**SSC14 VII-7**

# An Autonomous Proximity Navigation Testbed for Nanosatellites

Subhransu Mishra, Marin Kobilarov
Johns Hopkins University
3400 N Charles St, 223 Latrobe Hall, Baltimore, MD, 21218; (443) 693 7496
smishra9@jhu.edu, marin@jhu.edu

## ABSTRACT

The goal of this work is to develop techniques and tools to enhance small-spacecraft in-orbit autonomy. The focus is on developing perception and control algorithms to enable autonomous proximity operations, and performing preliminary validation in a laboratory air-bearing testbed. To achieve this, we have developed CubeSat engineering models equipped with a simple cold-gas propulsion system, on-board vision-based sensing and embedded computer for perception and control algorithms, and integrated them in the air-bearing testbed. Basic point-to-point autonomous navigation and obstacle avoidance tasks on such a planar environment have been successfully demonstrated.

## INTRODUCTION

Recent developments in the area of small satellites [1], [2], [3] have resulted in new classes of *pico-satellites* and *nano-satellites*, weighing less than 1 kg and 10 kg, respectively, that enable low-cost missions due to reduced size and weight requirements, unified standard resulting in collaborative world-wide (partially open-source) effort [4],[5], and increase in launch opportunities. Currently, there is active research in improving basic functionalities such as attitude control [6], as well as developing more advanced capabilities including formation flying [7] and proximity operations [8], [9].

This work is concerned with autonomous navigation in a close proximity (e.g. 10 meters) of a target object. Such navigation requires the ability of the spacecraft to construct a spatial representation of its surrounding and to generate obstacle-free trajectories to a desired state while optimizing a given performance metric such as fuel expended or time taken. Autonomous navigation thus inherently relies on real-time perception, spatial mapping, path planning, and trajectory control with real-time feedback from on-board sensing. In addition, hardware limitations such as control input bounds, mass and power budget further complicate the problem.

Our main focus is on the development of control algorithms for trajectory generation and obstacle avoidance, and the integration of these algorithms with perception algorithms for a complete navigation system. To validate such complex technology it is necessary to perform extensive laboratory testing of both the hardware components and software algorithms. Motivated by this need, we have also designed and built an air-bearing testbed together with a CubeSat engineering model for partial validation of these algorithms.

Employing frictionless air-bearing testbeds is a standard approach [10], [11], [12], [13] for system validation, typically allowing three (translation in the plane and rotation around the vertical axis) or five (translations in the plane and any rotation within a given range) degrees of freedom. While very common for larger spacecraft, few such testbeds exist specifically for CubeSat or smaller form-factor spacecraft. One reason has been the unavailability of inexpensive and easy-to-use propulsion system that is safe to use in a laboratory environment. We have, therefore, designed a simplified cold gas propulsion system suitable for safe testing in a laboratory environment without the need for special protective equipment. The approach is based on rapid-prototyping inspired by recent 3D-printed integrated propulsion systems [14], [15].

An overview of the testbed is given next, followed by a description of the hardware and software components, with particular focus on the perception algorithms, and the novel control and obstacle avoidance methods. Finally, we present experimental results of autonomous navigation through a mock-up air-table environment with obstacles.

## TESTBED OVERVIEW

The testbed consists of three main components: a CubeSat engineering model placed on an air-bearing base, a flat granite table, and a motion capture system for providing ground truth data. The setup permits three degree-of-freedom motions, two translational on the table and one rotational around the vertical axis. Mock-up obstacles representing other spacecraft or debris are positioned on the table. The testbed is enclosed in a housing assembled using aluminum framing and acrylic to create a clean environment with minimum external disturbances. Additional cameras are also employed to record experiments.

