Terrain-Relative Navigation with Neuro-Inspired Elevation Encoding

KRISTEN A. MICHAELSON
University of Texas at Austin, Austin, TX

FELIX WANG
Sandia National Laboratories, Albuquerque, NM

RENATO ZANETTI, Senior Member, IEEE
University of Texas at Austin, Austin, TX

***Abstract*— Terrain-relative navigation (TRN) encompasses a wide variety of algorithms that perform localization with respect to the terrain below a flying vehicle. In traditional approaches, measurements of the terrain are matched to a map carried onboard. This work presents a terrain-relative navigation filter with a position measurement inspired by neural activity associated with positioning in nature. The filter is shown to produce accurate position measurements that outperform popular optimization and template matching methods given poor prior knowledge of the position. The proposed method is also better-suited to distributed implementation than optimization-based methods.**

***Index Terms*— terrain-relative navigation, inertial navigation, kalman filter, neuro-inspired**

Manuscript received XXXXX 00, 0000; revised XXXXX 00, 0000; accepted XXXXX 00, 0000.

Corresponding author: K. Michaelson

Kristen A. Michaelson is with the Department of Aerospace Engineering & Engineering Mechanics at The University of Texas at Austin, Austin, TX 78705 USA (e-mail: kmichaelson@utexas.edu). Felix Wang is with Sandia National Laboratories, Albuquerque, NM 87123 USA (e-mail: felwang@sandia.gov). Renato Zanetti is with the the Department of Aerospace Engineering & Engineering Mechanics and the Oden Institute at The University of Texas at Austin, Austin, TX 78705 USA (e-mail: renato@utexas.edu).

I. Introduction

THE task of an autonomous navigation system is to provide timely, meaningful state estimates throughout the period of operation. Aided inertial navigation is a popular choice for autonomous vehicles with difficult-to-characterize dynamics. During aided inertial navigation, the state estimates are propagated forward in time using measurements from high-rate inertial measurements units (IMUs). Then, at less frequent intervals, they are updated using observations of the environment from other sensors. In most existing terrain-relative navigation (TRN) systems, position is deduced by comparing observations of the terrain to a terrain map carried onboard. In this work, we adapt a novel approach to terrain-relative localization [1] for use in a navigation filter.

Early TRN algorithms were developed for missile navigation [2], [3]. These methods are similar in principle to iterative closest point (ICP), an optimization-based point cloud matching algorithm [4]. If the terrain measurements can be represented as depth images, then image correlation techniques may also be used [5], [6]. In this article, we present an inertial navigation filter with a neuro-inspired terrain-relative position measurement. Many recently-introduced TRN algorithms use neural networks to localize spacecraft relative to craters and other features on the surface of the moon [7], [8]. Machine learning methods have also been proposed for hazard detection and guidance during spacecraft landing [9], [10], [11].

Instead of a neural network, we use a *phase candidate dictionary* built by applying a series of linear transformations to elevation contours on a digital elevation map (DEM). This representation is inspired by neural activity associated with positioning in animals [12], [13]. The resulting data structure is smaller than the DEM and better-suited to implementation on neuromorphic hardware than optimization-based methods. We improve on previous work [14] by introducing a novel position update that significantly reduces the amount of working memory required to process each terrain measurement. The new update scheme also includes a more elegant formulation of the online covariance computation and a new measurement rejection technique based on a simple image processing metric. Finally, we relax the assumption of perfectly-known altitude.

The remainder of this article is organized as follows: Section II details the biological inspiration behind the proposed approach and outlines the processes for encoding and decoding elevation values; Section III describes the inertial navigation filter; Section IV gives the estimation error results for a simulated trajectory and compares the positioning performance to two other approaches; and Section V presents conclusions and future work.

II. Background

A. Biological Inspiration

Scientists study navigation in animals by measuring electrical activity in their brains. Activity in the *grid cells*, a group of cells in the hippocampus, has been identified as a key component of self-localization. Fig. 1 shows data collected during three experiments with rats [12]. In each experiment, the rat roams freely around a square enclosure. The animal's path is shown in black. Spikes in grid cell activity are marked with colored points. Remarkably, the activity in each cell forms a distinct

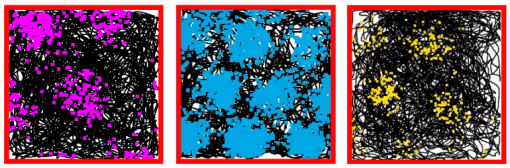
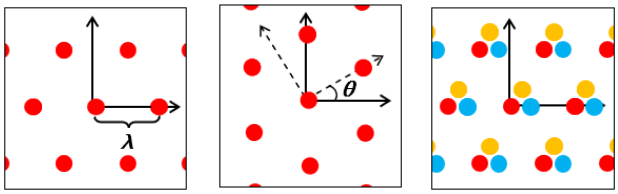


Fig. 1: Grid cell activity during experiments with rats. Adapted from [12] (see Supplementary Information, Fig. 3).

hexagonal grid pattern over the space. Activity peaks at the center of each cluster and is sparser elsewhere. Figs. 2 and 3 show examples of computational representations of hexagonal grids.

The brain contains hundreds of grid cells. Each fires with unique *scale*, *orientation*, and x- and y- *spatial phase offsets*. The scale is the distance between peaks in activity. The orientation is the angle of an axis connecting adjacent peaks with respect to an arbitrarily-defined frame. The spatial phase offsets define the locations of the peaks in the space. Because the grid activity is periodic, the spatial phase offsets are defined on the interval $[0, 2\pi)$. These four parameters fully define any hexagonal grid.



(a) Grid scale, λ (b) Grid orientation, θ (c) Phase, (ϕ_x, ϕ_y)

Fig. 2: The four grid parameters. The spatial phase offsets $(\phi_x, \phi_y) \in [0, 2\pi) \times [0, 2\pi)$ shift the grid along axes connecting adjacent peaks. The red, blue, and yellow points in Fig. 2c represent grids with the same scale and orientation, but different spatial phase offsets.

It is likely that grid cells fire in response to perceived environmental features [12]. In this work, we take inspiration from biological grid cell activity by associating distinct hexagonal grids with elevation values on a DEM. While a single hexagonal grid does not uniquely identify

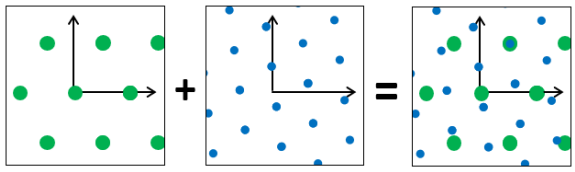


Fig. 3: Grid overlap for positioning

position, in concert, a number of grids can yield an accurate position estimate [1]. This concept is demonstrated in Fig. 3. The green grid (left) and the blue grid (center) both have a number of peaks throughout the navigation space. Neither provides a singular position estimate. However, when superimposed, a point of overlap emerges in the first quadrant (right). This provides evidence that the navigating agent is located at the intersecting point.

B. Reference Frame Conventions

A DEM is a matrix that encodes elevation values at regularly-spaced intervals. Fig. 4 shows a DEM of a 400×400 m section of the University of Texas at Austin campus [15]. The size of the DEM is 800×800 pixels. Each pixel in the DEM represents a single elevation value. In this work, the origin is placed at the top-left corner of the image. Fig. 5a shows the DEM in the pixel space.

