Acta Astronautica 126 (2016) 298–311

Contents lists available at ScienceDirect

Acta Astronautica

journal homepage: www.elsevier.com/locate/aa

LIDAR-based relative navigation with respect to non-cooperative objects

John O. Woods [1], John A. Christian *

Department of Mechanical and Aerospace Engineering, West Virginia University, Mortantown, WV 26506, United States

ARTICLE INFO

Article history:
Received 21 October 2015
Accepted 7 May 2016
Available online 12 May 2016

Keywords:
Relative navigation
Spacecraft rendezvous
LIDAR
OUR-CVFH

ABSTRACT

Most navigation solutions which make use of LIDAR for proximity operations with respect to non-cooperative objects rely on the iterative closest point, or ICP, algorithm. For correct convergence, ICP requires a good initial guess as to the 6 degree-of-freedom relative pose of a client object. Some solutions require manual pose initialization; and template matching — refined by ICP — was recently demonstrated as an automated solution for initialization. Additionally, some have used the output of one ICP iteration as the initial guess for the next, which is inherently dangerous (since bad ICP poses are propagated forward in time by the filter, by ICP, or by both; and because it introduces measurement errors that are correlated with the *a priori* state errors). We demonstrate the use of a method borrowed from personal robotics, OUR-CVFH (for Oriented, Unique, and Repeatable Clustered Viewpoint Feature Histograms), for rendezvous with a tumbling object in low earth orbit as well as an asteroid in a heliocentric orbit. Our strategy requires no initial pose estimate, and refines OUR-CVFH results with ICP; we demonstrate its utility as part of a full navigation solution with a dual-state inertial extended Kalman filter.

© 2016 IAA Published by Elsevier Ltd. All rights reserved.

1. Introduction

Light detection and ranging (LIDAR) sensors are increasingly being considered as the primary relative navigation sensor for rendezvous with both satellites and natural objects, such as asteroids and comets. Early LIDARdevices — such as the laser altimeters aboard MUSES-C (Hayabusa) [1], and Hayabusa-2 [2] — were essentially laser altimeters, and only measured range to (or altitude above) a single point. In contrast, modern scanning and flash LIDAR sensors are capable of returning the range to many points on the surface of an object in the form of a three-dimensional (3D) point cloud [3]. Recently, a number of flash LIDAR sensors have been tested during rendezvous with the International Space Station [4,5] and are planned for future operational use on missions to asteroids [6]. Flash LIDAR sensors have become increasingly popular because they produce all of the range measurements at the same time and, consequently, are not subject to the same kind of motion blur artifacts as scanning systems. Of note, however, is that the speed of some modern scanning LIDAR sensors is such that this kind of motion blur may be negligible for most practical rendezvous, proximity operations, and docking (RPOD) operations. The methods discussed in this paper apply equally well to both flash and scanning LIDAR sensors, as well as any other sensor systems that produce 3D point clouds.

Pose is defined as the six degree-of-freedom (6 DOF) relative attitude and relative position between the spacecraft and an observed object. In relative navigation, pose provides the mapping from the body frame of the observed object to the spacecraft's body frame, consisting of a rotation and a translation.

Most solutions for LIDAR-based pose determination with respect to a non-cooperative object make use of the iterative closest point (ICP) algorithm [7] or variants thereof. ICP attempts to compute pose by 'registering' or aligning two point clouds by finding correspondences between the clouds: it transforms one cloud so as to minimize the distance between the corresponding points, and then repeats the procedure until no further improvements are achievable.

Early ICP implementations computed the error metric directly from the point-to-point correspondences. The point-to-plane strategy, however, finds point-to-point correspondences and then computes its error metric on the basis of the distance to the tangent plane of the corresponding point (e.g., [8]). Both strategies presuppose that perfect correspondences exist between the two point clouds, which — at least in the case of point clouds sampled from objects via LIDAR — is asymptotically improbable.

While ICP is always guaranteed to converge, it requires a reasonable initial guess in order to converge on the 'correct' pose. In many cases, the output of one ICP execution is used as the guess for the next execution (e.g., [9]), but as illustrated in Fig. 1 an erroneous initialization value can cause incorrect pose information to

* Corresponding author.
E-mail address: john.christian@mail.wvu.edu (J.A. Christian).
[1] He is now a Senior Development Engineer at Intuitive Machines in Houston, TX.

http://dx.doi.org/10.1016/j.actaastro.2016.05.007
0094-5765/© 2016 IAA Published by Elsevier Ltd. All rights reserved.

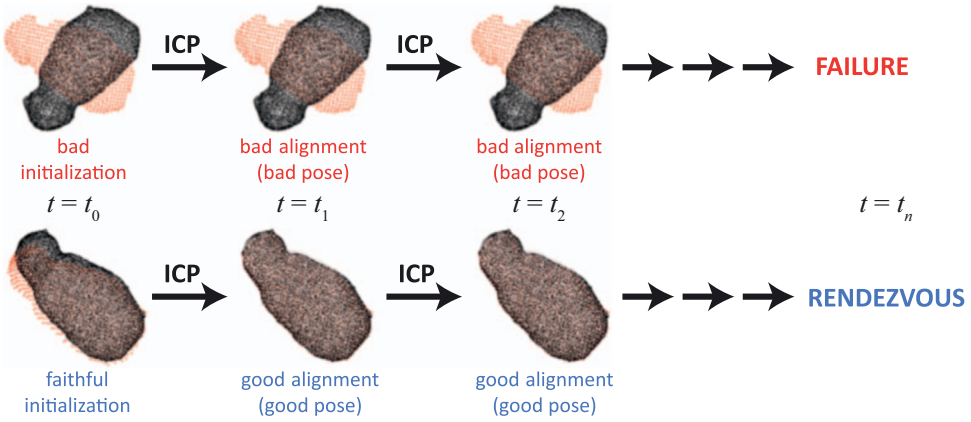

Fig. 1. ICP requires a reasonable initial guess in order to converge on a correct solution, and bad pose information can be propagated forward in a rendezvous. The challenge is illustrated using the asteroid 25143 Itokawa. In the top portion, an incorrect guess is provided, and ICP converges on a nearby local minimum rather than the absolute minimum (the correct pose). The bottom half of the figure shows initialization with an imperfect but near-correct initial guess, from which ICP is able to produce a correct pose, which serves as the initial guess for the next ICP call. In summary, a good initial guess is more likely to result in a successful rendezvous.

be propagated forward in a rendezvous — possibly creating a catastrophe. Pose initialization may also be assisted by the use of retroreflectors on a client object (termed *cooperative* rendezvous) [10,11], prior information from an attitude-only navigation filter (such as during launch), or by algorithmic means.

However, this manuscript focuses on the problem of non-cooperative rendezvous, such as with an asteroid or orbital debris, where the client object pose is not known and no navigation aids (fiducials such as reflectors) are available. A number of partial solutions exist, but thus far no generalizable approach — applicable for both natural and artificial objects — has been presented. Template matching, a 2D image recognition technique, has been investigated for servicing ENVISAT [12], but requires a great deal of training; provides a limited 3 DOF (attitude only) solution; and may fail when objects are partially occluded or outside the sensor field-of-view. Additionally, the strategy presented by Opromolla et al. is not demonstrated in the context of a navigation filter.

In this manuscript, we discuss several improvements for non-cooperative pose determination, applicable to both natural and unnatural objects. First, we present a 6 DOF pose initialization strategy based on Oriented, Unique, and Repeatable Clustered Viewpoint Feature Histograms (OUR-CVFH) [13]; this approach has the same runtime complexity as ICP (in many cases it runs more quickly), provides free object recognition, and is robust to many types of occlusion. Secondly, we demonstrate the techniques in two approach scenarios using a dual inertial-state multiplicative extended Kalman filter (MEKF)

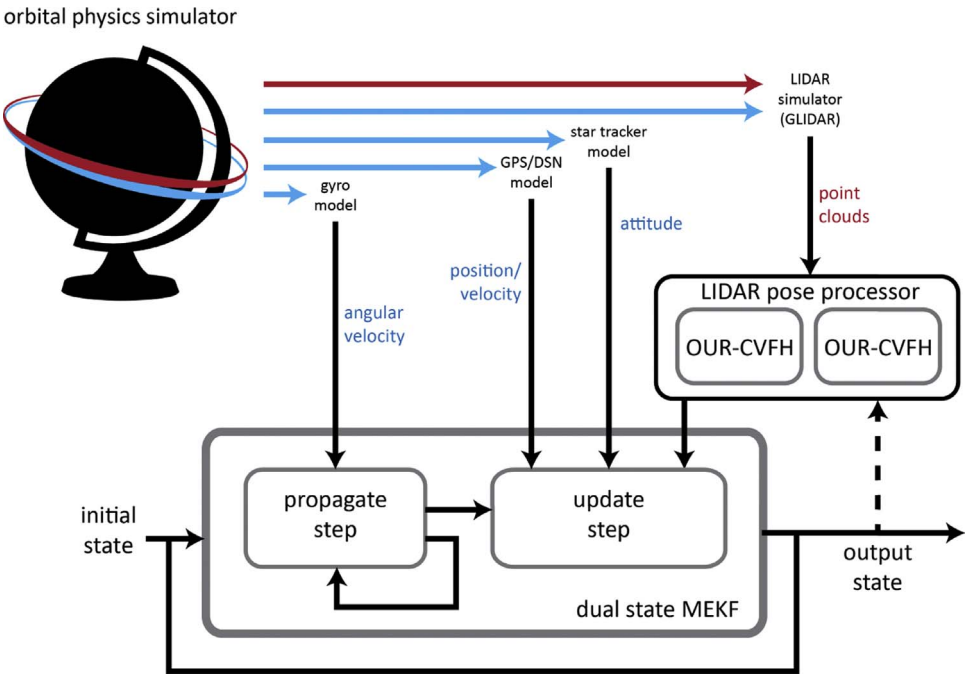

Fig. 2. The simulation consists of a physics simulator, which includes models for a gyroscope, a GPS or DSN receiver, a star tracker; a LIDAR simulator (GLIDAR), a pose processor, and a filter. Angular velocity measurements from the IMU's gyro are incorporated directly into the propagation, but other measurements are ingested via the update step. Blue arrows and fonts indicate information about the chaser vehicle; red represents the observed object, as well as sensor images thereof. The filter output state may be fed into the pose processor to provide initial guesses for ICP, but the primary initialization uses OUR-CVFH and requires no information from the filter. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

[14], and validate using Monte Carlo analyses.

2. Example system architecture

Fig. 2 shows the system architecture used in the present work. This configuration may be easily modified to include hardware in the loop; for example, a time-of-flight camera or Microsoft Kinect on a robot arm may be substituted (along with a different physics simulator).

In the present work, we assume that the navigation filter has access to measurements of the spacecraft's translational states, inertial attitude, and relative pose. In between updates from these sensors, the state is propagated through a dynamics model and angular velocity measurements from an inertial measurement unit (IMU).