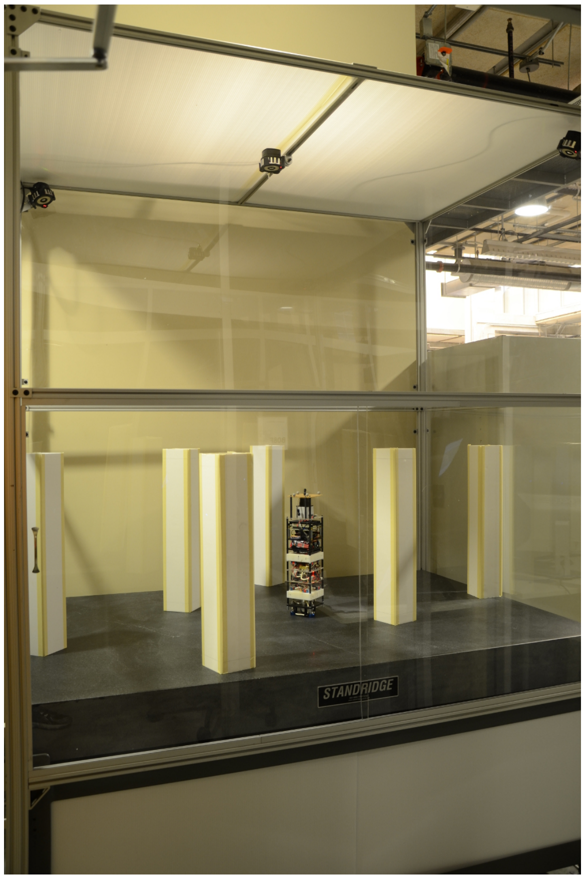

**Fig. 1: The JHU CubeSat Proximity Navigation Testbed.**

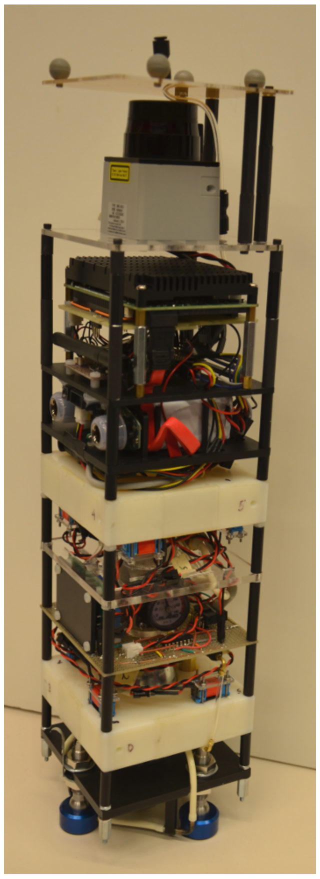

**Fig. 2: The CubeSat Engineering Model with Components Related to Autonomous Proximity Navigation.**

## Hardware Design

The CubeSat engineering model hardware consists of a propulsion subsystem, power subsystem, sensing and computing components. The details of each subsystem are discussed in the subsections below and a list of key hardware components is presented in the Table I.

The model was designed from ground up with the dimensions of 3U CubeSat in mind. We chose not to use any commercially available housing because it would need to be extensively modified to allow for assembly of non standard components. We followed a modular design that facilitates frequent assembly and disassembly, and allows introduction of modification with ease.

In addition to the essential components that makes up a CubeSat, the engineering model has a $CO_2$ canister and regulator on-board to feed the air-bearing setup and the propulsion system with $CO_2$ at 60 PSI. A switchable 2-way valve is integrated to choose the pressurized air supply between the on-board $CO_2$ canister and an external compressed air source. The external compressed air is provided through a flexible pipe which connects to a quick disconnect coupling at the top of the model. The external compressed air source is meant to be used only during testing phase as it it adds significant external forces to the model.

### *Propulsion Subsystem*

The propulsion subsystem consists of a 3D printed manifold, pneumatic solenoid valves and an one way valve. Although the design is based on other 3D printed propulsion system that can store and utilize liquid compressed gas, this design takes in $CO_2$ from a canister on-board which is same one used by the air-bearing components. We decided to use $CO_2$ to facilitate prolonged laboratory testing and ensure safety of people involved. Although the thrust varies from one thruster to another, experiments conducted put the the average maximum thrust at around 0.1N. The nozzles and the propellant manifold is designed and manufactured using a type of rapid prototyping technique called PolyJet Printing. PolyJet printed parts have high resolution and

**TABLE I: List of Key Autonomy Hardware Components.**

| Name | Description | Specification |
|---|---|---|
| Advantech MIO-2262 | High-level computer | CPU: Atom N2800 |
| ATmega 2560 | Low-level controller | CPU: 8-bit AVR<br>Architecture:RISC<br>Freq: 16 MHz<br>SRAM: 256 Kbytes |
| Hokuyo URG-04LX | Lidar | Max range: 1000 mm<br>Min range: 20 mm<br>Resolution: 2 mm |
| ASUS Xtion Pro | Depth sensor | Max range: 4.5 m |
| Minoru camera | Stereo camera | Baseline: 10 mm<br>FOV:72° |
| MPU 6000 | IMU unit | Interface: SPI<br>Accl range:$\pm 4g$<br>Accl res: 16 bit<br>Gyro Range:$\pm 500°$<br>Gyro res: 16 bit |
| Parker X-Valve | Solenoid Valves | Max pressure: 100 psi |
| Li-ion battery | On-board Power | Cells: 2<br>Capacity: 4000mAh |
| Newway S102501 | Air bearing | Flatness: 0.0005 mm<br>Size: 25 mm dia |

are non-porous which is a necessity for the manifold. Figure 3 shows the propulsion system assembly. A cutaway portion reveals the nozzle design.

### *Sensing and Computing Components*

The sensing and computing components includes several sensors, a low-level controller and a high level computer. The information flow is described in Figure 5 .

The processing for autonomy algorithm takes place on the The motion of the CubeSat model is captured externally using a commercial motion capture system and is used as ground-truth for comparison

*Sensors:* The engineering model is instrumented with several sensors for collecting the inertial measurement data, monitoring parameters like the pressure, temperature of CO2, total current consumption and battery voltage. In addition, there are sensors like laser scanner, depth cameras and stereo cameras which communicate with the high-level computer. Depth cameras provide depth information aligned with every pixel, in addition to the color information of an image pixel, that is robust

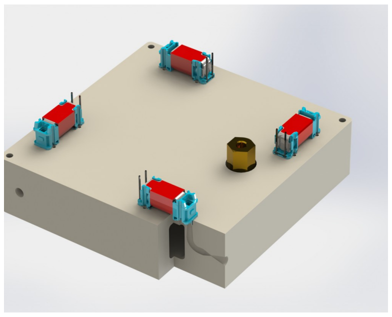

**Fig. 3: 3D Printed Propulsion Manifold Assembled with Low-ost Solenoid Pneumatic Valves.**

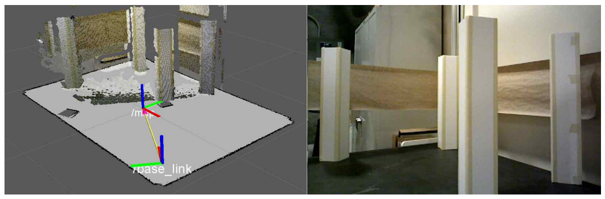

**Fig. 4: Left: Point Cloud from Depth Sensor Right: Camera View.**

to lighting condition and textures of objects in view.