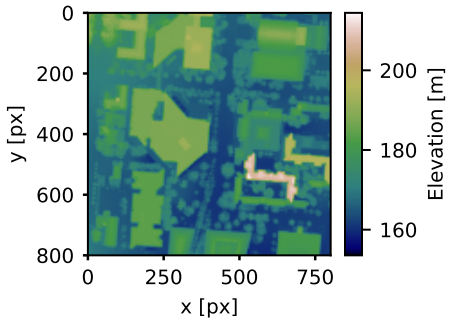
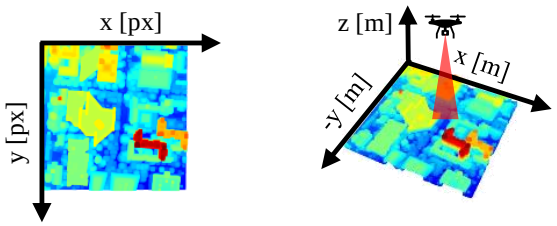


Fig. 4: Elevation map of the University of Texas at Austin campus

Navigation is performed in a local east-north-up (ENU) frame. The origin of the ENU frame is also placed at the top left corner of the map. The x-axis points in the same direction as the x-axis in the pixel space, and the y-axis points in the opposite direction. Fig. 16 shows the relationship between the two reference frames employed in this work.

C. Elevation Encoding

Rather than comparing measured elevations to a DEM, we use elevation measurements to perform lookups in the phase candidate dictionary. The phase candidate dictionary encodes the spatial phase offsets $(\phi_{x,g}, \phi_{y,g})$ that place a peak of hexagonal grid g at the (x, y) pixel



(a) Pixel space (b) ENU Frame

Fig. 5: Navigating over a terrain elevation map. The navigation reference frame is a local east-north-up (ENU) frame defined with its origin at the top left corner of the DEM. Points in the navigation frame are expressed in meters.

location of elevation e on the DEM. Grid g has scale λ_g pixels and orientation θ_g°, where $g = 1 \dots N$ and N is the number of grids in the phase candidate dictionary. The spatial phase offsets are represented by 50×50 binary *phase candidate matrices*. Fig. 6 illustrates the process of building the phase candidate dictionary. The entire process occurs offline before flight time; all the steps described in this section happen only once, and the resulting data structure is carried onboard instead of the DEM.

First, the DEM is divided into elevation contours. Next, a erosion-dilation filter is applied to each contour to remove features with poor spatial correlation. Fig. 6, Step 1 shows the outcome of this process for for elevation values between 179 m and 181 m on the map in Fig. 4. The resulting contour plot captures the flat roof of the building at the bottom-right corner of the DEM, a few outer walls, and sections of sloped roofs.

For each pixel on the contour, we determine the spatial phase offsets $(\phi_{x,g}, \phi_{y,g})$ that place a peak of grid g at (x, y) in the pixel space. The spatial phase offsets are computed using a shear transformation.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos(\theta_g) & -\sin(\theta_g + \frac{\pi}{6}) \\ \sin(\theta_g) & \cos(\theta_g + \frac{\pi}{6}) \end{bmatrix}^{-1} \begin{bmatrix} x \\ y \end{bmatrix} \quad (1)$$

$$\phi_{x,g} = \frac{2\pi}{\lambda_g}(x' \mod \lambda_g) \quad (2)$$

$$\phi_{y,g} = \frac{2\pi}{\lambda_g}(y' \mod \lambda_g) \quad (3)$$

This process is repeated for each pixel on the contour. The spatial phase offsets $(\phi_{x,g}, \phi_{y,g})$ are discretized into 50 bins of size $\frac{2\pi}{50}$ radians. A phase value between 0 and $1 \times \frac{2\pi}{50}$ falls in bin 0, a phase value between $1 \times \frac{2\pi}{50}$ and $2 \times \frac{2\pi}{50}$ falls in bin 1, and so on. These bins become row-column indices in the phase candidate matrix. The phase candidate matrix for grid ($\lambda_g = 295$ px, $\theta_g = 2^\circ$) is shown in Step 3 in Fig. 6. Each value in the phase candidate matrix is initialized to zero. Then, Eqs. 1-3 are applied to each pixel on the contour. The resulting spatial phase offsets are binned. If $\phi_{x,g}$ is placed in bin l, and $\phi_{y,g}$ is placed in bin k, then a 1 is placed at index (k, l) in

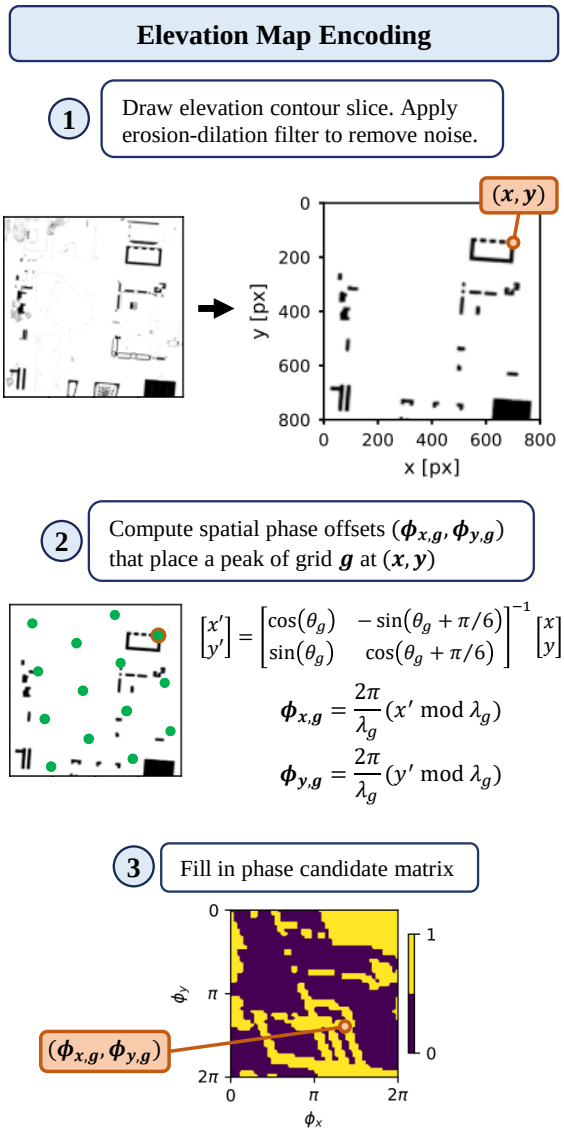


Fig. 6: Elevation encoding. Steps 1-3 are repeated for each elevation contour. Steps 2-3 are repeated for each grid. If we divide the DEM into M elevation contours and choose to encode N grids, then the phase candidate dictionary contains $M \times N$ phase candidate matrices.

the phase candidate matrix. If the value at index (k, l) is already equal to 1, nothing is changed.

A phase candidate matrix is computed for all the elevation contours for each grid in the phase candidate dictionary. Fig. 7 shows a set of phase candidate matrices for a grid with parameters ($\lambda_g = 360$ px, $\theta_g = 18^\circ$). While the information encoded in the phase candidate matrices does not translate directly to the physical space, it is clear that artifacts from the elevation contours are preserved. The erosion-dilation filter applied to each contour (Fig. 6, Step 1) helps avoid *saturating* the phase candidate matrices; if all the values in the phase candidate matrix are 1, then it contains no useful information. Phase candidate matrices are accessed in the phase candidate dictionary

using the lookup keys $[b, g]$, where b is the index of an elevation bin.

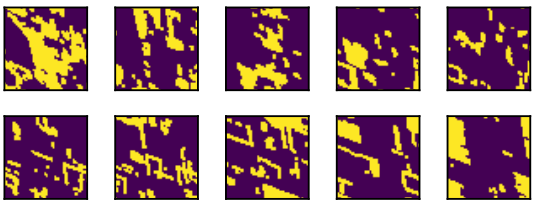


Fig. 7: Phase candidate matrices for grid ($\lambda_g = 360$ px, $\theta_g = 18°$) corresponding to 2 m elevation bins between 165 m and 185 m.

D. Decoding Elevation Measurements

The phase candidate dictionary is used to compute an (x, y) position in the ENU frame from elevation measurements. Fig. 8 shows an example of a group of elevation measurements. It contains 254 elevation values.

