Terrain-aided Navigation for an Underwater Glider

Brian Claus
Department of Ocean and Naval Architecture Engineering, Memorial University, St. John's, Newfoundland, Canada
Ralf Bachmayer
Department of Ocean and Naval Architecture Engineering, Memorial University, St. John's, Newfoundland, Canada
e-mail: bachmayer@mun.ca

Received 21 January 2014; accepted 7 November 2014

A terrain-aided navigation method for an underwater glider is proposed that is suitable for use in ice-covered regions or areas with heavy ship traffic where the glider may not be able to surface for GPS location updates. The algorithm is based on a jittered bootstrap algorithm, which is a type of particle filter that makes use of the vehicle's dead-reckoned navigation solution, onboard altimeter, and a local digital elevation model (DEM). An evaluation is performed through postprocessing offline location estimates from field trials that took place in Holyrood Arm, Newfoundland, overlapping a previously collected DEM. During the postprocessing of these trials, the number of particles, jittering variance, and DEM grid cell size were varied, showing that convergence is maintained for 1,000 particles, a jittering variance of 15 m^2, and a range of DEM grid cell sizes from the base size of 2 m up to 100 m. Using nominal values, the algorithm is shown to maintain bounded error location estimates with root-mean-square (RMS) errors of 33 and 50 m in two sets of trials. These errors are contrasted with dead-reckoned errors of 900 m and 5.5 km in those same trials. Online open-loop field trials were performed for which RMS errors of 76 and 32 m- were obtained during 2-h-long trials. The dead-reckoned error for these same trials was 190 and 90 m, respectively. The online open-loop trials validate the filter despite the large dead-reckoned errors, single-beam altitude measurements, and short test duration. © 2015 Wiley Periodicals, Inc.

1. INTRODUCTION

Over the past decade, underwater gliders, a type of autonomous underwater vehicle (AUV), have proven their ability to persistently monitor ocean processes in a wide range of conditions (Schofield et al., 2007). However, operational gaps still exist in regions where surface access is limited. Surface access is particularly challenging in regions with ice cover or in regions with heavy ship traffic. Sustained underwater observations in regions with ice cover are of particular importance to climate change research and polar exploration efforts, making tools to overcome these observational obstacles a key development in improving global climate change predictions (Rintoul et al., 2012).

Existing methods for underwater navigation may be grouped into geophysical, acoustic, inertial, and model-based techniques (Kinsey, Eustice, & Whitcomb, 2006; Paull, Saeedi, Seto, & Li, 2014; Stutters, Liu, Tiltman, & Brown, 2008). Of the available methods, only the acoustic baseline and geophysical-aided navigation methods provide bounded location estimates. For long-range vehicles traveling hundreds of kilometers, acoustic methods require either a very-low-frequency sound source, a net of multiple sound sources, or a surface vessel shadowing the AUV. In regions with periodic ice cover, a shadowing vehicle is often not practical, and in shipping lanes it can present a navigational hazard. Infrastructure costs for acoustic nets of standard long baseline (LBL) systems are prohibitively expensive once the vehicle's range extends past the nominal range of tens of kilometers. Low-frequency sound sources are perhaps the only alternative left for acoustic localization of long-range vehicles with ranges of up to hundreds of kilometers (Lee & Gobat, 2006; Webster, Lee, & Gobat, 2014). However, these systems are expensive to deploy and maintain, particularly in regions with ice cover, which tend to be remote. Terrain-aided methods have no infrastructure requirements, making them attractive for long-range vehicles from a cost perspective. They do have the limitations of requiring a detailed digital elevation model (DEM) of the region and considerable design effort to tailor the algorithms to a specific platform.

In general, geophysical techniques rely on feature variability in past measurements that are compared to a DEM. Statistical estimators are then used to generate an estimate of the current position of the vehicle given the prior measurement and position estimates. A few successful demonstrations of these techniques on real systems have been shown (Meduna, Rock, & McEwen, 2010; Morice, Veres, & McPhail, 2009; Nygren, 2008). The majority of these demonstrations use the existing fused navigation solution from the high-accuracy inertial navigation system (INS)

Author to whom all correspondence should be addressed: Brian Claus, bclaus@mun.ca

aided by a Doppler velocity log (DVL) as an input to the terrain-aided navigation (TAN) algorithm (Chen, Wang, McDonald-Maier, & Hu, 2013). This high-accuracy navigation update is then combined with the measurement update from a multibeam sonar or DVL that measures the bathymetry when combined with the vehicle depth given by the pressure sensor (Bergem, 1993; Nygren, 2005). However, it has recently been shown that lower-grade inertial sensors aided by a DVL may be used by directly including the inertial and DVL measurements in the filter used for the terrain-aided navigation (Meduna, 2011). This tight coupling was achieved through increasing the number of states in the terrain algorithm, increasing the reliability and accuracy. General limitations of bathymetric techniques are due to the range of the acoustic sensors, generally around 100 m, low terrain variability, which increases the uncertainty of the estimate, and the power requirements of the inertial and ranging sensors. Recent attempts have increased the robustness of terrain relative navigation techniques through adapting to changes in terrain variability and employing a series of statistical consistency checks (Dektor & Rock, 2012; Houts, Dektor, & Rock, 2013).

Most TAN algorithms use a preexisting DEM to bound the position error of an AUV through comparisons with the vehicle's water depth estimates, generating a location estimate that is then integrated back into the navigation solution. Alternatively, there has been some progress toward simultaneous localization and mapping (SLAM) methods on underwater vehicles. These techniques compile a DEM as the vehicle gathers water depth estimates, ensuring all the measurements are self-consistent, and using this to reduce the navigation error (Barkby, Williams, Pizarro, & Jakuba, 2011). Other geophysical parameters, such as the earth's magnetic field, have been proposed to augment TAN methods to increase their robustness, but they have yet to have a practical demonstration (Kato & Shigetomi, 2009; Teixeira & Pascoal, 2008).

This work proposes a TAN method that has been designed around a 200 m, Slocum Electric underwater glider operating in profiling modes. The Slocum Electric underwater glider is already equipped with the sensors needed for a rudimentary terrain-aided navigation scheme. As such, a brief overview of the vehicle is presented followed by a look at the theory for terrain-aided navigation using a jittered sequential importance resampling (SIR) or bootstrap algorithm. The SIR algorithm belongs to the class of algorithms that have been termed particle filters. The jittered bootstrap algorithm is then applied to the underwater glider. This application makes use of the existing dead-reckoning algorithm in the underwater glider for the navigation update. The measurement update is composed of a water depth model based on the ray-traced altitude, the vehicle depth, and the tidal estimate. This application of terrain-aided navigational techniques to an underwater glider using only a single-beam altimeter and a low-accuracy dead-reckoning algorithm is believed by the authors to be unique. The glider TAN algorithm is possible through simultaneously relaxing the accuracy requirements of the navigation estimate from the order of meters in the case of the prior art to tens or hundreds of meters and increasing the process noise added to the navigation update. These modifications are complementary as the increased navigation update process noise serves to increase the noise in the TAN estimate but also makes the method more robust to complete divergence.

The resulting algorithm is presented in detail with the necessary processing steps explained. It is then used to post-process location estimates for an underwater glider using data collected in offline field trials that took place in 2010 and 2012 in Holyrood Arm, Newfoundland, which overlap a ship-based multibeam bathymetric DEM collected by Memorial University's Marine Institute. The offline results include an analysis of the number of particles required for convergence, the effects of the jitter variance, and the impact of coarser DEM grids, and they show that the algorithm is capable of producing bounded location estimates for underwater gliders without surface access. Finally, the methodology is evaluated through online, open loop trials which took place in 2014 in Holyrood Arm, Newfoundland. In these trials, 1 h northward and southward missions were performed that show the functionality of the algorithm onboard an underwater glider.

2. UNDERWATER GLIDERS

The Slocum underwater glider uses active ballast changes as its main propulsive force (Rudnick, Davis, Eriksen, Fratantoni, & Perry, 2004). The ballast system is located in the nose of the vehicle and creates an upward or downward pitching moment, assisting in the change in pitch necessary to form a suitable glide path. A mass shifting mechanism attached to one of the battery packs acts as a vernier pitch adjustment mechanism to control the vehicle pitch to a precise angle. The cyclic positive and negative vertical motion due to the forces from the ballast system generates lift due to the wings and vehicle body, which moves the vehicle forward when the vehicle pitch is within a certain range of values. For the Slocum glider, the pitch values that produce the most forward movement are in the range of 20–30 deg.

Before a given deployment, a glider's mass is adjusted such that the vehicle is neutrally buoyant in seawater and the center of mass is directly below the center of buoyancy when the ballast system and the mass shifting mechanism are both in the center of their range. At this stage, the glider user plans the mission by selecting appropriate waypoints for the vehicle to navigate to, as well as other vehicle parameters such as desired pitch, surfacing conditions, and minimum altitude, among others. During the mission, the vehicle attempts to reach the waypoints by moving forward in its cyclic pattern and by steering using its rudder and magnetic compass. If the vehicle reaches a surfacing condition, such as hitting a waypoint or being too long underwater, it will surface, obtain a new GPS fix, reestablish the range and bearing to the waypoint, and continue on its way. In this navigational scheme, the vehicle tracks its progress to the waypoints, using dead-reckoning while underwater.

2.1. Dead Reckoning

Dead-reckoning systems keep track of the location of a vehicle using an initial location and adding the incremental displacements given by the product of the vehicle velocity and the time difference between measurements. The glider's dead-reckoning system uses the pressure and attitude sensors to estimate the horizontal velocity components. Toward that end, the vehicle's horizontal velocity v_h is computed by

$$v_h = \frac{v_z}{\tan(\xi)}, \tag{1}$$

where v_z is the vertical velocity as given by the first derivative of the vehicle depth, measured by the pressure sensor, and ξ is the glide path angle. The glide path angle ξ is composed of the vehicle pitch θ and angle of attack α as in

$$\xi = \theta + \alpha. \tag{2}$$

During the test deployments, the angle of attack was not included in the glide path estimate, resulting in the vehicle pitch being the same as the glide path angle. However, more recent versions of the vehicle software include an estimator that calculates an angle of attack based on the vehicle parameters, and measurements of the vehicle pitch and depth rate (Merckelbach, Smeed, & Griffiths, 2010). The horizontal vehicle velocity may be further decomposed into the horizontal velocity components and multiplied by the time between measurements ΔT to produce the state updates Δx and Δy as in

$$\Delta x = \Delta T v_h \sin(\psi + \delta), \tag{3}$$

$$\Delta y = \Delta T v_h \cos(\psi + \delta), \tag{4}$$

where ψ is the vehicle magnetic heading and δ is the magnetic declination. The resulting dead-reckoning equations add the state updates from Eqs. (3) and (4) to the prior state as in

$$x_{k+1} = x_k + \Delta x, \tag{5}$$

$$y_{k+1} = y_k + \Delta y, \tag{6}$$

where k is the time step and x and y give the vehicle location in the local coordinate frame. The dead-reckoning scheme for underwater gliders described above works well for profiling missions with ready access to the surface for GPS updates. However, as the vehicle has no direct measurement of its speed over ground, the accuracy of its velocity estimates is subject to *a priori* unknown and changing water velocities. The vehicle attempts to compensate for this by assuming these water velocities are solely responsible for the difference between the dead-reckoning location and the first GPS fix upon surfacing. This depth averaged water velocity estimate is then used as a corrective term to the vehicle velocity estimate in the next dive cycle. In highly stratified or dynamic areas of the ocean, this assumption breaks down and as such the dead-reckoning algorithm can be subject to significant error. Additionally, in regions where surface access is denied, this water velocity estimation method is not possible, resulting in a further degraded dead-reckoning solution.