*Low-level Controllers:* The low-level controller is an AVR microcontroller based board which controls the solenoid valves, communicates with sensors and high-level computer, and runs low level estimation and control algorithms.

*High-Level Computer:* MIO-2262, an Intel Atom N2800 based SBC, is used as the high-level computer. It interfaces to a pair of stereo vision cameras, PrimeSense depth camera and a low level controller board. The low-level controller is an AVR microcontroller based board which controls the solenoid valves, communicates with sensors and high-level computer, and runs low level estimation and control algorithms.

## SOFTWARE ARCHITECTURE BASED ON ROBOT OPERATING SYSTEM(ROS)

Our approach is to build upon and extend existing robotic algorithms, specifically related to robot perception, trajectory planning and control. The Robot Operating System (ROS) is a set of open-source robotic software libraries and tools which provide a subset of these required capabilities. We utilized the existing perception algorithms for mapping and localization in ROS and developed novel control and obstacle avoidance methods specifically for nanosatellites. These new algorithms are also integrated within the ROS framework which is also used for visualization and user interface. The details of the algorithms are discussed next.

### Perception Algorithm

The spacecraft sensor data is processed using perception algorithms that generate a map of the environment and localize the spacecraft with respect to this map. Data from the Inertial Measurement Unit (IMU), camera images and laser scans of satellite's surrounding, as well the thruster commands serve as inputs to the algorithm. We assume that the shape of obstacles is known, so that they can be easily identified in the constructed map. The pose of the physical obstacles is then employed by the control algorithm to navigate to the goal while avoiding collisions.

Different algorithms are employed based on the sensor type, i.e. whether a 3-D depth camera or a 2-D Lidar is used. For depth-cameras we use an open-source algorithm, RGBD-SLAM [16], based on 3-D feature matching and relative pose estimation. Local maps and their relative transformations are combined as a graph which is further optimized to produce a consistent global 3-D map. For laser scanners we employ a traditional occupancy-grid based simultaneous localization and mapping (SLAM) and particle filter (PF) localization. The map in the SLAM algorithm is a grid where the value of each cell is related to the probability of the grid being occupied. Multiple possible maps are maintained as particles in a probabilistic representation accounting for uncertainty. Our implementation is based on the open-source Gmapping [17] for mapping and AMCL [18] for localization.

### Control Algorithm

We employ gyroscopic obstacle avoidance algorithm combined with standard stabilization control laws to handle unexpected obstacles during trajectory execution. Our approach is motivated by the fact that gyroscopic forcing has desirable convergence properties under certain conditions. In particular, the system is guaranteed to avoid collisions and reach the goal as long as the obstacle is convex and the required controls do not saturate.

Avoiding collisions is a key requirement for autonomous proximity operations. The subject has a long history in robotics (see [19] and [20] for an overview). Standard methodologies include potential fields [21], artificial navigation functions [22], or dynamic window avoidance [23]. Gyroscopic avoidance [24], employed in this work, is another strategy based on steering away from obstacles without injecting additional energy (e.g. from repulsive potential forces) into the system. By designing the system to have minimum energy at a given goal state, the system is guaranteed to reach that state even in presence of obstacles.