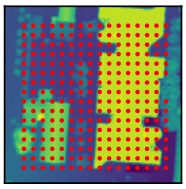


Fig. 8: A collection of elevation measurements. The vehicle is located roughly at the center of the points.

The phase candidate dictionary is used to compute the spatial phase offsets for each grid corresponding to the position of the vehicle above the map. This process is outlined in Fig. 9. First, the elevations are placed into bins using the same discretization as the phase candidate dictionary. Then, starting with grid g, a lookup is performed for each observed elevation. This produces 254 phase candidate matrices.

Since the aim is to estimate the location of the vehicle (not the location of each measured elevation), a roll operation must be applied to each phase candidate matrix. If the displacement between the vehicle's location and the location of a measured elevation is $\begin{bmatrix} dx & dy \end{bmatrix}^T$, then the phase shifts $d\phi_{x,g}$ and $d\phi_{y,g}$ are computed by applying Eqs. 1-3 to the vector $\begin{bmatrix} dx & dy \end{bmatrix}^T$. The phase shifts $d\phi_{x,g}$ and $d\phi_{y,g}$ are placed into $\frac{2\pi}{50}$-length bins; the same used to build the phase candidate matrices. The bin index of $d\phi_{y,g}$ is used to roll the rows of the phase candidate matrix, and the bin index of $d\phi_{x,g}$ is used to roll the columns.

Fig. 9, Step 1 shows a selection of the phase candidate matrices produced from the elevation measurements in Fig. 8 using the roll operation described above. Since

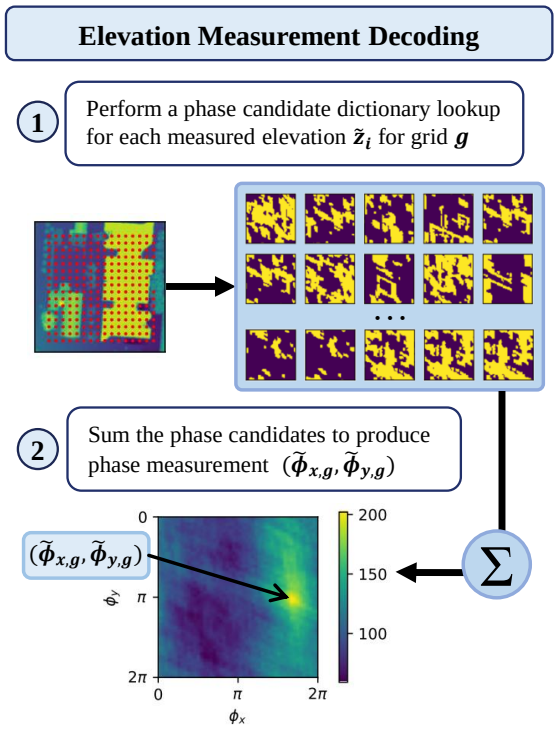


Fig. 9: Elevation measurement decoding scheme. Steps 1-2 are repeated for each grid in the phase candidate dictionary.

many of the measured elevations fall into the same bins, the phase candidate dictionary lookups produce many copies of the same phase candidate matrices. However, since they are shifted in the phase space according to their displacement relative to the vehicle, most of the phase candidate matrices in Fig. 9, Step 1 are unique. After the roll operation is applied to each phase candidate matrix, the phase candidate matrices are summed to produce a phase measurement for the grid. This summation operation produces a bright peak at the phase values common to each phase candidate matrix; the phase values that place a peak of grid g at the location of the vehicle in the pixel space. The phase measurement $(\tilde{\phi}_{x,g}, \tilde{\phi}_{y,g})$ is the `arg max` of the phase candidate sum.

III. Inertial Navigation

In Section IV, we simulate a constant-altitude flight above the terrain shown in Fig. 4. The navigation filter carries five states:

$$\mathbf{x} = \begin{bmatrix} \mathbf{r} \\ \mathbf{v} \\ \theta \end{bmatrix} \tag{4}$$

where $\mathbf{r} = \begin{bmatrix} r_x & r_y \end{bmatrix}^T$ is the x-y position of the vehicle in the ENU frame, $\mathbf{v} = \begin{bmatrix} v_x & v_y \end{bmatrix}^T$ is the x-y velocity, and θ is the heading angle. For the purposes of this work, the ENU frame is taken to be an inertial frame. The sensor

z-axis remains aligned with the ENU z-axis throughout the flight.

A. Dynamics Propagation

The dynamics are propagated using measurements from an onboard inertial measurement unit (IMU). The IMU measures acceleration $\tilde{\mathbf{a}}$ and angular rate $\tilde{\omega}$ in the body frame. The accelerometer measurement is

$$\tilde{\mathbf{a}} = T_e^b \mathbf{a} + \boldsymbol{\eta}_a \tag{5}$$

where $\mathbf{a} = \begin{bmatrix} a_x & a_y \end{bmatrix}$ is the true ENU-frame acceleration, T_e^b is a 2×2 direction cosine matrix that represents the the transformation from the ENU frame to the body frame, and $\boldsymbol{\eta}_a$ is an additive white Gaussian noise with power spectral density $PSD_a = 1.361 \times 10^{-6}$ m^2/s^3 [16].

Similarly, the gyroscope measurement is

$$\tilde{\omega} = \omega + \eta_g \tag{6}$$

where ω is the true angular rate and η_g is an additive white Gaussian noise with power spectral density $PSD_g = 6.250 \times 10^{-6}$ deg^2/s [16].

The state dynamics are $\dot{\mathbf{x}} = f(\mathbf{x})$, where

$$f(\mathbf{x}) = \begin{bmatrix} \mathbf{v} \\ \mathbf{a} \\ \omega \end{bmatrix}. \tag{7}$$

The estimated state is propagated in discrete time at the IMU rate:

$$\bar{\mathbf{r}}(k+1) = \bar{\mathbf{r}}(k) + \bar{\mathbf{v}}(k)\Delta t + \frac{1}{2}\hat{\mathbf{a}}(k)\Delta t^2 \tag{8}$$

$$\bar{\mathbf{v}}(k+1) = \bar{\mathbf{v}}(k) + \hat{\mathbf{a}}(k)\Delta t \tag{9}$$

$$\bar{\theta}(k+1) = \bar{\theta}(k) + \tilde{\omega}\Delta t \tag{10}$$

where

$$\hat{\mathbf{a}}(k) = \bar{T}_b^e(k)\tilde{\mathbf{a}} \tag{11}$$

$\bar{T}_b^e$ is a direction cosine matrix computed from the heading angle estimate $\bar{\theta}$, and Δt is the time between time step k and time step $k+1$.

