Gamma-SLAM: Visual SLAM in Unstructured Environments Using Variance Grid Maps

Tim K. Marks
Computer Science and Engineering Department
University of California, San Diego
La Jolla, California 92093-0404
e-mail: tkmarks@cs.ucsd.edu

Andrew Howard and Max Bajracharya
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive
Pasadena, California 91109
e-mail: andrew.howard@jpl.nasa.gov,
max.bajracharya@jpl.nasa.gov

Garrison W. Cottrell
Computer Science and Engineering Department
University of California, San Diego
La Jolla, California 92093-0404
e-mail: gary@cs.ucsd.edu

Larry H. Matthies
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive
Pasadena, California 91109
e-mail: larry.matthies@jpl.nasa.gov

Received 4 April 2008; accepted 22 November 2008

This paper describes an online stereo visual simultaneous localization and mapping (SLAM) algorithm developed for the Learning Applied to Ground Robotics (LAGR) program. The Gamma-SLAM algorithm uses a Rao–Blackwellized particle filter to obtain a joint posterior over poses and maps: the pose distribution is estimated using a particle filter, and each particle has its own map that is obtained through exact filtering conditioned on the particle's pose. Visual odometry is used to provide good proposal distributions for the particle filter, and maps are represented using a Cartesian grid. Unlike previous grid-based SLAM algorithms, however, the Gamma-SLAM map maintains a posterior distribution over the *elevation variance* in each cell. This variance grid map can capture rocks,

Published online in Wiley InterScience (www.interscience.wiley.com). • DOI: 10.1002/rob.20273

vegetation, and other objects that are typically found in unstructured environments but are not well modeled by traditional occupancy or elevation grid maps. The algorithm runs in real time on conventional processors and has been evaluated for both qualitative and quantitative accuracy in three outdoor environments over trajectories totaling 1,600 m in length.

1. INTRODUCTION

The DARPA Learning Applied to Ground Robotics (LAGR) program challenged participants to develop vision-based navigation systems for autonomous ground vehicles. Although the principal thrust of the LAGR program was learning, the ability to build and maintain accurate global maps was a key enabling technology. Toward this end, we have developed and tested a new algorithm for vision-based online simultaneous localization and mapping (SLAM), to be used in conjunction with a standard planner, that enables robust navigation in unstructured outdoor environments.

The Gamma-SLAM algorithm described in this paper makes use of a Rao–Blackwellized particle filter (RBPF) for maintaining a distribution over poses and maps (Doucet, de Freitas, Murphy, & Russell, 2000; Haehnel, Burgard, Fox, & Thrun, 2003; Montemerlo & Thrun, 2003). The core idea behind the RBPF approach is that the SLAM problem can be factored into two parts: finding the distribution over robot trajectories and finding the map conditioned on any given trajectory. This is done using a particle filter in which each particle encodes both a possible trajectory and a map conditioned on that trajectory.

Unlike most previous SLAM algorithms, which are designed to work on LADAR or stereo range data in structured environments, the Gamma-SLAM algorithm is designed to work in unstructured and heavily vegetated outdoor environments. This leads to two important design decisions. First, we use visual odometry (VO) (Howard, 2008) rather than wheel odometry to provide vehicle motion estimates. As we will show in Section 4, wheel odometry is unreliable in unstructured environments due to the complex nature of the wheel/terrain interaction (i.e., wheels tend to sink, slip, and skid). Second, the Gamma-SLAM grid map encodes the elevation variance within each cell, rather than cell occupancy or mean elevation. That is, unlike occupancy grids (with the underlying assumption that each cell in the world is either occupied or free) and elevation maps (with the underlying assumption that each cell in the world is a flat surface), our maps are based on the assumption that the points observed from each cell are drawn from a Gaussian distribution of elevations, and the statistic of interest is the *variance* of elevations in the cell. The memory required by a variance grid map is only slightly larger than that required by an occupancy grid or elevation map: each map cell contains two scalar values that are the sufficient statistics of the posterior distribution (see Section 5.3). The name Gamma-SLAM derives from the fact that the posterior distribution over the inverse variance is a gamma distribution (Raiffa & Schlaifer, 2000).

The use of variance grid maps derives from our experience operating robots in LAGR-relevant environments, where the key landmarks for localization are trees, bushes, and grasses. The variance grid captures more information about these types of landmarks than pure elevation or binary occupancy grids and is more robust to changes in viewpoint and lighting conditions than appearance-based models. Variance grid maps also have a secondary advantage that is of particular relevance to the LAGR program: in vegetated environments, elevation variance is a key feature for classifing terrain traversability. Intuitively, areas with low variance (such as mown grass) are much more traversable than areas with high variance (such as bushes and trees). Positive step edges (such as rocks or fences) also generate large variances, because points in these cells are effectively drawn from multiple distributions. On the other hand, negative step edges (such as cliffs) are generally unobservable. Mean elevation, in contrast, tells us relatively little about traversability; rather, it is the change in elevation between cells that is significant, and whereas such changes may effectively capture step edges, they are less likely to capture the difference between short grass (which is traversable) and tall grass (which is not).

