Research Article

INTERNATIONAL JOURNAL OF ADVANCED ROBOTIC SYSTEMS

International Journal of Advanced Robotic Systems
January-February 2018: 1–12

DOI: 10.1177/1729881417753691
journals.sagepub.com/home/arx
SAGE

Handrail detection and pose estimation for a free-flying robot

Dong-Hyun Lee[1], Brian Coltin[2], Theodore Morse[2], In-Won Park[2], Lorenzo Flückiger[2] and Trey Smith[3]

Abstract

We present a handrail detection and pose estimation algorithm for the free-flying Astrobee robots that will operate inside the International Space Station. The Astrobee will be equipped with a single time-of-flight depth sensor and a compliant perching arm to grab the International Space Station handrails. Autonomous perching enables a free-flying robot to minimize power consumption by holding its position without using propulsion. Astrobee is a small robot with many competing demands on its computing, power, and volume resources. Therefore, for perching, we were limited to using a single compact sensor and a lightweight detection algorithm. Moreover, the handrails on the International Space Station are surrounded by various instruments and cables, and the lighting conditions change significantly depending on the light sources, time, and robot location. The proposed algorithm uses a time-of-flight depth sensor for handrail perception under varying lighting conditions and utilizes the geometric characteristics of the handrails for robust detection and pose estimation. We demonstrate the robustness and accuracy of the algorithm in various environment scenarios.

Keywords

Space robots, free-flyers, object detection, pose estimation

Date received: 23 May 2017; accepted: 26 November 2017

Topic: Vision Systems
Topic Editor: Yangquan Chen
Associate Editor: Liang Sun

Introduction

The NASA Astrobee Project is developing the next-generation free-flying robots that will operate inside the International Space Station (ISS) alongside the astronauts.[1] Astrobee's primary objective is to provide a zero-g research facility for guest scientists. The Astrobees will replace the SPHERES robots that have been among the most-used facilities on the ISS since they arrived in 2006, hosting experiments on topics ranging from magnetic propulsion,[2] to simulated satellite inspection,[3] to studying the dynamics of tethers in zero-g.[4]

Astrobee will carry on the SPHERES tradition, while opening up new areas of research with its greatly expanded capabilities, which include improved autonomy, better support for guest science hardware add-ons, a built-in suite of cameras, and a robot arm. If you have new ideas about how to use zero-g robots, the Astrobee Research Facility (https://www.nasa.gov/astrobee) is currently seeking guest scientists to begin experiments on the ISS in late 2018. Experiments already under development for Astrobee include an Radio-Frequency Identification (RFID) reader for tracking equipment on ISS, a new gripper using a gecko-inspired adhesive that could enable perching on flat surfaces, and an investigation using an arm for acrobatic

[1] Department of Electrical Engineering, Kumoh National Institute of Technology, Gumi, Gyeongbuk, South Korea
[2] Stinger Ghaffarian Technologies, Inc., Moffett Field, California, USA
[3] Intelligent Robotics Group, NASA Ames Research Center, Moffett Field, California, USA

Corresponding author:
Dong-Hyun Lee, Department of Electrical Engineering, Kumoh National Institute of Technology, 61 Daehak-ro, Gumi, Gyeongsangbuk-do, South Korea.
Email: donglee@kumoh.ac.kr

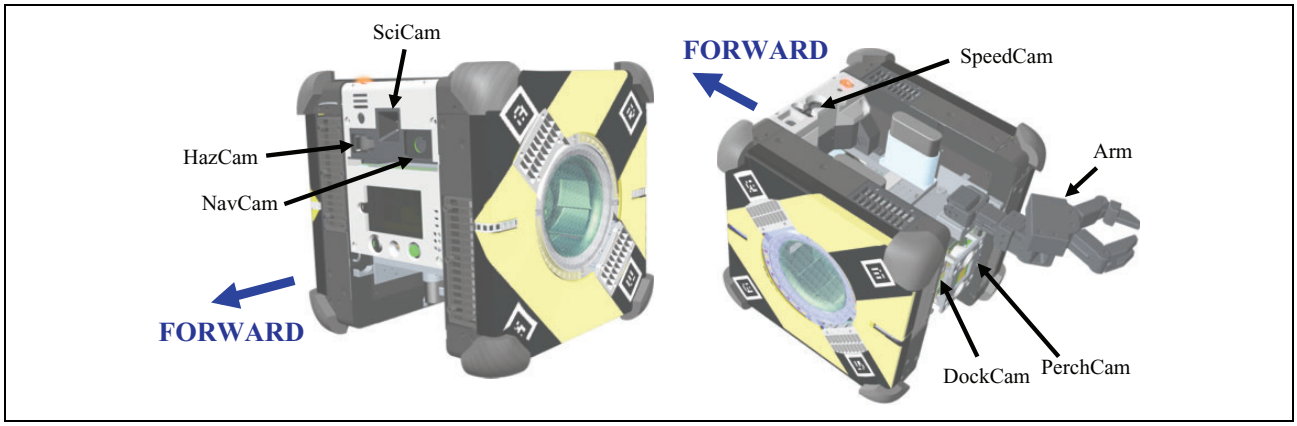

Figure 1. Astrobee robot geometry.

Table 1. Astrobee pose estimation modes.

Mode	Purpose	Sensor	Information
General-purpose	Navigate throughout the ISS	NavCam camera (forward facing)	Recognize features on walls from prior map; supplement with relative updates from visual odometry
Docking	Improve accuracy and robustness during docking approach	DockCam camera (aft facing)	Recognize fiducials on docking station
Perching	Improve accuracy and robustness during perching approach	PerchCam LIDAR (aft facing)	Recognize 3-D geometry of wall and handrail
(All)	Supplement other sensing to ensure smooth high-rate pose updates	IMU	6-DOF linear acceleration and angular velocity

ISS: International Space Station; IMU: Inertial Measurement Unit; DOF: Degree Of Freedom.

maneuvers that could reduce fuel usage for future extravehicular free-flyers.

In addition to supporting guest science, the Astrobees can be teleoperated as free-flying cameras to improve situation awareness for flight controllers. They can also perform surveys with a variety of environment sensors, for example, monitoring CO_2 concentration, radiation, and noise levels. By using robots to sample rather than astronauts, more data can be collected more often with less crew time. Better information will inform future improvements to the life support systems the astronauts depend on.

The Astrobees have a variety of pose estimation modes that rely on different sensors as shown in Figure 1 and serve different operational needs as shown in Table 1. The core pose estimator is implemented as an extended Kalman filter (EKF) that can accept different inputs.[5] This article focuses on the perching mode.

There are various types of handrails on the ISS as shown in Figure 2 and the Astrobee will be equipped with a handrail perching system, which consists of a time-of-flight (ToF) depth sensor and a compliant perching arm, to hold its position for minimizing power consumption, recording videos, and monitoring the ISS interior. Since Astrobee is a small robot with many competing demands on its computing, power, and volume resources, we limited ourselves to using a single compact sensor and a robust, lightweight detection algorithm for perching.

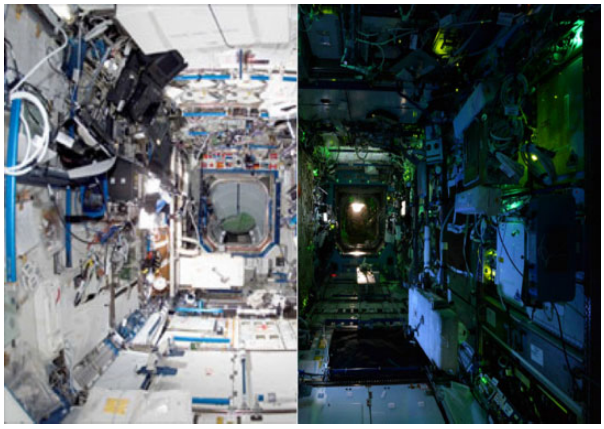

Figure 2. The lighting condition inside the US Destiny laboratory during waking and sleep periods.

In general, perching is an important ability for flying robots, such as unmanned aerial vehicles (UAVs) and micro aerial vehicles (MAVs), to minimize power consumption while performing stationary tasks in a fixed location such as video streaming and telecommunication relay as shown in Figure 3. For real-time perching, a fast handrail detection algorithm is required to robustly detect the handrail and estimate its relative pose to the robot.