The discrete-time covariance propagation is

$$P_{k+1} = \Phi P_k \Phi^T + BQB^T, \tag{12}$$

where $\Phi = e^{F\Delta t}$, and F is the Jacobian of the continuous-time error dynamics:

$$F = \begin{bmatrix} \mathbf{0}_{2\times2} & I_{2\times2} & \mathbf{0}_{2\times1} \\ \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} & \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}\hat{\mathbf{a}}(k) \\ \mathbf{0}_{1\times2} & \mathbf{0}_{1\times2} & 0 \end{bmatrix}. \tag{13}$$

The skew-symmetric term in Eq. 13 manifests as a result of the partial derivative of direction cosine matrix $\hat{T}_b^e$ with respect to the heading angle. B is the Jacobian of the error dynamics with respect to the noises η_a and η_g:

$$B = \begin{bmatrix} -\frac{1}{2}\hat{T}_b^e\Delta t & \mathbf{0}_{2\times1} \\ -\hat{T}_b^e & \mathbf{0}_{2\times1} \\ \mathbf{0}_{1\times2} & 1 \end{bmatrix}, \tag{14}$$

and

$$Q = \begin{bmatrix} PSD_a I_{2\times2} & \mathbf{0}_{2\times1} \\ \mathbf{0}_{1\times2} & PSD_g \end{bmatrix}\Delta t. \tag{15}$$

B. Sensor Measurements

A LIDAR measures range, azimuth, and elevation to points on the terrain below the vehicle. In this section, elevation ϵ refers to an angle between the vehicle and a point on the terrain. The range is measured in meters, and the azimuth and elevation are measured in radians.

$$\tilde{\mathbf{y}}_i = \begin{bmatrix} \rho_i \\ \alpha_i \\ \epsilon_i \end{bmatrix} + \boldsymbol{\gamma}_i \tag{16}$$

$$\boldsymbol{\gamma}_i \sim \mathcal{N}\Big(\mathbf{0}_{3\times1}, \begin{bmatrix} 0.25^2 & 0 & 0 \\ 0 & (0.01\pi/180)^2 & 0 \\ 0 & 0 & (0.01\pi/180)^2 \end{bmatrix}\Big) \tag{17}$$

The field of view (FOV) of the sensor is 20°. The LIDAR measurements are first converted to a point cloud in the body frame. Then they are rotated into the inertial frame using an onboard magnetometer. The location of a point in the body frame is computed from the LIDAR measurement as

$$\tilde{\mathbf{r}}_{i,b} = \begin{bmatrix} \tilde{\rho}_i \cos(\tilde{\epsilon}_i)\cos(\tilde{\alpha}_i) \\ \tilde{\rho}_i \cos(\tilde{\epsilon}_i)\sin(\tilde{\alpha}_i) \\ \tilde{\rho}_i \sin(\tilde{\epsilon}_i) \end{bmatrix} \tag{18}$$

where $\tilde{\rho}$, $\tilde{\epsilon}$, and $\tilde{\alpha}$ are the noisy range, azimuth, and elevation measurements respectively. The magnetometer measures the heading of the vehicle in radians. The magnetometer measurement is [17]:

$$\tilde{\theta} = \theta + \xi \tag{19}$$

$$\xi \sim \mathcal{N}\left(0, \left(\frac{2.5}{3}\pi/180\right)^2\right) \tag{20}$$

where θ is the true heading angle. The magnetometer measurement is not used to update the vehicle heading directly; rather, it is used to place the LIDAR point cloud in the inertial frame. The inertial-frame point cloud is $\tilde{\mathbf{r}}_{i,e} = \begin{bmatrix} \tilde{x}_{i,e} & \tilde{y}_{i,e} & \tilde{z}_{i,e} \end{bmatrix}^T$, where:

$$\tilde{\mathbf{r}}_{i,e} = \begin{bmatrix} R_b^e(\tilde{\theta}) & \mathbf{0}_{2\times1} \\ \mathbf{0}_{1\times2} & 1 \end{bmatrix}\tilde{\mathbf{r}}_{i,b} + \begin{bmatrix} 0 \\ 0 \\ \hat{Z} \end{bmatrix} \tag{21}$$

where $R_b^e(\tilde{\theta})$ is the 2×2 rotation matrix that places points from the body frame into the ENU frame, and $\hat{Z}$ is the estimated altitude of the vehicle (see Sec. D). Note that Eq. 21 does not compute the x and y values of $\tilde{\mathbf{r}}_{i,e}$ in the inertial frame; rather, the pair $(\tilde{x}_{i,e}, \tilde{y}_{i,e})$ represents the inertial-frame offset between the vehicle and the elevation value $\tilde{z}_{i,e}$. The inertial-frame offsets $(\tilde{x}_{i,e}, \tilde{y}_{i,e})$ and the elevation values $\tilde{z}_{i,e}$ are used to estimate phase values in the manner described in Section D. A pair of phase measurements $(\tilde{\phi}_{x,g}, \tilde{\phi}_{y,g})$ is computed for each grid in the phase candidate dictionary.

C. Measurement Update

The state update is applied in two stages: first, *a priori* phase values are computed from the current position estimate. A Kalman update is applied using the measurements

$(\tilde{\phi}_{x,g}, \tilde{\phi}_{y,g})$. Then, the new phase values are combined in a weighted update scheme to form an *a posteriori* position estimate. The new position estimate is used to update the other filter states.

The relationship between the position of the vehicle and the corresponding phase values for each grid is linear and can be computed offline.

$$\begin{bmatrix}\bar{\phi}_{x,g}\\ \bar{\phi}_{y,g}\end{bmatrix} = M_g \begin{bmatrix}\bar{r}_x\\ \bar{r}_y\end{bmatrix} \mod 2\pi \tag{22}$$

The matrix M_g encodes the transformation between the inertial frame and the phase space of grid g:

$$M_g = \frac{2\pi}{\lambda_g d}\begin{bmatrix}\cos(\theta_g) & -\sin(\theta_g+\frac{\pi}{6})\\ \sin(\theta_g) & \cos(\theta_g+\frac{\pi}{6})\end{bmatrix}^{-1}\begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix} \tag{23}$$

where d is the discretization of the DEM in meters per pixel.

The *a priori* phase covariance for grid g is

$$\bar{P}_{\phi_g\phi_g} = M_g \bar{P}_{rr} M_g^T. \tag{24}$$

The innovation is

$$\boldsymbol{\nu}_{\phi_g} = \left(\begin{bmatrix}\tilde{\phi}_{x,g}\\ \tilde{\phi}_{y,g}\end{bmatrix} - \begin{bmatrix}b_{\phi_x}\\ b_{\phi_y}\end{bmatrix}\right) - \begin{bmatrix}\bar{\phi}_{x,g}\\ \bar{\phi}_{y,g}\end{bmatrix} \tag{25}$$

where $\begin{bmatrix}b_{\phi_x} & b_{\phi_y}\end{bmatrix}^T$ is the phase measurement bias, and the values in $\boldsymbol{\nu}_{\phi_g}$ are adjusted to fall within the interval $[-\pi, \pi)$.

The phase measurements are computed in the manner described in Section D. Given the discretization of the phase candidate matrices, each measured phase value is an integer multiple of $\frac{2\pi}{50}$. This means the phase measurement error is uniformly distributed over the interval $(-\frac{2\pi}{50}, 0] \times (-\frac{2\pi}{50}, 0]$. The phase measurements can be modeled as

$$\begin{bmatrix}\tilde{\phi}_{x,g}\\ \tilde{\phi}_{y,g}\end{bmatrix} = \begin{bmatrix}\phi_{x,g}\\ \phi_{y,g}\end{bmatrix} + \begin{bmatrix}b_{\phi_x}\\ b_{\phi_y}\end{bmatrix} + \boldsymbol{\zeta} \tag{26}$$

where $\begin{bmatrix}\phi_{x,g} & \phi_{y,g}\end{bmatrix}^T$ are the true phase values, and

$$\begin{bmatrix}b_{\phi_x}\\ b_{\phi_y}\end{bmatrix} = \begin{bmatrix}-\frac{\pi}{50}\\ -\frac{\pi}{50}\end{bmatrix} \tag{27}$$

is the mean of the uniform distribution. The phase measurement noise $\boldsymbol{\zeta}$ can be approximated as zero-mean Gaussian with the covariance of the uniform distribution.

$$R_{\phi\phi} = \frac{1}{3}\left(\frac{\pi}{50}\right)^2 I_{2\times 2} \tag{28}$$

The phase update is then

$$W = H\bar{P}_{\phi_g\phi_g}H^T + R_{\phi\phi} \tag{29}$$

$$K = \bar{P}_{\phi_g\phi_g}H^T W^{-1} \tag{30}$$

$$\begin{bmatrix}\hat{\phi}_{x,g}\\ \hat{\phi}_{y,g}\end{bmatrix} = \begin{bmatrix}\bar{\phi}_{x,g}\\ \bar{\phi}_{y,g}\end{bmatrix} + K\boldsymbol{\nu}_{\phi_g} \tag{31}$$

and the updated phase covariance is

$$\hat{P}_{\phi_g\phi_g} = (I-KH)\bar{P}_{\phi_g\phi_g}(I-KH)^T + KR_{\phi\phi}K^T \tag{32}$$

where $H = I_{2\times 2}$.