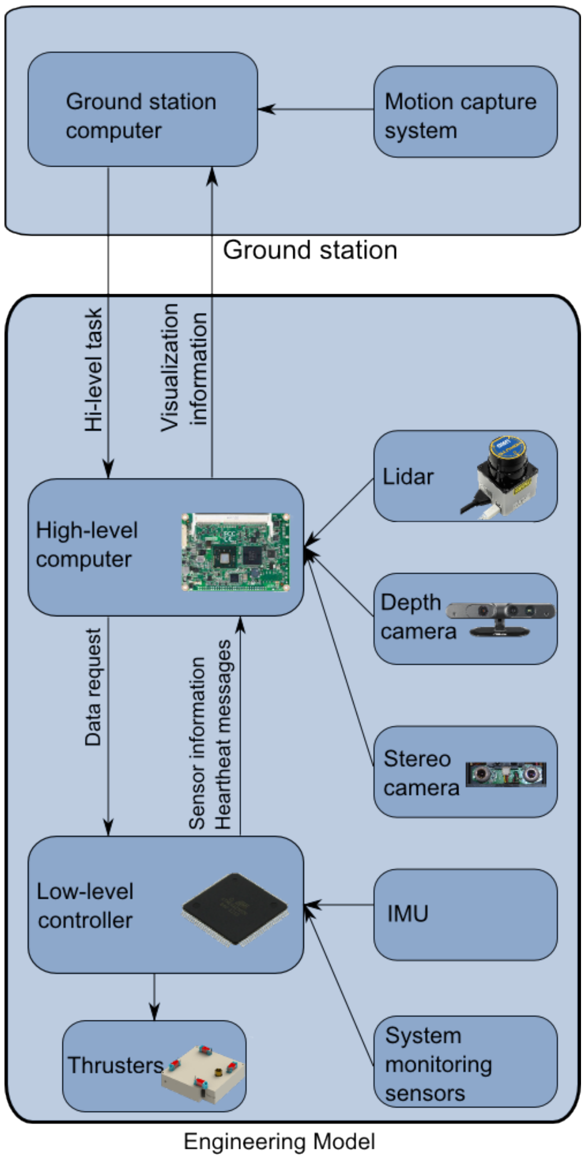

**Fig. 5: Sensor, Actuator, and Computing Components.**

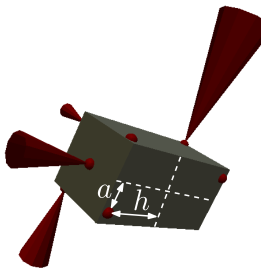

**Fig. 6: The Ten-thruster Configuration for Full 6-Dof Control.**

***System Model***

The robot is modeled as a single rigid body with position $\boldsymbol{x} = (x, y, z) \in \mathbb{R}^3$ and orientation matrix $R \in \mathrm{SO}(3)$ defined with respect to a moving frame attached to a nearby moving reference (e.g. a target), such as an RSW frame [25], [26]. The *body-fixed* angular velocity is denoted by $\boldsymbol{\omega} \in \mathbb{R}^3$. The vehicle has mass $m$ and principal moments of rotational inertia $J_x, J_y, J_z$ forming the inertia tensor $\mathbb{J} = \mathrm{diag}(J_x, J_y, J_z)$.

The spacecraft is actuated with $c$ thrusters producing forces denoted by $\boldsymbol{u} \in U$ where $U \subset \mathbb{R}^c$ is a bounded set. The equations of motion are

$$\dot{R} = R\widehat{\boldsymbol{\omega}}, \tag{1}$$

$$\begin{pmatrix} \mathbb{J}\dot{\boldsymbol{\omega}} \\ m\ddot{\boldsymbol{x}} \end{pmatrix} = \begin{pmatrix} \mathbb{J}\boldsymbol{\omega} \times \boldsymbol{\omega} + \boldsymbol{\tau}_{\text{ext}} \\ \boldsymbol{f}_{\text{ext}} \end{pmatrix} + \begin{bmatrix} \mathbf{1} & \mathbf{0} \\ \mathbf{0} & R \end{bmatrix} B\boldsymbol{u}, \tag{2}$$

where the map $\widehat{\cdot} : \mathbb{R}^3 \to \mathfrak{so}(3)$ is defined by

$$\widehat{\boldsymbol{\omega}} = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}, \tag{3}$$

and $B$ is a constant thruster allocation matrix such that $\mathrm{rank}(B) = 6$ and $\{B\boldsymbol{u} \mid \boldsymbol{u} \in U\} \subset \mathbb{R}^6$ is an open set containing the origin.

For our satellite equipped with ten thrusters illustrated in Figure 6 the thruster allocation matrix is defined by

$$B = \begin{bmatrix} 0 & -h & 0 & h & h & 0 & -h & 0 & 0 & 0 \\ h & 0 & -h & 0 & 0 & h & 0 & -h & 0 & 0 \\ a & a & a & a & -a & -a & -a & -a & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 1 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & 1 & -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix},$$

where $a > 0$ and $h > 0$ are the lateral and vertical offsets of each of the side thrusters. A standard approximation of external forces is

$$\boldsymbol{f}_{\text{ext}} = m \begin{bmatrix} 2\omega_c \dot{z} \\ -\omega_c^2 y \\ -2\omega_c \dot{x} + 3\omega_c^2 z \end{bmatrix},$$

with constant $\omega_c > 0$ denoting the circular orbit angular rate. The dominating torques are due to gravity gradients defined by

$$\boldsymbol{\tau}_{\text{ext}} = 3\omega_c^2 R\boldsymbol{e}_z \times \mathbb{J}R\boldsymbol{e}_z,$$

where $\boldsymbol{e}_z = (0, 0, 1)$. Given the time-scales considered in this work (e.g. maneuvers lasting on the order of a few minutes) the effect of such forces is negligible.

*Obstacle constraints* require that the spacecraft must not collide with obstacles denoted by $\mathcal{O}_1, ..., \mathcal{O}_{n_o} \subset \mathbb{R}^3$. Assume that the vehicle is occupying a region $\mathcal{A}(R, \boldsymbol{x}) \subset \mathbb{R}^3$, and let $\mathbf{prox}(\mathcal{A}_1, \mathcal{A}_2)$ be the Euclidean distance between two sets $\mathcal{A}_{1,2}$ that is negative in the case of intersection. Obstacle avoidance requires that

$$\min_i \mathbf{prox}\left(\mathcal{A}(R, \boldsymbol{x}), \mathcal{O}_i\right) > 0. \tag{4}$$

We use a standard collision checking algorithm implemented by Proximity Query Package (PQP) [27] to compute $\mathbf{prox}$.

***Gyroscopic Obstacle Avoidance***

The goal of gyroscopic steering is to modify the dynamical system so that obstacles are avoided without violating its stability properties. This is accomplished by adding a gyroscopic force term, i.e. that does not inject energy into the system. Our development in this context can be regarded as an extension of [24] to 3-D workspaces and to vehicles with more realistic dynamics.

Assume that while executing a reference trajectory $\boldsymbol{x}_r : [0, T] \to \mathbb{R}^3$ at time $t < T$ the vehicle encounters an unexpected obstacle blocking its path. Obstacle avoidance is performed by first assigning a new desired position state $(\boldsymbol{x}_d, \dot{\boldsymbol{x}}_d)$, for instance $\boldsymbol{x}_d = \boldsymbol{x}_r(t_g)$ for some $t < t_g \leqslant T$ and then steering away from the obstacle and towards $\boldsymbol{x}_d$. This is accomplished by defining the error term

$$\Delta\boldsymbol{x} = \boldsymbol{x} - \boldsymbol{x}_d$$

and the control law

$$\boldsymbol{f}_d = -k_x \Delta\boldsymbol{x} - k_v \Delta\dot{\boldsymbol{x}} - \boldsymbol{f}_{\text{ext}} + \Gamma\dot{\boldsymbol{x}}, \tag{5}$$

where $k_x, k_v > 0$ and $\Gamma \in \mathbb{R}^{3\times 3}$ is a skew-symmetric matrix, i.e. such that $\Gamma = -\Gamma^T$.