Figure 3. Artist's concept of the Astrobee robot taking video of an astronaut activity while perching.

We propose a robust handrail detection and pose estimation algorithm on the ISS. All the handrails on the ISS have the same thickness and are made of similar material, but they vary as to length, mounting geometry, and background wall texture. There are major similarities between the handrails on the ISS and the other handrails in our daily lives such as handles in doors, refrigerators, and cabinets: they are cylindrical and attached to flat surfaces. Thus, the proposed algorithm can also be applied to various tasks for autonomous robots to perch or manipulate handrails. In fact, detecting the handrails on the ISS is quite challenging due to the cluttered environmental condition of the ISS with varying lighting conditions. To simplify our algorithm and particularly our testing approach, we focus on the 70% of handrails that are "typical," that is, length > 50 cm and mounted in the middle of a wall. Most of the remaining handrails are less suitable for perching in any case, because they are found near narrow hatchways where Astrobee would be more likely to obstruct astronaut motion. The task of the handrail estimation is to confirm that a target handrail exists and provide a relative pose estimate between handrail and gripper that is sufficiently accurate (3 cm) and stable to enable robust grasping. After successful perching, the arm joints act as a pan/tilt base to point cameras fixed on the robot body; to provide the maximum range of motion, the desired perched geometry has the arm link normal to the plane of the wall.

The handrail estimator complements the general-purpose visual pose estimator that provides less accuracy but works anywhere in the ISS US segment.[5] The perching process works as follows: The user views a 3-D model of the ISS with known handrail locations marked and selects a handrail to perch on. Astrobee then moves to a target pose near the handrail using its general-purpose estimator. Once at the target pose, the handrail should be in the ToF sensor field of view and in the vertical orientation that is compatible with the gripper. The robot then switches from general-purpose navigation to visual servoing on the handrail to complete the perching approach.

The task of handrail detection and pose estimation is an instance of the general problem of recognizing 3-D objects and estimating their poses.[6,7] There have been numerous methods for 3-D object recognition such as 3-D keypoint detection.[8–10] However, most of them focus on identifying multiple objects using RGB-D sensors.[11–13] Since our goal is to detect a handrail and estimate its pose with a computationally limited processor and a single depth sensor, the previous 3-D object recognition approaches with the RGB-D sensors are not suitable for this problem. The Robonaut 2 at the NASA Johnson Space Center has been developing a mobile system and the end effectors of the legs have a sensor package for handrail detection.[14,15] However, the system uses both a camera and a ToF sensor for handrail detection.

The handrail detection problem also has similarity with the handrail perception problem for autonomous door and drawer opening.[16–18] However, most work in handle detection assumes that the handle is small enough to be shown its whole shape within the view area of sensors and the door does not have other attached objects on its surface except for the handle. Unlike the doors and cabinets where only handles are attached, the walls in the ISS have considerable numbers of cables and instruments attached along with handrails. Moreover, due to the length of the handrails and the limited field of view of the sensor, only part of the handrail is within the viewing field of the sensor as the robot approaches to the handrail for perching.

Unlike the other approaches that use both an RGB camera and a depth sensor, this article focuses on the approach with a single depth sensor for detecting the handrails in a cluttered environment with varying lighting conditions. We are using the CamBoard Pico Flexx sensor (PMD Technologies), which is much smaller (68 mm × 17 mm × 7.25 mm) than other off-the-shelf RGB-D sensors with similar range and resolution (224 × 171 pixel) that were available at the time the sensor was selected. For robust handrail detection and pose estimation, we utilize the geometric relationship between the handrails and the walls on the ISS as well as the geometric characteristics of the handrails. The proposed approach can be applied to handle and knob detection for autonomous door and drawer opening. It can be also used in various micro UAVs to perform a wide variety of tasks that require real-time autonomous navigation and aerial gripping with lower computational and energy requirements.[19–21]

Perching arm

As a part of the Astrobee robotic system, a compliant, detachable perching arm is being developed to support long duration tasks. This arm will grab ISS handrails to hold its position without using propulsion or navigation to minimize power consumption.

The prototype of the Astrobee perching arm in Figure 4 is developed to verify the grasping functionality and the range of

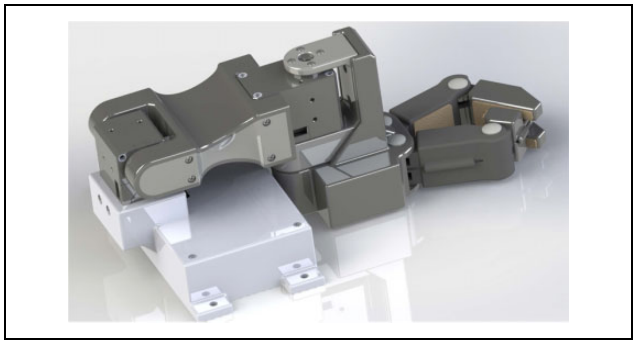

Figure 4. The Astrobee perching arm preliminary design (stowed configuration).

pan-tilt motion.[22] The arm consists of two Dynamixel AX-12A motors and the tendons in the gripper are connected to a Pololu metal gearmotor (Pololu Robotics and Electronics). Dynamixel motors are directly controlled from a BeagleBone board and the Pololu motor is controlled using a Baby Orangutan B-328 board, where the desired commands are sent from the BeagleBone board via serial. The length and mass of the Astrobee perching arm are 24 cm and 315 g, respectively.

Handrail estimation

We consider handrails that are attached on the plane since all the handrails on the ISS must be attached on the wall and most of the handrails in general are also installed on the flat surfaces, such as door knob and refrigerator handle. Moreover, estimating the plane that the handrail is attached is essential for the robot to estimate its relative orientation to the handrail for perching. After the plane estimation, the algorithm filters out false-positive handrail points by using the geometric constraints between the plane and the handrail as well as the geometric characteristics of the handrail, such as the gap distance between the plane and the handrail as well as the width of the handrail. After the outlier rejection, the algorithm estimates a line from the filtered point cloud and clusters the line to determine the handrail that contains the most suitable handrail points.

The overall procedure of the proposed algorithm can be summarized as follows:

1. *Plane estimation*: From a point cloud (Figure 5(a)), estimate a wall (plane; Figure 5(b)).
2. *Handrail outlier filtering*: Filter out false-positive handrail points.
   (a) *Gap distance test*: Ignore points that are too close to the wall to be part of a handrail (Figure 5(c)).
   (b) *Cross point test*: Ignore points whose local neighborhood points do not satisfy the handrail geometry (Figure 5(d)).
   (c) *Side width test*: Ignore points whose local neighborhood points do not satisfy the handrail width (Figure 5(e)).

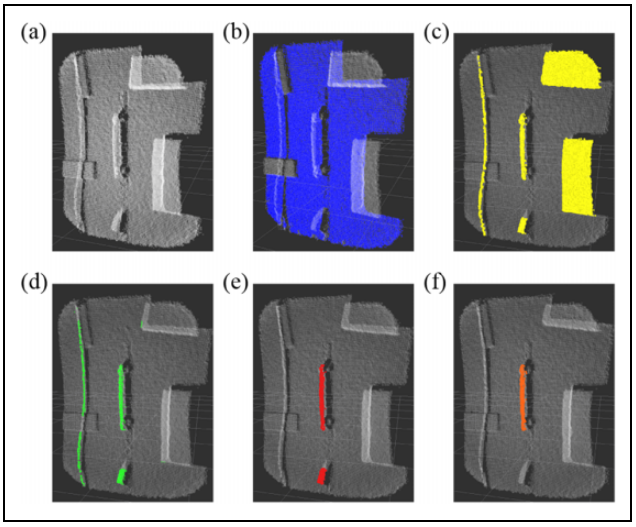

Figure 5. The handrail detection process. (a) A measured point cloud from the depth sensor. (b) The plane inliers of the estimated plane. (c) The remaining points from the plane outliers after gap distance test. (d) The remaining points after the cross point test. (e) The remaining points after the side width test. (f) The remaining points after line estimation and clustering.