The updated phase values form an ensemble of position estimates in the physical space. The position estimates are

$$\begin{bmatrix}\hat{r}_{x,g}\\ \hat{r}_{y,g}\end{bmatrix} = M_g^{-1}\left(\begin{bmatrix}\hat{\phi}_{x,g}\\ \hat{\phi}_{y,g}\end{bmatrix} + 2\pi\begin{bmatrix}n_{\phi_{x,g}}\\ n_{\phi_{y,g}}\end{bmatrix}\right) \tag{33}$$

where the vector $\begin{bmatrix}n_{\phi_x} & n_{\phi_y}\end{bmatrix}^T$ is the vector of divisors of the modulo operation in (22). The error covariance of each position estimate is

$$\hat{P}_{rr,g} = M_g^{-1}\hat{P}_{\phi_g\phi_g}(M_g^{-1})^T. \tag{34}$$

Fig. 10a shows an example of grids with updated phase values $(\hat{\phi}_{x,g}, \hat{\phi}_{y,g})$ superimposed over the map space. Peaks of each grid form a tight cluster at the true location of the vehicle. The lack of grid activity in the area surrounding the cluster is by construction; the minimum grid scale is 100 m. Fig. 10b shows the cluster in more detail. Each point $\begin{bmatrix}\hat{r}_{x,g} & \hat{r}_{y,g}\end{bmatrix}$ is shown with a 1σ error covariance ellipse computed from $\hat{P}_{rr,g}$.

The updated position estimate $\begin{bmatrix}\hat{r}_x & \hat{r}_y\end{bmatrix}^T$ is computed using a weighted sum of the points $\begin{bmatrix}\hat{r}_{x,g} & \hat{r}_{y,g}\end{bmatrix}^T$. The weights are proportional to the likelihoods of the phase measurements:

$$w_g = \frac{\mathcal{N}(\boldsymbol{\nu}_{\phi_g}; \mathbf{0}_{2\times 1}, H\bar{P}_{\phi_g\phi_g}H^T + R_{\phi_g\phi_g})}{\sum_{g\in G}\mathcal{N}(\boldsymbol{\nu}_{\phi_g}; \mathbf{0}_{2\times 1}, H\bar{P}_{\phi_g\phi_g}H^T + R_{\phi_g\phi_g})} \tag{35}$$

where $\mathcal{N}(\mathbf{x}; \mathbf{m}, P)$ represents the evaluation of a Gaussian with mean $\mathbf{m}$ and covariance P at point $\mathbf{x}$, and G is the set of measured grids (see Section E).

The updated position estimate is

$$\hat{\mathbf{r}} = \sum_{g\in G} w_g \hat{\mathbf{r}}_g \tag{36}$$

and the updated covariance is

$$\hat{P}_{rr} = \sum_{g\in G} w_g\left(\hat{P}_{rr,g} + \hat{\mathbf{r}}_g\hat{\mathbf{r}}_g^T - \hat{\mathbf{r}}\hat{\mathbf{r}}^T\right). \tag{37}$$

The rest of the states are updated using the same update technique as the Q-method EKF [18]. The Q-method EKF update can be applied when the measurement is only a function of a handful of states. Since position is measured directly, we partition the state vector into $\bar{\mathbf{x}}_1 \triangleq \bar{\mathbf{r}}$ and $\bar{\mathbf{x}}_2 \triangleq \begin{bmatrix}\bar{\mathbf{v}}^T & \bar{\theta}\end{bmatrix}^T$. Then,

$$\hat{\mathbf{x}}_2 = \bar{\mathbf{x}}_2 + \bar{P}_{21}\bar{P}_{11}^{-1}(\hat{\mathbf{x}}_1 - \bar{\mathbf{x}}_1) \tag{38}$$

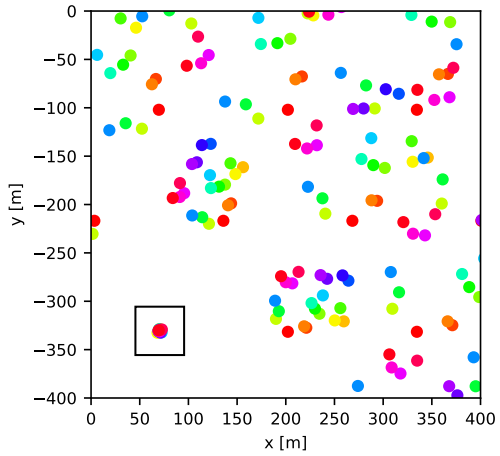
$$\hat{P}_{21} = \bar{P}_{21}\bar{P}_{11}^{-1}\hat{P}_{11} \tag{39}$$

$$\hat{P}_{22} = \bar{P}_{22} + \bar{P}_{21}(\bar{P}_{11}^{-1}\hat{P}_{11}\bar{P}_{11}^{-1} - \bar{P}_{11}^{-1})\bar{P}_{12} \tag{40}$$

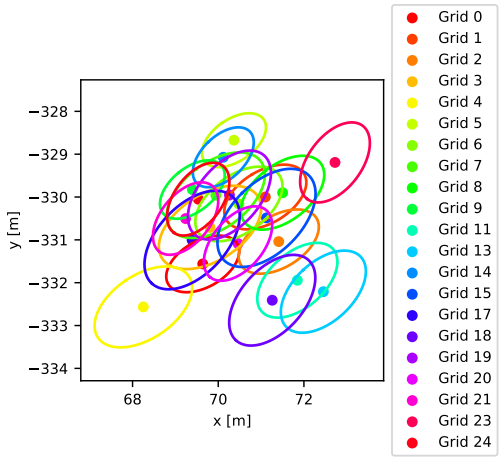
where $\hat{\mathbf{x}}_1 = \hat{\mathbf{r}}$ and $\hat{P}_{11} = \hat{P}_{rr}$.

D. Altitude Estimation

In Section B, LIDAR measurements are transformed from the body frame to the inertial frame. This transformation requires knowledge of the altitude of the vehicle, $\hat{Z}$, in order to correctly bin the measured elevations $\tilde{z}_{i,e}$ for the phase candidate dictionary lookups (Eq. 21). We



(a) Grids with updated phase values



(b) Enhanced view of updated position values $\begin{bmatrix}\hat{r}_{x,g} & \hat{r}_{y,g}\end{bmatrix}^T$ with 1σ covariance ellipses $\hat{P}_{rr,g}$.

Fig. 10: Updated grids. A tight cluster of activity appears at the vehicle position in the lower-left corner.

assume the true altitude of the vehicle (Z) is unknown, but an altitude estimate is available:

$$\tilde{Z} = Z + \delta \tag{41}$$

where the error δ is modeled as zero-mean Gaussian with standard deviation σ_Z.

We process the decoding algorithm (Fig. 9) several times for different values of $\hat{Z}$. We can estimate Z by subtracting reasonable outcomes of δ from $\tilde{Z}$ and checking whether the resulting phase measurements produce a distinct cluster of activity near the estimated position of the vehicle.