Let $d(R, \boldsymbol{x}) > 0$ and $\boldsymbol{n}(R, \boldsymbol{x}) \in \mathbb{R}^3$ denote the distance and the unit vector to the detected obstacle, respectively, computed by the $\mathbf{prox}$ function. We then set

$$\Gamma = \frac{k_g v_{\max}}{d(R, \boldsymbol{x})} \widehat{\boldsymbol{c}}, \tag{6}$$

for some fixed $k_g, v_{\max} > 0$, and $\boldsymbol{c} \in \mathbb{R}^3$ is computed according to

$$\begin{aligned} \boldsymbol{c} &= \begin{cases} 0 & \text{if } |\beta| \geqslant \pi/2, \\ \frac{\boldsymbol{c}'}{\|\boldsymbol{c}'\|} & \text{if } |\beta| < \pi/2, \end{cases} \\ \boldsymbol{c}' &= \frac{\boldsymbol{n} \times \dot{\boldsymbol{x}}}{\|\dot{\boldsymbol{x}}\|}, \\ \beta &= \mathrm{sgn}(\arcsin\|\boldsymbol{c}'\|) \arccos\frac{\boldsymbol{n}^T\dot{\boldsymbol{x}}}{\|\dot{\boldsymbol{x}}\|}. \end{aligned} \tag{7}$$

Here $\beta \in [-\pi, \pi]$ is angle between direction of collision and direction of motion, and $\boldsymbol{c}$ plays the role of an axis around which the velocity vector $\dot{\boldsymbol{x}}$ rotates to avoid the obstacle.

### *Control Law*

We next design a controller to track desired position $\boldsymbol{x}_d(t)$ and attitude $R_d(t)$ during a time interval $[0,T]$, which must be chosen so that the required control inputs do not saturate. Asymptotic stability can be achieved by setting the controls to

$$\boldsymbol{u} = (B^T B)^{-1} B^T \begin{bmatrix} \boldsymbol{\tau}_d \\ R^T \boldsymbol{f}_d \end{bmatrix}, \tag{8}$$

where $\boldsymbol{f}_d$ is defined in (5) and $\boldsymbol{\tau}_d$ is a standard attitude tracking control law. For instance, following [28] such a control law is obtained by

$$\begin{aligned}\boldsymbol{\tau}_d =& -\operatorname{skew}(K_R \Delta R)^{\vee} - K_\omega(\boldsymbol{\omega} - \Delta R^T \boldsymbol{\omega}_d) \\ &+ \boldsymbol{\omega} \times \mathbb{J}(\Delta R^T \boldsymbol{\omega}_d) + \mathbb{J}(\Delta R^T \dot{\boldsymbol{\omega}}_d),\end{aligned} \tag{9}$$

where $\Delta R := R_d^T R$, $\operatorname{skew}(A) := (A - A^T)$, the operator $\check{\cdot}$ is the inverse of $\hat{\cdot}$ defined in (3), and $K_R, K_\omega$ are positive definite matrices.

The complete algorithms is summarized below:

**Algorithm 1:** $\boldsymbol{u} = \texttt{Track}\,(\boldsymbol{x}(t), R(t), \boldsymbol{x}_d(t), R_d(t))$

```
parameters: gains $k_x, k_v, K_R, K_\omega, k_g$
1 $\boldsymbol{f}_d = -k_x \Delta \boldsymbol{x} + k_v \Delta \dot{\boldsymbol{x}} - \boldsymbol{f}_{\text{ext}}(\boldsymbol{s}) + \Gamma \dot{\boldsymbol{x}}$
2 $\boldsymbol{\tau}_d = -\operatorname{skew}(K_R \Delta R)^{\vee} - K_\omega(\boldsymbol{\omega} - \Delta R^T \boldsymbol{\omega}_d) + \boldsymbol{\omega} \times \mathbb{J}(\Delta R^T \boldsymbol{\omega}_d) + \mathbb{J}(\Delta R^T \dot{\boldsymbol{\omega}}_d)$,
3 $\boldsymbol{u} = (B^T B)^{-1} B^T \begin{bmatrix} \boldsymbol{\tau}_d \\ R^T \boldsymbol{f}_d \end{bmatrix}$,
4 return control inputs $\boldsymbol{u}$
```

### *Simulations in 3-D*

Figure 7 shows an example scenario requiring the satellite to avoid a large obstacle to stabilize at the origin. The stabilization of a group of seven satellites performing a simulated segmented mirror assembly task is considered next. The goal is to reconfigure from a free-flying mode to a latched configuration without incurring any collisions. Figure (8) shows a few snapshots of the scenario. The algorithm exhibited good behavior and no collision happened before the final latching.

## Experimental Evaluation

The experiment was designed to simulate autonomous navigation of a single 3U CubeSat in the proximity of other mock-up spacecraft and objects treated as obstacles. The ground station, connected to the high-level on-board-computer, commands a desired goal position to the model which then autonomously navigates to the commanded position while avoiding the obstacles. We present, in Figure 9 and 10, the results as navigation and obstacle avoidance maneuvers for two goal position.