3. *Line estimation and clustering*: Estimate handrail candidate sets from the rest of the points and cluster them (Figure 5(f)).
4. *Handrail selection*: Determine the line set that contains the most suitable handrail points for grasping.

Plane estimation and gap distance test

The handrail consists of one beam and two legs as shown in Figure 6(b) so that the handrail can be modeled as a combination of a long cylinder for a handrail beam and two short cylinders for two legs that connect the beam to the wall as shown in Figure 6(a). In this cylinder model, the diameter and the height of the handrail beam, b_D and b_H, are, respectively, set to 3.5 and 55.0 cm (or 105.5 cm for a longer handrail), and the handrail leg diameter, l_D, and height, l_H, are set to 3.5 and 6.5 cm, respectively. To utilize the geometric characteristics of the handrail on the wall, the proposed algorithm first estimates the plane coefficients and classifies plane inlier set, $\mathrm{C}_{\mathrm{Plane}}$ from a point cloud using Random sample consensus (RANSAC).[23] Since the gap between the plane and the handrail is fixed and the points measured from the handrail are within the plane outliers, the distance between the plane and the handrail points must be between l_H and $l_H + b_D$, which is defined as the gap distance. The gap distance set, $\mathrm{C}_{\mathrm{Gap}}$, which contains the points that are within the range of the gap distance, is defined as

$$\mathrm{C}_{\mathrm{Gap}} = \{\mathbf{p} | l_H \leq d_g(\mathbf{p}) \leq l_H + b_D, \mathbf{p} \in \mathrm{C} \backslash \mathrm{C}_{\mathrm{Plane}}\} \quad (1)$$

where $\mathbf{p} = [p_x \quad p_y \quad p_z]^{\mathrm{T}}$ is the 3-D position of the measured point to the depth sensor frame, C is the set of all

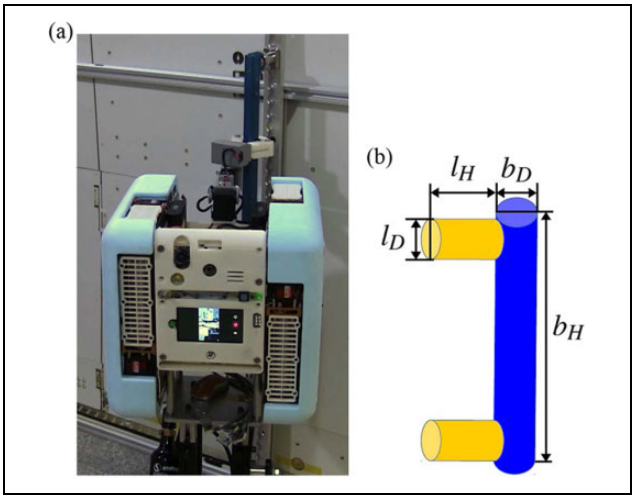

Figure 6. (a) Astrobee prototype perching on a handrail. (b) The handrail is modeled as a combination of one long cylinder (handrail beam) and two short cylinders (handrail legs).

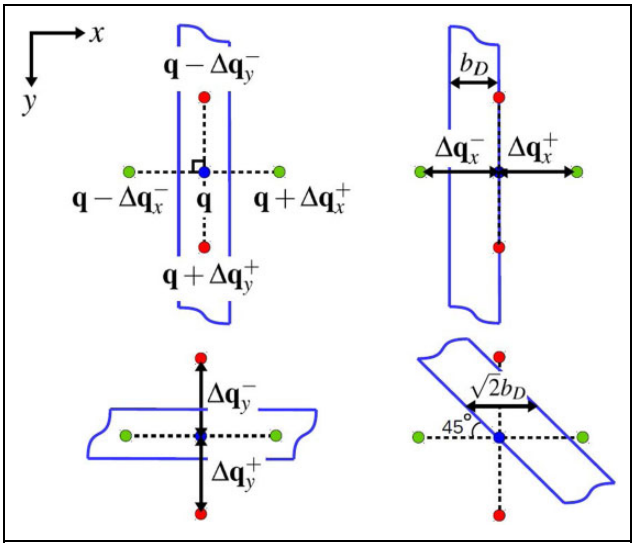

Figure 7. The illustration of the cross point test on the 2-D depth image. The top-left and top-right figures show that the *x*-axis cross points, $\mathbf{q}+\Delta\mathbf{q}_x^+$ and $\mathbf{q}-\Delta\mathbf{q}_x^-$, are on the wall if the testing point, *q*, is on the handrail. Likewise, the bottom-left figure shows that the *y*-axis cross points, $\mathbf{q}+\Delta\mathbf{q}_y^+$ and $\mathbf{q}-\Delta\mathbf{q}_y^-$, are on the wall if *q* is on the handrail. In the case of the bottom-right figure, all the cross points are on the wall.

points in the measured point cloud from the depth sensor, and $d_g(\mathbf{p})$ is the gap distance between the estimated plane and $\mathbf{p}$.

Cross point test

The handrail beam is projected as a thick line in a 2-D depth image. In the cross point test, four cross points are defined for each testing point in the 2-D depth image and tested whether they satisfy the geometric relationship between the handrail and the wall. Figure 7 shows the four cross points of the testing point in the 2-D depth image where $\mathbf{q}$ is the corresponding 2-D image point of the 3-D testing point, $\mathbf{p}$, $\mathbf{q}+\Delta\mathbf{q}_x^+$ and $\mathbf{q}-\Delta\mathbf{q}_x^-$ are the two *x*-axis cross points in the depth image, and $\mathbf{q}+\Delta\mathbf{q}_y^+$ and $\mathbf{q}-\Delta\mathbf{q}_y^-$ are the two *y*-axis cross points in the depth image. As shown in Figure 7, either the two *x*-axis cross points or the two *y*-axis cross points should be located on the wall if the testing point is located on the surface of the handrail beam for any handrail beam angles.

The displacement vectors, $\Delta\mathbf{q}_x^+$, $\Delta\mathbf{q}_x^-$, $\Delta\mathbf{q}_y^+$, and $\Delta\mathbf{q}_y^-$ are defined as

$$\begin{aligned}
\Delta\mathbf{q}_x^+ &= (\max(d_c\cos(\theta) - (l_H + d_c)\sin(\theta), d_c)/p_z)\mathbf{u}_x \\
\Delta\mathbf{q}_x^- &= (\max(d_c\cos(\theta) + (l_H + d_c)\sin(\theta), d_c)/p_z)\mathbf{u}_x \\
\Delta\mathbf{q}_y^+ &= (\max(d_c\cos(\phi) - (l_H + d_c)\sin(\phi), d_c)/p_z)\mathbf{u}_y \\
\Delta\mathbf{q}_y^- &= (\max(d_c\cos(\phi) + (l_H + d_c)\sin(\phi), d_c)/p_z)\mathbf{u}_y
\end{aligned} \tag{2}$$

where

$$\theta = \tan^{-1}\left(\frac{\alpha_x}{|\alpha_z|}\right), \phi = \tan^{-1}\left(\frac{\alpha_y}{|\alpha_z|}\right), d_c = \sqrt{2}b_D + \delta(\delta = b_D)$$

where α_x, α_y, and α_z are the plane coefficients in $\alpha = [\alpha_x \quad \alpha_y \quad \alpha_z \quad \alpha_w]$, α is the plane coefficient vector estimated from the points in $\mathrm{C}_{\mathrm{Plane}}$, p_z is the *z* value of the testing point $\mathbf{p}$, and $\mathbf{u}_x$ and $\mathbf{u}_y$ are the unit vectors of *x* and *y* axes of the 2-D depth image frame, respectively. In order to avoid selecting the cross points on the handrail leg, the displacement vectors are the functions of l_H, θ, and ϕ. If both θ and ϕ are zero, all the displacement vectors are set to d_c.