Fig 11 shows the result of this process for a vehicle flying at $Z = 500$ m given a noisy altitude estimate of $\tilde{Z} = 500.1$ m with $\sigma_Z = 30$ cm. Since a range of $[-1, 1]$ m covers all reasonable outcomes of δ, we plot the full set of grids in the phase candidate dictionary using the raw phase measurements (Section D) calculated using $\hat{Z} =$

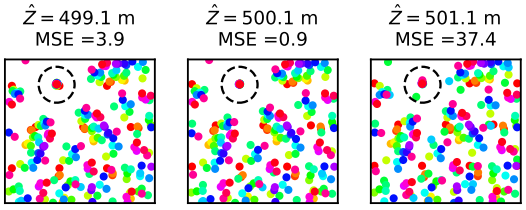


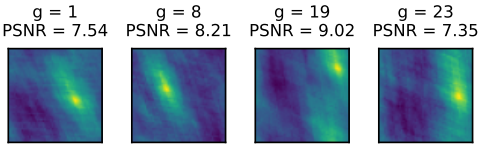
Fig. 11: Grid alignment for three altitude estimates

$\tilde{Z}-1$, $\hat{Z} = \tilde{Z}$, and $\hat{Z} = \tilde{Z}+1$. We then compute the mean-squared error of all grid peaks within a 50 m radius of the estimated position of the vehicle. The estimate $\hat{Z}$ with the smallest MSE is chosen as the altitude estimate, and the corresponding phase values are used in the measurement update (Section B).

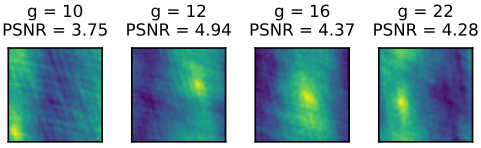
For higher values of σ_Z, more altitude estimates must be tested. However, these will be bounded by the number of elevation bins on the map; any altitude estimate that places elevations $\tilde{z}_{i,e}$ outside values that exist on the map need not be considered. A step size of 1 m (or one half of the elevation bin size) was found to work well for the navigation scenario in Sec. IV.

E. Measurement Rejection

A phase candidate sum is computed from the LIDAR measurement for each grid in the phase candidate dictionary. Fig. 12 shows a set of phase candidate sums for the measurement update in Fig. 10.



(a) Examples of accepted phase candidate sums



(b) Rejected phase candidate sums

Fig. 12: Phase candidate sums for the measurement in Fig. 8. Grids 10, 12, 16, and 22 are rejected based on the criteria in Eq. 42. The corresponding phase candidate sums are visually noisier and have larger areas of increased activity than the accepted phase candidate sums.

The four phase candidate sums in Fig. 12a each have a clear, bright peak. The phase candidate sums in Fig. 12b are more ambiguous. This means that the set of elevation values in the LIDAR measurement do not

uniquely determine a phase value for the corresponding grid. Phase candidate sums that do not have a clear maximum should not be used in the position estimate.

A rejection criterion is defined for phase candidate sums based on the peak signal-to-noise ratio, a measure traditionally used to quantify image reconstruction quality. The peak signal-to-noise ratio is

$$PSNR = 10\log_{10}\left[\frac{\max(I)^2}{\frac{1}{N^2}\sum_{i=0}^{N}\sum_{j=0}^{N} I(i,j) - \tilde{I}(i,j)}\right] \tag{42}$$

where I is an $N \times N$ noise-free image, and $\tilde{I}$ is its noisy counterpart. Eq. 42 is evaluated for each phase candidate sum, where $\tilde{I}$ is the phase candidate sum itself, and I is an "ideal" phase candidate sum with the number of elevation measurements at the (i,j) index of the maximum of $\tilde{I}$ and zeros everywhere else. I is ideal in the sense that it represents perfect evidence for phase measurement $\begin{bmatrix}\tilde{\phi}_{x,g} & \tilde{\phi}_{y,g}\end{bmatrix}^T$. In reality, most phase candidate sums contain areas of increased activity close to the maximum (see Fig. 12).

A $PSNR$ value of 5.0 was chosen as the acceptance threshold for the filter in Section IV. Phase candidate sums with $PSNR > 5.0$ were used as phase measurements, and phase candidate sums with $PSNR \leq 5.0$ were rejected.

IV. Results

The filter is tested on data from a simulated flight over a section of the University of Texas at Austin campus (see Fig. 4). The vehicle flies at an altitude of 500 m with a constant speed of 10 m/s. As it flies, its heading points in the direction of the velocity vector. The IMU propagation rate is 100 Hz. Fig. 13 shows the trajectory.

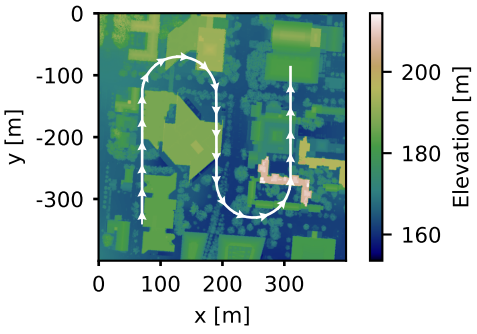


Fig. 13: Trajectory for the simulated flight

The initial uncertainty is 10 m in position, 1 m/s in velocity, and 5 degrees in heading.

$$P_0 = \begin{bmatrix} 10^2 I_{2\times 2} & 0 & 0 \\ 0 & 1.0^2 I_{2\times 2} & 0 \\ 0 & 0 & (5.0\frac{\pi}{180})^2 \end{bmatrix} \tag{43}$$

A pair of LIDAR and magnetometer measurements are recorded every 2 seconds. At each measurement time, a noisy altitude estimate $\tilde{Z}$ with noise $\sigma_Z = 30$ cm is available. The altitude $\hat{Z}$ is predicted using the method described in Sec. D.

The phase candidate dictionary was built using a DEM with resolution 0.5 m/px and an elevation bin resolution of 2 m. It contains 25 grids. The grids have randomly-generated scales λ_g between 200 and 400 px (100 and 200 m) and orientations θ_g between 0° and 24°. Table I lists the grid parameters used in this study.

Grid	λ_g [px]	θ_g [°]
0	265	0°
1	300	1°
2	295	2°
3	380	3°
4	365	4°
5	240	5°
6	330	6°
7	375	7°
8	320	8°
9	250	9°
10	200	10°
11	315	11°
12	230	12°
13	340	13°
14	245	14°
15	400	15°
16	220	16°
17	395	17°
18	360	18°
19	350	19°
20	290	20°
21	280	21°
22	215	22°
23	305	23°
24	275	24°

TABLE I: Table of grid values

Figure 14 shows the set of updated grids for six LIDAR measurements recorded along the trajectory in Fig. 13. In each figure, a box indicates the position of the vehicle. The grids at $t = 50$ s are a good example of measurement rejection. The terrain below the vehicle in that area of the map is mostly vegetation. As a result, many of the phase candidate sums do not have a clear maximum. The associated grids are are not included in the position update (see Section E).