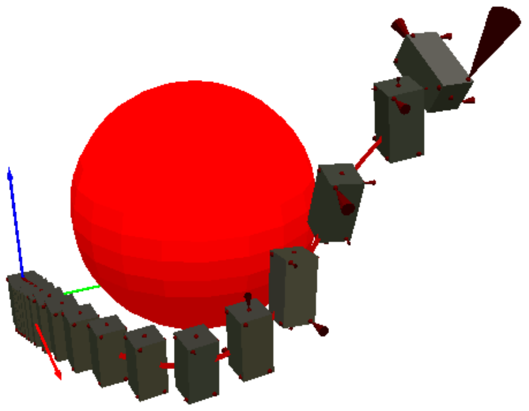

**Fig. 7: A Conceptual Nano-satellite with Ten Thrusters Avoiding a Spherical Obstacle and Stabilizing at the Origin.**

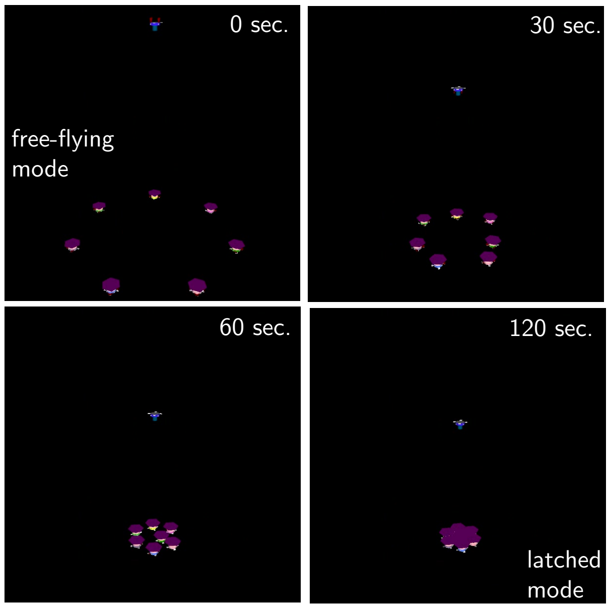

**Fig. 8: Reconfiguration of a Fleet of Spacecraft Forming a Segmented Mirror. Each Vehicle Employs Gyroscopic Obstacle Avoidance to Avoid Collisions with Other Mock-up Spacecraft and Arriving Stably at its Latching Configuration.**