2.2. Water Depth Measurement Model

The water depth estimate from the glider combines the depth of the vehicle given by the pressure sensor with the altitude of the vehicle given by the altimeter. The altimeter is a 170 kHz, narrow beam sonar (3°), located in the nose of the vehicle and angled at 26° forward of the vertical such that at the nominal dive angle, the altimeter points straight down. A first-generation 200 m Slocum glider has a 300 psi pressure transducer ported in the rear bulkhead and connected to the sensor located in the aft of the glider. Second generation (G2) Slocum gliders have a 2,000 psi pressure transducer for the vehicle control ported in the same manner. The G2's higher pressure rating degrades the accuracy of the depth estimates, requiring the use of the conductivity, temperature, and depth (CTD) sensor's values instead. The locations and orientations of these devices relative to the center of buoyancy for a first-generation Slocum glider are illustrated in Figure 1.

Since the altimeter and pressure transducer are not colocated, their vertical separation must be accounted for in the water depth estimate. Additionally, as the density of water changes, the speed and direction of the sound through the water change as well. To correct for this, a simple ray-tracing procedure, shown in Algorithm 1, is performed that uses Snell's law to compute the path based on the sound

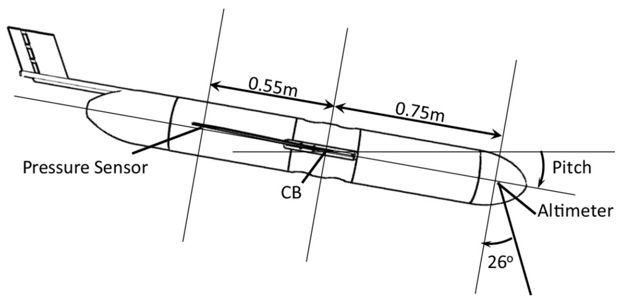

Figure 1. Locations of the altimeter and pressure transducer relative to each other and the center of buoyancy, where CB is the center of buoyancy.

speed profile obtained from the glider's CTD sensor and the initial beam angle (Hodges, 2010).

Algorithm 1 Glider Altitude Ray Tracing

```
 1: [$\hat{z}_{rt,k}$,$\Delta\mathbf{x}_{rt,k}$]=RAYTRACE [$z_{g,k}$,$z_{a,k}$,$\phi_{g,k}$,$\theta_{g,k}$,$\psi_{g,k}$,SV($z$)]
 2: initialize $\hat{z}_{rt,k}$ to $z_{g,k}$
 3: initialize $x$ and $t$ to zero
 4: set increment: $\Delta z = 0.1$
 5: compute beam angle from vertical:
    $\theta_v = \arccos[\cos\phi_{g,k}\cos(\theta_{g,k} - 26^\circ)]$
 6: compute one-way travel time: $t_{1\text{way}} = \frac{z_{a,k}}{1{,}500}$
 7: compute c: $c$ =LINEARINT[SV($z$),$\hat{z}_{rt,k}$]
 8: compute a: $a = \sin\frac{\theta_v}{c}$
 9: while $t < t_{1\text{way}}$ do
10:     increment water depth estimate: $\hat{z}_{rt,k} += \Delta z$
11:     update x: $x += \Delta z\tan(\theta_v)$
12:     update t: $t += \frac{\sqrt{\Delta z\tan(\theta_v)+\Delta z^2}}{c}$
13:     update c: $c$ =LINEARINT[$SV(z)$,$\hat{z}_{rt,k}$]
14:     update angle: $\theta_v = \arcsin(ac)$
15: end while
16: compute beam heading:
    $\psi_{b,k} = \arctan\left(\frac{\sqrt{2z_{rt,k}^2(1-\cos\phi_{g,k})}}{(\hat{z}_{a,k}-z_{g,k})\tan(\theta_{g,k}-26^o)}\right)$
17: compute x offset: $\Delta x_{rt,k} = -x\sin(\psi_{g,k} + \psi_{b,k})$,
18: compute y offset: $\Delta y_{rt,k} = -x\cos(\psi_{g,k} + \psi_{b,k})$
```

The vehicle altimeter assumes a uniform sound speed of 1,500 m/s and reports an altitude, $z_{a,k}$. The ray tracing is performed by backing the travel time out of this initial estimate, using the glider roll $\phi_{g,k}$ and pitch $\theta_{g,k}$ to compute the beam angle from vertical θ_v, and using the vehicle depth $z_{g,k}$ as the starting depth. Initial estimates of the speed of sound c are computed through linear interpolation, which is used to compute the ray-tracing constant a. The algorithm then iterates the water depth estimate by Δz until the one-way travel time is exceeded, keeping track of the horizontal distance x the beam travels. The results of this calculation are the ray-traced water depth, $\hat{z}_{rt,k}$, and the measurement location offsets, $\Delta\mathbf{x}_{rt,k}$, which are computed using the horizontal beam distance and the combined beam and vehicle headings $\psi_{b,k}$ and $\psi_{g,k}$.

The tidal variation is also included in the water depth model to account for the time-varying signal of the water depth. The tidal correction, $z_{T,k}$, may be either historical measurements or from a predictive model. In the offline field trials, historical measurements from the St. John's, Newfoundland station in the Canadian Tides and Water Levels Data Archive were used (DFO, 2013). In the online trials, the tidal component was not included due to the short duration of those tests.

The resulting water depth measurement model is given as

$$\mathbf{z}_k = \hat{z}_{rt,k} + x_{ap}\sin(\theta_{g,k}) + z_{T,k} + z_b, \tag{7}$$

where $\hat{z}_{a,k}$, $\theta_{g,k}$, $z_{g,k}$, and $z_{T,k}$ are the ray-traced altitude, vehicle pitch, glider depth, and tidal signal at time step k. The distance from the pressure sensor to the altimeter along the vehicle axis is x_{ap} and the DEM depth bias is z_b, which is defined later in Section 4.

3. TERRAIN-AIDED NAVIGATION

The general terrain-aided navigation (TAN) problem attempts to localize a body using *a priori* digital elevation models (DEMs), some knowledge of the body's movements, and measurements that relate the body to the DEM. One set of solutions to these types of problems are broadly based on sequential importance sampling methods, also known as particle filters. A visual explanation of the particle filter is given by Fox, Hightower, Liao, Schulz, & Borriello (2003), and more theoretical treatments by Sanjeev Arulampalam, Maskell, Gordon, & Clapp (2002); Ristic, Arulampalam, & Gordon (2004), and Doucet, Godsill, & Andrieu (2000). Another helpful treatment is presented by Simon (2006), which provides some history of the particle filter and places it in the context of other estimation techniques.

The generic particle filter algorithm draws many guesses, or particles of where the most recent water depth measurement might be, and it compares the measurement to the DEM for each particle location. These particles are drawn according to an importance density function, which attempts to allocate particles to the important part of the state space based on all of the prior states and all of the prior measurements. The choice of importance density function is a significant consideration in the design of a particle filter with respect to a particular application. Each particle's location has an associated value from the DEM that is compared to the water depth measurement to evaluate its weight. These weights are normalized such that the sum of all the weights is equal to 1, shaping them into a probability distribution. The state estimate is then given by the sum of the product of each particle's location with its weight, essentially computing the centroid of the particle cloud.

The classic particle filter algorithm presents several difficulties with its implementation. The first is related to the choice of the importance density function, which for the original formulation requires complete knowledge of the entire set of states and measurements. While it has been shown that an optimal form of the importance density can be approximated, it is only usable if analytic forms of the state transition probabilities are available such that the integrals have closed-form solutions (Doucet et al., 2000). This difficulty has led to suboptimal forms being used, such as the transitional prior, which simply applies the state update to the prior particle locations.

However, in using the transitional prior, as the prior densities accumulate, the weights of the particles become concentrated on very few particles, with the majority of particles having little weight and therefore contributing very little to the state estimate. This concentration of particle weights is termed sample impoverishment or degeneracy. To help improve this situation, the particles are often resampled such that those particles with very little weight are discarded and the particles with a lot of weight are divided into more. This process leads to another problem in which the particle cloud becomes very small and no longer provides any corrective behavior, termed particle collapse. One method used to deal with particle collapse is through the jittering or roughening of the particles in which their locations have some process noise added to spread them back out in the state space (Gordon, Salmond, & Smith, 1993).

The result of these simplifications and fixes is termed the sequential importance resampling method. In this method, resampling is only done when needed as determined by some metric. When resampling is performed at every time step, the method is often termed the bootstrap method (Gordon et al., 1993). In the bootstrap method, the evaluation of the weights is simplified as they will have equal value after resampling, however the frequent resampling also accentuates the particle collapse, requiring stronger jittering. The addition of jittering to the algorithm essentially introduces additional process noise into the estimator. This increased process noise has been shown to improve the robustness of the estimator to complete failures, allowing reconvergence after periods of sparse measurements, flat terrain, or DEM artifacts (Bar-Shalom, Li, & Kirubarajan, 2004; Houts et al., 2013). The improved robustness comes at the expense of higher estimator noise, which for a survey grade AUV is problematic but is less of an issue to localization for underwater gliders.

The jittered bootstrap method is used in the remainder of this work and is presented in Algorithm 2 and illustrated in Figure 2.

This algorithm takes as inputs the prior particles $\{\mathbf{x}_{k-1}^i\}_{i=1}^N$, the state update $\Delta\mathbf{x}_k$, and the water depth estimate $\mathbf{z}_k$ at time step k, where N is the number of particles and i is the particle index. The outputs of the method are the state estimate $\hat{\mathbf{x}}_k$ and the particle locations $\{\mathbf{x}_k^i\}_{i=1}^N$, which are then saved for the next iteration. The operation of the algorithm begins with computing a particle jitter $\mathbf{r}_k$ based on a normal distribution with zero mean and σ_j^2 variance. Particle i for time step k is then drawn by updating the prior particle location $\mathbf{x}_{k-1}^i$ with the state update $\Delta\mathbf{x}_k$ and applying the particle jitter computed previously. The weight of each particle $\tilde{w}_k^i$ is then evaluated as the probability of the water depth estimate given the particle's location. This process is repeated N times, drawing all of the particles and computing their weights. These weights are then normalized by the sum of the weights s_w. The particles are then resampled and

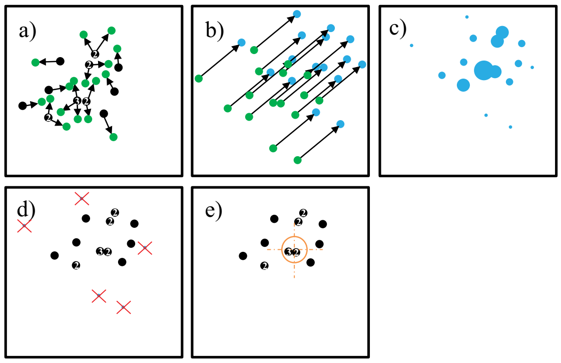

Figure 2. An illustration of the jittered bootstrap algorithm with two state variables for a small number of particles, where (a) shows the jittering of the particles; (b) shows the particles before and after the state update, as indicated by the arrows, is applied; (c) shows the particle weights, where larger particles indicate a higher weight based on how closely the water depth measurement matches the DEM; (d) shows the resampling process, where small particles are discarded as indicated by the crosses and large particles divided; and (e) shows the state estimation as indicated by the cross hair. Particles with numbers overlaid represent a stack of particles.