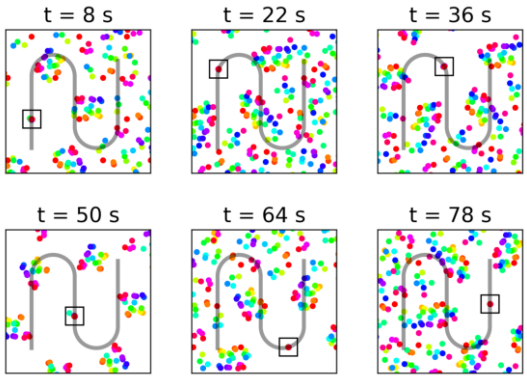


Fig. 14: Grid alignment for points along the trajectory

Figure 15 shows the state estimation error for 100 Monte Carlo runs. The measurement covariance is com-

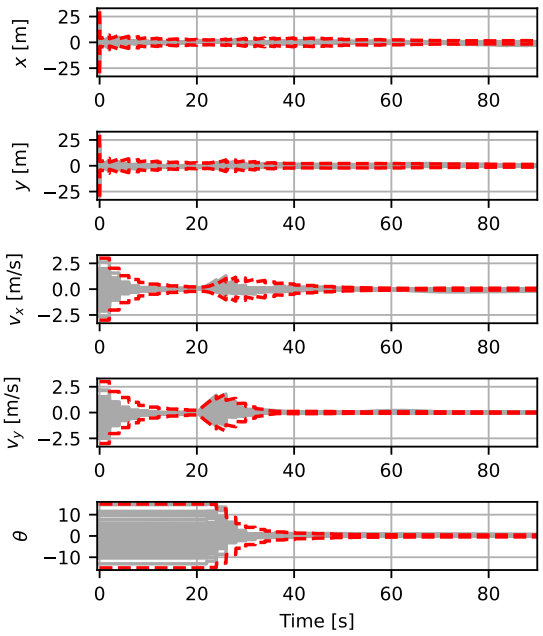


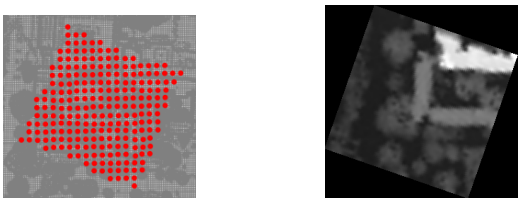
Fig. 15: Estimation error results (gray) and mean 3σ covariance bounds (red)

puted online using the method described in Section C, and phase measurements are rejected using the criteria in Section E. While slightly conservative, the filter successfully maintains custody of the vehicle. The 3σ position error after convergence is on the order of 2.5 m. The 3σ velocity error is on the order of 0.15 m/s, and the 3σ heading error is on the order of 1°.

A. Comparison study

We provide context for these results by comparing them to position measurements generated with two other methods: ICP and image template matching. ICP computes a six degree-of-freedom transformation that aligns a measured point cloud to a reference point cloud. Both point-to-point [4] and point-to-plane [19] registration were computed using Open3D [20]. Image template matching was conducted in MATLAB using the 2D fast Fourier transform (FFT). Depth images were generated at $3\times$ the resolution of the point clouds used in the proposed approach and ICP; this was the minimum resolution required to reliably produce matches. Fig. 16a shows an example of the point cloud measurements used to test ICP and the proposed approach. Fig. 16b shows an example depth image.

The sensor measurement noise characteristics were the same as those used in the filter (see Section B). All four algorithms were given the true altitude, $Z = 500$ m, so the altitude estimation process in Sec. D was not needed for the proposed approach. Each algorithm was initialized with a rotation from a noisy magnetometer measurement



(a) ICP (b) FFT

Fig. 16: Data for comparison study

and position estimate

$$\tilde{\mathbf{r}} = \mathbf{r} + \boldsymbol{\zeta} \tag{44}$$

where $\mathbf{r}$ is the true (x, y) position of the sensors in the ENU frame and $\boldsymbol{\zeta}$ is a random vector with magnitude 30 m, the 3σ initial position error in the filter (see Eq. 43).

For each point cloud measurement, the proposed approach was carried out as described in Section C. Given the same point cloud measurement, a transformation from the LIDAR point cloud to the DEM was computed using ICP. Each ICP algorithm was initialized with position value $\tilde{\mathbf{r}}$ and given a 50 m maximum search radius. The (x, y) values from the ICP transformation result were taken as the position measurement. Similarly, FFT-based image template matching was conducted using the depth image and the DEM. A 50 m search radius was also enforced for image template matching, even though it is a global approach.

The position error is $||\mathbf{r} - \hat{\mathbf{r}}||$, where $\hat{\mathbf{r}}$ is the position estimate computed by each algorithm. Positioning error was computed for measurements generated from 1000 random positions and heading angles. All four algorithms' position errors are concentrated near zero. However, for the very sparse measurement scenario under consideration, the proposed approach vastly outperforms ICP and image template matching. Fig. 17 shows a zoomed-in view of the tails of each error distribution. The point-to-point ICP error distribution has a much thicker tail than that of the proposed approach. Point-to-plane ICP actually achieves an error value near zero (< 1 m) at about the same frequency as the proposed approach, but there is also a second mode at roughly 15 m of error. These results highlight a fundamental shortcoming of ICP; poor initialization can lead to convergence at local minima.

The proposed approach is more robust to poor initialization, since the only requirement for producing an accurate position measurement is that the values $\begin{bmatrix} n_{\phi_x} & n_{\phi_y} \end{bmatrix}^T$ in (33) are correct. This is true as long as the error in the prior position estimate is no greater than the scale of the smallest grid in the phase candidate dictionary. In cases where the initial position estimate is very poor or unknown, previous works have demonstrated the accuracy of global position estimation using sums of two-dimensional sinusoids with peaks placed at the grid peaks [14], [21]. In principle, any ambiguous values $\begin{bmatrix} n_{\phi_x} & n_{\phi_y} \end{bmatrix}^T$ could be computed from a raw phase measurement by plotting

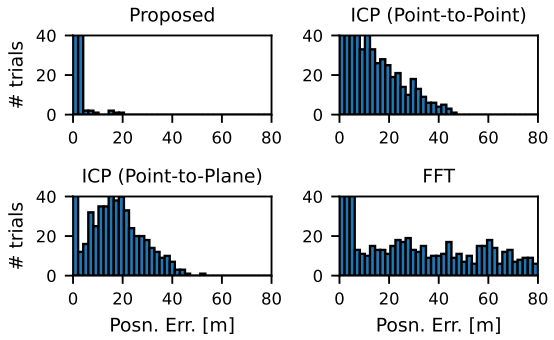


Fig. 17: Position measurement errors

all the grid peaks and finding the tightest cluster (similar to Sec. D).

The LIDAR measurements are simulated using elevation values on the DEM. This gives ICP and image template matching an advantage, since they compute position values by finding a transformation that matches the two. The DEM contains vegetation and other small features that may change over time. The proposed approach provides accurate position values *despite* the presence of these features, since they are removed when the phase candidate dictionary is encoded (see Fig. 6, Step 1). This suggests that the proposed approach may be more robust to differences between real-world measured LIDAR point clouds and the DEM than traditional scan-matching techniques.

Finally, we address the difference in memory requirement between the three approaches. The phase candidate dictionary is made up of 50×50 binary phase candidate matrices. The phase candidate matrices are encoded for 25 grids using 31 elevation bins of size 2 m. Therefore, the size of the phase candidate dictionary is $50 \times 50 \times 25 \times 31$ bits, or about 0.24 MB. If the elevation values in the DEM are encoded at half-precision, then the size of the DEM is $800 \times 800 \times 16$ bits, or about 1.25 MB. Thus, the proposed approach achieves acceptable position accuracy with an onboard data structure five times smaller than the DEM.

V. Conclusion

This article presents a terrain-relative navigation filter with a novel position update inspired by navigation in nature. Instead of matching elevation measurements to an onboard terrain map, the filter uses simple dictionary lookups to compute phase values from LIDAR measurements. Due to the linear relationship between the physical space and the phase space, the position measurement uncertainty can be computed online at measurement time. We also introduce an altitude estimation technique for cases where the true altitude is unknown but the altitude uncertainty is well-characterized.

This iteration of the work was completed with an eye toward hardware implementation. The elevation bins are large; 2 m gives plenty of room for noisy altitude estimates and large LIDAR range errors. A strict phase measurement rejection criterion and likelihood weighting allow the filter to deliver accurate position estimates despite the relatively small size of the phase candidate dictionary. Large grid scales make tight clusters of activity obvious, which allows for robust initialization when there is little or no prior position information.