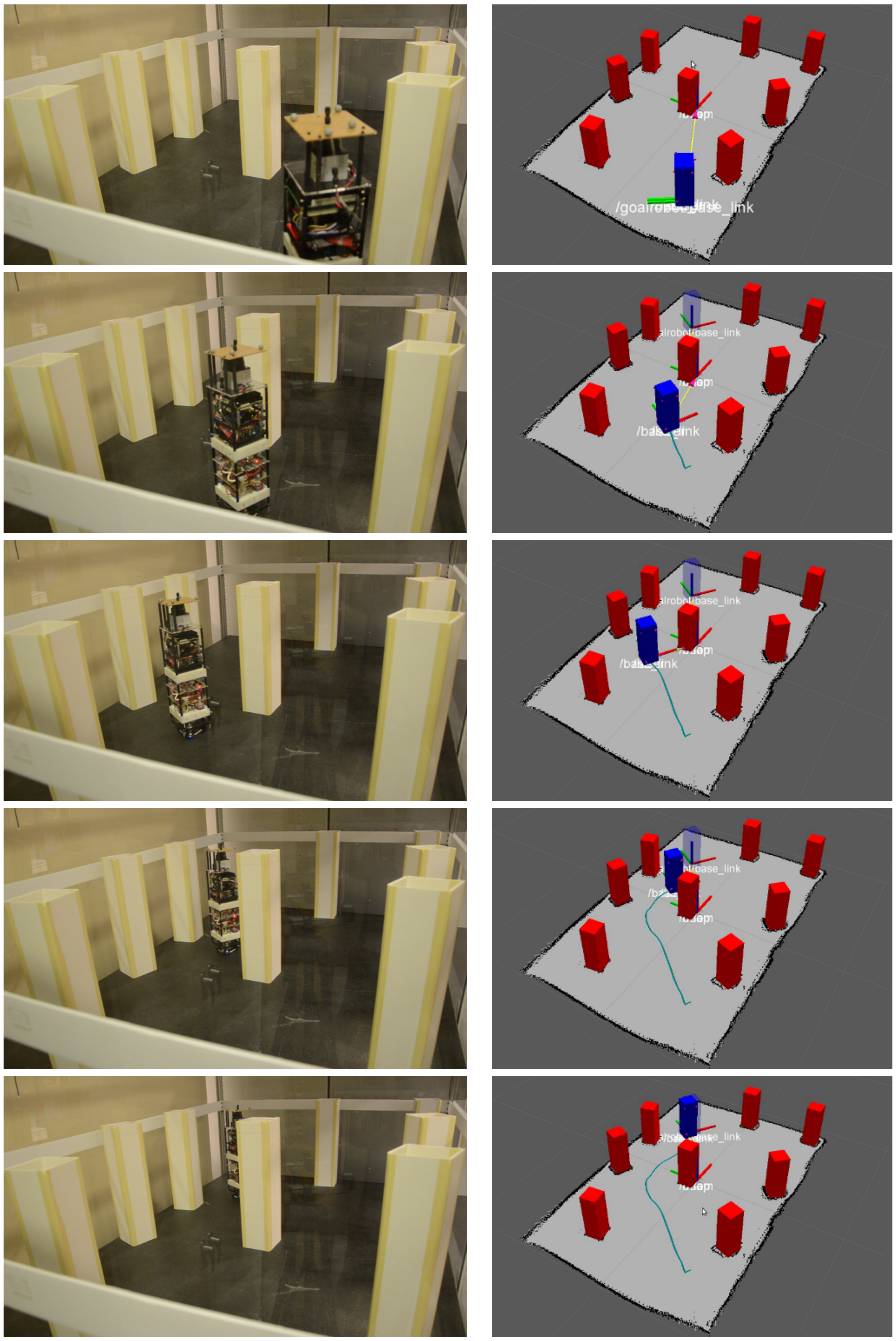

**Fig. 9: Obstacle Avoidance Manouver-Scenario 1**

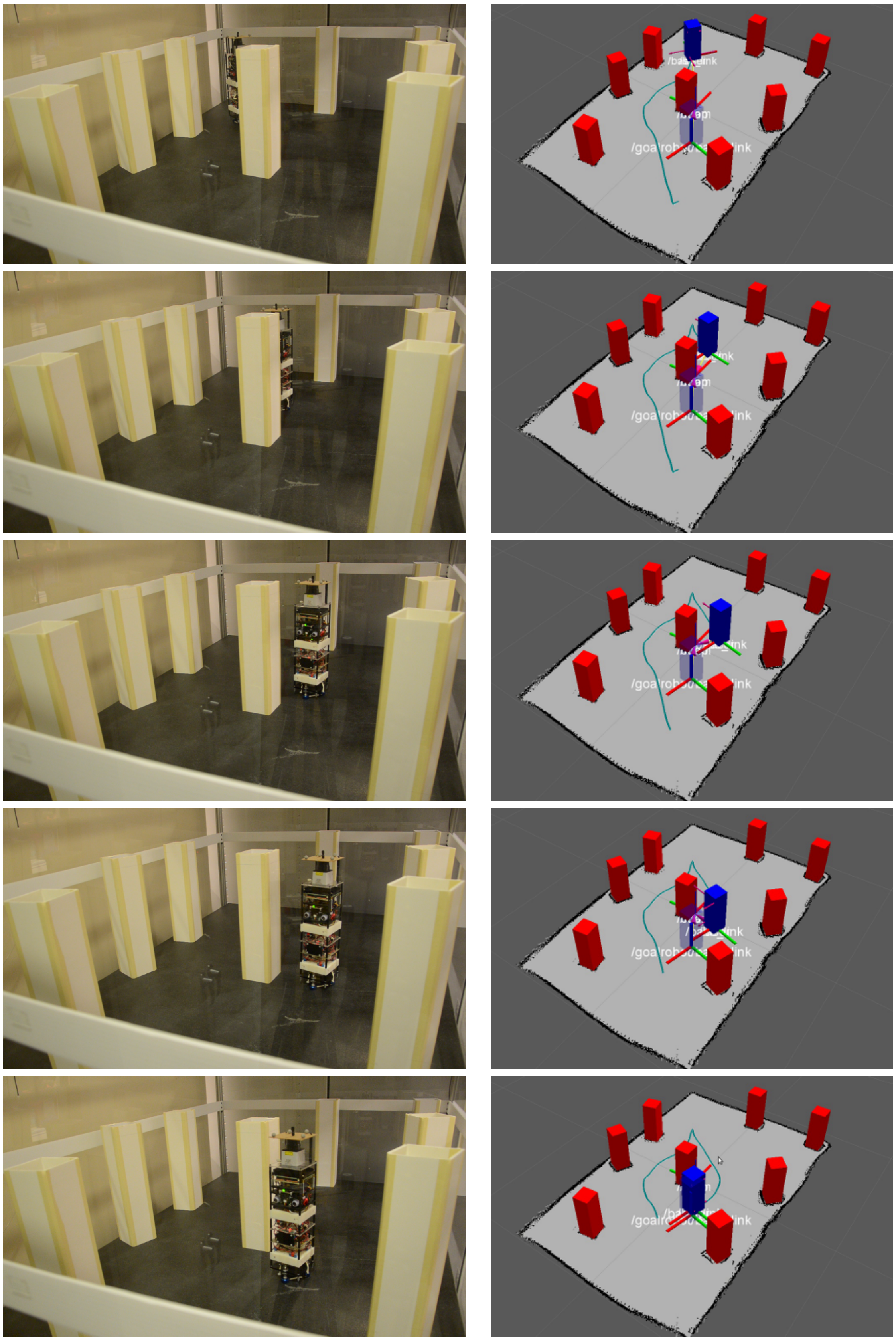

**Fig. 10: Obstacle Avoidance Manouver-Scenario 2**

## Conclusion and Future Work

This work focused on the development and demonstration of autonomous proximity navigation capabilities for nano-satellites. The reported initial developments are based on a laboratory air-bearing testbed in which we were able to implement and test spacecraft perception, control, and obstacle avoidance algorithms. To achieve this we assumed a planar environment and employed a simplistic cold-gas propulsion system and embedded system components (which are yet to be validated in a space environment) with CPU computing power capable of executing current robotic perception and control algorithms. Future work will focus on employing 3-DOF attitude control, a more realistic propulsion system, and performing integration with standard CubeSat subsystems. Finally, our goal is to employ the developed algorithms to enable future applications across multiple spacecraft in mission-specific contexts such as satellite docking and servicing, debris removal, or small-body sampling.