The inertial translational states are provided by a Global Positioning System (GPS) receiver in low-Earth orbit scenarios and through a Deep Space Network (DSN) solution in deep space scenarios. In both cases, the measurement provided to the on-board navigation filter are assumed to be position and velocity of the spacecraft at a given time — sometimes referred to as position–velocity–time (PVT) measurements. For scenarios in low-Earth orbit, it is assumed that the GPS receiver will process the raw GPS pseudorange or carrier phase measurements to produce the desired PVT measurements. For the deep space applications, the raw DSN observables are assumed to be processed on the ground and the position and velocity are then uplinked to the spacecraft.

The inertial attitude is provided by a star tracker and the relative pose is provided by the OUR-CVFH/ICP solution generated from a LIDAR image.

Measurement models for each sensor and the nature of their implementation in a navigation filter is discussed in Section 4.4 and Appendix A.

3. Pose estimation with LIDARS using oriented, unique, and repeatable clustered viewpoint feature histograms (OUR-CVFH)

The OUR-CVFH algorithm is part of a family of methods based on *point feature histograms* (PFH [15,16]). These methods were developed for use in personal robotics, the idea being that a robot, sent to retrieve some object, could simultaneously

1. recognize the object amid clutter, and
1. determine its pose in order to grasp it effectively.

3.1. Point feature histogram space

In general, feature histograms attempt to describe a point cloud — a set of points sampled from a surface — in terms of surface features surrounding the sampled points. In the original method (PFH, for point feature histograms), for example, the feature space consists of concatenated histograms describing the pairwise pan, tilt, and yaw angles, and pairwise distance, between every pair of normals on a surface patch [15] (called a PFH descriptor hereafter). The FPFH descriptor is similar, but neglects the pairwise distance [16]. Each of these elements (pan, tilt, etc.), when present, is allotted 45 bins — a design decision which appears to be arbitrary, but which ultimately is dictated by its inclusion in the reference implementation, the Point Cloud Library (PCL).[2]

The viewpoint feature histogram (VFH, [17]) method contributed an additional component to the descriptor space: a 128-bin histogram quantizing the distribution of angles between the centroid–viewpoint vector and each normal vector. In the clustered variants, the centroid is computed for each cluster instead of for the entire object (a 128-bin histogram for CVFH [18] and a 64-bin histogram for OUR-CVFH [13]). This so-called VFH component is roll-invariant.

The clustered variants (CVFH and OUR-CVFH) also removed the roll-invariance, providing the clusters with reproducible orientations — hence, making the camera rotation observable. For OUR-CVFH, this portion consists of thirteen 8-bin histograms, each of which describes the distribution of Euclidean points in a given cluster through a SGURF, or *semi-global unique reference frame*, computed based on the spatial distribution of the clustered points [13].

Since OUR-CVFH ideally produces a single descriptor per cluster, the FPFH components (the roll, pitch, and yaw histograms) must be summed across all points in the cluster. In cases where SGURF orientations are ambiguous, each cluster may produce multiple features — one for each possible orientation.

Finally, there is a fallback option for views where no clusters of sufficient size are detectable. In training on an object, not only are OUR-CVFH descriptors computed; but also VFH descriptors for the entire view [18] (Aldoma et al. suggested including camera roll histograms, which could further improve performance with some relative client attitudes, but these were not implemented in the PCL 3D recognition pipeline, so we do not discuss them further). Thus, for sensor images encountered without viable OUR-CVFH descriptors, matches may be found from among the training data based on the full image.

3.2. Definitions

A first step in the OUR-CVFH procedure is the estimation of normals and curvature, generally determined by eigenvalue decomposition. The visible points are filtered, removing those with high curvature.

Next, points are clustered into *patches* based on their pairwise Euclidean distance and the angles between their normals; patches which are too small or too large are discarded. We describe a patch mathematically as a set of points $\{\mathbf{p}_i\}$ (identified through a k nearest neighbors search with respect to some query point) with centroid $\mathbf{p}_c$ and centroid of the normals $\mathbf{n}_c$, and with corresponding normals represented by $\{\mathbf{n}_i\}$.

For each point $\mathbf{p}_i$, the pan, tilt, yaw, and distance are computed as

$$\cos(\alpha) = \hat{\mathbf{v}}^\top \mathbf{n}_i \tag{1}$$

$$\cos(\phi) = \hat{\mathbf{u}}^\top \mathbf{r}_{ic} \tag{2}$$

$$\theta = \arctan\left(\hat{\mathbf{w}}^\top \mathbf{n}_i,\ \hat{\mathbf{u}}^\top \mathbf{n}_i\right) \tag{3}$$

$$d = \left\|\mathbf{p}_c - \mathbf{p}_i\right\|, \tag{4}$$

with

$$\mathbf{r}_{ic} = \frac{(\mathbf{p}_c - \mathbf{p}_i)}{d}$$

as the normalized radius from the query point to a patch point.

The basis $\langle \hat{\mathbf{u}}, \hat{\mathbf{v}}, \hat{\mathbf{w}} \rangle$ is a Darboux frame defined for a patch point $\mathbf{p}_i$ as

[2] The Point Cloud Library is an open-source C/C++ library for loading, processing, interpreting, and displaying 3D point cloud data. More information may be found online at http://pointclouds.org

$$\hat{\mathbf{u}} = \mathbf{n}_c$$
$$\hat{\mathbf{v}} = \frac{\mathbf{r}_{ic} \times \mathbf{n}_i}{\| \mathbf{r}_{ic} \times \mathbf{n}_i \|}$$
$$\hat{\mathbf{w}} = \hat{\mathbf{u}} \times \hat{\mathbf{v}}.$$

Once computed, m-bin histograms (arbitrarily defined with $m=45$ per [17]) are determined for each of the pairwise angles and inter-point distances, and these histograms are concatenated together to form what the authors call *simplified point feature histograms* (here, these are denoted as $\mathrm{S}(\mathbf{p})$) for each point $\mathbf{p}$.

Ideally, and in the original PFH technique, the angle (and distance) components were calculated pairwise for *all* points, not with respect to the centroid $\mathbf{p}_c$. However, to reduce the worst-case complexity from $O(nk^2)$ to $O(nk)$ (where n is the number of surface normals in the cloud), the full histogram for each point is approximated as

$$\mathrm{F}(\mathbf{p}_c) \approx \mathrm{S}(\mathbf{p}_c) + \frac{1}{k}\sum_{i=0}^{k-1}\frac{1}{w_{ic}}\mathrm{S}(\mathbf{p}_i), \tag{5}$$

where weighting w_{ic} is defined as the distance between $\mathbf{p}_c$ and $\mathbf{p}_i$. Additionally, Rusu et al. imply that the use of strategic caching of nearest neighbors reduces the complexity to $O(n)$. $\mathrm{F}(\mathbf{p})$ represents the FPFH descriptor, although the FPFH representation is not used in the PCL implementation of the OUR-CVFH descriptor.

Further, VFH, CVFH, and OUR-CVFH include an additional angle in the descriptor

$$\cos(\beta) = \mathbf{n}_i^{\top}\frac{\mathbf{p}_c}{\| \mathbf{p}_c \|} \tag{6}$$

These angles are split into a 128-bin histogram for VFH and CVFH, which is concatenated onto the 4×45-bin histogram already constructed from point–centroid angles and distances. Thus, for CVFH, the result is a 308-bin histogram for each patch — or, if no patches are found, a single, representative 308-bin VFH for the whole point cloud.

The final step is to remove the roll invariance by creating a 6-DOF descriptor that captures camera roll. CVFH does this by appending a 90-bin histogram of viewpoint-normal angles to its existing 308-bin histogram for the best n matches. OUR-CVFH takes a different approach. Instead of appending a histogram of viewpoint-normal angles, OUR-CVFH replaces the pairwise distance histogram (the fourth 45-bin histogram) and the first half of the 128-bin β histogram with the SGURF histogram (permissible because all normals are assumed to be oriented in the direction of the viewpoint). Between one and four SGURFS are computed for each cluster [13] on the basis of the distributions of points in the cluster, allowing the points to be binned into one of eight octants. Each octant is itself divided into 13 bins representing the distance from the reference frame's center, for a total of 104 bins. This results in a 303-bin OUR-CVFH histogram (consisting of a 45-bin α histogram, 45-bin ϕ histogram, 45-bin θ histogram, 104-bin SGURF histogram, and 64-bin β histogram). When multiple SGURFS are found, a separate 303-bin histogram is used to represent each one.

3.3. Training procedure

In order to perform recognition, OUR-CVFH must be trained on objects of interest. For this purpose, the Point Cloud Library includes a training procedure which encapsulates client objects in an icosahedron. A camera is then placed at each vertex (or on each face) of the icosahedron, oriented toward the center, and the object is rendered as a point cloud. OUR-CVFH is run on the cloud, producing at least a histogram for each view (ideally a histogram or histograms for each cluster, depending on how many SGURFS are found).

The PCL-provided procedure repeats four times (once for each possible right-angle camera rotation about the boresight), which may not be necessary given the roll disambiguation provided by the use of SGURFS. However, this procedure ran quickly and produced extremely small amounts of data — and as the search for matching histograms is such a small part of the OUR-CVFH method — we elected not to modify the training procedure.

4. The dual inertial state multiplicative extended Kalman filter

Rooted in Bayesian probability, a navigation filter is responsible for maintaining knowledge of the state (such as position) of one or more objects, represented as (1) the *expectation* of the distribution of possible states given *prior evidence* or prior information, and (2) the covariance of the states. When new evidence is available, a *posterior probability* may be calculated, represented in the form of an updated estimated state and covariance.

Since the objects of interest obey some set of physical dynamics, the filter can *propagate* its prior state and covariance as time passes; and the filter *updates* its knowledge using sensor measurements as those become available. The outputs of the update, then, are the posterior states and covariance. The last part of the update, which takes place after all new measurements are ingested, is called the *reset*. The reset relabels the posterior knowledge as prior knowledge in preparation for further propagation.

The filter design chosen here is a dual inertial state multiplicative extended Kalman filter. We now describe this filter architecture in detail, beginning with a review of the specific extended Kalman filter (EKF) framework used. This review is followed by a brief description of the "multiplicative" version of the EKF and a discussion of why this version is necessary for the present application. Then, finally, we describe the dual inertial state formulation and present all of the necessary equations for implementation.

4.1. Review of the extended Kalman filter (EKF)

In the present work, the navigation filter takes the form of an EKF, with conventions and implementation consistent with the approaches outlined by Gelb [19] and Maybeck [20].

Suppose we have a system whose state is described by $\mathbf{x}$ and whose evolution over time is governed by the following nonlinear dynamics:

$$\dot{\mathbf{x}} = f(\mathbf{x}, t) + \mathbf{w} \tag{7}$$

where $\mathbf{w}$ is zero mean white noise. Further suppose we can obtain some noisy measurements, $\tilde{\mathbf{y}}_k$, at time t_k which are related to the state by some nonlinear measurement model,

$$\tilde{\mathbf{y}}_k = h(\mathbf{x}_k) + \boldsymbol{\nu} \tag{8}$$

where $\boldsymbol{\nu}$ is zero mean Gaussian noise with covariance $\mathbf{R}$.