Algorithm 2 Jittered Bootstrap

```
1:  [$\hat{\mathbf{x}}_k$,$\{\mathbf{x}_k^i\}_{i=1}^N$] =BOOTSTRAP [$\{\mathbf{x}_{k-1}^i\}_{i=1}^N$,$\Delta\mathbf{x}_k$,$\mathbf{z}_k$]
2:  for $i = 1$ TO $N$ do
3:      compute jitter: $\mathbf{r}_k = \mathcal{N}(0, \sigma_j^2)$
4:      state update: $\mathbf{x}_k^i = \mathbf{x}_{k-1}^i + \Delta\mathbf{x}_k + \mathbf{r}_k$
5:      compute weights: $\tilde{w}_k^i = p(\mathbf{z}_k|\mathbf{x}_k^i)$
6:  end for
7:  calculate total weight: $s_w = \sum_{i=1}^{N} \tilde{w}_k^i$
8:  for $i = 1$ TO $N$ do
9:      normalize weights: $w_k^i = \frac{\tilde{w}_k^i}{s_w}$
10: end for
11: resample: [$\{\mathbf{x}_k^i\}_{i=1}^N$] =RESAMPLE[$\{\mathbf{x}_k^i,w_k^i\}_{i=1}^N$]
12: estimate state: $\hat{\mathbf{x}}_k = \frac{1}{N}\sum_{i=1}^{N} \mathbf{x}_k^i$
```

the state estimate $\hat{\mathbf{x}}_k$ is computed through the sum of the particles divided by the total number of particles.

The resampling algorithm selected for this work is the systematic resampling method as outlined in Algorithm 3 (Kitagawa, 1996).

This method provides a fast and simple way to represent the probability density through evenly weighted particles, requiring the particle locations and weights as inputs and providing the resampled particles as outputs. The algorithm operates by first taking the cumulative sum, $\{c_k\}_{i=1}^N$, of the particle weights, forming an increasing set of values from 0 to 1. A random number r is then drawn from a

Algorithm 3 Systematic Re-Sampling

```
1: $[\{\mathbf{h}_k^i\}_{i=1}^N]$=RESAMPLE$[\{\mathbf{x}_k^i, w_k^i\}_{i=1}^N]$
2: cumulative sum: $\{c_k\}_{i=1}^N = \text{CUSUM}(\{w_k^i\}_{i=1}^N)$
3: for $i = 0$ TO $N$ do
4:     draw random number on [0 1]: $r$ =RAND(0,1)
5:     find where it belongs: $s_k^i$ =FIND$(r, \{c_k\}_{i=1}^N)$
6:     store particle: $\mathbf{h}_k^i = \{\mathbf{x}_k^i\}_{i=1}^N(s_k^i)$
7: end for
```

uniform distribution on the interval [0,1] and the index s_k^i of the cumulative sum found in which the random number is equal to the cumulative sum. The particle at that index is then stored for output. In this way, particles with high weight are divided into many particles as they occupy a large portion of the cumulative sum, and particles with small weights are discarded as they occupy a negligible portion of the cumulative sum.

4. DIGITAL ELEVATION MODEL

The DEM used in this work is constructed from ship-based multibeam surveys done by Memorial University's Marine Institute. The DEM is gridded by latitude and longitude at an equivalent distance of roughly 2 m. Since the DEM is generated from data collected by a ship, an error model is used with variance σ^2_{DEM} as in Lagadec, Cecchi, Rixen, & Grasso (2010),

$$\sigma^2_{\text{DEM}} = \frac{1}{2}\sqrt{1 + (0.023 z_{\text{DEM}})^2}, \tag{8}$$

where z_{DEM} is the water depth of a particular location of the DEM. The value of the DEM at each particle's location is then obtained through bilinear interpolation of the gridded data. The DEM was found to have a depth bias of several meters, which is accounted for in the water depth estimate as z_b. The depth bias is due to a combination of a misregistration of the multibeam data during processing, errors in the vehicle pressure sensor, and errors in the tidal estimates.

Two flags are inserted into the DEM, one at the shoreline and one at the DEM boundary, to allow for decisions to be made when particles return one of these flags. The shoreline flag prevents values for the DEM being returned where there is no water, but it does not transition the algorithm to a dead-reckoned solution. The DEM boundary flag allows the algorithm to check if there is a new DEM to load, and if not, it reverts to the dead-reckoned navigation solution.

5. GLIDER TAN

An unmodified Slocum Electric underwater glider has rudimentary versions of all of the components necessary for an implementation of the jittered bootstrap method shown in Algorithm 2. It computes a dead-reckoned navigation solution that may be used for the state update as in Eqs. (5) and (6), and it has an altimeter and pressure sensor that, when combined, allows for a water depth estimate as in Eq. (7). The only other inputs to the bootstrap algorithm are the prior particles, which once initialized are propagated from time step to time step, and the DEM. After initialization of the method, the algorithm computes a location estimate for every time step corresponding to each successful altimeter reading. The complete glider TAN method is shown in Algorithm 4 as an extension of the bootstrap method shown in Algorithm 2 with the initialization procedure as follows:

1. get initial Lat/Lon: $\hat{\text{Lat}}_{k-1}$,$\hat{\text{Lon}}_{k-1} = \text{Lat}_{\text{GPS}}$,$\text{Lon}_{\text{GPS}}$
2. set local mission coordinate frame: $\hat{\mathbf{x}}_{k-1}$ ={0,0}
3. set prior particles: $\{\mathbf{x}_{k-1}^i\}_{i=1}^N = \mathbf{0}$
4. wait for first altimeter reading

The initialization procedure requires an initial location, which, on an underwater glider, is obtained by the GPS while the vehicle is at the surface before the mission begins. This initial location sets the origin for the local mission coordinate (LMC) reference frame in units of meters. The prior particles are initialized to zero, relying on the jittering of the particles to spread the particles by the appropriate amount. The glider TAN algorithm is now ready to iterate through each time step, where a time step corresponds to a successful altimeter reading.

The glider TAN algorithm's inputs are the prior particles $\{\mathbf{x}_{k-1}^i\}_{i=1}^N$, the prior state estimate $\hat{\mathbf{x}}_{k-1}$ in LMC and in latitude $\hat{\text{Lat}}_{k-1}$ and longitude $\hat{\text{Lon}}_{k-1}$, the state update $\Delta\mathbf{x}_k$ from the glider, the glider depth $z_{g,k}$, and the glider altitude $z_{a,k}$. Other inputs include $\phi_{g,k}$, $\theta_{g,k}$, and $\psi_{g,k}$, which are the glider's roll, pitch, and heading, the sound velocity profile SV(z), and the DEM.

The outputs of the algorithm are the glider TAN state estimate $\hat{\mathbf{x}}_k$ in LMC, and in latitude $\hat{\text{Lat}}_k$ and longitude $\hat{\text{Lon}}_k$, and the particles $\{\mathbf{x}_k^i\}_{i=1}^N$.

The algorithm begins by ray-tracing the altitude to obtain the ray-traced water depth $\hat{z}_{rt,k}$ and the water depth measurement offsets $\Delta\mathbf{x}_{rt,k}$ using Algorithm 1. The water depth estimate z_k is then computed according to Eq. (7) by adding the pressure sensor and altimeter vertical separation, the tidal signal, and the DEM depth bias to the ray-traced water depth. For the offline field trials, the DEM depth bias was computed by taking the average water depth estimate error over the entire set of measurements. However, an estimate of the DEM depth bias for the online trials was computed using the values from a set of initialization profiles.

The algorithm next enters the particle update and weight computation loop, which first checks the dead-reckoning flag. If the algorithm's dead-reckoning flag is not set, it proceeds to compute the particle's jitter $\mathbf{r}_k$, which is randomly generated using a normally distributed pseudorandom number generator based on the Mersenne Twister with variance σ_j^2 and zero mean (Matsumoto & Nishimura,

Algorithm 4 Glider TAN

```
1: [$\hat{\mathbf{x}}_k$,$\{\mathbf{x}_k^i\}_{i=1}^N$,$\hat{\text{Lat}}_k$,$\hat{\text{Lon}}_k$]=gTAN[$\{\mathbf{x}_{k-1}^i\}_{i=1}^N$,$\hat{\mathbf{x}}_{k-1}$,
   $\hat{\text{Lat}}_{k-1}$,$\hat{\text{Lon}}_{k-1}$,$\Delta\mathbf{x}_k$,$z_{g,k}$,$z_{a,k}$,$\phi_{g,k}$,$\theta_{g,k}$,$\psi_g$,SV($z$),DEM]
2: Ray trace altitude: [$\hat{z}_{rt,k}$,$\Delta\mathbf{x}_{rt,k}$] =RAYTRACE
   [$z_{g,k}$,$z_{a,k}$,$\phi_{g,k}$,$\theta_{g,k}$,$\psi_{g,k}$,SV($z$)]
3: Compute water depth estimate:
   $z_k = \hat{z}_{rt,k} + x_{ap}\sin(\theta_{g,k}) + z_{T,k} + z_b$
4: for $i = 0$ TO $N$ do
5:   if Not Dead-Reckoning then
6:     Compute jitter: $\mathbf{r}_k = \mathcal{N}(0, \sigma_j^2)$
7:     Compute particle location in LMC:
       $\mathbf{x}_k^i = \mathbf{x}_{k-1}^i + \Delta\mathbf{x}_k + \mathbf{r}_k + \Delta\mathbf{x}_{rt,k}$
8:     Convert LMC to Lat/Lon:
       [Lat$_k^i$,Lon$_k^i$]=LMC2LL[$\mathbf{x}_k^i$,$\hat{\mathbf{x}}_{k-1}$,$\hat{\text{Lat}}_{k-1}$,$\hat{\text{Lon}}_{k-1}$]
9:     if Lat$_k^i$,Lon$_k^i$ within DEM bounds then
10:      get DEM water depth:
           $z_{\text{DEM},k}$=BILINEAR[Lat$_k^i$,Lon$_k^i$,DEM]
11:      if Shoreline Flag then
12:        Set DEM water depth to zero: $z_{\text{DEM},k}$=0
13:      end if
14:      if Map Bounds Flag then
15:        Check for new DEM
16:        Set Dead-Reckon Flag
17:      end if
18:      compute DEM variance:
           $\sigma_{\text{DEM}}^2 = 1/2\sqrt{1 + (0.023z_{\text{DEM},k})^2}$
19:      compute weight: $\tilde{w}_k^i$ =
           NORMPDF[$z_k$,$z_{\text{DEM},k}$,$\sigma_{\text{DEM}}^2$]
20:    else
21:      Set Dead-Reckon Flag
22:    end if
23:  end if
24: end for
25: if Not Dead-Reckoning then
26:   compute particle weight sum: $s_w = \sum_{i=1}^{N} \tilde{w}_k^i$
27:   for $i = 1$ TO $N$ do
28:     normalize weights: $w_k^i = \frac{\tilde{w}_k^i}{s_w}$
29:   end for
30:   resample: $\{\mathbf{x}_k^i\}_{i=1}^N$=RESAMPLE[$\{\mathbf{x}_k^i, w_k^i\}_{i=1}^N$]
31:   compute state estimate in LMC: $\hat{\mathbf{x}}_k = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_k^i$
32:   convert LMC to Lat/Lon:
        [$\hat{\text{Lat}}_k$,$\hat{\text{Lon}}_k$]=LMC2LL[$\hat{\mathbf{x}}_k$,$\hat{\mathbf{x}}_{k-1}$,$\hat{\text{Lat}}_{k-1}$,$\hat{\text{Lon}}_{k-1}$]
33: else
34:   compute state estimate in LMC: $\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_{k-1} + \Delta\mathbf{x}_k$
35:   convert LMC to Lat/Lon:
      [$\hat{Lat}_k$,$\hat{Lon}_k$]=LMC2LL[$\hat{\mathbf{x}}_k$,$\hat{\mathbf{x}}_{k-1}$,$\hat{\text{Lat}}_{k-1}$,$\hat{\text{Lon}}_{k-1}$]
36:   Reset particles: $\{\mathbf{x}_k^i\}_{i=1}^N = \hat{\mathbf{x}}_k$
37:   Reset Dead-Reckon Flag
38: end if
```