One remaining challenge is implementing a full six degree-of-freedom state estimator. In general, terrain-aided navigation algorithms that rely on elevation measurements benefit from the range sensor pointing downward at all times; relaxing this assumption and accounting for the correlation between pitch/roll angles and measured elevation is left as future work. We would also like to decrease the field of view of the LIDAR. Like other TRN methods, the proposed approach suffers when the measured terrain is insufficiently distinguishable from other parts of the map. Measurement uniqueness can always be improved by including more of the terrain; the challenge is to localize with respect to an image that represents the smallest distinguishable subregion.

Power considerations are an important factor to adopting a given onboard navigation approach. One of the main advantages to adopting a more neuro-inspired approach to localization is that it becomes a better fit for implementation on emerging neuromorphic hardware platforms. As contrasted with more traditional von Neumann architectures, neuromorphic platforms are compute accelerators developed according to neural design principles such as distributed event-based processing and processing in memory architectures [22], [23]. Furthermore, these platforms have demonstrated significant power efficiency in accelerating both neural and non-neural algorithms [24], [25]. Similar to the improved throughput achieved through other compute accelerators such as GPUs, there are latency advantages to neuromorphic solutions as well, where there is an algorithmic tradeoff toward distributed computation over simpler serial components, as contrasted with singular complex serial computation.

Acknowledgment

This work was supported by a Sandia National Laboratory grant to the University of Texas at Austin. The authors would like to thank Kyle Morgenstein for his early work on the project and Tucker Haydon for his advice on numerous topics, as well as his care and attention during the review and approval process.

REFERENCES

[1] F. Wang, C. Teeter, S. Luca, S. Musuvathy, and J. B. Aimone, "Localization through grid-based encodings on digital elevation models," in *ICONS*, 2022.

[2] J. P. Golden, "Terrain contour matching (tercom): a cruise missile guidance aid," in *Image processing for missile guidance*, vol. 238. SPIE, 1980, pp. 10–18.

[3] L. Hostetler and R. Andreas, "Nonlinear kalman filtering techniques for terrain-aided navigation," *IEEE Transactions on Automatic Control*, vol. 28, no. 3, pp. 315–323, 1983.

[4] P. J. Besl and N. D. McKay, "Method for registration of 3-d shapes," in *Sensor fusion IV: control paradigms and data structures*, vol. 1611. Spie, 1992, pp. 586–606.

[5] R. A. Hewitt, T. P. Setterfield, and N. Trawny, "Lidar-based map relative localization performance analysis for landing on europa," in *2021 IEEE Aerospace Conference (50100)*. IEEE, 2021, pp. 1–13.

[6] T. P. Setterfield, R. A. Hewitt, P.-T. Chen, A. T. Espinoza, N. Trawny, and A. Katake, "Lidar-inertial based navigation and mapping for precision landing," in *2021 IEEE Aerospace Conference (50100)*. IEEE, 2021, pp. 1–19.

[7] R. E. Gold, S. G. Catalan, B. A. Jones, and R. Zanetti, "Extending capabilities of crater navigation and timing for autonomous lunar orbital operations," in *Space Imaging Workshop*, 2022.

[8] L. Downes, T. J. Steiner, and J. P. How, "Deep learning crater detection for lunar terrain relative navigation," in *AIAA SciTech 2020 Forum*, 2020, p. 1838.

[9] R. Moghe and R. Zanetti, "A deep learning approach to hazard detection for autonomous lunar landing," *The Journal of the Astronautical Sciences*, vol. 67, no. 4, pp. 1811–1830, 2020.

[10] Driver, Travis*, Tomita, Kento*, K. Ho, and P. Tsiotras, "Deep monocular hazard detection for small body landing," in *AAS/AIAA Space Flight Mechanics Meeting*, 2023, pp. 1–17, *These authors contributed equally to this work.

[11] A. Scorsoglio, A. D'Ambrosio, L. Ghilardi, B. Gaudet, F. Curti, and R. Furfaro, "Image-based deep reinforcement meta-learning for autonomous lunar landing," *Journal of Spacecraft and Rockets*, vol. 59, no. 1, pp. 153–165, 2022.

[12] C. Barry, R. Hayman, N. Burgess, and K. J. Jeffery, "Experience-dependent rescaling of entorhinal grids," *Nature neuroscience*, vol. 10, no. 6, pp. 682–684, 2007.

[13] W. Dorrell, P. E. Latham, T. E. Behrens, and J. C. Whittington, "Actionable neural representations: Grid cells from minimal constraints," *arXiv preprint arXiv:2209.15563*, 2022.

[14] K. Michaelson, F. Wang, and R. Zanetti, "Terrain-relative navigation with neuro-inspired elevation encoding," in *ION PLANS*, 2023.

[15] P. Passalacqua, "Austin, TX, rapid response, 2015 airborne lidar survey," National Center for Airborne Laser Mapping (NCALM), 2015.

[16] *Ultra-high performance inertial measurement unit (IMU): STIM300 Product Brief*, Sensonor, Aug 2017.

[17] *Data sheet: BMM150*, Bosch, Apr 2020, rev. 1.4.

[18] T. Ainscough, R. Zanetti, J. Christian, and P. D. Spanos, "Q-method extended kalman filter," *Journal of Guidance, Control, and Dynamics*, vol. 38, no. 4, pp. 752–760, 2015.

[19] S. Rusinkiewicz and M. Levoy, "Efficient variants of the icp algorithm," in *Proceedings third international conference on 3-D digital imaging and modeling*. IEEE, 2001, pp. 145–152.

[20] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A modern library for 3D data processing," *arXiv:1801.09847*, 2018.

[21] T. Solstad, E. I. Moser, and G. T. Einevoll, "From grid cells to place cells: a mathematical model," *Hippocampus*, vol. 16, no. 12, pp. 1026–1031, 2006.

[22] C. D. Schuman, T. E. Potok, R. M. Patton, J. D. Birdwell, M. E. Dean, G. S. Rose, and J. S. Plank, "A survey of neuromorphic computing and neural networks in hardware," *arXiv preprint arXiv:1705.06963*, 2017.

[23] J. B. Aimone, "Neural algorithms and computing beyond moore's law," *Communications of the ACM*, vol. 62, no. 4, pp. 110–110, 2019.

[24] M. Davies, A. Wild, G. Orchard, Y. Sandamirskaya, G. A. F. Guerra, P. Joshi, P. Plank, and S. R. Risbud, "Advancing neuromorphic computing with loihi: A survey of results and outlook," *Proceedings of the IEEE*, vol. 109, no. 5, pp. 911–934, 2021.

[25] J. D. Smith, A. J. Hill, L. E. Reeder, B. C. Franke, R. B. Lehoucq, O. Parekh, W. Severa, and J. B. Aimone, "Neuromorphic scaling advantages for energy-efficient random walk computations," *Nature Electronics*, vol. 5, no. 2, pp. 102–112, 2022.

Kristen A. Michaelson obtained a B.S. in Mechanical Engineering from Brown University in 2016 and an M.S. in Aerospace Engineering from the University of Texas at Austin in 2020. She is currently a Ph.D. student in Dr. Zanetti's Nonlinear Estimation and Autonomy Research group. Kristen's research interests include nonlinear estimation, terrain-relative navigation, and particle filers.

Felix Wang Felix Wang is a Senior Member of Technical Staff in the Cognitive and Emerging Computing Department at Sandia National Laboratories. Prior to Sandia, he obtained a PhD in Electrical and Computer Engineering from the University of Illinois at Urbana-Champaign in 2018, with a focus in signal processing. Felix's research interests include biologically inspired algorithms, neuromorphic and neural computing, representation learning, and distributed computation.

Renato Zanetti (Senior Member, IEEE) is an Assistant Professor of Aerospace Engineering at The University of Texas at Austin. Prior to joining UT he worked as an engineer at the NASA Johnson Space Center and at Draper Laboratory. Renato's research interests include nonlinear estimation, onboard navigation, and autonomous aerospace vehicles.