The set of points that pass the cross point test, $\mathrm{C}_{\mathrm{Cross}}$, is then defined as

$$\mathrm{C}_{\mathrm{Cross}} = \mathrm{C}_{x,\mathrm{Cross}} \cup \mathrm{C}_{y,\mathrm{Cross}} \tag{3}$$

where

$$\begin{aligned}
\mathrm{C}_{x,\mathrm{Cross}} &= \{\mathbf{p} | d_g(\mathbf{p}_x^+) < d_{g,\min} \wedge d_g(\mathbf{p}_x^-) < d_{g,\min}, \mathbf{p} \in \mathrm{C}_{\mathrm{Gap}}\} \\
\mathrm{C}_{y,\mathrm{Cross}} &= \{\mathbf{p} | d_g(\mathbf{p}_y^+) < d_{g,\min} \wedge d_g(\mathbf{p}_y^-) < d_{g,\min}, \mathbf{p} \in \mathrm{C}_{\mathrm{Gap}}\}
\end{aligned}$$

where $d_{g,\min}$ is the threshold gap distance, $\mathbf{p}_x^+$, $\mathbf{p}_x^-$, $\mathbf{p}_y^+$, and $\mathbf{p}_y^-$ are the corresponding 3-D cross points of $\mathbf{q}+\Delta\mathbf{q}_x^+$, $\mathbf{q}-\Delta\mathbf{q}_x^-$, $\mathbf{q}+\Delta\mathbf{q}_y^+$, and $\mathbf{q}-\Delta\mathbf{q}_y^-$, respectively. The sets $\mathrm{C}_{x,\mathrm{Cross}}$ and $\mathrm{C}_{y,\mathrm{Cross}}$ represent the set of points that their *x*-axis cross points and *y*-axis cross points are close to the plane, respectively.

Side width test

There are various types of cables scattered around the handrails on the ISS as shown in Figure 2. Since the cables have

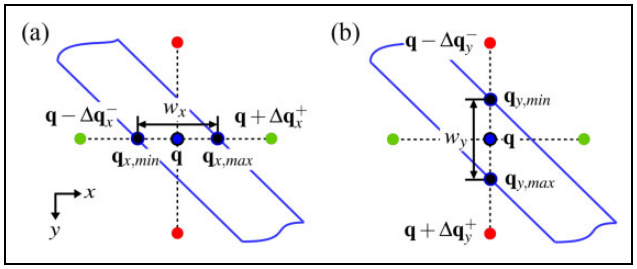

Figure 8. The illustration of the side width test on the 2-D depth image. (a) The two edge points, $\mathbf{q}_{x,\min}$ and $\mathbf{q}_{x,\max}$, are used to measure the side width in the *x*-axis direction. (b) The two edge points, $\mathbf{q}_{y,\min}$ and $\mathbf{q}_{y,\max}$, are used to measure the side width in the *y*-axis direction.

Algorithm 1. Handrail estimation algorithm.

1: $\mathrm{C}_{\mathrm{Line}} = \varnothing$
2: $[\ \mathrm{C}_1 \quad \mathrm{C}_2\ ] = \mathit{CandidateEsti}\ (\mathrm{C}_{\mathrm{Side}})$
3: **for** $n = 1$ to 2 **do**
4: $\quad [\ \mathrm{C}_n \quad \beta_n\ ] = \mathit{Cluster}(\mathrm{C}_n, d_r)$
5: $\quad$ **if** $b_H(\mathrm{C}_n) < b_{H,\min} \vee b_H(\mathrm{C}_n) > b_{H,\max}$ **then**
6: $\quad\quad \mathrm{C}_n = \varnothing$
7: $\quad$ **end if**
8: **end for**
9: **if** $\mathrm{C}_1 \neq \varnothing \vee \mathrm{C}_2 \neq \varnothing$ **then**
10: $\quad n = \underset{n\in\{1,2\}}{\operatorname{argmin}}\left(|\psi_p - \tan^{-1}\left(\frac{|\beta_{nx}|}{|\beta_{ny}|}\right)|\right)$
11: $\quad \mathrm{C}_{\mathrm{Line}} = \mathrm{C}_n$
12: **end if**

similar shape to the handrails, some portion of the points on the cables could pass both the gap distance test and the cross point test, which could be considered as the false-positive handrail points. Since the thickness or width of the cables are different from that of the handrail, the side width test evaluates whether the width of the local neighborhood of the testing point is within the range of the handrail width.

In order to filter out the points from the cables, the *x*- and *y*-axes direction side width tests are applied to the points in $\mathrm{C}_{x,\mathrm{Cross}}$ and $\mathrm{C}_{y,\mathrm{Cross}}$, respectively. Figure 8 shows the process of the side width test in a 2-D depth image. In the case of the *x*-axis direction side width test, the two edge points of $\mathbf{q}$ alongside with the *x*-axis direction, denoted as $\mathbf{q}_{x,\min}$ and $\mathbf{q}_{x,\max}$, are calculated for each point in $\mathrm{C}_{x,\mathrm{Cross}}$ as shown in Figure 8(a). Likewise, the *y*-axis direction side width test calculates the two edge points, $\mathbf{q}_{y,\min}$ and $\mathbf{q}_{y,\max}$, of each point in $\mathrm{C}_{y,\mathrm{Cross}}$ as shown in Figure 8(b). Then the side width set, $\mathrm{C}_{\mathrm{Side}}$, is defined as

$$\mathrm{C}_{\mathrm{Side}} = \mathrm{C}_{x,\mathrm{Side}} \cup \mathrm{C}_{y,\mathrm{Side}} \tag{4}$$

where

$$\mathrm{C}_{x,\mathrm{Side}} = \{\mathbf{p} | b_D - \delta < |\mathbf{p}_{x,\max} - \mathbf{p}_{x,\min}| < d_c, \mathbf{p} \in \mathrm{C}_{x,\mathrm{Cross}}\}$$
$$\mathrm{C}_{y,\mathrm{Side}} = \{\mathbf{p} | b_D - \delta < |\mathbf{p}_{y,\max} - \mathbf{p}_{y,\min}| < d_c, \mathbf{p} \in \mathrm{C}_{y,\mathrm{Cross}}\}$$

where $\mathbf{p}_{x,\min}$, $\mathbf{p}_{x,\max}$, $\mathbf{p}_{y,\min}$, and $\mathbf{p}_{y,\max}$ are the corresponding 3-D points of the 2-D depth image points $\mathbf{q}_{x,\min}$, $\mathbf{q}_{x,\max}$, $\mathbf{q}_{y,\min}$, and $\mathbf{q}_{y,\max}$ in Figure 8, respectively.

Line estimation

The handrail is further modeled as a line such that the line estimation process estimates the group of points that are within a threshold distance of the estimated line model. If there is a single handrail within a depth sensor view area, a basic RANSAC algorithm can be applied to estimate a line regardless of its angle with respect to the angle of the gripper. In the case of multiple handrails in the view area, the RANSAC algorithm simply selects one line model that has more inlier points than other line models. However, for the purpose of handrail grasping, the line model that has smaller angle with the handrail gripper must be selected as the line model of the target handrail. As shown in Figure 2, the handrails are either parallel or perpendicular to each other. The proposed line estimation algorithm utilizes this geometric relationship between the handrails by estimating the two most salient set of points that are perpendicular to each other and selecting the one that has smaller angle difference between its line vector and the gripper angle of the perching arm.

Algorithm 1 describes the overall handrail estimation process where *CandidateEsti* $(\mathrm{C}_{\mathrm{Side}})$ is the handrail candidate estimation algorithm (algorithm 2) which returns two sets, C_1 and C_2, of the estimated lines that are perpendicular to each other. The clustering function in line 4, denoted as $\mathit{Cluster}(\mathrm{C}_n, d_r)$, clusters points in C_n that the distance between the points are within the cluster distance, d_r, and returns the clustered set, C_n, and its line coefficients, β_n. Since the height of the handrail beam is fixed, the height of the estimated line in C_n, denoted as $b_H(\mathrm{C}_n)$, is compared with the minimum handrail beam height, $b_{H,\min}$, and the maximum handrail beam height, $b_{H,\max}$, as shown in line 5. If either C_1 or C_1 is not an empty set, the one which has the minimum angle difference between the gripper angle, ψ_p, is selected as the final set for the handrail, $\mathrm{C}_{\mathrm{Line}}$, as shown in lines 10 and 11.