1998). The particle's location is then computed in LMC through adding the state update $\Delta\mathbf{x}_k$, the particle jitter, and the water depth measurement offsets to the prior particle's locations. To interface with the DEM, the LMC location is converted to latitude and longitude by assuming the change in the LMC location, indicated by Δx and Δy, is equal to the length of an arc, as in

$$\text{Lat}_k = \text{Lat}_{k-1} + (\Delta y)/R_e, \qquad (9)$$

$$\text{Lon}_k = \text{Lon}_{k-1} + (\Delta x)/[R_e \cos(\text{Lat}_{k-1})], \qquad (10)$$

where R_e is the radius of the earth.

Once the particle's latitude and longitude are computed, the particle's location may be checked against the general DEM bounds. If it is within the bounds, the DEM water depth $z_{\text{DEM},k}$ is retrieved through bilinear interpolation, otherwise the dead-reckoning flag is set. If the DEM water depth is equal to the shoreline flag, then it is zeroed. If it is equal to the map bounds flag, then the dead-reckoning flag is set. Otherwise, the DEM variance is σ_{DEM}^2 computed as in Eq. (8). The particle weight is then computed by comparing the water depth estimate to the DEM water depth through a normal probability density function with the DEM variance. This procedure is repeated for all N particles.

The algorithm now moves on to check once more if the dead-reckoning flag is not set before computing the particle weight sum s_w and using it to normalize the weights. The particles are next resampled using the systematic resampling method described in Algorithm 3. After the resampling process, all of the particles have equal weight, allowing the state estimate to be computed in LMC using the mean of the particle locations. The state estimate is then converted to latitude and longitude, and the algorithm is done for this iteration.

If the dead-reckoning flag is set, meaning that at least one of the particles is outside of the map bounds, the algorithm skips directly to the dead-reckoning state estimation. In this case, the state estimates in LMC, latitude, and longitude are computed using only the state update, and the particles states are reset to the dead-reckoned state estimate. The dead-reckoning flag is then reset in case, for the next iteration, all of the particles are back within the map bounds.

6. OFFLINE FIELD TRIALS

The glider TAN algorithm was evaluated offline through two sets of field trials that overlapped the region of the DEM in Holyrood Arm of Conception Bay, Newfoundland. These experiments took place in October 2010 and October 2012 using a 200 m electric Slocum underwater glider. In the 2010 trials, the glider flew straight out of Holyrood Arm and past the boundary of the DEM for a total distance of approximately 12 km, and in the 2012 trials the vehicle flew in overlapping rectangles up Holyrood Arm for a total distance of approximately 91 km, as illustrated in Figure 3.

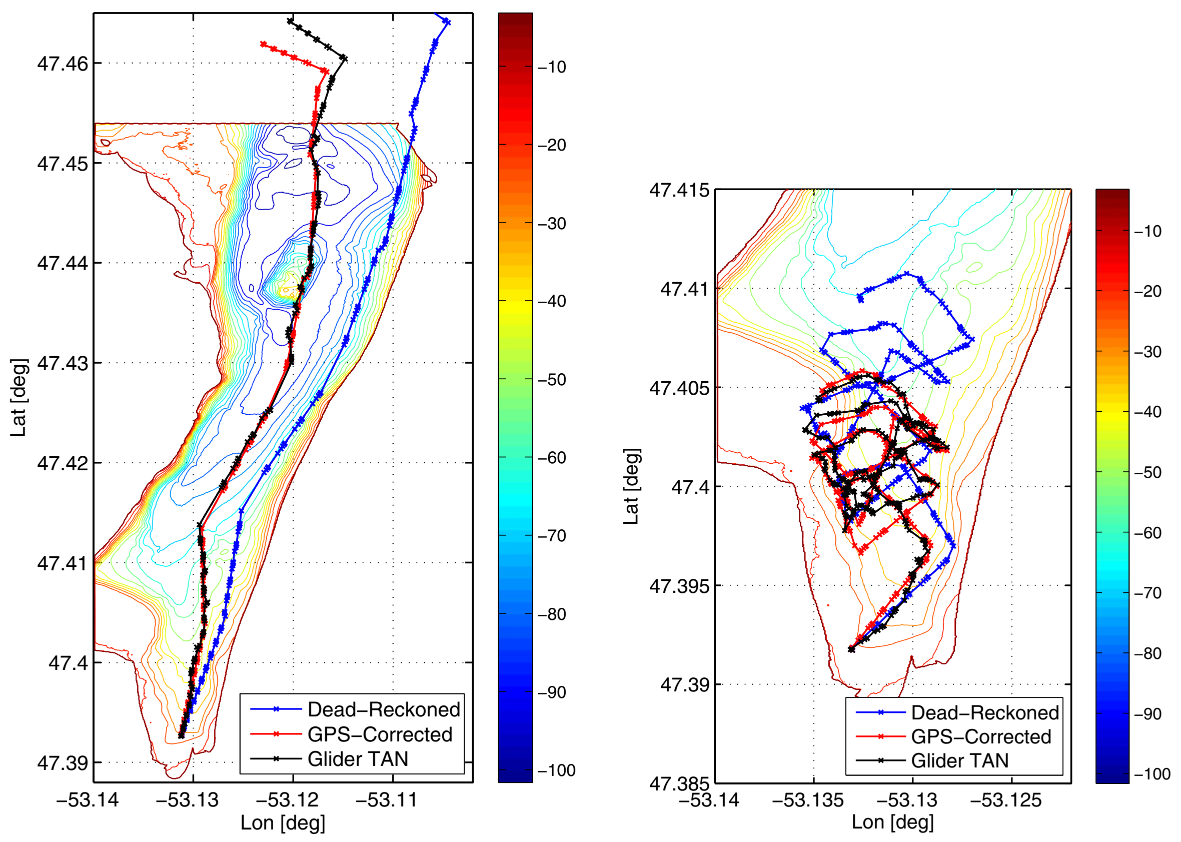

Figure 3. Location estimates from the glider TAN algorithm (black) against the GPS-corrected dead-reckoned locations (red) and the dead-reckoned locations (blue) from the 2010 offline field trials (left) and from the first 10 km 2012 offline field trials (right).

In both experiments, the glider recorded its navigation data to allow for the glider TAN algorithm to be evaluated through post-processing. As no independent localization method, such as an ultrashort baseline system, was available, the glider was programmed to surface approximately every hour and correct for the drift in its position estimate. The glider's recorded dead-reckoned locations were then able to be postprocessed using these GPS updates, as illustrated in Figure 4.

The GPS-corrected glider locations are used as the baseline locations for comparison of the performance of the glider TAN algorithm. This method of baseline comparison is most accurate at the locations of the GPS updates during surfacing events, with the uncertainty increasing to a maximum halfway between updates. It should also be noted that because the glider does not record data, in particular attitude and altitude, during surfacing events the surface drift, that is, the surfacing GPS location minus the diving GPS location, is removed from the offline glider TAN postprocessing. Otherwise, the glider TAN algorithm uses the GPS information only for initialization of the algorithm prior to the first dive.

Moreover, since the altimeter is oriented at a 26° angle from the vertical, altitude measurements are only acquired on the downward glide due to the shallow grazing angle on the upward glide, as is illustrated in Figure 5.

The period between altitude measurements is not constant, generally being around 30 s, decreasing when it approaches the seafloor to a minimum of 4 s. This behavior is due to the vehicle's altimeter filter, which attempts to reject bad values and limit the power use of the device. The histogram of the time between altimeter measurements is shown in Figure 6, with longer periods associated with the gap in measurements due to the climbing segments.

The large gaps in measurements when the glider is climbing are dependent on the depth of the profiles the glider is performing. For the field trials in Holyrood Arm, the maximum profile depth was around 100 m, limiting the maximum time between measurement updates to around 20 min. The nonconstant frequency of the altitude measurements during the downward glide followed by the large amount of time during the upward glide with no altitude measurements creates a unique challenge for a TAN algorithm. The structure of the bootstrap algorithm with jittering

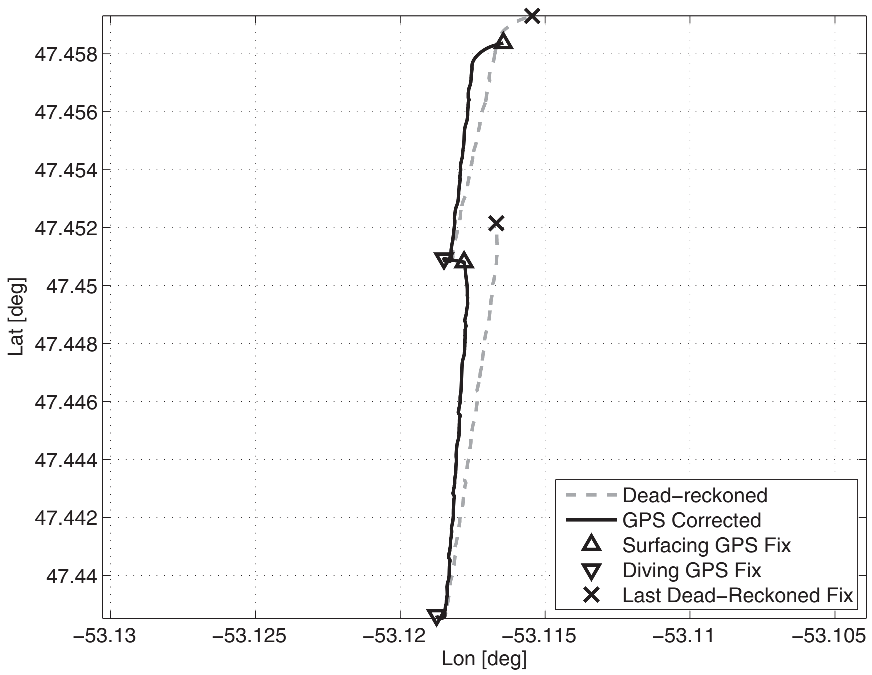

Figure 4. GPS-corrected dead-reckoned location estimates computed from the dead-reckoned estimates using the difference between the last dead-reckoned estimate and the GPS fix upon surfacing applied as a constant disturbance from the diving GPS fix to the last dead-reckoned estimate.

is well suited to this problem as it makes no assumptions about the frequency of the measurements. Additionally, because jittering and resampling are performed at every time step, the particle distribution rapidly adjusts to an accurate representation of the prior density function. This behavior is particularly helpful in maintaining convergence during large measurement update gaps due to a climbing section and in reconvergence after the vehicle leaves the bounds of the map.