The paper is structured as follows. We first review some related SLAM approaches (Section 2), briefly describe the LAGR robot hardware and software (Section 3), and present key details of the VO algorithm (Section 4). The Gamma-SLAM formalism

and algorithm are described in Sections 5 and 6, along with experimental results from three different outdoor environments (Section 7). We demonstrate an efficient online method for georeferencing the SLAM solution using global positioning system (GPS) data (Section 8) and conclude with a discussion of planner integration and future work (Section 9).

2. RELATED WORK

There are now a wide variety of approaches to solving the problem of SLAM; for a recent review, see Bailey and Durrant-Whyte (2006) and Durrant-Whyte and Bailey (2006). Many of the current approaches to SLAM use laser range scans for their observations. Many of these laser-based systems (such as that of Montemerlo & Thrun, 2003) are *landmark based*, in that each particle's map consists of a posterior distribution over the locations of a number of salient landmarks. Others (such as those of Grisetti, Tipaldi, Stachniss, Burgard, & Nardi, 2007, and Haehnel et al., 2003) are *grid based*, meaning that each particle's map is a dense occupancy grid containing the posterior probability that each cell in the grid is occupied. In addition to SLAM algorithms that use laser range measurements, there are a growing number of examples of vision-based SLAM, some of which use stereo vision (Dailey & Parnichkun, 2006; Elinas, Sim, & Little, 2006; Se, Barfoot, & Jasiobedzki, 2005; Sim, Elinas, & Little, 2007; Sim, & Little, 2006). All of the existing stereo vision SLAM algorithms are landmark based. One of these systems (Sim, Elinas, Griffin, Shyr, & Little, 2006; Sim et al., 2007; Sim & Little, 2006) uses a landmark-based representation for SLAM and then at each time step uses the maximum-likelihood SLAM trajectory to generate an occupancy grid for navigation. In that system, the SLAM algorithm itself is solely landmark based and makes no use of the occupancy grid map. To our knowledge, our Gamma-SLAM algorithm is the first stereo vision SLAM algorithm that is built upon a grid-based representation. Furthermore, most current approaches to SLAM address either indoor environments or structured outdoor environments. Gamma-SLAM is one of the first to succeed in off-road outdoor environments such as those featured in the LAGR program. We speculate that this may be because the appearance-based landmark representations used in previous stereo vision SLAM algorithms are less appropriate for natural terrain or are less robust to changes in three-dimensional (3D) viewpoint and lighting conditions than Gamma-SLAM's variance grid maps.

Most existing grid-based SLAM systems (such as that of Haehnel et al., 2003) use laser range finders and a binary occupancy grid. Occupancy grids, which are often used in structured indoor environments, are based on an underlying assumption that each cell in the world is either occupied (nontraversable) or free (unoccupied). Using observations of each cell, the SLAM system of Haehnel et al. (2003) infers the posterior probability that the cell is occupied versus unoccupied, which is a Bernoulli distribution. Another type of grid map is an elevation map, which corresponds to a representation of the horizontal surfaces of an environment (Kümmerle, Triebel, Pfaff, & Burgard, 2007). Elevation maps are based on an underlying assumption that each cell in the world is a flat surface, in which all samples from a cell have the same elevation, or height, up to additive Gaussian noise.[1] Each cell of an elevation map, which is constructed using the Kalman filter update rule, contains a Gaussian posterior distribution over the height of the surface in that cell. The mean of this Gaussian is the estimate of the height of the surface, and the variance of the Gaussian measures the uncertainty of this height estimate. In contrast, the variance grid maps that we introduce in this paper assume that the points observed in each cell are drawn from a Gaussian distribution over elevations. A variance grid map contains a posterior distribution over the variance of elevations (rather than the mean elevation) in each cell.

In recent extensions of elevation maps, which have been used in systems for outdoor localization and for mapping and loop closing (Kümmerle et al., 2007; Pfaff, Triebel, & Burgard, 2007; Triebel, Pfaff, & Burgard, 2006), a cell is allowed to contain one or more horizontal surfaces or a vertical surface. These extensions still assume that each cell is a planar patch (or multiple levels of planar patches, or a vertical surface), which makes them well suited to structured outdoor environments such as urban roads and overpasses. Map matching in these systems (Pfaff et al.,

[1] Although they are usually thought of as modeling flat surfaces, in which each cell has a single elevation up to Gaussian observation noise, elevation maps can also be thought of as modeling each cell as a Gaussian distribution over elevations. Either way, elevation maps estimate the posterior distribution over the *mean* elevation in each cell. In contrast, *variance grid maps*, which we introduce in this paper, estimate the posterior distribution over the *variance* of elevations in each cell.

