constant length from the vehicle to the path in the forward direction. By commanding this heading, the UAV will approach a tangent velocity to the virtual path. This method works well for straight line paths but shows decreasing performance for more complex shapes.

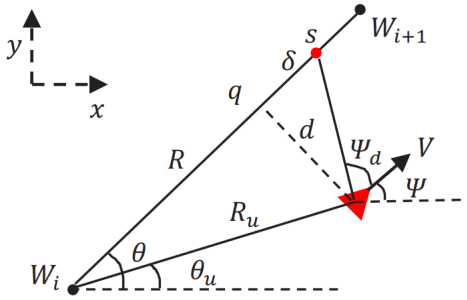


Figure 6: The vehicle is represented by the red triangle traveling with velocity V. The virtual path is piece-wise linear through the way-points, W_i. The algorithm generates a constant-length line path to the virtual point s and the heading commands a heading change proportional to Ψ_d [15]

Navigational vector fields are constructed by developing vector valued functions that depend on the current position of the robot and return the direction in which it should travel. This kind of navigation can be termed as an attractive field method. This uses the analogy of a charged particle near an attractive or repulsive field. The motion of the particle can be predicted with knowledge of the vector field generated by the environment. Extending this analogy to a UAV, obstacles or walls would generate repulsive fields, while a desirable path would generate an attractive field [22]. Inside the ISS, Astrobee would assign walls and important equipment repulsive fields and with superposition, the sum of these fields would generate the entire vector field. When Astrobee is placed in the environment, the vector valued function returns the commanded direction vector which is passed to the lower level controller. Difficulties with this method include generating these vector fields on the fly and local minima.

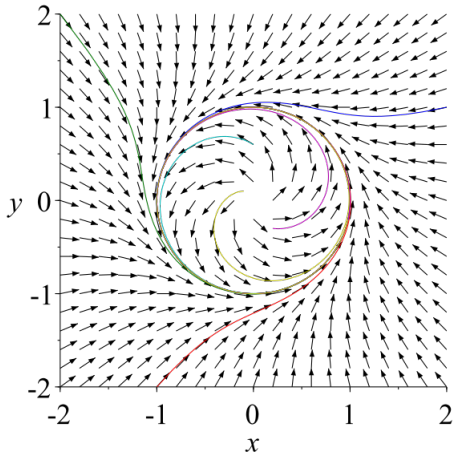


Figure 7: An example vector field for an orbital path around a central point. In this case, a circular vector field pushes a vehicle parallel to the vector at its current position [23].

2.4.2 Trajectory Optimization

Convex programming is very popular for trajectory optimization, given its feasibility for real-time implementation. [7] proposes a minimum-jerk smooth trajectory generation technique, which is capable of avoiding known obstacles, by using Semidefinite Programming (SDP) and Quadratic Programming (QP). More recently, the *Guaranteed Sequential Trajectory Optimization via Sequential Convex Programming* (GuSTO) algorithm, which uses Sequential Convex Programming (SCP), has been proven to outperform the state-of-the-art approaches in many aspects [24]. Such algorithms are computationally efficient and have the potential to be implemented in real-time on free-flying space robots like Astrobee.

3 Timeline

The following outlines the steps and milestones we will consider to complete this project. The major items indicate subprojects that will be worked on asynchronously and can be combined at any time. The deliverables are improved MIMO adaptive robust control designs using Youla Parameterization and H_∞-Optimization and a comparison of the robustness and performance of the developed controllers with that of the existing controller by varying the properties of the tools to be carried. Another deliverable required for testing performance is a path planning algorithm for marker following.

I. Record Trajectory Data

   a. Retrieve the pose estimates of Astrobee as a function of time

II. Marker Following

   a. Put a fiducial in the air lock for recognition (simulating a marker attached to an astronaut's sleeve)
      i. Acquire the marker via tele-operation
      ii. Acquire the marker via waypoint planning

b. Move a marker along a trajectory in the U.S. Lab (small movements or "noise" and larger movements going from one side of a module to another)
   i. Assign constant relative position to marker for waypoint following
   ii. Using attractive-repulsive fields, follow marker
   iii. Using GuSTO, follow marker

III. Controller Design

a. Change Astrobee model properties by adding a tool with unexpected weight and inertia to the arm → model the multibody dynamics using *Kane's Method*
   i. Fly same path as (II) and compare to built-in controller
b. Design Youla robust controller to account for added mass
c. Determine tool physical bounds for performance guarantee

4 Contributions

I. Tammer

a. Scenario development
b. Dynamics Modeling of Astrobee
c. Operating Astrobee simulator through various scenarios (baseline with arm stowed, arm unstowed, and arm gripping a tool)
d. Trajectory Optimization via Convex Programming (if time permits)

II. Jason

a. Reactive path planning
b. Astrobee model description variation
c. Multivariable Adaptive Robust Control Design: Youla Parameterization, H_∞-Optimization
d. Trajectory Optimization via Convex Programming (if time permits)

III. Abhi

a. Multibody Dynamics Modeling of Astrobee
b. Plant Linearization → State-Space Realization → MIMO Transfer-Function Modeling
c. Multivariable Adaptive Robust Control Design: Youla Parameterization, H_∞-Optimization
d. Robustness and Performance Analysis of the Closed-Loop System; Comparison
e. Trajectory Optimization via Convex Programming (if time permits)

5 References

[1] J. L. Broyan, M. K. Ewert, and P. W. Fink, "Logistics reduction technologies for exploration missions," in *AIAA SPACE 2014 Conference and Exposition*, 2014, p. 4334.