6.1. Software Implementation

The software for the glider TAN postprocessing was written using MATLAB®. Postprocessing the field trials from 2010 and 2012 required computing the baseline locations using the GPS-corrected dead-reckoning and computing the dead-reckoning solutions with no GPS influence. The glider uses a correction algorithm to compensate for water velocities based on the difference between the dead-reckoned and first GPS location upon surfacing. This correction is embedded in the dead-reckoning solutions the vehicle computes, requiring the MATLAB® code to strip out this effect to obtain the state updates independent of any GPS influence. The code then initializes the particle filter and runs through Algorithm 4 for every altimeter measurement. Location errors are computed as the difference between the GPS-corrected dead-reckoning locations and the location estimates produced by the glider TAN algorithm.

6.2. Parameter Tuning

The glider TAN algorithm has several variables that require tuning to allow the algorithm to retain its convergence. These parameters include the number of particles N and the jittering variance σ_j^2. The grid cell size of the DEM was also varied to investigate its impact on the convergence of the algorithm. The jittering variance was tested first using a relatively large number of particles, $N = 1000$, and the highest resolution DEM, which is gridded at 2 m. These tests were run five times for a range of levels between 2 and 30 m^2, with the results shown in Figure 7.

The results of these tests show that there is a minimum jittering variance required for the algorithm to achieve convergence in both cases of around 10 m^2. The best values for the jittering variance in the 2010 trials appear to be in the range of 16–26 m^2, while for the 2012 trials the best jittering variances are in the range of 10–14 m^2 with the RMS errors increasing steadily past this point. The differences in the RMS errors between the two trials for the jittering variance tests are attributed to the trials operating in different regions of the DEM and to variations in the accuracy of the dead-reckoning algorithm.

The effect of the grid size of the DEM was investigated next by regridding the DEM to a range of values from 2 m

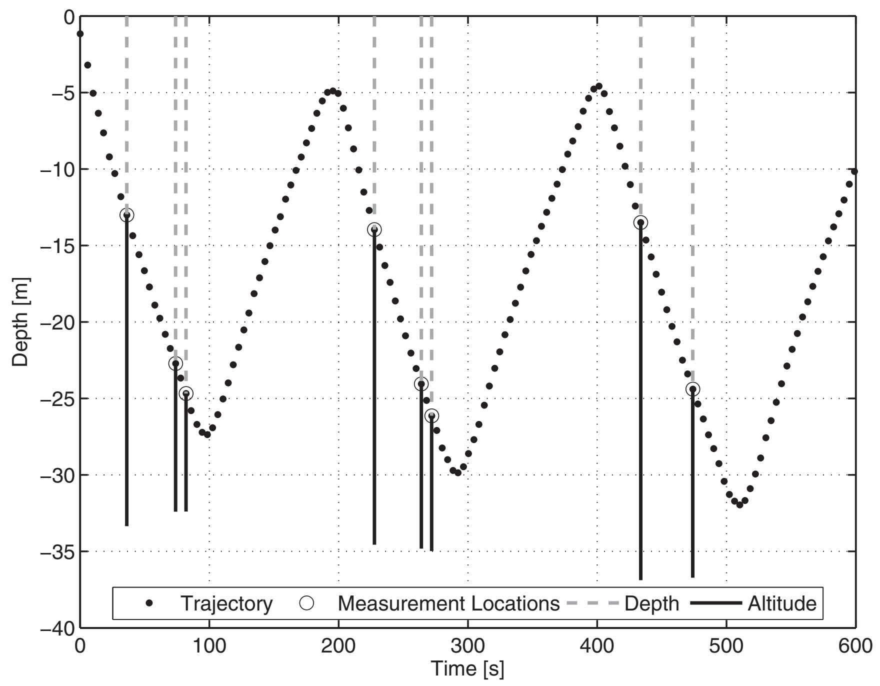

Figure 5. A sequence of measurements illustrating the construction of the water depth estimate using the vehicle depth and altitude measurements.

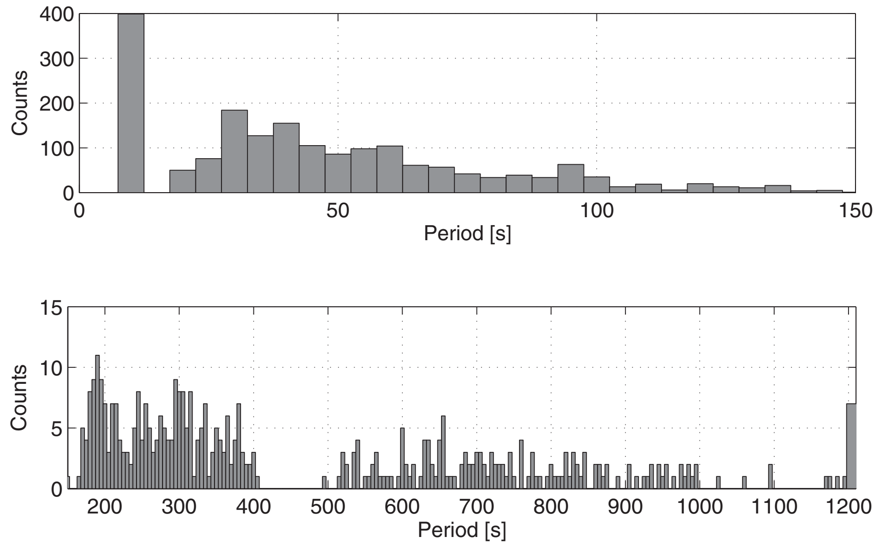

Figure 6. Histogram of the period between altimeter measurements in 5 s bins for periods between close together pings (top) and far apart pings (bottom) where the large gaps in measurements are due to the climbing portion of the profile.

up to 180 m. The DEM was regridded by taking the mean of all of the points in the multibeam survey data that fell within each grid cell. The tests were run for 1,000 particles and a jittering variance of 10, 15, and 20 m^2, as shown in Figure 8.

For both trials, the algorithm maintained convergence for grid cell sizes ranging from 2 to 100 m. The different values for the jittering variance are insignificant in the case of the 2010 trials but show a slight preference for 15 m^2 in the 2012 trials at the smaller grid cell sizes. In both cases,

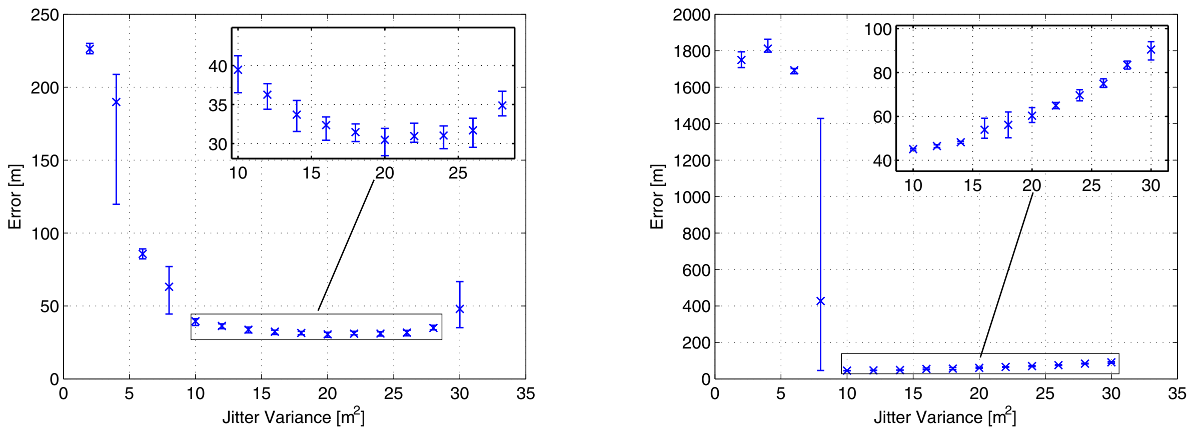

Figure 7. RMS errors of the glider TAN algorithm for different values of the jittering variance for the 2010 offline field trials (left) and the 2012 offline field trials (right), where the number of particles was 1,000 and the DEM was gridded at 2 m. The cross marks the mean RMS error and the upper and lower bars represent the maximum and minimum RMS error over a total of five Monte Carlo runs at each level. Insets show a closeup of the area bounded by the box in each case.

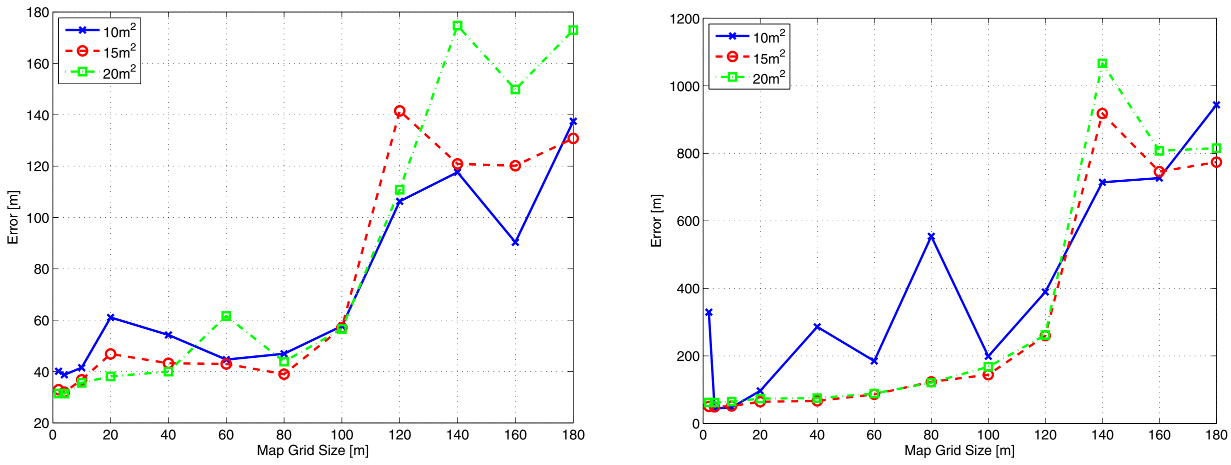

Figure 8. Average RMS errors of the glider TAN algorithm over five runs for different values of the grid cell size for the 2010 offline field trials (left) and the 2012 offline field trials (right), where the number of particles was 1,000 and the jittering variance was 10, 15, and 20 m^2.

the tests with a jittering variance of 10 m^2 showed a higher probability of divergence and no significant improvement in the RMS error.