Algorithm 2 shows the handrail candidate estimation process that estimates two perpendicular line models and returns the two inlier sets of the line models. The algorithm first samples two random points from $\mathrm{C}_{\mathrm{Side}}$ and generates a sample line vector, β'. The inliers that lie on the line with a point $\mathbf{p}'_1$ and a vector β' are collected in C' as shown in lines 5 through 12. The angle of the new line, ψ', is calculated from β' and compares which set is close to C' by checking the angle difference between ψ' and their previous angles, which are denoted as ψ_1 and ψ_2. If C' is close to C_n for $n \in \{1, 2\}$ and has more points than C_n, the set and the angle are updated by the new ones

Algorithm 2. Handrail candidates estimation algorithm.

1: **function** *CandidateEsti*(C_{Side})
2: $C_1, C_2 = \varnothing, \psi_1, \psi_2 = 0$
3: **for** N iterations **do**
4: Set $C' = \varnothing$ and randomly sample $\mathbf{p}_1'$ and $\mathbf{p}_2'$ from C_{Side}
5: **for** $\forall \mathbf{p}_i \in C_{Side}$ **do**
6: **if** $\frac{|(\mathbf{p}_1' - \mathbf{p}_2') \times (\mathbf{p}_2' - \mathbf{p}_i)|}{|\mathbf{p}_1' - \mathbf{p}_2'|} < d_{g,\min}$ **then**
7: $C' = C' \cup \{\mathbf{p}_i\}$
8: **end if**
9: **end for**
10: $\psi' = \tan^{-1}\left(\frac{|\beta'_x|}{|\beta'_y|}\right)$ where $\beta' = \frac{\mathbf{p}_1' - \mathbf{p}_2'}{|\mathbf{p}_1' - \mathbf{p}_2'|}$
11: $n = \underset{n \in \{1,2\}}{\text{argmin}}(|\psi' - \psi_n|), m = 3 - n$
12: **if** $|C'| > |C_n|$ **then**
13: $C_n = C', \psi_n = \psi'$
14: **if** $|C_n| > |C_m|$ **then**
15: $\psi_m = \pi/2 - \psi_n$
16: **end if**
17: **end if**
18: **end for**
19: return $[\, C_1 \quad C_2 \,]$
20: **end function**

as shown in line 16. In order to estimate two perpendicular line models, the dominant set, which has more points than the other, is selected first and then the angle of the nondominant set is calculated by subtracting the angle of the dominant set from $\pi/2$ as shown in lines 17 through 19. By updating the angle of the nondominant set based on the angle of the dominant set, the handrail candidate estimation algorithm is able to estimate the most salient perpendicular lines.

Localization

The overview of the EKF-based localization for Astrobee using a single camera was described by Coltin et al.[5] In order to maintain the same EKF localization process for the depth sensor observations, a collection of depth sensor features from the handrail and the wall behind the handrail is used. The difference of the measurement models between the sparse map landmarks and the depth sensor observations is that the residuals of the sparse map landmarks are fixed to two dimensions, while the dimensions of the residuals from the depth sensor observations change depending on the state of the handrail.

Handrail pose estimation

The origin of the handrail frame in the camera frame, ${}^{C}\mathbf{p}_H$, is calculated by selecting the medians of x-, y-, and z-axes values of the points in M_L. The unit vectors of the handrail frame, $\mathbf{u}_x$, $\mathbf{u}_y$, and $\mathbf{u}_z$, in the camera frame are defined as

$$\begin{aligned} \mathbf{u}_x &= \mathbf{v}_P/|\mathbf{v}_P| \\ \mathbf{u}_y &= \mathbf{u}_z \times \mathbf{u}_x \\ \mathbf{u}_z &= \frac{(\mathbf{v}_L - (\mathbf{v}_L \cdot \mathbf{v}_P)\mathbf{v}_P)}{(|\mathbf{v}_L - (\mathbf{v}_L \cdot \mathbf{v}_P)\mathbf{v}_P|)} \\ \mathbf{C}({}^{H}_{C}\bar{\mathbf{q}})^{\mathrm{T}} &= [\, \mathbf{u}_x \quad \mathbf{u}_y \quad \mathbf{u}_z \,] \end{aligned} \tag{5}$$

where ${}^{C}_{H}\bar{\mathbf{q}}$ is the unit quaternion describing the rotation from the handrail frame to the camera frame, and $\mathbf{C}({}^{C}_{H}\bar{\mathbf{q}})$ is the rotational matrix corresponding to ${}^{C}_{H}\bar{\mathbf{q}}$. Note that the measured point cloud from the handrail often shows curved shape due to the handrail surface characteristics and the measurement noise, and it causes inaccurate 3-D handrail pose estimation. For the robust estimation, we utilize two geometric characteristics of the handrail, that is, the handrail is a straight pole shape and the plane is parallel to the handrail by defining $\mathbf{u}_z$ as the projected vector of $\mathbf{v}_L$ to the plane.

Measurement model

The measurement of feature j in the i th camera frame, $\mathbf{z}_i^{(j)}$, and its expected value, $\hat{\mathbf{z}}_i^{(j)}$, are defined as

$$\begin{aligned} \mathbf{z}_i^{(j)} &= {}^{C_i}\mathbf{p}_{l_j} + \mathbf{n}_i^{(j)} \\ \hat{\mathbf{z}}_i^{(j)} &= {}^{C_i}\hat{\mathbf{p}}_{l_j} = \mathbf{C}({}^{C_i}_{G}\hat{\bar{\mathbf{q}}})({}^{G}\mathbf{p}_{l_j} - {}^{G}\hat{\mathbf{p}}_{C_i}) \end{aligned} \tag{6}$$

where ${}^{C_i}\mathbf{p}_{l_j}$ is the position of the feature j from the i th camera frame and $\mathbf{n}_i^{(j)}$ is the 3×1 measurement noise vector. The estimated quaternion and position of the i th camera pose in the global frame, ${}^{C_i}_{G}\hat{\bar{\mathbf{q}}}$ and ${}^{G}\hat{\mathbf{p}}_C$, are calculated as

$$\begin{aligned} {}^{C_i}_{G}\hat{\bar{\mathbf{q}}} &= {}^{C_i}_{B}\bar{\mathbf{q}} \otimes {}^{B}_{G}\hat{\bar{\mathbf{q}}} \\ {}^{G}\hat{\mathbf{p}}_C &= {}^{G}\hat{\mathbf{p}}_B + \mathbf{C}({}^{B}_{G}\hat{\bar{\mathbf{q}}})^{\mathrm{T}}{}^{B}\mathbf{p}_C \end{aligned} \tag{7}$$

where the feature positions in the global frame and the handrail frame, ${}^{G}\mathbf{p}_{l_j}$ and ${}^{H}\mathbf{p}_{l_j}$, are calculated as

$$\begin{aligned} {}^{G}\mathbf{p}_{l_j} &= {}^{G}\mathbf{p}_H + \mathbf{C}({}^{H}_{G}\bar{\mathbf{q}})^{\mathrm{T}}{}^{H}\mathbf{p}_{l_j} \\ {}^{H}\mathbf{p}_{l_j} &= {}^{H}\mathbf{p}_{C_i} + \mathbf{C}({}^{H}_{C_i}\bar{\mathbf{q}})^{\mathrm{T}}{}^{C_i}\mathbf{p}_{l_j} \end{aligned} \tag{8}$$

In the case where either of the end points of the handrail is detected, the residual of feature j in the i th camera frame, $\mathbf{r}_i^{(j)} = \mathbf{z}_i^{(j)} - \hat{\mathbf{z}}_i^{(j)}$, is linearized as

$$\mathbf{r}_i^{(j)} \simeq \mathbf{H}_{\delta\theta_i}^{(j)}\delta\theta_i + \mathbf{H}_{\mathbf{p}_i}^{(j)}{}^{G}\tilde{\mathbf{p}}_{C_i} \tag{9}$$

where

$$\begin{aligned} \mathbf{H}_{\delta\theta_i}^{(j)} &= \lfloor \mathbf{C}({}^{C_i}_{G}\hat{\bar{\mathbf{q}}})({}^{G}\mathbf{p}_{l_j} - {}^{G}\hat{\mathbf{p}}_{C_i}) \times \rfloor \\ \mathbf{H}_{\mathbf{p}_i}^{(j)} &= -\mathbf{C}({}^{C_i}_{G}\hat{\bar{\mathbf{q}}}) \end{aligned} \tag{10}$$

are Jacobians of $\delta\theta_i$ and ${}^{G}\tilde{\mathbf{p}}_{C_i}$, respectively.