As mentioned above, typical navigation filters consist of a propagation step and an update step. Thus, within the EKF framework, we propagate the best estimate of the state, $\hat{\mathbf{x}}$, according to

$$\dot{\hat{\mathbf{x}}} = f(\hat{\mathbf{x}}, t) \tag{9}$$

and the corresponding state transition matrix (STM) by

$$\boldsymbol{\Phi}(t, t_0) = \mathbf{F}(t)\,\boldsymbol{\Phi}(t, t_0) \tag{10}$$

where $\boldsymbol{\Phi}(t_0, t_0) = \mathbf{I}$ and

$$\mathbf{F}(t) = \left.\frac{\partial f(\mathbf{x}, t)}{\partial \mathbf{x}}\right|_{\mathbf{x}=\hat{\mathbf{x}}}. \tag{11}$$

We use the STM to advance the state covariance from one measurement update time at t_k to the subsequent measurement update time at t_{k+1},

$$\mathbf{P}_{k+1} = \mathbf{\Phi}(t_{k+1}, t_k)\mathbf{P}_k\mathbf{\Phi}^T(t_{k+1}, t_k) + \mathbf{Q} \tag{12}$$

where $\mathbf{Q}$ describes the inflation of the state covariance due to process noise (which comes from $\mathbf{w}$).

The state estimate at t_k and its corresponding covariance may then be updated according to

$$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + \mathbf{K}_k\left(\tilde{\mathbf{y}}_k - h(\hat{\mathbf{x}}_k)\right) \tag{13}$$

$$\mathbf{P}_k^+ = \left(\mathbf{I} - \mathbf{K}_k\mathbf{H}_k\right)\mathbf{P}_k^-\left(\mathbf{I} - \mathbf{K}_k\mathbf{H}_k\right)^\top + \mathbf{K}_k\mathbf{R}\mathbf{K}_k^\top \tag{14}$$

where $\mathbf{H}_k$ is the measurement sensitivity matrix and $\mathbf{K}_k$ is the Kalman gain,

$$\mathbf{H}_k = \left.\frac{\partial h(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}=\hat{\mathbf{x}}_k} \tag{15}$$

$$\mathbf{K}_k = \mathbf{P}_k^-\mathbf{H}_k^\top\left(\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^\top + \mathbf{R}\right)^{-1}. \tag{16}$$

In order to accurately model a rendezvous between two objects in orbit — one actuated and with sensors, and the other non-cooperative and possibly tumbling — the filter must estimate the 6-DOF state (translational and rotational) for both the spacecraft and the observed object. In each case, the position portions of the state vector are propagated using the velocity estimate, and velocity is propagated via a dynamical model (here, gravitational acceleration, which depends on the object's position in the inertial frame). The attitude states require the introduction of the MEKF.

4.2. The multiplicative extended Kalman filter (MEKF)

Because we aim to estimate both translational and attitude states, we have implemented a multiplicative EKF (MEKF) [14,21] — where the attitude states undergo a multiplicative update and all other states undergo the standard EKF update.

The standard convention is to keep two quantities in the filter for each object's attitude: an attitude quaternion and, as part of the state vector, a three-parameter attitude error (e.g., [22]). The former is propagated, and the latter updated; the covariance describes the attitude error rather than the propagated attitude. At the end of each update, during the reset, the prior attitude is set equal to the quaternion product of the error and the posterior attitude, and the attitude error starts over at zero.

Just as position requires velocity for its propagation, attitude requires angular velocity. For the chaser spacecraft, the angular velocity used for propagation comes directly from the gyroscope, and a gyroscope bias term is propagated according to the dynamics

$$\dot{\mathbf{b}}_s = -\frac{1}{\tau_s}\mathbf{b}_s + \boldsymbol{\nu}_s, \tag{17}$$

where $\boldsymbol{\nu}_s$ is a zero-mean Gaussian white-noise process. The observed object's angular velocity — also included in the propagation — obeys torque-free rigid body dynamics

$$\dot{\boldsymbol{\omega}}_c = \mathbf{J}^{-1}\left(\mathbf{J}\boldsymbol{\omega}_c \times \boldsymbol{\omega}_c\right), \tag{18}$$

where $\mathbf{J}$ is the observed object's inertia tensor. The object's angular velocity is used to propagate its respective attitude quaternion.

We chose to implement a dual inertial state filter. That is, the filter will estimate the full 6 DOF inertial state of both the spacecraft and the observed object. In contrast, other filter formulations may choose to include relative states, rather than inertial states, in the filter.

4.3. Implementation of the dual inertial state MEKF

Putting all of the above components together, the 24×1 state vector for the present problem is defined as

$$\mathbf{x} = \begin{bmatrix}\mathbf{r}_s^\top & \dot{\mathbf{r}}_s^\top & \mathbf{a}_s^\top & \mathbf{b}_s^\top & \mathbf{r}_c^\top & \dot{\mathbf{r}}_c^\top & \mathbf{a}_c^\top & \boldsymbol{\omega}_c^\top\end{bmatrix}^\top. \tag{19}$$

where $\mathbf{r}_s$ is the spacecraft inertial position, $\mathbf{a}_s$ is the three-parameter representation to the spacecraft attitude error, $\mathbf{b}_s$ is the bias of the spacecraft's gyro, $\mathbf{r}_c$ is the observed object (or client object) inertial position, $\mathbf{a}_c$ is the three-parameter representation to the observed object's attitude error, and $\boldsymbol{\omega}_c$ is the observed object's angular velocity. The three-parameter attitude representations used here are described in Appendix A.1.

As discussed above, we propagate the state estimate by Eq. (9), where

$$\mathbf{F}(t) = \left.\frac{\partial f}{\partial \mathbf{x}}\right|_{\mathbf{x}=\hat{\mathbf{x}}} \tag{20}$$

$$\mathbf{F}(t) = \begin{bmatrix}\mathbf{F}_s & \mathbf{0}_{12\times 12} \\ \mathbf{0}_{12\times 12} & \mathbf{F}_c\end{bmatrix}. \tag{21}$$

The two components of $\mathbf{F}(t)$ are

$$\mathbf{F}_s = \begin{bmatrix}\mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} \\ G(\hat{\mathbf{r}}_s) & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & -\left[\hat{\boldsymbol{\omega}}_s \times\right] & -\mathbf{I} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0}\end{bmatrix} \tag{22}$$

$$\mathbf{F}_c = \begin{bmatrix}\mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} \\ G(\hat{\mathbf{r}}_c) & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & -\left[\hat{\boldsymbol{\omega}}_c \times\right] & \mathbf{I} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{J}^{-1}\left(\left[\mathbf{J}\hat{\boldsymbol{\omega}}_c \times\right] - \left[\hat{\boldsymbol{\omega}}_c \times\right]\mathbf{J}\right)\end{bmatrix}, \tag{23}$$

where $\hat{\boldsymbol{\omega}}_s$ is the spacecraft angular velocity and comes from the gyro, each $\mathbf{0}$ and $\mathbf{I}$ is 3×3, and the position-dependent gravity model is

$$G(\mathbf{r}) = -\frac{\mu}{r^3}\left(\mathbf{I} - \frac{3}{r^2}\mathbf{r}\mathbf{r}^\top\right). \tag{24}$$

The spacecraft gyro is assumed to follow the Farrenkopf model [23]:

$$\tilde{\boldsymbol{\omega}}_s = \boldsymbol{\omega}_s + \mathbf{b}_s + \boldsymbol{\nu}_g \tag{25}$$

where $\mathbf{b}_s$ is the gyro bias and $\boldsymbol{\nu}_g$ is the zero mean noise. Thus, the best estimate of the spacecraft's body rate is simply

$$\hat{\boldsymbol{\omega}}_s = \tilde{\boldsymbol{\omega}}_s - \hat{\mathbf{b}}_s \tag{26}$$

It is now possible to also propagate the STM, $\mathbf{\Phi}(t_{k+1}, t_k)$, according to Eq. (10). If the time step $(\Delta t = t_{k+1} - t_k)$ is relatively small, the STM propagation may be approximated using only the first few terms of a Taylor series expansion

$$\mathbf{\Phi}(t_{k+1}, t_k) \approx \mathbf{I} + \mathbf{F}\,\Delta t + \frac{1}{2}\mathbf{F}^2\,\Delta t^2 + \cdots \tag{27}$$

unless they can be analytically determined. The only term for which an analytic solution has been included is in sub-matrix (3, 4), which is the solution to the differential equation in Eq. (17),

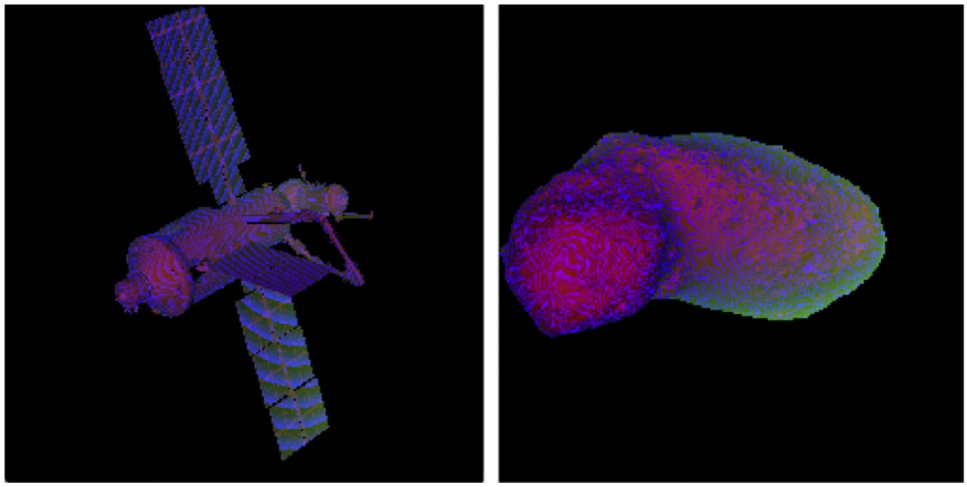

Fig. 3. An open-source 3D point cloud simulator called GLIDAR was used to generate simulated LIDAR images of the ISS FCB (left) and asteroid 25143 Itokawa (right). GLIDAR was developed by the authors of this paper to quickly simulate 3D point clouds of various objects [25]. It is implemented in C++ and the source code is available on GitHub at https://github.com/WVU-ASEL/glidar.

$$\dot{\mathbf{b}}_c(\Delta t) = \mathbf{I}\exp\left\{-\frac{\Delta t}{\tau_c}\right\}. \tag{28}$$

4.4. Measurement models

The filter admits three types of sensor measurements: pose in the sensor (LIDAR) frame, using ICP or OUR-CVFH; GPS/DSN-based positions and velocities in the inertial frame; and attitude measurements in the star tracker frame.