## References

[1] D. Sakoda and J. A. Horning, "Overview of the nps spacecraft architecture and technology demonstration satellite, npsat1," 2002.

[2] R. Sandau, "Status and trends of small satellite missions for earth observation," *Acta Astronautica*, vol. 66, no. 1, pp. 1–12, 2010.

[3] S. P. Neeck, T. J. Magner, and G. E. Paules, "Nasa's small satellite missions for earth observation," *Acta Astronautica*, vol. 56, no. 1, pp. 187–192, 2005.

[4] H. Heidt, J. Puig-Suari, A. Moore, S. Nakasuka, and R. Twiggs, "Cubesat: A new generation of picosatellite for education and industry low-cost space experimentation," 2000.

[5] R. Nugent, R. Munakata, A. Chin, R. Coelho, and J. Puig-Suari, "The cubesat: The picosatellite standard for research and education," *Aerospace Engineering*, vol. 805, pp. 756–5087, 2008.

[6] D. S. Sanders, D. L. Heater, S. R. Peeples, P.-H. A. Huang, *et al.*, "Pushing the limits of cubesat attitude control: A ground demonstration," 2013.

[7] E. Caillibot, C. Grant, and D. Kekez, "Formation flying demonstration missions enabled by canx nanosatellite technology," 2005.

[8] M. Kobilarov and S. Pellegrino, "Trajectory planning for cubesat short-time-scale proximity operations," *Journal of Guidance, Control, and Dynamics*, vol. 37, no. 2, pp. 566–579, 2014.

[9] C. Becker, R. Howard, and J. Rakoczy, "Smartphone video guidance sensor for small satellites," 2013.

[10] W. R. Wilson, L. L. Jones, and M. A. Peck, "A multimodule planar air bearing testbed for cubesat-scale spacecraft," *Journal of Dynamic Systems, Measurement, and Control*, vol. 135, no. 4, p. 045001, 2013.

[11] D. Miller, A. Saenz-Otero, J. Wertz, A. Chen, G. Berkowski, C. Brodel, S. Carlson, D. Carpenter, S. Chen, S. Cheng, *et al.*, "Spheres: a testbed for long duration satellite formation flying in micro-gravity conditions," in *Proceedings of the AAS/AIAA space flight mechanics meeting*. Clearwater, Florida, January, 2000, pp. 167–179.

[12] J. Prado, G. Bisiacchi, L. Reyes, E. Vicente, F. Contreras, M. Mesinas, and A. Juárez, "Three-axis air-bearing based platform for small satellite attitude determination and control simulation," *Journal of Applied Research and Technology*, vol. 3, no. 03, 2005.

[13] B. N. Agrawal and R. E. Rasmussen, "Air-bearing-based satellite attitude dynamics simulator for control software research and development," in *Aerospace/Defense Sensing, Simulation, and Controls*. International Society for Optics and Photonics, 2001, pp. 204–214.

[14] D. Hinkley, "A novel cold gas propulsion system for nanosatellites and picosatellites," 2008.

[15] S. Arestie, E. G. Lightsey, and B. Hudson, "Development of a modular, cold gas propulsion system for small satellite applications," *Journal of Small Satellites*, vol. 1, no. 2, pp. 63–74, 2012.

[16] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, "Rgb-d mapping: Using kinect-style depth cameras for dense 3d modeling of indoor environments," *The International Journal of Robotics Research*, vol. 31, no. 5, pp. 647–663, 2012.

[17] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," *Robotics, IEEE Transactions on*, vol. 23, no. 1, pp. 34–46, 2007.

[18] D. Fox, W. Burgard, F. Dellaert, and S. Thrun, "Monte carlo localization: Efficient position estimation for mobile robots," *AAAI/IAAI*, vol. 1999, pp. 343–349, 1999.

[19] J.-C. Latombe, *Robot Motion Planning*. Kluwer Academic Press, 1991.

[20] H. Choset, K. M. Lynch, S. Hutchinson, G. A. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, June 2005.

[21] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *Int. J. Rob. Res.*, vol. 5, no. 1, pp. 90–98, Apr. 1986. [Online]. Available: http://dx.doi.org/10.1177/027836498600500106

[22] E. Rimon and D. Koditschek, “Exact robot navigation using artificial potential functions,” *Robotics and Automation, IEEE Transactions on*, vol. 8, no. 5, pp. 501 –518, oct 1992.

[23] D. Fox, W. Burgard, and S. Thrun, “The dynamic window approach to collision avoidance,” *Robotics Automation Magazine, IEEE*, vol. 4, no. 1, pp. 23 –33, mar 1997.

[24] D. E. Chang and J. E. Marsden, “Gyroscopic forces and collision avoidance with convex obstacles,” *Nonlinear Dynamics and Control*, no. 295, pp. 145–160, 2003.

[25] D. Vallado, *Fundamentals of Astrodynamics and Applications*. Primis, 1997.

[26] M. Kobilarov and S. Pellegrino, “Short-time trajectory planning for cubesat proximity operations,” 2012, to appear in Journal of Guidance, Control, and Dynamics.

[27] S. Gottschalk, M. C. Lin, and D. Manocha, “OBB-Tree: A hierarchical structure for rapid interference detection,” *Eurographics/ACM SIGGRAPH Symposium on Computer Animation*, vol. 30, pp. 171–180, 1996.

[28] F. Bullo and A. Lewis, *Geometric Control of Mechanical Systems*. Springer, 2004.