As the robot approaches close to the handrail, none of the handrail end points would leave the field of view of the sensor so that the relative position of the robot along the handrail line vector, $\mathbf{u}_z$ in equation (5), is undetermined. In this case, the measured feature positions in the camera frame are represented in the handrail frame and then the z-axis value is dropped from the residuals. The measurement of feature j of the i th camera image in the handrail frame is defined as

$$\mathbf{z}_i^{'(j)} = {}^H\mathbf{p}_{l_j} + \mathbf{n}_i^{'(j)} \tag{11}$$

The linearized residual for the new measurement is then calculated as

$$\mathbf{r}_i^{'(j)} \simeq \mathbf{H}_{\delta\theta_i}^{'(j)}\delta\theta_i + \mathbf{H}_{\mathbf{p}_i}^{'(j)}{}^G\tilde{\mathbf{p}}_{C_i} \tag{12}$$

where

$$\begin{aligned}\mathbf{H}_{\delta\theta_i}^{'(j)} &= [\mathbf{I}_2 \quad \mathbf{0}_{2\times 1}]\mathbf{C}({}^H_{C_i}\bar{\mathbf{q}})\mathbf{H}_{\delta\theta_i}^{(j)} \\ \mathbf{H}_{\mathbf{P}_i}^{'(j)} &= [\mathbf{I}_2 \quad \mathbf{0}_{2\times 1}]\mathbf{C}({}^H_{C_i}\bar{\mathbf{q}})\mathbf{H}_{\mathbf{P}_i}^{(j)}\end{aligned} \tag{13}$$

By multiplying 2×3 matrix, $[\mathbf{I}_2 \quad \mathbf{0}_{2\times 1}]$, the z-axis value, which is the same handrail axis value, in the residual is dropped.

Target pose estimation

On the ISS, the astronauts change the location of the handrails daily for their convenience. Thus, the locations of the handrails are not fixed in the ISS such that the pose of the handrail that the Astrobee is targeting should be estimated in a global frame. The center location of the handrail can be easily estimated if the depth sensor detects both ends of a handrail. However, since the viewing angle of the depth sensor is limited to $62^\circ \times 45^\circ$ (horizontal angle $\times$ vertical angle), both ends of the handrail cannot always be detected as the robot comes closer to the handrail for perching. In order to estimate the handrail target pose with limited viewing angle of the sensor, the end point detection algorithm first detects the end points of the handrail, and different strategies are used for transition between the handrail states, which are defined based on the number of handrail end points.

Since the perching arm is capable of grasping the vertical handrail, which is parallel to the z-axis of the Astrobee body frame, we consider the top end point and the bottom end point of the vertical handrail. In order to select the top end point and the bottom end point, the points in $\mathbf{S}_L$ that are represented to the robot frame, and then the one which has the highest z value is set to the top end point and the other one which has the lowest z value is defined as the bottom end point. Based on the number of detected end points, the four handrail states are defined as follows: NH for *no handrail found*, BE (Figure 9(a)) for *both end points of the handrail detected*, OE (Figure 9(b)) for *only one end point is detected*, and NE (Figure 9(c)) for *handrail found but none of the end points is detected*.

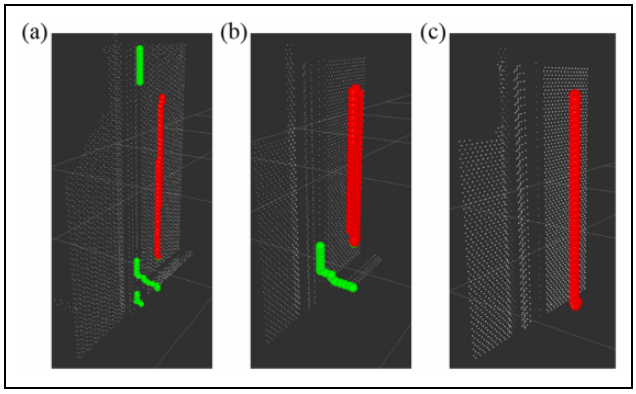

Figure 9. In (a), both ends of the handrail are detected (BE). In (b), only one end is detected (OE). In (c), the handrail is found but no end point is detected (NE).

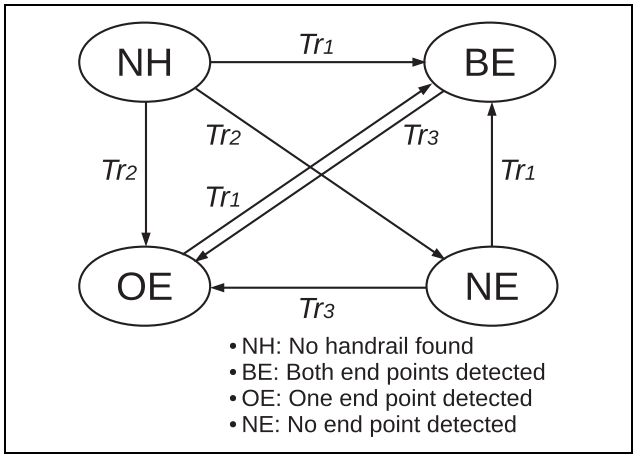

Figure 10. The transition of handrail state is based on the number of detected handrail end points.

The transition of handrail state is shown in Figure 10. There are three different strategies for three different transitions, Tr^1, Tr^2, and Tr^3. In the case of Tr^1, where both end points of the handrail are detected, the handrail center position is defined as the center position between the end points. In the case of Tr^2, where the handrail is not found in the previous state and either one or no end point is detected, the average value of the line inliers is used as the handrail center position. In the case of Tr^3, where the handrail center position is already determined in the previous state and only one end point is detected in the current state, the x and y values of the handrail center position in the global frame are updated by those of the handrail end point. This is based on the assumption that the handrail that we are interested in is vertical to the wall so that the x and y values of the top-end, center, and the bottom-end positions are the same in the global frame. For the rest of transitions that are not in the figure, the handrail center pose remains in its previous value.

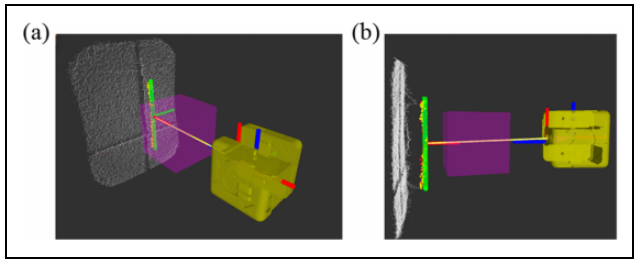

Figure 11. The visualization of the estimated handrail and the target pose of the robot. The green cylinder represents the estimated handrail and the semitransparent cube shows the target pose of the Astrobee for handrail perching.

The estimated handrail target pose is visualized as shown in Figure 11. The handrail target pose is defined by the handrail center position, $\mathbf{p}_o$, from algorithm 2, and the distance between the center of the robot and the end effector of the perching arm, d_c. The target position, $\mathbf{p}_g$, is then defined as $\mathbf{p}_g = \mathbf{p}_c + d_c\mathbf{v}_P$, where $\mathbf{v}_P$ is the normal vector of the plane. The target orientation is the same as the handrail orientation in the global frame. It is defined by three unit vectors, a line vector, $\mathbf{v}_L$, a normal vector of the plane, $\mathbf{v}_P$, and the cross product of them, $\mathbf{v}_o(\mathbf{v}_o = \mathbf{v}_L \times \mathbf{v}_P)$. The rotation matrix of the target is then defined as $\mathbf{R}_o = [\mathbf{v}_P\mathbf{v}_o\mathbf{v}_L]$.