4.4.1. GPS and DSN

The GPS (or DSN) measurements are the most straightforward, since they are direct observations of states within the filter. This is because we do not assume that the filter processes the raw GPS or DSN observables, but instead simply processes the position–velocity–time (PVT) output of the GPS receiver or the equivalent product produced on the ground from a DSN tracking pass. GPS is used for Earth orbit applications and DSN is used for deep space applications. For simplicity, it is assumed that the GPS receiver or DSN antenna is at the center of the spacecraft. The measurement model is simply

$$\mathbf{y}_{\text{GPS}} = \mathbf{y}_{\text{DSN}} = \begin{bmatrix}\mathbf{r}_s\\ \dot{\mathbf{r}}_s\end{bmatrix} \tag{29}$$

Likewise, it is trivial to compute the corresponding measurement sensitivity matrix,

$$\mathbf{H}_{\text{GPS}} = \mathbf{H}_{\text{DSN}} = \begin{bmatrix}\mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0}\end{bmatrix}. \tag{30}$$

4.4.2. LIDAR

The LIDAR provides two measurements, a relative translation and a relative rotation, both in the sensor frame. We begin with the relative position of the observed object with respect to the LIDAR as expressed in the LIDAR frame, which is simply

$$\mathbf{p}_{C/L} = \mathbf{T}_L^S\left(\mathbf{T}_S^I(\mathbf{r}_c - \mathbf{r}_s) - \mathbf{p}_L\right) \tag{31}$$

where $\mathbf{T}_s^i$ is the rotation matrix from the inertial frame to the spacecraft body frame, $\mathbf{T}_l^s$ is the rotation from the spacecraft body frame to the LIDAR sensor frame, and $\mathbf{p}_l$ is the position of the LIDAR on the spacecraft. Assuming small angles, we can incorporate the attitude error states and write the measurement model as

$$\mathbf{p}_{C/L} = \mathbf{T}_L^S\left(\left(\mathbf{I} - \left[\mathbf{a}_s\times\right]\right)\hat{\mathbf{T}}_S^I(\mathbf{r}_c - \mathbf{r}_s) - \mathbf{p}_L\right) \tag{32}$$

Likewise, the relative attitude measurement from the LIDAR is the attitude quaternion describing the rotation from the observed object's body frame to the LIDAR sensor frame,

$$\bar{\mathbf{q}}_L^C = \bar{\mathbf{q}}_L^S \otimes \bar{\mathbf{q}}_S^I \otimes \bar{\mathbf{q}}_I^C, \tag{33}$$

where $\otimes$ is the quaternion multiplication operator and the superscript and the subscript convention on the attitude quaternions are the same as used for rotation matrices. Note that $\bar{\mathbf{q}}_s^i$ and $\bar{\mathbf{q}}_c^i$ are the two inertial attitudes from the filter and $\bar{\mathbf{q}}_l^s$ describes the alignment of the LIDAR on the spacecraft. A detailed discussion of the LIDAR attitude measurement model and the derivation of the measurement sensitivity matrix is provided in Appendix A.2.

Finally, if the LIDAR measurement vector is given by $\mathbf{y}_l^\top = [\mathbf{p}_{c/l}^\top\ \mathbf{a}_l^\top]$, then taking the partial derivatives of the above measurement models yields the following measurement sensitivity matrix:

$$\mathbf{H}_L = \begin{bmatrix}-\mathbf{T}_L^S\hat{\mathbf{T}}_S^I & \mathbf{0} & \mathbf{T}_L^S\left[\hat{\mathbf{T}}_S^I\left(\mathbf{r}_c - \mathbf{r}_s\right)\times\right] & \mathbf{0} & \mathbf{T}_L^S\hat{\mathbf{T}}_S^I & \mathbf{0} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \hat{\mathbf{T}}_L^S & \mathbf{0} & \mathbf{0} & \mathbf{0} & -\hat{\mathbf{T}}_L^S\hat{\mathbf{T}}_S^L\hat{\mathbf{T}}_I^c & \mathbf{0}\end{bmatrix}. \tag{34}$$

4.4.3. Star tracker

The star tracker provides an attitude quaternion describing the rotation from the inertial frame to the star tracker sensor frame, $\bar{\mathbf{q}}_t^i$. This measurement is given by

$$\bar{\mathbf{q}}_T^I = \bar{\mathbf{q}}_T^S \otimes \bar{\mathbf{q}}_S^I \tag{35}$$

where $\bar{\mathbf{q}}_s^i$ is the inertial attitude of the spacecraft (one of the states being estimated) and $\bar{\mathbf{q}}_t^s$ describes the orientation of the star tracker on the spacecraft. A detailed discussion of the star tracker measurement model and the corresponding derivation of the measurement sensitivity matrix is provided in Appendix A.3. The result is the following measurement sensitivity matrix for use in the MEKF:

$$\mathbf{H}_T = \begin{bmatrix}\mathbf{0} & \mathbf{0} & \hat{\mathbf{T}}_T^S & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0}\end{bmatrix}. \tag{36}$$

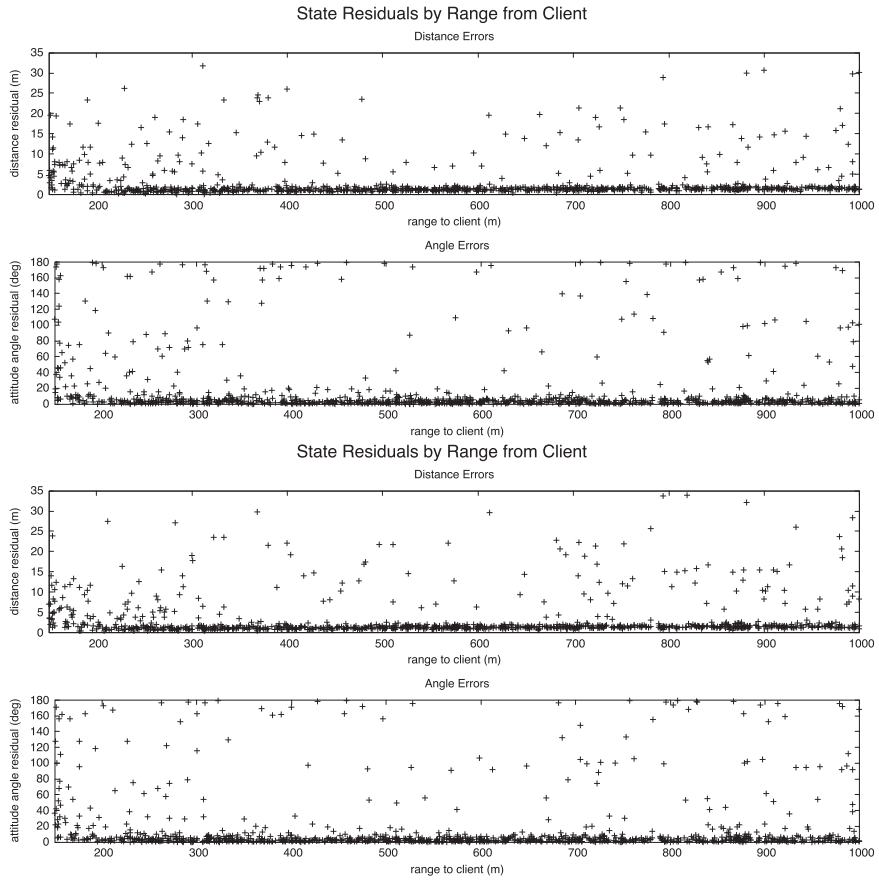

Fig. 4. The addition of noise does not noticeably affect Itokawa OUR-CVFH results as the client range varies. The top pair of state residuals show how far off OUR-CVFH (as refined by ICP) was for a set of randomly generated LIDAR images of Itokawa. The third and fourth plots are for the same set of LIDAR images, but with additive noise (zero-mean, negative, 10 cm).

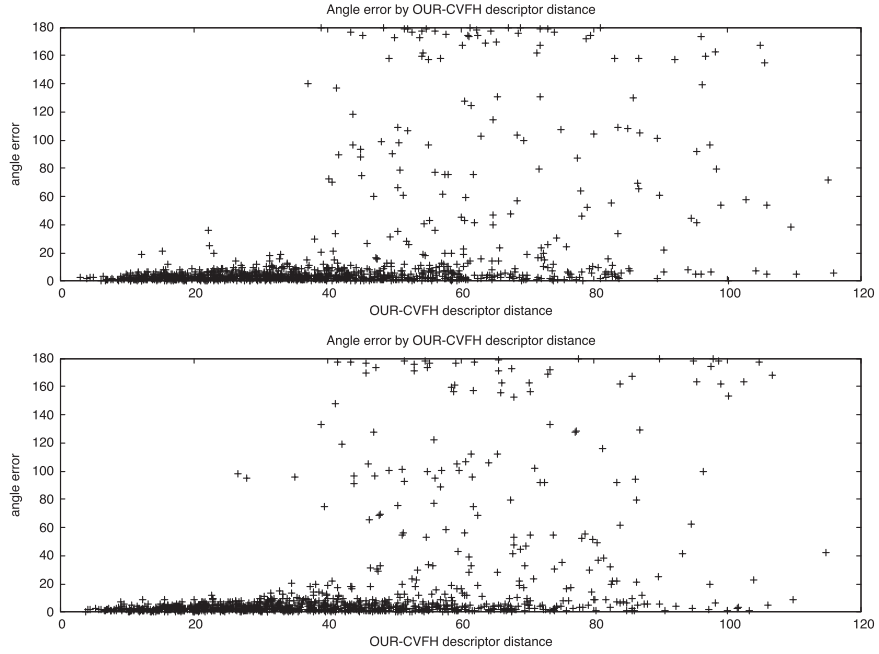

Fig. 5. Itokawa OUR-CVFH attitude errors may be limited by establishing a maximum allowable descriptor distance. The two plots are from the 1000-run Monte Carlo with and without noise, respectively, for 25143 Itokawa. The consequence of setting such a threshold is that our selected pose strategy produces fewer updates (see also Fig. 6).

4.5. Pose estimation and fault detection strategy

Most existing strategies for LIDAR-based pose estimation rely exclusively on ICP. An early goal of this work was to make use of OUR-CVFH as an independent check on ICP, each iteration of which is typically dependent upon the previous iteration's output. As such, we designed the following strategy for converting LIDAR images into pose estimates via two concurrently running, identical processes.