The number of particles for convergence was also tested for different numbers particles between 100 and 2,000, as shown in Figure 9. In these tests, the jittering variance was 15 m^2 and the grid cell size was 2 m.

In general, for particle filter algorithms, more particles provide a better estimate of the underlying probability density function and thereby more confidence at the expense of processing time. Finding the number of particles to use for a particular application becomes an exercise in determining the minimum number of particles required to reliably retain convergence of the algorithm. In this case, 0.105 s was required for each time step at 1,000 particles for the offline postprocessing. For the five Monte Carlo runs examining the number of particles needed for convergence during the 2010 and 2012 field trials, the average RMS error levels out at around 500 particles in both cases. To ensure convergence, 1,000 particles were selected for nominal use.

6.3. Results

The offline glider TAN location estimates are computed through 100 Monte Carlo simulations with the RMS and peak errors shown in Figure 10.

Convergence was maintained in all of the 100 Monte Carlo simulations of the glider TAN algorithm for a jittering variance of 15 m^2, 1,000 particles, and a grid cell size of 2 m

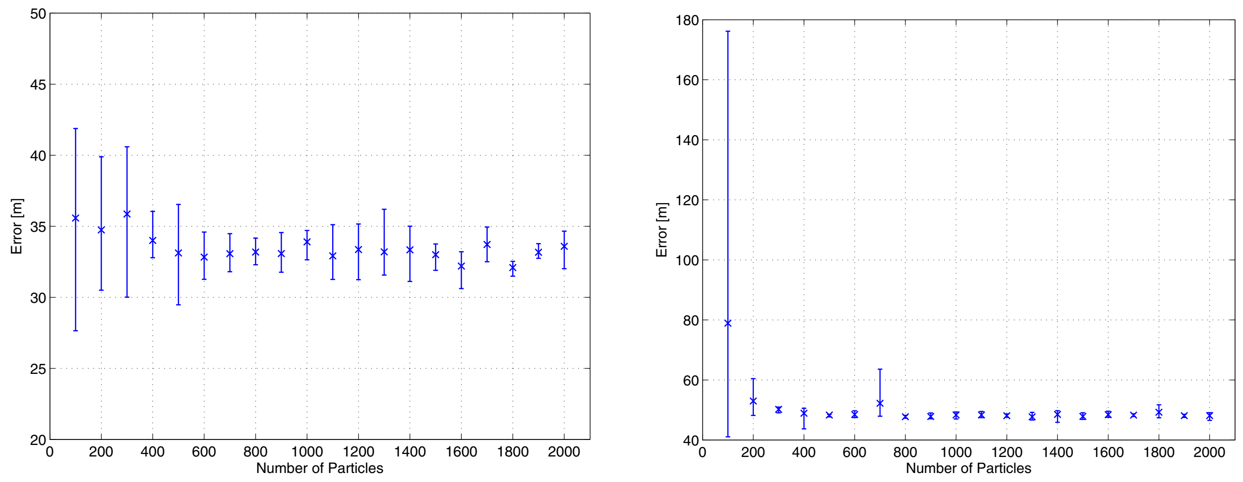

Figure 9. RMS errors of the glider TAN algorithm for different values of the number of particles for the 2010 offline field trials (left) and the 2012 offline field trials (right) where the jittering variance was 15 m^2 and the grid cell size was 2 m. The cross marks the mean RMS error and the upper and lower bars represent the maximum and minimum RMS error over a total of five Monte Carlo runs at each level.

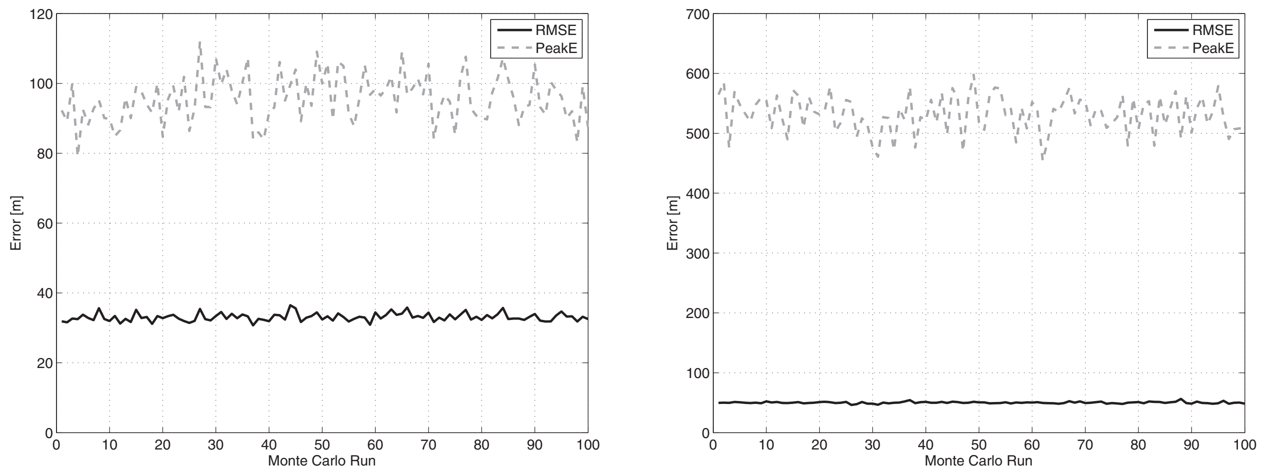

Figure 10. RMS error and peak error from 100 Monte Carlo simulations of the glider TAN algorithm for the 2010 offline trials (left) and the 2012 offline trials (right) with a jittering variance of 15 m^2, 1,000 particles, and a grid cell size of 2 m.

for both trials. The average RMS error in the 2010 trials was 33 m, with an average peak error of 96 m. For the 2012 trials, the average RMS error was 50 m, with an average peak error of 532 m. A comparison of the mean error with the dead-reckoned location estimates along with the Monte Carlo lower bound (MCLB) is shown in Figure 11, where the MCLB is the minimum value over the set of Monte Carlo runs.

The improvement of the glider TAN location estimates over the dead-reckoned location estimates is shown by the bounded error location estimates provided. In the 2010 trials, the dead-reckoned error reaches around 900 m during the time the algorithm is within the bounds of the DEM, while the glider TAN error at the same time step is only around 44 m. In the 2012 trials, the dead-reckoned error reaches over 5.5 km, while at the end of the mission the glider TAN error is only 16 m. Additionally, while the glider TAN algorithm shows occasional periods of divergence, the algorithm is able to reconverge shortly after. These periods of divergence along with a closer look at the mean error, MCUB, MCLB, and DEM flags are illustrated in Figure 12.

In the 2010 trials, the main divergence is after the map bounds flag goes high due to the vehicle, leaving the bounds of the DEM. At this point, the dead reckoned location estimates take over. For the 2012 trials, the glider TAN algorithm maintains convergence during most of the

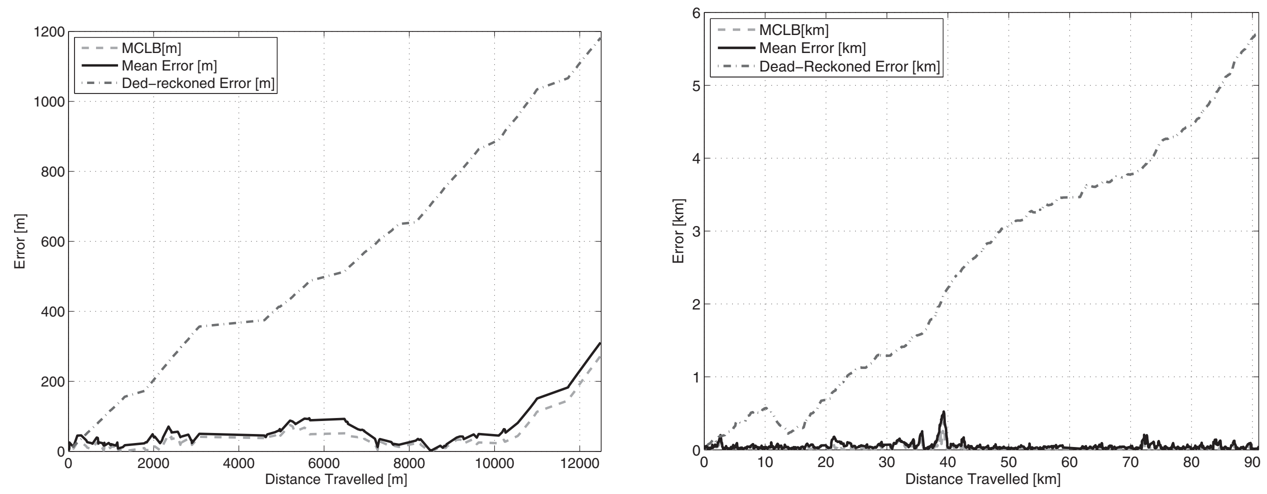

Figure 11. Improvement of the glider TAN algorithm over the dead-reckoned error computed from 100 Monte Carlo simulations of the glider TAN algorithm from the 2010 offline trials (left) and the 2012 offline trials (right).

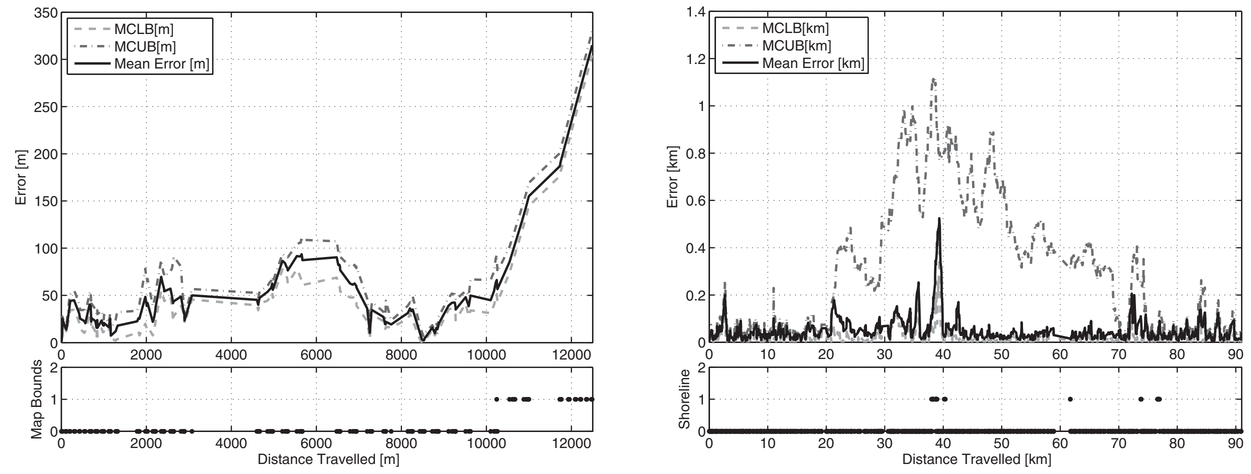

Figure 12. Mean RMS error, Monte Carlo upper bound (MCUB), and Monte Carlo lower bound (MCLB) from 100 Monte Carlo simulations of the glider TAN algorithm from the 2010 offline trials (top left) with the map bounds flag shown (bottom left). The mean RMS error, MCUB, and MCLB from the 2012 offline trials (top right) with the shoreline flag are shown (bottom right). The flag markers are shown at every altimeter reading, with a 1 indicating a particle is out of bounds.

deployment. Occasional spikes in the error are noticeable, with the largest occurring around 38 km into the mission and reaching 532 m. This large error corresponds with the shoreline flag and is due the glider being in shallow water during this portion of the trials. In shallow water, the glider's 200 m pump is not fast enough to keep the vehicle's speed up, resulting in the dead reckoning algorithm deteriorating on the vehicle and many altimeter measurements being performed in the same region. In effect, the vehicle thinks it is moving faster than it is and the water depth estimates are not changing very much during this time. However, once the vehicle leaves the proximity of the shoreline, the algorithm quickly reconverges. Large errors due to shallow water can be prevented during online trials through planning the mission with a suitable buffer around these areas.