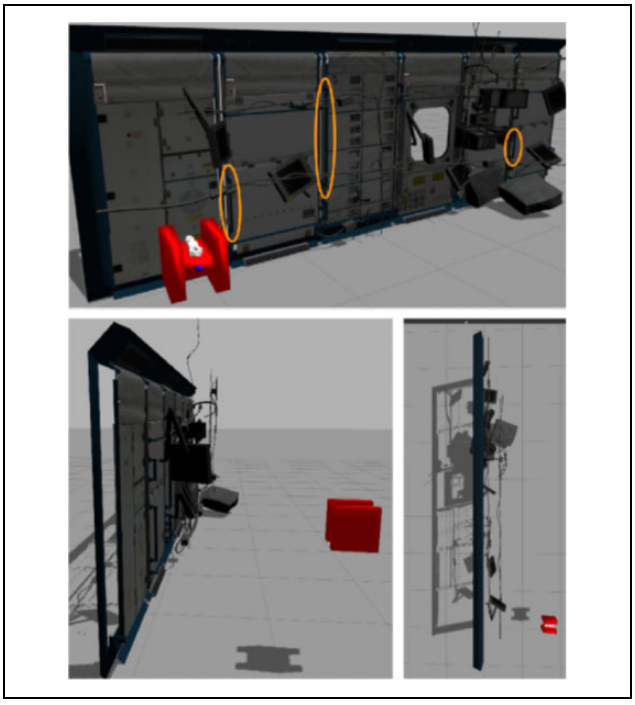

Figure 12. Simulation environment with ISS 3-D model and Gazebo simulator. ISS: International Space Station.

Experiments

Simulation results

The proposed algorithm is tested in a simulated ISS environment using Gazebo simulator to demonstrate the robust handrail detection and target pose estimation. As shown in Figure 12, cables and instruments, which may cause false-positive and false-handrail detection, are attached on the ISS wall.

In the simulation, the free-flyer moves laterally from left to right, facing the ToF depth sensor to the ISS wall (see supplementary video 1). Figure 13(a), (c), and (e) shows that the robot is located in front of the first, second, and third handrails, respectively. The results of handrail detection and the target pose estimation of the first, second, and the third handrails are visualized using Rviz visualization tool as shown in Figure 13(b), (d), and (f). In Rviz, the handrail is represented as a translucent green-colored cylinder with red-colored handrail inlier points inside. The side points of the handrail inlier points are represented as yellow-colored points. The blue-colored points near both ends of the handrail are the top point set and the bottom point set. The target pose of the robot is represented as a translucent purple cube. As shown in Figure 13(b), (d), and (f), the algorithm was able to robustly detect handrails and estimate target poses in a cluttered ISS environment.

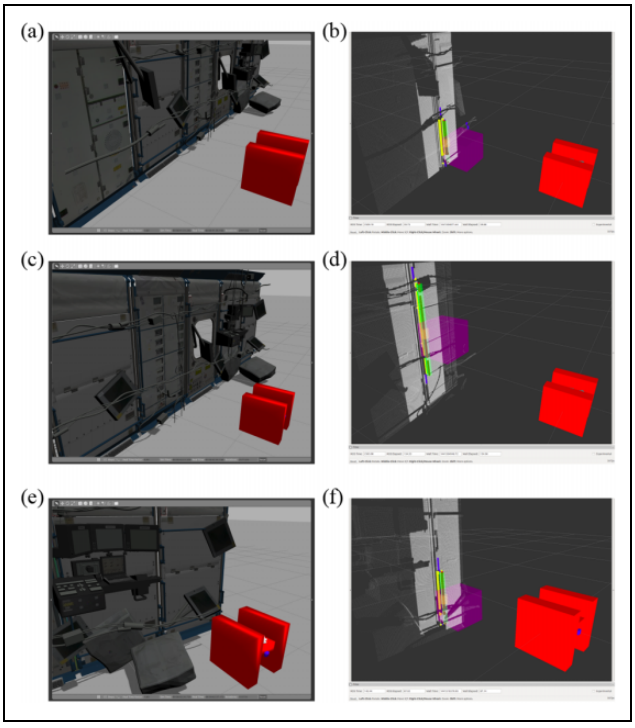

Figure 13. The robot encounters three handrails in the simulator as shown in (a), (c), and (e), and the proposed algorithm detects the corresponding handrails as shown in (b), (d), and (f).

Robust handrail pose estimation

In this experiment, the handrail pose estimation of the proposed algorithm was evaluated and compared with the basic algorithm, which estimates a plane first and then estimates a line from the plane outliers using a basic RANSAC algorithm. Figure 14 shows the target handrail

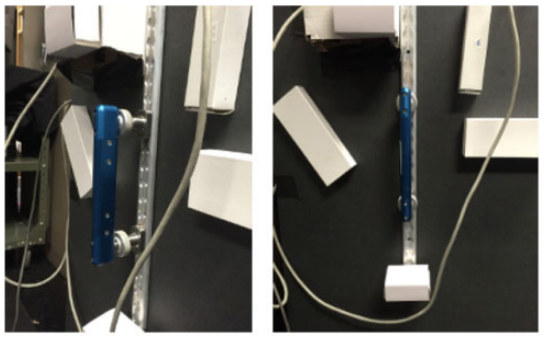

Figure 14. The experimental setup for the handrail pose estimation in static condition.

Table 2. Depth sensor configuration.

Test ID	*x* (cm)	*y* (cm)	*z* (cm)	*roll* (°)	*pitch* (°)	*yaw* (°)
1	60	0	0	0	0	0
2	60	0	0	30	0	0
3	60	15	0	0	0	30
4	60	15	0	30	0	30
5	60	0	15	0	0	30
6	60	0	15	30	0	30
7	60	15	15	0	30	30
8	60	15	15	30	30	30

attached on the wall and the other objects and cables are located around the handrail. For handrail pose estimation with different 3-D pose configurations, the experiment was conducted with eight different depth sensor poses as shown in Table 2 and 50 depth measurements were conducted for each pose.

Figure 15 compares the pose error for the basic and proposed algorithms, showing the mean and standard deviation across the eight different viewing geometries specified in Table 2. The result shows that the proposed algorithm has lower error for pose estimation as well as handrail beam height estimation than the basic algorithm. The large errors in *z* position and *roll* rotation of the basic algorithm were due to the objects above and below the handrail as well as the thin cables as shown in Figure 14 since the basic algorithm does not perform any of the handrail width checking process, side point testing process, and the interpolation process. On the other hand, the proposed algorithm was able to filter out the objects and cables around the handrail so that the 3-D pose of the handrail was robustly estimated.

Robot pose estimation based on handrail detection

In this experiment, the depth sensor was installed on the robot and the 2-D robot pose was estimated using the proposed algorithm while the robot moves around on the top of a microgravity simulating surface as shown in Figure 16

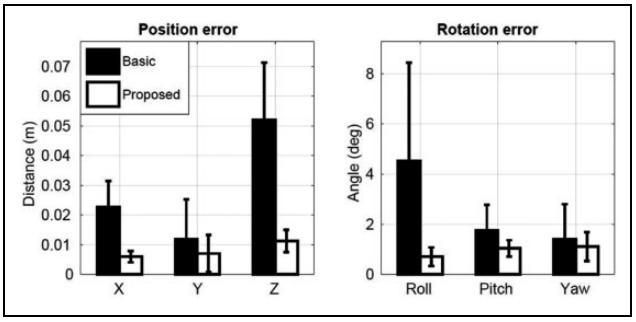

Figure 15. The mean and standard deviation across the eight different viewing geometries.

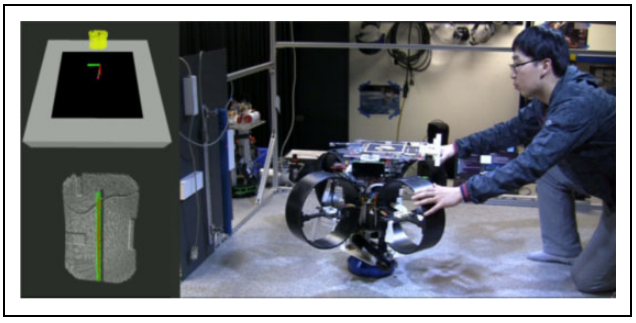

Figure 16. The experimental setup for the robot pose estimation in dynamic condition.