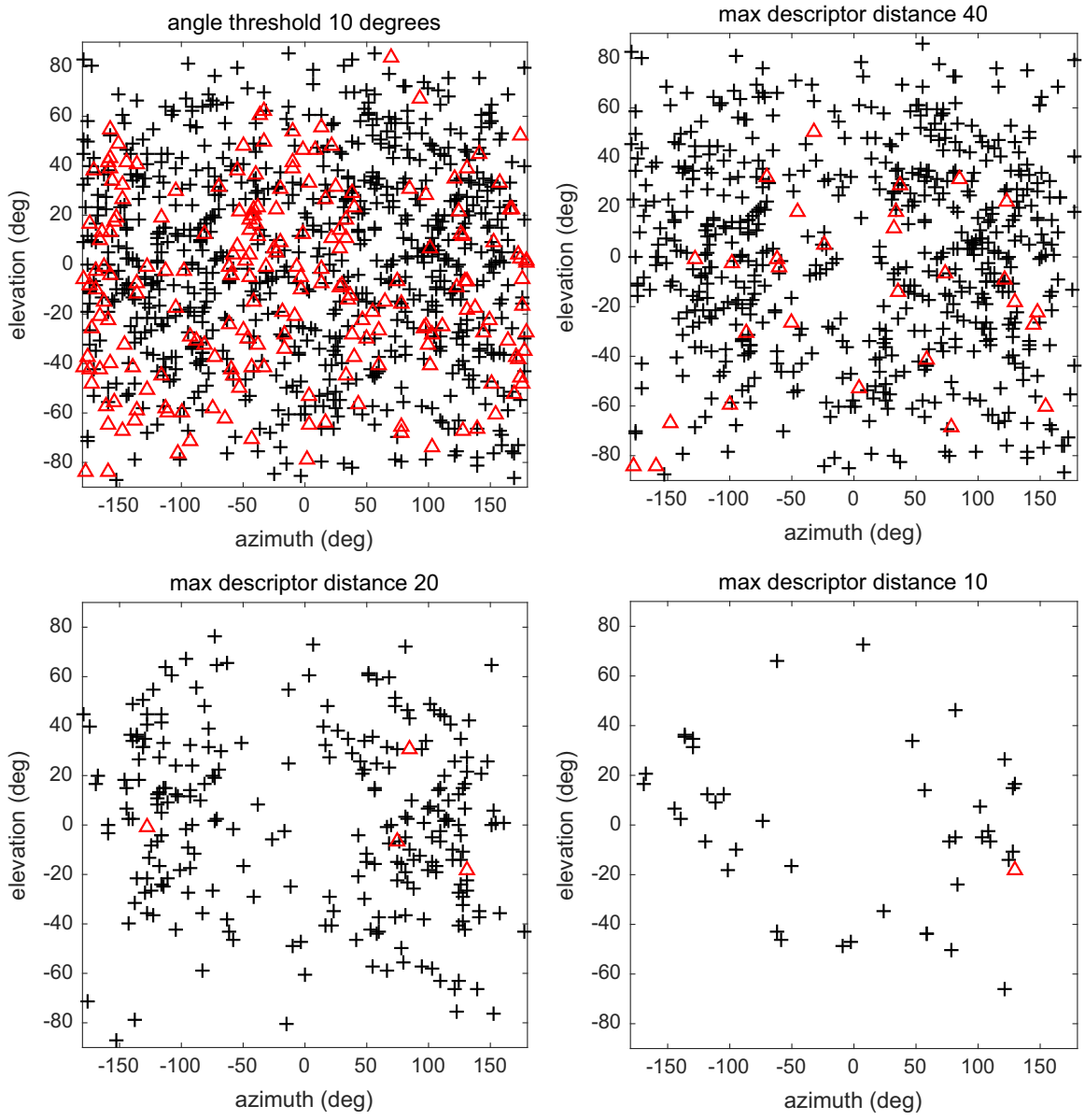

Fig. 6. Setting a restriction on OUR-CVFH descriptor distance limits the number of bad attitude updates sent to the filter. These plots show the azimuth and elevation of each of the 1000 attitude quaternions used to test OUR-CVFH. Black crosses indicate a "correct" pose, within 10 degrees of the true attitude; red triangles indicate an "incorrect" attitude. The upper left plot shows the full distribution of attitudes used in the Monte Carlo simulation; the upper right graph provides only those with a descriptor distance less than 40. The bottom plots show additional limitations on descriptor distances — and demonstrate that if the threshold is set too low, no updates will be available for certain asteroid attitudes. These results come from the noisy LIDAR images, but do not differ substantially from the noiseless simulations. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

The procedure for each process is as follows:

1. Receive a LIDAR image and the time it was recorded, t_{lidar}.
2. Obtain a pose estimate by one of the following means:
   (a) Run OUR-CVFH on the LIDAR image.
   (b) Ask the filter to provide a propagated pose estimate for time t_{lidar}.
3. Refine the estimate using ICP.
4. Send the refined pose, the ICP convergence score, and t_{lidar} to the filter.
5. Repeat.

In all cases, a pose process obtains its first pose estimate using OUR-CVFH. Since this strategy is generally longer running, we planned for one process to typically be running ICP while the other ran OUR-CVFH. If, at any time, the combination of OUR-CVFH and ICP produced a better convergence score than the ICP-only pose process, the ICP process would reinitialize with OUR-CVFH again — and the OUR-CVFH process would proceed into ICP-only mode. Similarly, either process would reinitialize if its update failed the filter's residual edit check — which meant that in some cases, OUR-CVFH could be running simultaneously in both threads. We attempted to stagger these processes, starting the first one several seconds before the second. If two instances requested a pose estimate from the filter for the same t_{lidar} at the same time, one of the two would get a new image from the LIDAR and run OUR-CVFH rather than continuing with ICP.

We found, however, that our ICP-only strategy produced a growing oscillation in the filter between client velocity and angular velocity states, which quickly led to filter divergence. In essence, using the filter as input for ICP amplified the errors by inducing an artificial correlation between these states — arising

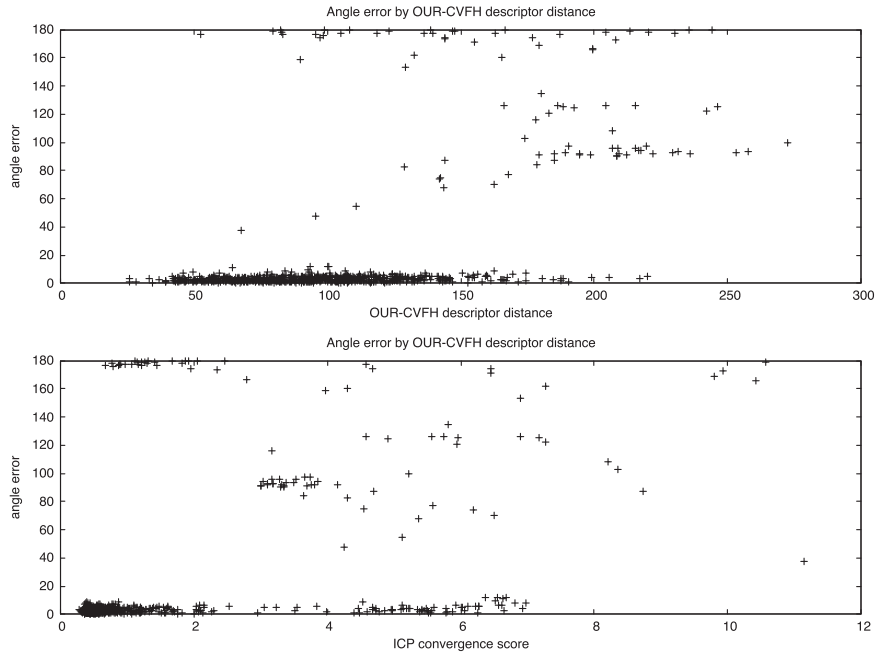

Fig. 7. For the ISS module, we found that ICP convergence score — along with descriptor distance — was effective for eliminating false positives. The displayed plots are for noiseless LIDAR images; the noisy plots are not shown, but are again similar. The effects of OUR-CVFH descriptor distance (top) and ICP convergence score (bottom) on attitude error are correlated.

from the fundamental assumption that for some measurement and state

$$\tilde{\mathbf{y}} = h(\mathbf{x}) + \boldsymbol{\nu} \tag{37}$$

$$\hat{\mathbf{x}} = \mathbf{x} + \mathbf{e}, \tag{38}$$

respectively, the errors $\boldsymbol{\nu}$ and $\mathbf{e}$ are uncorrelated,

$$\mathrm{E}\left[\boldsymbol{\nu}\mathbf{e}^{\mathrm{T}}\right] = \mathbf{0}. \tag{39}$$

We settled on a strategy which used ICP only to refine OUR-CVFH results (and continued to run them staggered in two concurrent processes). In other words, rather than using OUR-CVFH as a check on ICP, we use OUR-CVFH (with refinement by ICP) as the sole method for relative pose computation. We also imposed a requirement that no out-of-order updates be incorporated; that is, if one process finished its computation late, its results would not be admitted.

To characterize the performance of the OUR-CVFH with ICP refinement strategy, we tested it on 1000 attitudes (generated based on [24]) and distances, drawing uniformly, using models of both Itokawa and the ISS functional cargo block (FCB) module (Fig. 3). Point clouds of these objects at each attitude were generated using GLIDAR [25]. We repeated the analysis using the same point clouds with additive noise, and found the results to be generally indistinguishable (Fig. 4). Our analysis demonstrated that the vast majority of erroneous pose solutions can be detected by establishing a maximum allowable OUR-CVFH descriptor distance and/or maximum allowable ICP convergence score. The performance of such a scheme is shown for Itokawa in Figs. 5 and 6 and for the ISS FCB module in Figs. 7 and 8.

5. Simulation of LIDAR-based relative navigation during an asteroid rendezvous

In order to better evaluate the techniques outlined in previous sections of this paper, we assessed the end-to-end navigation system performance using a simulated rendezvous with the asteroid 25143 Itokawa. We focus on the nearer portion of the approach, where the LIDAR is likely to produce sufficient resolution that 6 DOF relative navigation is possible and OUR-CVFH can determine a useful pose.

The entire navigation architecture and physics simulator was written in C++ using the Point Cloud Library (PCL), with a smattering of BASH and Ruby scripts. We performed all Monte Carlo analyses on a Dell Precision T7610 with dual Intel Xeon E5-2630 v2 processors, an NVidia Quadro K4000 video card, and 32 gigabytes of RAM, running Ubuntu Linux.

Details of the simulation framework and a discussion of the results are now provided.

5.1. Summary of simulation framework

We utilized a simple Newtownian physics model to describe the spacecraft dynamics, considering only one gravitating body with no perturbations from solar radiation pressure, magnetic fields, etc. For the purposes of simplicity, we also chose to focus this simulation along a coasting arc, along which the spacecraft performs no translational maneuvers. The physics simulator propagated at 50 Hz, which is the assumed frequency of the spacecraft's gyroscope.

While a single DSN updates is assumed to occur at the beginning of the rendezvous, no additional DSN measurements were provided during the relatively short time period of the simulation. A star tracker provided updates at 5 Hz. In place of a LIDAR sensor, we utilized GLIDAR, an OpenGL-based 3D sensor simulator developed at West Virginia University [25], which generated new sensor images at the same frequency as the physics propagation. To generate a pose measurement, one of the two parallel OUR-CVFH threads would take the latest LIDAR image and compute the pose using the scheme described in Section 4.5. As a result, pose estimates were available to the filter at a much slower rate due to the finite run-time of OUR-CVFH and ICP — typically resulting in only a few measurements per second. We also tested the filter separately from the pose algorithms using "pass-through" no-latency pose updates at a rate of one every three seconds.