The MCUB and MCLB represent the maximum and minimum value observed at a time step over the entire set of Monte Carlo runs. The MCUB and MCLB give an indication of the absolute worst and best possible scenario.

A visual representation of the performance of the glider TAN algorithm for the 2010 trials and for the first 10 km of the 2012 trials is presented in Figure 3. These figures highlight the improvement provided by the glider TAN algorithm over the pure dead-reckoning location estimates, and they reinforce online localization efforts.

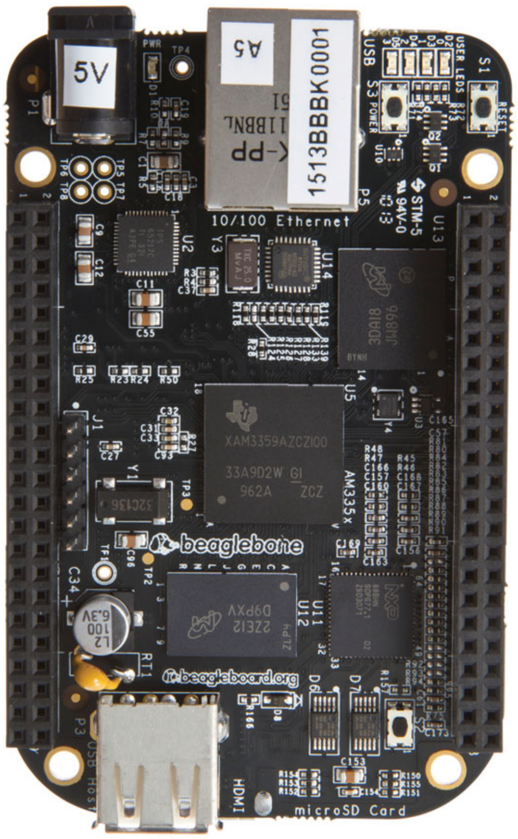

Figure 13. The Beagle Bone Black 1 GHz ARM Cortex-A8 processor with 512 Mb of RAM and 4 Gb of onboard flash.

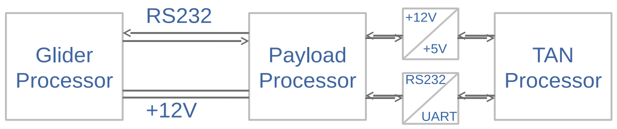

Figure 14. Diagram of the glider terrain-aided navigation hardware integration showing the power and communication connections to the science computer.

7. INTEGRATION

As the microprocessors onboard underwater gliders have minimal computational ability, an additional single board computer was deemed necessary for the integration of the algorithm into the glider. For this work, the Beagle Bone Black (BBB) (Figure 13) single board computer was selected for use, which has a 1 GHz ARM Cortex-A8 processor with 512 Mb of RAM, 4 Gb of onboard flash, and a standard set of embedded peripheral options.

The BBB was loaded with Ubuntu 13.04, allowing the particle filter to be programmed in C/C++. The BBB is powered through a separate switching regulator from the standard power pins in the payload module, which supplies 10–15 V, consuming on average 0.5 W. The communication interface connects from the 5 V UART on the BBB through a logic level converter to the standard RS232 port on the science computer of the glider. In this way, the BBB connects to the glider as any payload or science sensor would, as illustrated in Figure 14.

The particle filter program that runs on the BBB is configured to run as a background process once the operating system has booted. The UART and processor options are also configured at boot. The particle filter code accepts an initialization command and an update command from the vehicle. The initialization command sets the reference location for the local mission coordinate frame and resets the particle locations to this initial location. The update command computes one iteration of the particle filter and sends back to the vehicle a location update as well as a status flag. The status flag indicates if the location update is nominal, near shore, or near the map bounds. The computation time for the update command is approximately 10 ms, improving on the MATLAB implementation by over 10 times, and this is more than adequate for the fastest possible update rate of 4 s.

The particle filter program on the BBB is controlled by the science processor. The science processor runs a glider TAN "proglet," which requests the attitude, altitude, depth, dead-reckoned latitude and longitude, GPS latitude and longitude, and the local mission coordinate locations from the glider computer. The transmission of these variables from the glider processor to the science processor is triggered upon their being updated on the glider processor. Whenever the GPS latitude and longitude variables are updated on the science processor, it sends the initialization command to the BBB. In this way, the best navigation data are always used. Subsequent to the first initialization, any updates to the altitude variable trigger the transmission of the update command to the BBB. The BBB then computes a location based on the TAN particle filter and sends a location update and status flag back to the science computer, where the open-loop locations are logged for future analysis. While requesting variables from the glider processor is straightforward, sending variables back requires modification of the glider processor's source code, which was not possible at this time. For this reason, closed-loop trials were not performed for this work.

8. ONLINE FIELD TRIALS

Online tests of the glider TAN method were performed in Holyrood Arm of Conception Bay, Newfoundland during September, 2014. During these tests, the glider was flown in a northward and southward straight line segment for roughly 1 h in each direction, and the glider TAN processor was allowed to compute open-loop location estimates. An initial set of three profiles was performed to extract

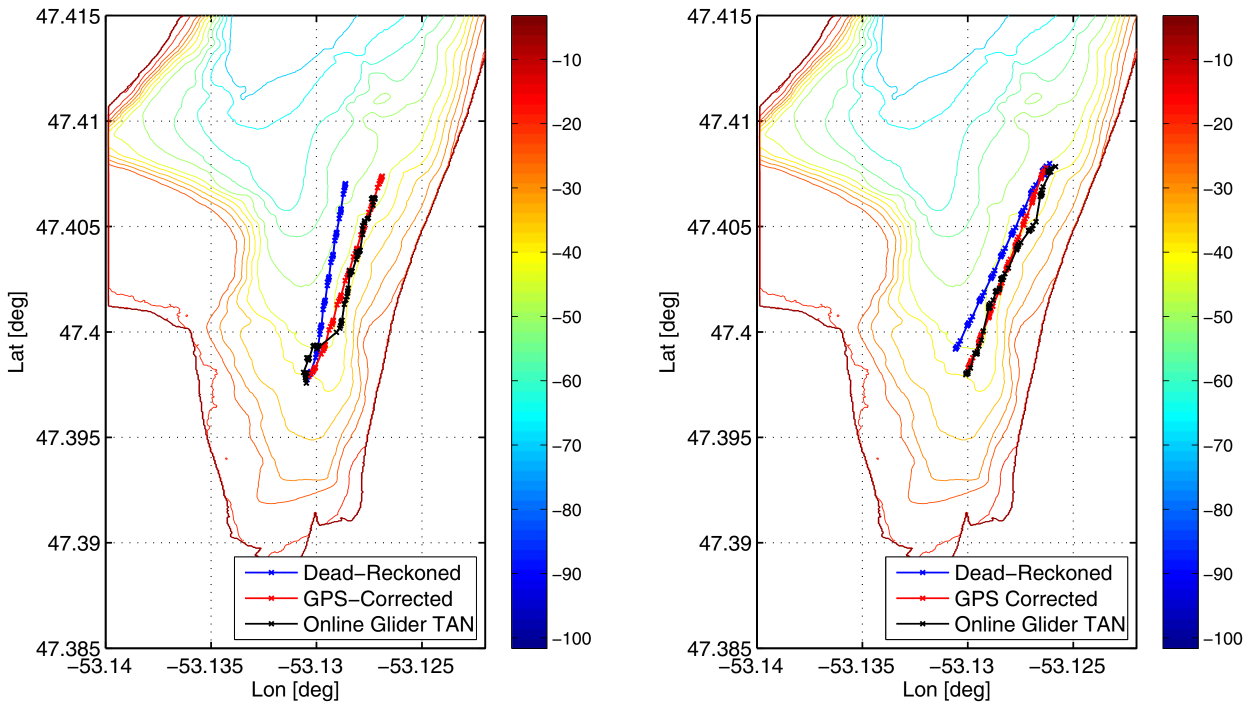

Figure 15. Online open-loop location estimates from the glider TAN algorithm (black) against the GPS locations (red) and the dead-reckoned locations (blue) from the 2014 online trials northward leg (left) and southward leg (right).

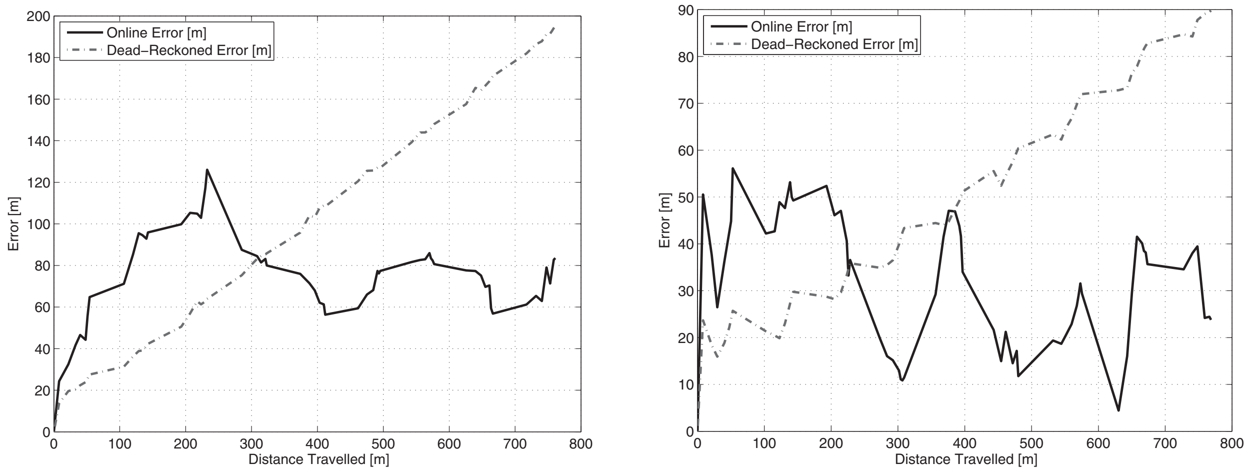

Figure 16. Improvement of the glider TAN algorithm over the dead-reckoned error during the 2014 online trials northward leg (left) and southward leg (right).

the depth bias and confirm the generation of glider TAN location estimates. These estimates were recorded on the Science computer for analysis later. The vehicle was configured to not correct for water velocities, which requires multiple surfacing events, as this is most representative of a longer distance closed-loop mission. The results of the open-loop glider TAN locations are shown relative to the dead-reckoning location estimates and GPS locations in Figure 15.