2007; Triebel et al., 2006) uses an iterated closest point (ICP) algorithm, and they are currently much too slow for real-time SLAM.

3. THE MOBILE ROBOT PLATFORM

The Gamma-SLAM algorithm was designed for and tested on the LAGR Herminator robot (Jackel, Krotkov, Perschbacher, Pippine, & Sullivan, 2007). The Herminator is a differential drive vehicle with powered wheels at the front and caster wheels at the back. It can traverse moderately rough terrain, including sand, dirt, grass, and low shrubs and climb inclines of up to 20 deg (depending on the surface type). The top speed is around 1 m/s. The onboard sensors include wheel encoders, GPS, an inertial measurement unit (IMU), and two stereo camera pairs with a baseline of 12 cm and a combined field of view of approximately 140 deg. The camera pairs are individually calibrated using a planar calibration target, and the stereo pairs are individually surveyed into the vehicle frame.

The Herminator employs a distributed computing architecture, with one computer dedicated to each of the stereo camera pairs, and a third computer dedicated to mapping and planning. In the Jet Propulsion Laboratory (JPL) LAGR system (Howard et al., 2007), the *eye* computers perform stereo ranging and VO and project terrain geometry data into a series of vehicle-centric grid maps. These maps are passed to the *planner* computer, where they are fused into the global map used for path planning. The Gamma-SLAM system described in this paper runs on the planner computer, using VO for motion estimates and vehicle-centric grid maps for observations.

4. VISUAL ODOMETRY

VO estimates the incremental vehicle pose by tracking features across image frames and integrating the induced frame-to-frame camera motion. This estimate has the same basic properties as the estimate derived from wheel odometry; specifically, it is defined relative to an arbitrary origin and is prone to drift over time. Compared with wheel odometry, however, VO is much more accurate and is not affected by wheel/terrain interactions such as slipping, sliding, or skidding. This fact is demonstrated in dramatic fashion in Figure 1, which plots the pose estimates from VO and wheel odometry in a high-slip environment. In this experiment, the vehicle was driven around the Mars Yard at the JPL over flat sandy terrain, up a steep sand bank, and back down to the starting point. Note the divergence between the visual and wheel odometry estimates when the vehicle arrives at the sand bank and becomes bogged (with the wheels spinning but the vehicle making no forward progress). The final position error when the vehicle returns to the initial location is approximately 19 m for wheel odometry and less than 0.30 m for VO.

VO also serves a secondary purpose: to determine the extent to which the wheels are slipping. We define the *slip ratio* as $(s_w - s_v) / |s_w|$, where s_w and s_v are the vehicle speeds determined by wheel odometry and VO, respectively. Figure 1, for example, shows the slip ratio computed for the Mars Yard experiment; the ratio approaches 1 around frame 1,400 as the vehicle attempts to scale the sand bank and the wheels start spinning in place. In the JPL LAGR system, the slip ratio is used by the vehicle planner to detect and abort actions that are not making progress (e.g., driving forward when the vehicle is stuck on a low branch or rock). It can also be used by learning algorithms as a training signal, such that these algorithms can learn the relationship between wheel slip, surface type, and slope (Angelova, Matthies, Helmick, Sibley, & Perona, 2006).

The basic VO algorithm proceeds as follows. For each sequential pair of stereo camera frames:

1. Detect features in each frame (corner detection).
2. Match features between frames (sum of absolute differences over local windows).
3. Find the largest set of self-consistent matches (inliers).
4. Find the frame-to-frame motion that minimizes the inlier reprojection error.

This algorithm is illustrated in Figure 2.