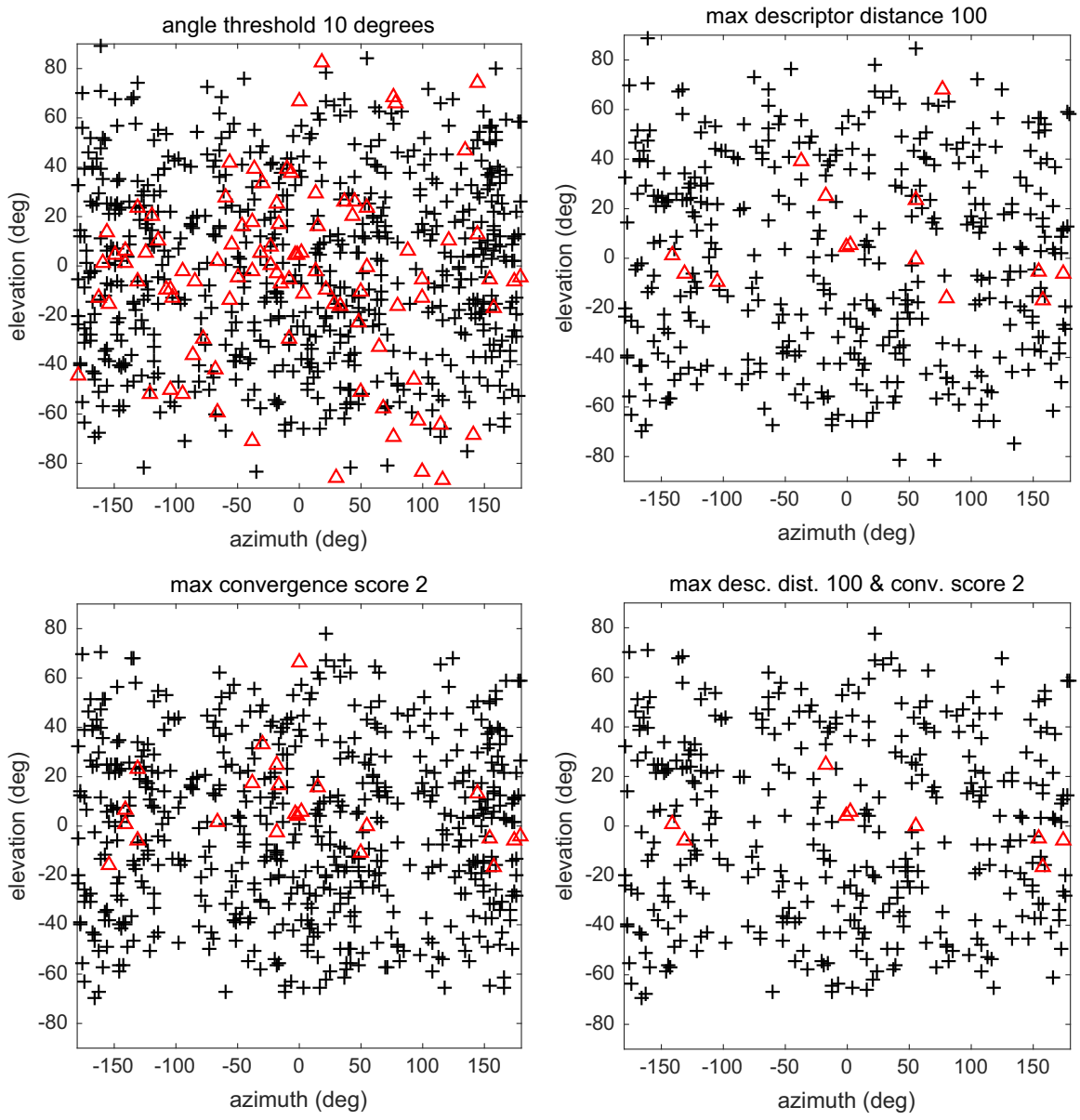

Fig. 8. Convergence score alone eliminates false positives more efficiently than descriptor distances or a combination (for the ISS module). As shown in Fig. 7, both OUR-CVFH descriptor distances and ICP convergence scores can be used to maximize precision and recall. However, we find that using convergence score alone is more efficient than the combination. Black crosses indicate true positives, and red triangles denote false positives, with the threshold between the two set at 10 degrees. Each of these marks shows the azimuth and elevation of a camera placement for testing OUR-CVFH in a 1000-run Monte Carlo analysis.

We ran the rendezvous simulation 1000 times, with identical physical constants (rate of approach, initial positions, etc.), but with a varying initial filter state and with noisy sensor measurements. Body rotation rates for Itokawa were arbitrarily chosen and non-zero (different from the true rotation rate of Itokawa) in order to demonstrate performance relative to a tumbling asteroid. We assume that a 3 DOF (translation-only) filter has just handed off to our 6 DOF filter at the beginning of each run, but without carrying over any of the state covariances. Each approach was run for 300 s.

The simulations function under the assumptions that the star tracker always has an unoccluded view of the stars. No forces other than gravity of the central body (earth or sun) were utilized, and no control is permitted.

The starting range for the Itokawa approach was 550 m, closing at 0.4 m/s; and it always begins with a DSN position and velocity update. LIDAR measurements were required to be within 4σ to be admitted by the filter, and other measurements within 3σ.

5.2. Discussion of results

We provide the navigation filter results from the 1000-run Monte Carlo simulation of an approach to Itokawa in Figs. 9 and 10. The time-history of state errors for each of the individual Monte Carlo runs are indicated by the black lines, while the shaded gray region depicts the filter's 3σ covariance.

The constant position errors and growing velocity covariance for the spacecraft in Fig. 9 occurs because DSN is the only source of absolute translational state information and there is only one such measurement at the beginning of the rendezvous; there are no additional DSN updates during the 300-s period of this simulation.

The asteroid position errors and velocity covariance (Fig. 10) exhibit the same behavior as spacecraft, and for the same reason. Relative to the state errors shown here, the error in the LIDAR measurements is quite small (on the order of centimeters instead of meters). As a result the asteroid's inertial position and velocity

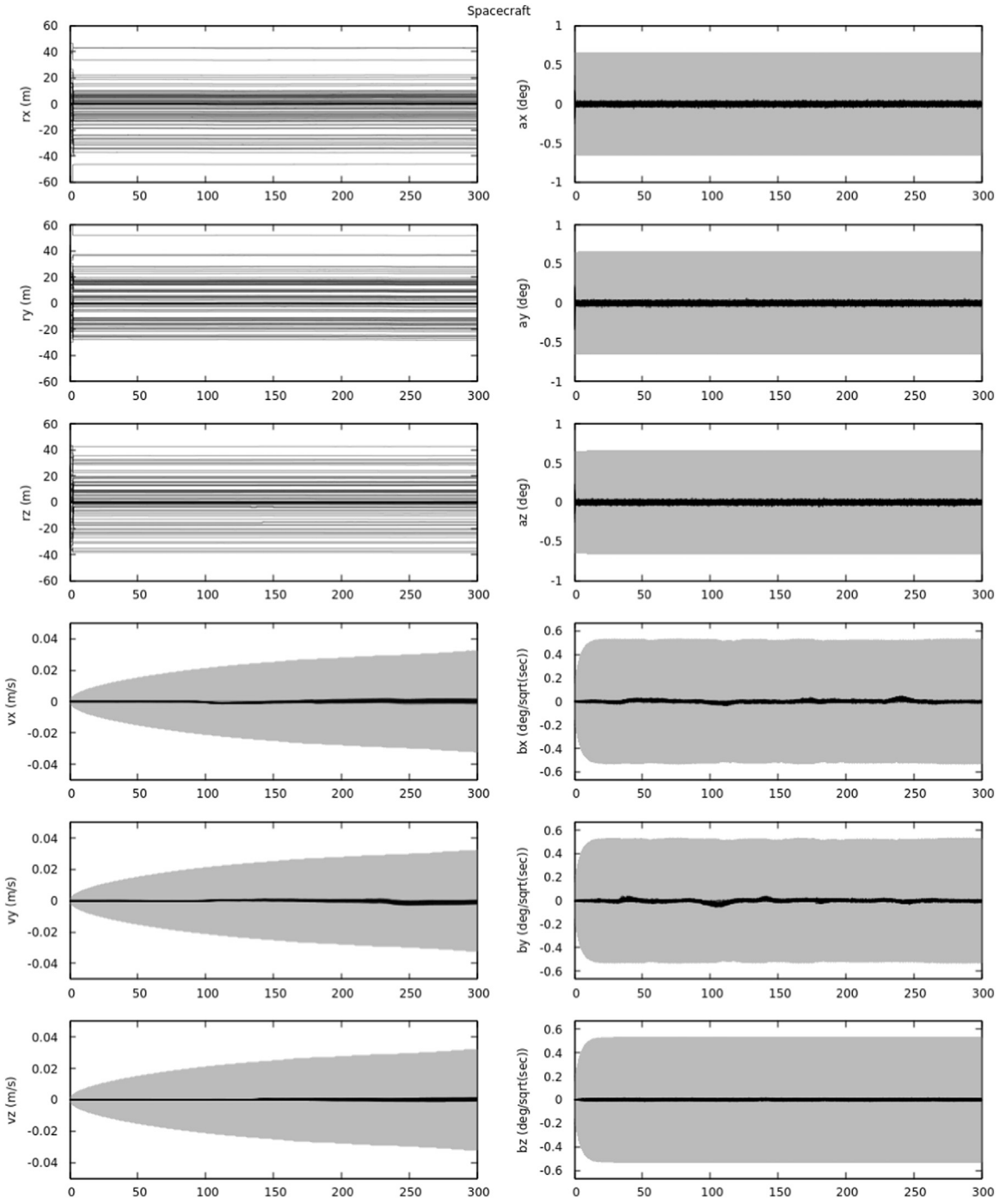

Fig. 9. The 12 filter states for the spacecraft include position, velocity, attitude error, and gyroscope bias. Position $\mathbf{r}$ and velocity $\mathbf{v}$ are in the inertial frame fixed at the center of the earth; attitude error ($\mathbf{a}$) is in the spacecraft body frame. The x-axis shows simulation time, in seconds. The black lines represent individual Monte Carlo runs; the lighter area represents the 3σ covariance from the first run. The position and the velocity are unobservable, and assume the use of DSN for position and velocity just prior to $t=0$; the covariance for position is not shown, but resembles that of velocity.

errors are effectively the same as those of the spacecraft.

The asteroid's attitude and body rate estimates, however, display more interesting behavior because of the nonlinear dynamics and the higher frequency of measurements yielding insight into these states (star tracker and LIDAR). Good convergence is seen on the estimate of both the inertial asteroid attitude and body rate. The growing and shrinking of the covariance for the asteroid attitude and body rate occur because the pose observability of the asymmetric and tumbling asteroid changes with time. Of the 1000 Monte Carlo cases, only once was the filter not able to converge — likely a consequence of the 4σ residual threshold established.

6. Conclusion

In this work, we demonstrated a complete, end-to-end solution for spacecraft rendezvous using LIDAR-based pose measurements. While others have examined the issue of pose computation using LIDAR-generated point clouds, little or no published work has demonstrated these techniques within the context of a functioning navigation filter.