During the northward trial, the GPS-corrected estimates are slightly east of the dead-reckoned estimates likely due to some local tidal current. The online glider TAN estimates converge to the correct water depth after a short period with some along track error.

During the southward trial, the GPS-corrected estimates are again slightly east of the dead-reckoned estimates. As these trials occurred immediately following the northward leg, this further supports an easterly tidal current, albeit one that is decreasing in magnitude. For these trials, the online glider TAN estimates again converge to the vehicle track after a short period, this time with less along track error.

The error between the GPS-corrected location estimates and the online glider TAN estimates is shown against the dead-reckoned error in Figure 16.

The online glider TAN errors in both the northward and the southward leg improve upon the dead-reckoned error. Specifically, the RMS error in the northward leg was 76 m and the southward leg was 32 m. The dead-reckoned error growth rates were 25.6% and 11.6% of distance traveled for the northward and southward legs, respectively. The larger RMS error of the online glider TAN estimates in the northward leg is therefore attributed to this larger dead-reckoning error growth rate. These large error growth rates are due to the vehicles' slow speed relative to the water currents during the tests and the dead-reckoning computation relying on the speed through water instead of the speed over ground.

9. CONCLUSIONS

An underwater glider terrain-aided navigation (TAN) algorithm has been presented for a 200 m, Slocum electric underwater glider based on a particle filter known as the jittered bootstrap filter. The glider TAN algorithm allows for location estimates to be computed through comparisons of the water depth estimate measured by the vehicle with a digital elevation model (DEM) of the bathymetry. The method makes use of the underwater glider's altimeter, pressure sensor, dead-reckoning solution, and attitude. The algorithm has been shown to be suitable for postprocessing of underwater glider data collected in water depths, which allow for altimeter measurements but in which GPS updates are denied. Additionally, the method was validated through online trials in which a more powerful single board computer was integrated to run the filter.

The method was evaluated through two sets of offline field trials that took place in October 2010 and 2012 in Holyrood Arm of Conception Bay, Newfoundland, which overlaps a DEM collected previously through a ship-based survey by Memorial University's Marine Institute. During these trials, the underwater glider was allowed to obtain GPS updates during its periodic surfacings to provide a baseline to compare against the glider TAN location estimates. The data collected during these trials were then used to compute location estimates using the glider TAN algorithm, which uses the first predive GPS location for initialization. The algorithm was evaluated for a range of values of the jittering variance, the DEM grid cell size, and the number of particles. It was found that the algorithm finds good convergence for a jittering variance of 15 m^2, 1,000 particles, and for DEM grid cell sizes ranging from the base grid cell size of 2 m up to a cell size of 100 m.

Using nominal values for the jittering variance, the number of particles, and the base grid cell size, 100 Monte Carlo simulations were run to confirm the convergence at those values. In both the 2010 and 2012 trials, the algorithm maintained its convergence for all 100 runs. During the runs for the 2010 trials, the peak error was 96 m and the RMS error was 33 m. The total distance traversed over the DEM was approximately 9 km and the error before the vehicle left the bounds of the DEM was 44 m compared the dead-reckoned error of 900 m. During the runs for the 2012 trials, the peak error was 532 m and the RMS error was 50 m. The total distance traveled by the vehicle was approximately 91 km and the error at the end of the mission was 16 m, compared with the dead-reckoned error of 5.5 km. The peak error of 532 m during the 2012 trials was found to correspond to a period of shallow water profiles near shore, and it was attributed to an increase in the dead-reckoning error rate and many water depth estimates in the same location. These trials show the glider TAN algorithm's utility in postprocessing bounded location estimates of glider data in GPS-denied areas.

Online, open-loop trials were performed in September of 2014 in Holyrood Arm, in which independent, hour-long northward and southward tests were run. For these trials, a Beagle Bone Black (BBB) single board computer was integrated into the payload bay of the underwater glider. The BBB runs the filter, programmed in C/C++, in around 10 ms while consuming just over 0.5 W on average. During the northward and southward legs of the online trials, the RMS errors were 76 and 32 m, respectively. The larger error in the northward leg was attributed to the large dead-reckoned error growth rate 25.6% of distance traveled. In both cases, the glider TAN estimates provided bounded error location estimates that improved on the dead-reckoned estimates in spite of the short duration of the tests and larger than normal dead-reckoning error growth rate.

ACKNOWLEDGMENTS

This work was supported by the Natural Sciences and Engineering Research Council (NSERC) through the NSERC Canadian Field Robotics Network (NCFRN), the Research Development Corporation, the Marine Institute, and Memorial University of Newfoundland.

REFERENCES

Bar-Shalom, Y., Li, X. R., & Kirubarajan, T. (2004). Estimation with applications to tracking and navigation: Theory algorithms and software. John Wiley & Sons.

Barkby, S., Williams, S. B., Pizarro, O., & Jakuba, M. V. (2011). A featureless approach to efficient bathymetric slam using distributed particle mapping. Journal of Field Robotics, 28(1), 19–39.

Bergem, O. (1993). Bathymetric navigation of autonomous underwater vehicles using a multibeam sonar and a Kalman filter with relative measurement covariance matrices. Ph.D. thesis, University of Trondheim.

Chen, L., Wang, S., McDonald-Maier, K., & Hu, H. (2013). Towards autonomous localization and mapping of AUVs: A survey. International Journal of Intelligent Unmanned Systems, 1(2), 97–120.

Dektor, S., & Rock, S. M. (2012). Improving robustness of terrain-relative navigation for AUVs in regions with flat terrain. In IEEE AUV 2012, Southampton, England.

DFO (2013). Archived tidal data for station 905. Accessed on Dec. 12th, 2013.

Doucet, A., Godsill, S., & Andrieu, C. (2000). On sequential Monte Carlo sampling methods for Bayesian filtering. Statistics and Computing, 10(3), 197–208.

Fox, D., Hightower, J., Liao, L., Schulz, D., & Borriello, G. (2003). Bayesian filtering for location estimation. Pervasive Computing, IEEE, 2(3), 24–33.

Gordon, N., Salmond, D., & Smith, A. F. M. (1993). Novel approach to nonlinear/non-Gaussian Bayesian state estimation. Radar and Signal Processing, IEE Proceedings F, 140(2), 107–113.

Hodges, R. (2010). Underwater acoustics: Analysis, design and performance of sonar (chap. 5, pp. 84–91). John Wiley and Sons, Ltd.

Houts, S. E., Dektor, S. G., & Rock, S. M. (2013). A robust framework for failure detection and recovery for terrain-relative navigation. In Unmanned Untethered Submersible Technology 2013, Portsmouth, NH.

Kato, N., & Shigetomi, T. (2009). Underwater navigation for long-range autonomous underwater vehicles using geomagnetic and bathymetric information. Advanced Robotics, 23(7-8), 787–803.

Kinsey, J. C., Eustice, R. M., & Whitcomb, L. L. (2006). A survey of underwater vehicle navigation: Recent advances and new challenges. In IFAC Conference of Manoeuvering and Control of Marine Craft, Lisbon, Portugal. Invited paper.

Kitagawa, G. (1996). Monte Carlo filter and smoother for non-Gaussian nonlinear state space models. Journal of Computational and Graphical Statistics, 5(1), 1–25.

Lagadec, J., Cecchi, D., Rixen, M., & Grasso, R. (2010). Terrain navigation for underwater autonomous gliders. In MREA10, Lerici, Italy, October 18–22.

Lee, C. M., & Gobat, J. I. (2006). Acoustic navigation and communication for high-latitude ocean research workshop. Eos, Transactions American Geophysical Union, 87(27), 268–268.

Matsumoto, M., & Nishimura, T. (1998). Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. ACM Transactions in Modelling and Computer Simulations, 8(1), 3–30.

Meduna, D. (2011). Terrain relative navigation for sensor-limited systems with application to underwater vehicles. Ph.D. thesis, Stanford University.

Meduna, D., Rock, S., & McEwen, R. (2010). Closed-loop terrain relative navigation for AUVs with non-inertial grade navigation sensors. In Autonomous Underwater Vehicles (AUV), 2010 IEEE/OES (pp. 1–8).

Merckelbach, L., Smeed, D., & Griffiths, G. (2010). Vertical water velocities from underwater gliders. Journal of Atmospheric Oceanic Technology, 27(3), 547–563.

Morice, C., Veres, S., & McPhail, S. (2009). Terrain referencing for autonomous navigation of underwater vehicles. In Oceans 2009–Europe (pp. 1–7).

Nygren, I. (2005). Terrain navigation for underwater vehicles. Ph.D. thesis, KTH Electrical Engineering, Stockholm, Sweden.

Nygren, I. (2008). Robust and efficient terrain navigation of underwater vehicles. In Position, Location and Navigation Symposium, 2008 IEEE/ION (pp. 923–932).

Paull, L., Saeedi, S., Seto, M., & Li, H. (2014). AUV navigation and localization: A review. IEEE Journal of Oceanic Engineering, 39(1), 131–149.

Rintoul, S. R., Sparrow, M., Meredith, M. P., Wadley, V., Speer, K., Hofmann, E., Summerhayes, C., Urban, E., Bellerby, R., Ackley, S., et al. (2012). The Southern Ocean observing system: Initial science and implementation strategy. SCAR and SCOR.

Ristic, B., Arulampalam, S., & Gordon, N. (2004). Beyond the Kalman filter: Particle filters for tracking applications. Artech House.

Rudnick, D., Davis, R., Eriksen, C., Fratantoni, D., & Perry, M. (2004). Underwater gliders for ocean research. Marine Technology Society Journal, 38, 73–84.

Sanjeev Arulampalam, M., Maskell, S., Gordon, N., & Clapp, T. (2002). A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. IEEE Transactions on Signal Processing, 50(2), 174–188.

Schofield, O., Kohut, J., Aragon, D., Creed, L., Graver, J., Haldeman, C., Kerfoot, J., Roarty, H., Jones, C., Webb, D., and Glenn, S. (2007). Slocum gliders: Robust and ready. Journal of Field Robotics, 24(6), 473–485.

Simon, D. (2006). Optimal state estimation: Kalman, H infinity, and nonlinear approaches. John Wiley & Sons.

Stutters, L., Liu, H., Tiltman, C., & Brown, D. (2008). Navigation technologies for autonomous underwater vehicles. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, 38(4), 581–589.

Teixeira, F.C., & Pascoal, A. M. (2008). Geophysical navigation of autonomous underwater vehicles using geomagnetic information. In 2nd IFAC Workshop, Navigation, Guidance and Control of Underwater Vehicles.

Webster, S., Lee, C., & Gobat, J. (2014). Preliminary results in under-ice acoustic navigation for seagliders in Davis Strait. In Oceans 2014 (pp. 1–5).