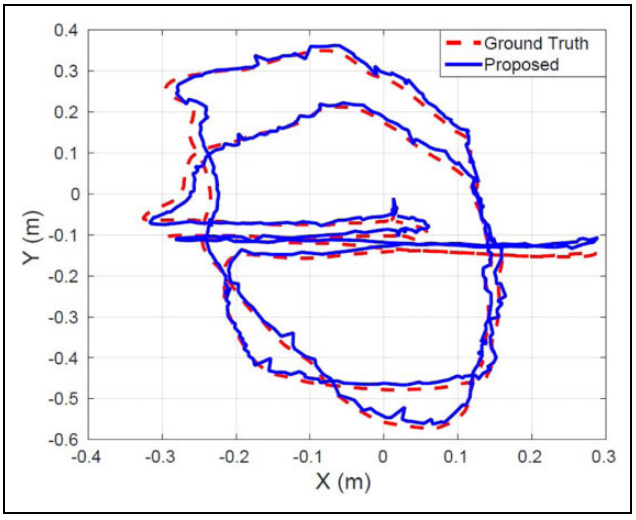

Figure 17. The experiment result of the robot pose estimation in 2-D environment.

(see supplementary video 2). The ground truth and the estimated 2-D position of the robot are shown in Figure 17 where the dotted line and the solid line represent the ground truth and the estimated positions of the robot, respectively. The mean and the standard deviation of the absolute error in the robot position estimation for the *x*-axis are 0.9 cm and 0.7, respectively, and for the *y*-axis are 1.7 cm and 1.2, respectively. Although the graph shows some

jitters due to the measurement noise of the point cloud as well as the handrail pose estimation error, the mean and the standard deviation of the absolute error in the robot position estimation are within the range of the perching arm gripper, which is about 3 cm. Thus, the proposed algorithm satisfies the handrail perching requirement.

Conclusion

In this article, we presented a robust handrail detection and pose estimation algorithm for handrail grasping of a free-flying robot. Since the proposed algorithm makes use of the geometric assumption that the handrails have cylinder types of shape and they are attached on a flat surface, it is capable of detecting and estimating various types of handrails with similar geometric constraints. The handrails on the ISS are the most challenging types among the other handrails since there are many cables and instruments attached together next to the handrails as well as varying lighting conditions. The proposed algorithm uses three tests, that is, gap distance test, cross point test, and side width test, to filter out false-positive handrail points and estimates the relative pose of the handrail to the robot body based on the estimated plane and line from the point cloud. The experimental results show that the proposed algorithm is able to filter out objects around the handrail and robustly estimate the handrail pose with various 3-D poses in static and dynamic conditions.

Future work includes the estimation of the target pose for the handrail perching and the state of the gripper on the handrail based on the kinematic model of the perching arm. The proposed algorithm will also be used for the EKF-based robot localization such that the Astrobee will use the handrail pose estimation as the measurement data of its pose during handrail perching process.

Acknowledgement

The authors would like to thank the Astrobee engineering team and the NASA Human Exploration Telerobotics 2 project for supporting this work.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was supported by the NASA Game Changing Development Program (Space Technology Mission Directorate), ISS SPHERES Facility (Human Exploration and Operations Mission Directorate), the Ministry of Science, ICT and Future Planning, Korea, under the Information Technology Research Center support program (IITP-2017-2014-0-00639) supervised by the Institute for Information and communications Technology Promotion (IITP), and the Kumoh National Institute of Technology.

Supplemental material

Supplementary material for this article is available online.

References

1. Bualat M, Barlow J, Fong T, et al. Astrobee: developing a free-flying robot for the international space station. In: *AIAA SPACE 2015 Conference and Exposition*, Pasadena, California, 31 August–September 2015, Vol. 4643, pp. 1–10. American Institute of Aeronautics and Astronautics.
2. Porter AK, Alinger DJ, Sedwick RJ, et al. Demonstration of electromagnetic formation flight and wireless power transfer. *J Spacecr Rockets* 2014; 51: 1914–1923.
3. Tweddle BE, Muggler E, Saenz-Otero A, et al. The SPHERES VERTIGO Goggles: vision based mapping and localization onboard the international space station. In: *Proceedings International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS)*, Turin, Italy, 4 September 2012, pp. 1–7.
4. Mantellato R, Lorenzini E, Sternberg D, et al. Simulation of a tethered microgravity robot pair and validation on a planar air bearing. *Acta Astronautica* 2016; 138: 579–589.
5. Coltin B, Fusco J, Moratto Z, et al. Localization from visual landmarks on a free-flying robot. In: *2016 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, Daejeon, South Korea, 9 October 2016, pp. 4377–4382. IEEE.
6. Collet A, Martinez M, and Srinivasa SS. The moped framework: object recognition and pose estimation for manipulation. *Int J Robot Res* 2011; 30(10): 1284–1306.
7. Hao Q, Cai R, Li Z, et al. Efficient 2D-to-3D correspondence filtering for scalable 3D object recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, Oregon, USA, 23 Jun 2013, pp. 899–906. IEEE.
8. Lo TWR and Siebert JP. Local feature extraction and matching on range images: 2.5 D SIFT. *Comput Vis Image Underst* 2009; 113(12): 1235–1250.
9. Rusu RB, Bradski G, Thibaux R, et al. Fast 3D recognition and pose using the viewpoint feature histogram. In: *2010 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, Taipei, Taiwan, 18 October 2010, pp. 2155–2162. IEEE.
10. Hua J, Lai Z, Dong M, et al. Geodesic distance-weighted shape vector image diffusion. *IEEE Trans Vis Comput Graph* 2008; 14(6): 1643–1650.
11. Lai K, Bo L, Ren X, et al. A large-scale hierarchical multi-view RGB-D object dataset. In: *2011 IEEE international conference on robotics and automation (ICRA)*, Shanghai, China, 9 May 2011, pp. 1817–1824. IEEE.
12. Choi C and Christensen HI. RGB-D object tracking: a particle filter approach on GPU. In: *2013 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, Tokyo, Japa, 3 November 2013, pp. 1084–1091. IEEE.
13. Gupta S, Girshick R, Arbeláez P, et al. Learning rich features from RGB-D images for object detection and segmentation. In: *European conference on computer vision*, Zurich, Switzerland, 6 September 2014, pp. 345–360. Springer.

14. Badger J, Hulse A, Taylor R, et al. Model-based robotic dynamic motion control for the robonaut 2 humanoid robot. In: *2013 13th IEEE-RAS international conference on humanoid robots (humanoids)*, Atlanta, USA, 15 October 2013, pp. 62–67. IEEE.
15. Ahlstrom TD, Diftler ME, Berka RB, et al. Robonaut 2 on the international space station: status update and preparations for IVA mobility. *AIAA SPACE 2013 Conference and Exposition*, San Diego, CA, 10 September 2013, pp. 1–14. American Institute of Aeronautics and Astronautics.
16. Rusu RB, Meeussen W, Chitta S, et al. Laser-based perception for door and handle identification. In: *ICAR 2009. International conference on advanced robotics, 2009*, Munich, Germany, 22 June 2009, pp. 1–8. IEEE.
17. Meeussen W, Wise M, Glaser S, et al. Autonomous door opening and plugging in with a personal robot. In: *2010 IEEE international conference on robotics and automation (ICRA)*, Anchorage, USA, 3 May 2010, pp. 729–736. IEEE.
18. Rühr T, Sturm J, Pangercic D, et al. A generalized framework for opening doors and drawers in kitchen environments. In: *2012 IEEE international conference on robotics and automation (ICRA)*, St Paul, USA, 14 May 2012, pp. 3852–3858. IEEE.
19. Bachrach A, Prentice S, He R, et al. RANGE–robust autonomous navigation in GPS-denied environments. *J Field Robot* 2011; 28(5): 644–666.
20. Shen S, Michael N, and Kumar V. Autonomous multi-floor indoor navigation with a computationally constrained MAV. In: *2011 IEEE international conference on robotics and automation (ICRA)*, Shanghai, China, 9 May 2011, pp. 20–25. IEEE.
21. Ghadiok V, Goldin J, and Ren W. Autonomous indoor aerial gripping using a quadrotor. In: *2011 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, San Francisco, USA, 25 September 2011, pp. 4645–4651. IEEE.
22. Park IW, Smith T, Wong S, et al. Developing a 3-DOF compliant perching arm for a free-flying robot on the international space station. In: *Proceedings IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, Munich, Germany, 24 August 2017, pp. 1135–1141. IEEE.
23. Fischler MA and Bolles RC. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun ACM* 1981; 24(6): 381–395.