Additionally, we demonstrated the insufficiency of the iterative closest point (ICP) algorithm alone as a method for determining pose. Even if an initial guess can be provided for ICP, and the

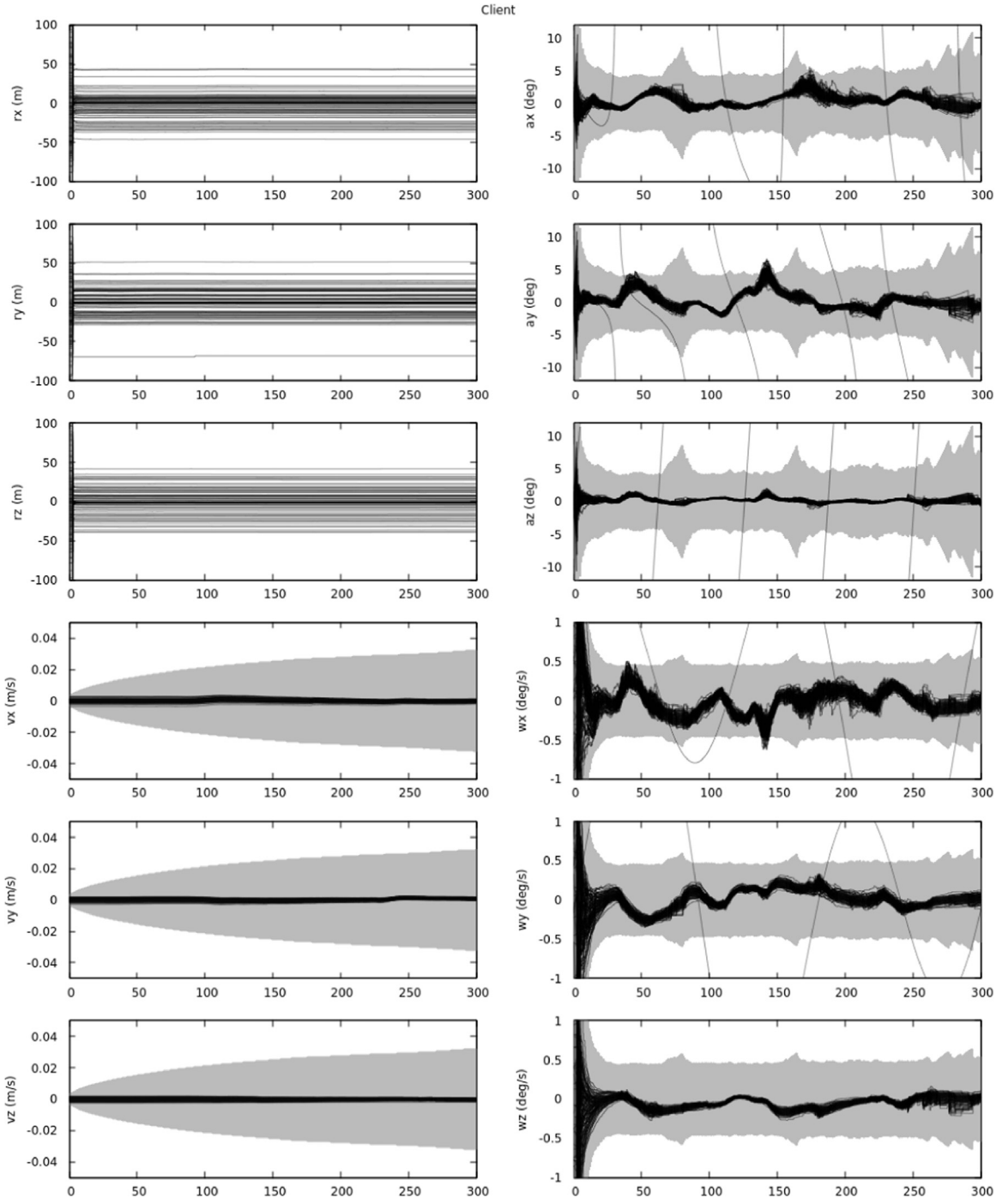

Fig. 10. The 12 asteroid filter states include position, velocity, attitude error, and body rate (angular velocity). Position and velocity are in the inertial frame, as in Fig. 9, and attitude error is in the asteroid body frame. One Monte Carlo instance failed to accept any measurements at all, and can be observed in the attitude plots and two of the angular velocity plots. The *x*-axis shows simulation time, in seconds. Again, the lighter region describes the 3σ covariance, but is not included for the position plots. Additionally, the covariance differs substantially more between Monte Carlo runs for the asteroid attitude and body rate states than it does for position and velocity; we have elected to display only the covariance for the first Monte Carlo run, but note that the regions where the uncertainty increases are the same from run to run.

propagation of incorrect pose solutions avoided, the correlated nature of the resulting errors may be incompatible in some cases with the assumptions employed in modern navigation filters. We introduced a solution that uses the Oriented, Unique, and Repeatable Clustered Viewpoint Feature Histograms (OUR-CVFH) algorithm to circumvent these issues and reliably estimate pose with respect to non-cooperative objects without the need of a good *a priori* guess of the pose.

Acknowledgments

The authors wish to thank Dr. Thomas Evans, Jordan Sell, and Andrew Rhodes of West Virginia University for many thoughtful conversations and support throughout the preparation of this manuscript. The authors also express their appreciation to the developers of the open source Point Cloud Library, which was used extensively in this work. This research was supported by NASA

Goddard Space Flight Center through a contract with the Satellite Servicing Capabilities Office (contract NNG14CR58C, subcontract METSB0043).

Appendix A. Attitude measurements and their relation to attitude states

A.1. Preliminaries

Let us begin with an attitude quaternion

$$\bar{\mathbf{q}} = \begin{bmatrix} q_s \\ \mathbf{q} \end{bmatrix} \tag{A.1}$$

where q_s is the scalar part of the quaternion and $\mathbf{q}$ is the vector part of the quaternion. If this is an attitude quaternion, then it is unity norm, $\| \bar{\mathbf{q}} \| = 1$. We will use the convention $\bar{\mathbf{q}}_b^a$ to represent an attitude quaternion that describes a rotation from frame A to frame B. The corresponding rotation matrix is given by $\mathbf{T}_b^a$, and these can be related according to

$$\mathbf{T} = \left(q_s^2 - \mathbf{q}^T\mathbf{q}\right)\mathbf{I}_{3\times 3} + 2\mathbf{q}\mathbf{q}^T - 2q_s\left[\mathbf{q}\times\right]. \tag{A.2}$$

Additionally, the quaternion may be related to an angle vector according to

$$\bar{\mathbf{q}} = \begin{bmatrix} q_s \\ \mathbf{q} \end{bmatrix} = \begin{bmatrix} \cos(\theta/2) \\ \sin(\theta/2)\mathbf{e}_\theta \end{bmatrix} \tag{A.3}$$

where $\mathbf{e}_\theta$ is the axis of rotation and θ is the magnitude of rotation about that axis. We may also write the quaternion in terms of twice the Gibbs vector, $\mathbf{a}$,

$$\mathbf{a} = 2\mathbf{g} = 2\tan\left(\frac{\theta}{2}\right)\mathbf{e}_\theta. \tag{A.4}$$

Thus, for small angles, we see that

$$\bar{\mathbf{q}} \approx \begin{bmatrix} 1 \\ (\theta/2)\mathbf{e}_\theta \end{bmatrix} \approx \begin{bmatrix} 1 \\ (1/2)\mathbf{a} \end{bmatrix}. \tag{A.5}$$

Returning to regular attitude quaternions (not necessarily small angles), we define quaternion multiplication using the non-Hamiltonian convention,

$$\bar{\mathbf{p}} \otimes \bar{\mathbf{q}} = \begin{bmatrix} q_s p_s - \mathbf{q}^T\mathbf{p} \\ q_s\mathbf{p} + p_s\mathbf{q} - \mathbf{p}\times\mathbf{q} \end{bmatrix}, \tag{A.6}$$

such that the order of operations matches that of rotation matrix multiplication,

$$\bar{\mathbf{q}}_c^a = \bar{\mathbf{q}}_c^b \otimes \bar{\mathbf{q}}_b^a \quad \longleftrightarrow \quad \mathbf{T}_c^a = \mathbf{T}_c^b\mathbf{T}_b^a. \tag{A.7}$$

All of this puts us in a position to make one final useful observation. Consider the quaternion triple product $\bar{\mathbf{q}}_b^a \otimes \delta\bar{\mathbf{q}} \otimes \bar{\mathbf{q}}_a^b$.

Multiplying using Eq. (A.6), using the fact that $\| \mathbf{q} \| = 1$, and applying Eq. (A.2) yields,

$$\bar{\mathbf{q}}_b^a \otimes \delta\bar{\mathbf{q}} \otimes \bar{\mathbf{q}}_a^b = \begin{bmatrix} \delta q \\ \mathbf{T}_b^a\delta\mathbf{q} \end{bmatrix}. \tag{A.8}$$

A.2. LIDAR

Recall that the raw LIDAR relative attitude measurement is given by

$$\bar{\mathbf{q}}_l^c = \bar{\mathbf{q}}_l^s \otimes \bar{\mathbf{q}}_s^i \otimes \bar{\mathbf{q}}_i^c \tag{A.9}$$

Further, briefly recall that the spacecraft attitude error is defined as

$$\bar{\mathbf{q}}_s^i = \delta\bar{\mathbf{q}}_s \otimes \hat{\bar{\mathbf{q}}}_s^i \tag{A.10}$$

and the observed object's (or client's) attitude error is defined as

$$\bar{\mathbf{q}}_c^i = \delta\bar{\mathbf{q}}_c \otimes \hat{\bar{\mathbf{q}}}_c^i \tag{A.11}$$

We define the LIDAR attitude error as

$$\delta\bar{\mathbf{q}}_l = \bar{\mathbf{q}}_l^c \otimes \hat{\bar{\mathbf{q}}}_c^l \tag{A.12}$$

or, after expansion into known and estimated quantities,

$$\delta\bar{\mathbf{q}}_l = \bar{\mathbf{q}}_l^s \otimes \bar{\mathbf{q}}_s^i \otimes \bar{\mathbf{q}}_i^c \otimes \hat{\bar{\mathbf{q}}}_c^i \otimes \hat{\bar{\mathbf{q}}}_i^s \otimes \hat{\bar{\mathbf{q}}}_s^l \tag{A.13}$$

Now, substituting Eqs. (A.10) and (A.11), and recognizing that we are not carrying the LIDAR misalignment as a state ($\bar{\mathbf{q}}_l^s = \hat{\bar{\mathbf{q}}}_l^s$),

$$\delta\bar{\mathbf{q}}_l = \hat{\bar{\mathbf{q}}}_l^s \otimes \delta\bar{\mathbf{q}}_s \otimes \hat{\bar{\mathbf{q}}}_s^i \otimes \hat{\bar{\mathbf{q}}}_i^c \otimes \delta\bar{\mathbf{q}}_c^{-1} \otimes \hat{\bar{\mathbf{q}}}_c^i \otimes \hat{\bar{\mathbf{q}}}_i^s \otimes \hat{\bar{\mathbf{q}}}_s^l \tag{A.14}$$