[2] NASA, "International space station facts and figures," 2019. [Online]. Available: https://www.nasa.gov/feature/facts-and-figures

[3] M. G. Bualat, T. Smith, E. E. Smith, T. Fong, and D. Wheeler, "Astrobee: A new tool for iss operations," in *2018 SpaceOps Conference*, 2018, p. 2517.

[4] T. Smith, J. Barlow, M. Bualat, T. Fong, C. Provencher, H. Sanchez, and E. Smith, "Astrobee: A new platform for free-flying robotics on the international space station," 2016.

[5] B. Coltin, J. Fusco, Z. Moratto, O. Alexandrov, and R. Nakamura, "Localization from visual landmarks on a free-flying robot," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 4377–4382.

[6] M. Bualat, J. Barlow, T. Fong, C. Provencher, and T. Smith, "Astrobee: Developing a free-flying robot for the international space station," in *AIAA SPACE 2015 Conference and Exposition*, 2015, p. 4643.

[7] M. Watterson, T. Smith, and V. Kumar, "Smooth trajectory generation on se (3) for a free flying space robot," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 5459–5466.

[8] I.-W. Park, T. Smith, H. Sanchez, S. W. Wong, P. Piacenza, and M. Ciocarlie, "Developing a 3-dof compliant perching arm for a free-flying robot on the international space station," in *2017 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*. IEEE, 2017, pp. 1135–1141.

[9] S. Dubowsky and E. Papadopoulos, "The kinematics, dynamics, and control of free-flying and free-floating space robotic systems," *IEEE Transactions on robotics and automation*, vol. 9, no. 5, pp. 531–543, 1993.

[10] T. R. Kane and D. A. Levinson, *Dynamics, theory and applications*. McGraw Hill, 1985.

[11] J. C. Doyle, "Guaranteed margins for lqg regulators," *IEEE Transactions on automatic Control*, vol. 23, no. 4, pp. 756–757, 1978.

[12] D. Noll, "A generalization of the linear quadratic gaussian loop transfer recovery procedure (lqg/ltr)."

[13] C. Liu, W.-H. Chen, and J. Andrews, "Tracking control of small-scale helicopters using explicit nonlinear mpc augmented with disturbance observers," *Control Engineering Practice*, vol. 20, no. 3, pp. 258–268, 2012.

[14] J. D. Boskovic and R. K. Mehra, "Stable multiple model adaptive flight control for accommodation of a large class of control effector failures," in *Proceedings of the 1999 American Control Conference (Cat. No. 99CH36251)*, vol. 3, 1999, pp. 1920–1924 vol.3.

[15] S. A. H. Tabatabaei, A. Yousefi-koma, M. Ayati, and S. S. Mohtasebi, "Three dimensional fuzzy carrot-chasing path following algorithm for fixed-wing vehicles," in *2015 3rd RSI International Conference on Robotics and Mechatronics (ICROM)*. IEEE, 2015, pp. 784–788.

[16] F. Assadian, A. K. Beckerman, and J. Velazquez Alcantar, "Estimation design using youla parametrization with automotive applications," *Journal of Dynamic Systems, Measurement, and Control*, vol. 140, no. 8, 2018.

[17] Z. Liu, F. Assadian, and K. Mallon, "Vehicle yaw rate and sideslip estimations: A comparative analysis of siso and mimo youla controller output observer, linear and nonlinear kalman filters, and kinematic computation," *Journal of Mechanical and Aerospace Engineering*, vol. 1, no. 1, 2019.

[18] C. Philippe, A. Annaswamy, G. Balas, J. Bals, S. Garg, A. Knoll, K. Krishnakumar, M. Maroni, R. Osterhuber, and Y. C. Yeh, "Apollo as the catalyst for control technology."

[19] B. Rubí, R. Pérez, and B. Morcego, "A survey of path following control strategies for uavs focused on quadrotors," *Journal of Intelligent & Robotic Systems*, pp. 1–25, 2019.

[20] D. Cabecinhas, R. Cunha, and C. Silvestre, "Rotorcraft path following control for extended flight envelope coverage," in *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*. IEEE, 2009, pp. 3460–3465.

[21] H. G. De Marina, Y. A. Kapitanyuk, M. Bronz, G. Hattenberger, and M. Cao, "Guidance algorithm for smooth trajectory tracking of a fixed wing uav flying in wind flows," in *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2017, pp. 5740–5745.

[22] Y.-b. Chen, G.-c. Luo, Y.-s. Mei, J.-q. Yu, and X.-l. Su, "Uav path planning using artificial potential field method updated by optimal control theory," *International Journal of Systems Science*, vol. 47, no. 6, pp. 1407–1420, 2016.

[23] Yapparina, "File:vector field and trajectories of a simple limit cycle.svg," 2018. [Online]. Available: https://commons.wikimedia.org/wiki/File:Vector_field_and_trajectories_of_a_simple_limit_cycle.svg

[24] R. Bonalli, A. Cauligi, A. Bylard, and M. Pavone, "Gusto: Guaranteed sequential trajectory optimization via sequential convex programming," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 6741–6747.