Applying Eq. (A.8)

$$\delta\bar{\mathbf{q}}_l = \hat{\bar{\mathbf{q}}}_l^s \otimes \delta\bar{\mathbf{q}}_s \otimes \hat{\bar{\mathbf{q}}}_s^i \otimes \begin{bmatrix} \delta q_c \\ -\hat{\mathbf{T}}_i^c\delta\mathbf{q}_c \end{bmatrix} \otimes \hat{\bar{\mathbf{q}}}_i^s \otimes \hat{\bar{\mathbf{q}}}_s^l, \tag{A.15}$$

and applying Eq. (A.8) again,

$$\delta\bar{\mathbf{q}}_l = \hat{\bar{\mathbf{q}}}_l^s \otimes \delta\mathbf{q}_s \otimes \begin{bmatrix} \delta q_c \\ -\hat{\mathbf{T}}_s^i\hat{\mathbf{T}}_i^c\delta\mathbf{q}_c \end{bmatrix} \otimes \hat{\bar{\mathbf{q}}}_s^l. \tag{A.16}$$

Recognizing that both $\delta\bar{\mathbf{q}}_s$ and $\delta\bar{\mathbf{q}}_c$ are assumed to be small, the above is approximately the same as

$$\delta\bar{\mathbf{q}}_l \approx \hat{\bar{\mathbf{q}}}_l^s \otimes \begin{bmatrix} 1 \\ \delta\mathbf{q}_s - \hat{\mathbf{T}}_s^i\hat{\mathbf{T}}_i^c\delta\mathbf{q}_c \end{bmatrix} \otimes \hat{\bar{\mathbf{q}}}_s^l \tag{A.17}$$

and one final application of Eq. (A.8) yields

$$\delta\bar{\mathbf{q}}_l \approx \begin{bmatrix} 1 \\ \hat{\mathbf{T}}_l^s\delta\mathbf{q}_s - \hat{\mathbf{T}}_l^s\hat{\mathbf{T}}_s^i\hat{\mathbf{T}}_i^c\delta\mathbf{q}_c \end{bmatrix} \tag{A.18}$$

The errors have already been assumed small, so substitute using Eq. (A.5) for the three-parameter attitude representation

$$\begin{bmatrix} 1 \\ \frac{\mathbf{a}_l}{2} \end{bmatrix} \approx \begin{bmatrix} 1 \\ \hat{\mathbf{T}}_l^s\frac{\mathbf{a}_s}{2} - \hat{\mathbf{T}}_l^s\hat{\mathbf{T}}_s^i\hat{\mathbf{T}}_i^c\frac{\mathbf{a}_c}{2} \end{bmatrix}, \tag{A.19}$$

which yields the following equation for the LIDAR attitude measurement residual

$$\mathbf{a}_l \approx \hat{\mathbf{T}}_l^s\mathbf{a}_s - \hat{\mathbf{T}}_l^s\hat{\mathbf{T}}_s^i\hat{\mathbf{T}}_i^c\mathbf{a}_c. \tag{A.20}$$

From here the elements in the measurement sensitivity matrix are evident.

A.3. Star tracker

Recall that the raw star tracker measurement is given by

$$\bar{\mathbf{q}}_t^i = \bar{\mathbf{q}}_t^s \otimes \bar{\mathbf{q}}_s^i. \tag{A.21}$$

We will define the star tracker attitude error as

$$\delta\bar{\mathbf{q}}_t = \bar{\mathbf{q}}_t^i \otimes \hat{\bar{\mathbf{q}}}_i^t \tag{A.22}$$

or, after expansion into known and estimated quantities,

$$\delta\bar{\mathbf{q}}_t = \bar{\mathbf{q}}_t^s \otimes \bar{\mathbf{q}}_s^i \otimes \hat{\bar{\mathbf{q}}}_i^s \otimes \hat{\bar{\mathbf{q}}}_s^t. \tag{A.23}$$

Now, substituting Eq. (A.10) and recognizing that we are not carrying the star tracker misalignment as a state ($\bar{\mathbf{q}}_t^s = \hat{\bar{\mathbf{q}}}_t^s$),

$$\delta\bar{\mathbf{q}}_t = \hat{\bar{\mathbf{q}}}_t^s \otimes \delta\bar{\mathbf{q}}_s \otimes \hat{\bar{\mathbf{q}}}_s^i \otimes \hat{\bar{\mathbf{q}}}_i^s \otimes \hat{\bar{\mathbf{q}}}_s^t \tag{A.24}$$

which simplifies to

$$\delta\bar{\mathbf{q}}_t = \hat{\bar{\mathbf{q}}}_t^s \otimes \delta\bar{\mathbf{q}}_s \otimes \hat{\bar{\mathbf{q}}}_s^t. \tag{A.25}$$

Applying Eq. (A.8), we quickly see that

$$\delta\bar{\mathbf{q}}_t = \begin{bmatrix} \delta q_s \\ \hat{\mathbf{T}}_t^s \delta\mathbf{q}_s \end{bmatrix}. \tag{A.26}$$

Assuming that the errors are small and substituting in for our three-parameter attitude error representations (see Eq. (A.5))

$$\begin{bmatrix} 1 \\ \frac{\mathbf{a}_t}{2} \end{bmatrix} \approx \begin{bmatrix} 1 \\ \hat{\mathbf{T}}_t^s \frac{\mathbf{a}_s}{2} \end{bmatrix} \tag{A.27}$$

which yields the following equation for the star tracker measurement residual:

$$\mathbf{a}_t \approx \hat{\mathbf{T}}_t^s \mathbf{a}_s. \tag{A.28}$$

From here the elements in the measurement sensitivity matrix are evident.

References

[1] T. Mukai, A.M. Nakamura, T. Sakai, Asteroidal surface studies by laboratory light scattering and LIDAR on HAYABUSA, Adv. Space Res. 37 (January (1)) (2006) 138–141.

[2] N. Namiki, T. Mizuno, M. Mita, K. Kawahara, H. Kunimori, H. Senshu, R. Yamada, H. Noda, M. Shizugami, N. Hirata, H. Ikeda, S. Abe, K. Matsumoto, S. Oshigami, F. Yoshida, N. Hirata, H. Miyamoto, S. Sasaki, H. Araki, S. Tazawa, Y. Ishihara, M. Kobayashi, K. Wada, H. Demura, J. Kimura, M. Hayakawa, N. Kobayashi, Development of Hayabusa-2 LIDAR, in: 45th Lunar and Planetary Science Conference, The Woodlands, TX, US, 2014.

[3] J.A. Christian, S. Cryan, A survey of LIDAR technology and its use in spacecraft relative navigation, in: AIAA Guidance and Control Conference, Boston, MA, 2013.

[4] J.a. Christian, H. Hinkel, C.N. D'Souza, S. Maguire, M. Patangan, The Sensor Test for Orion RelNav Risk Mitigation (STORRM) development test objective, in: Guidance, Navigation, and Control Conference, no. August, Portland, OR, 2011.

[5] C. Dennehy, Relative Navigation Light Detection and Ranging (LIDAR) Sensor Development Test Objective (DTO) Performance Verification, NASA, Technical Report NASA/TM-2013-217992, 2013.

[6] P.J. Llanos, M. Di Domenico, J. Gil-Fernandez, Advanced GNC technologies for proximity operations in missions to small bodies, in: AAS Guidance and Control Conference, Breckenridge, CO, 2013.

[7] P.J. Besl, N.D. McKay, A method for registration of 3-D shapes, in: SPIE Sensor Fusion IV: Control Paradigms and Data Structures, vol. 1611, 1992, pp. 586–606.

[8] Y. Chen, G. Medioni, Object modeling by registration of multiple range images, in: IEEE International Conference on Robotics and Automation, Sacramento, CA, 1991, pp. 2724–2729.

[9] F. Aghili, M. Kuryllo, G. Okouneva, C. English, Fault–tolerant pose estimation of space objects, in: 2010 IEEE/ASME International Conference on Advanced Intelligent Mechatronics, Ieee, Montreal, Canada, July 2010, pp. 947–954.

[10] J. Christian, S. Robinson, C. D'Souza, J. Ruiz, Cooperative relative navigation of spacecraft using flash light detection and ranging sensors, J. Guid. Control Dyn. 37 (2) (2014) 452–465.

[11] S. Robinson, J. Christian, Pattern design for 3d point matching, NAVIGATION: J. Inst. Navig. 62 (3) (2015) 189–203.

[12] R. Opromolla, G. Fasano, G. Rufino, M. Grassi, Uncooperative pose estimation with a LIDAR-based system, Acta Astronaut. 110 (2015) 287–297.

[13] A. Aldoma, F. Tombari, R.B. Rusu, M. Vincze, OUR-CVFH—Oriented, Unique and Repeatable Clustered Viewpoint Feature Histogram for object recognition and 6DOF pose estimation, in: A. Pinz, T. Pock, H. Bischof, F. Leberl (Eds.), Pattern Recognition, Springer, Berlin, Heidelberg, 2012, pp. 113–122.

[14] E. Lefferts, F. Markley, M. Shuster, Kalman filtering for spacecraft attitude estimation, J. Guid. Control Dyn. 5 (5) (1982) 417–429.

[15] R. Rusu, N. Blodow, Z. Marton, M. Beetz, Aligning point cloud views using persistent feature histograms, in: 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, September 2008, pp. 3384–3391.

[16] R.B. Rusu, N. Blodow, M. Beetz, Fast Point Feature Histograms (FPFH) for 3D registration, in: 2009 IEEE International Conference on Robotics and Automation, May 2009, pp. 3212–3217.

[17] R.B. Rusu, G. Bradski, R. Thibaux, J. Hsu, Fast 3D recognition and pose using the Viewpoint Feature Histogram, in: 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, October 2010, pp. 2155–2162.

[18] A. Aldoma, M. Vincze, N. Blodow, D. Gossow, S. Gedikli, R.B. Rusu, G. Bradski, W. Garage, CAD-model recognition and 6DOF pose estimation using 3D cues, in: IEEE International Conference on Computer Vision Workshops, 2011, pp. 585–592.

[19] A. Gelb, Applied Optimal Estimation, MIT Press, Cambridge, MA, 1974.

[20] P.S. Maybeck, Stochastic Models, Estimation and Control, vol. 1, Academic Press, New York, NY, 1979.

[21] F.L. Markley, J.L. Crassidis, Fundamentals of Spacecraft Attitude Determination and Control, Springer, New York, NY, 2014.

[22] F.L. Markley, Attitude error representations for Kalman filtering, J. Guid. Control Dyn. 26 (2) (2003) 311–317.

[23] R. Farrenkopf, Analytic steady-state accuracy solution for two common spacecraft attitude estimators, J. Guid. Control 1 (1) (1978) 282–284.

[24] K. Shoemake, Uniform random rotations, Graph. Gems III (1992) 124–132.

[25] J. Woods, J. Christian, Glidar: an opengl-based, real-time, and open source 3d sensor simulator for testing computer vision algorithms, J. Imaging 2 (1) (2016).