The inlier detection step (3) is the key distinguishing feature of the algorithm. The feature-matching stage inevitably produces some incorrect correspondences, which, if left intact, will unfavorably bias the frame-to-frame motion estimate. A common solution to this problem is to use a robust estimator, such as RANSAC (Fischler & Bolles, 1981), that can tolerate some number of false matches. In our algorithm, however, we adopt an approach described in Hirschmuller, Innocent, and Garibaldi (2002) and exploit stereo range data at the inlier detection stage. The core intuition is that the 3D locations of

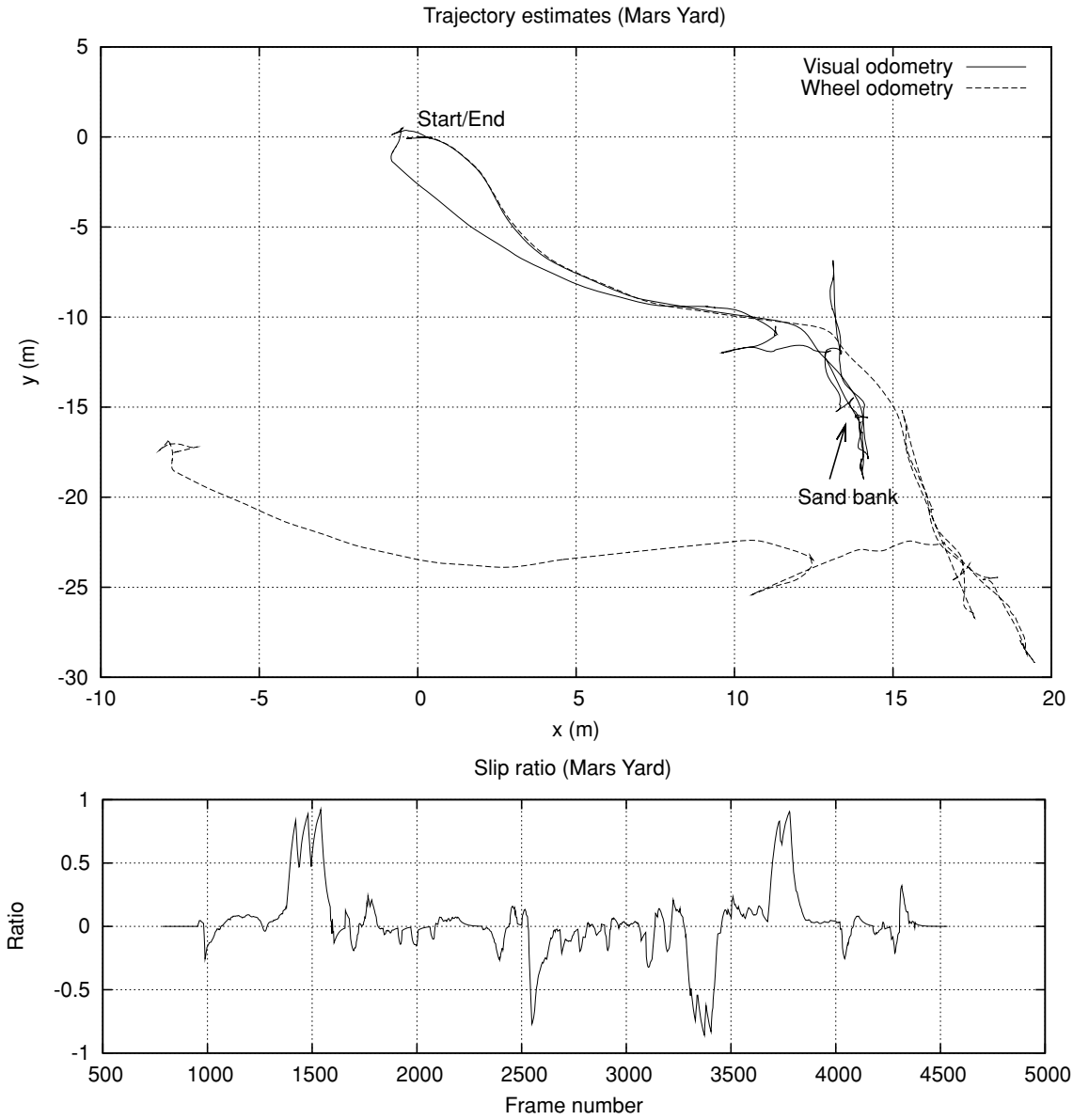

Figure 1. Top: Comparison of wheel odometry and VO in a high-slip environment (JPL Mars Yard). The solid and dashed tracks show the position estimates from VO and wheel odometry, respectively. VO correctly estimates that the robot's initial and final positions are identical. Bottom: Slip ratio derived from the visual and wheel odometry; a ratio of 1 or −1 indicates that the wheels are spinning in place.

features must obey a rigidity constraint and that this constraint can be used to identify sets of features that are mutually consistent (a clique) prior to computing the frame-to-frame motion estimate. This approach, which can be described as *inlier detection* rather than *outlier rejection*, offers a number of advantages. The algorithm is extremely robust (even when false matches outnumber true matches), can handle dynamic scenes (because each independent motion gives rise to a separate set of self-consistent features), and is insensitive to self-shadow effects (the vehicle's shadow is just another independent motion).

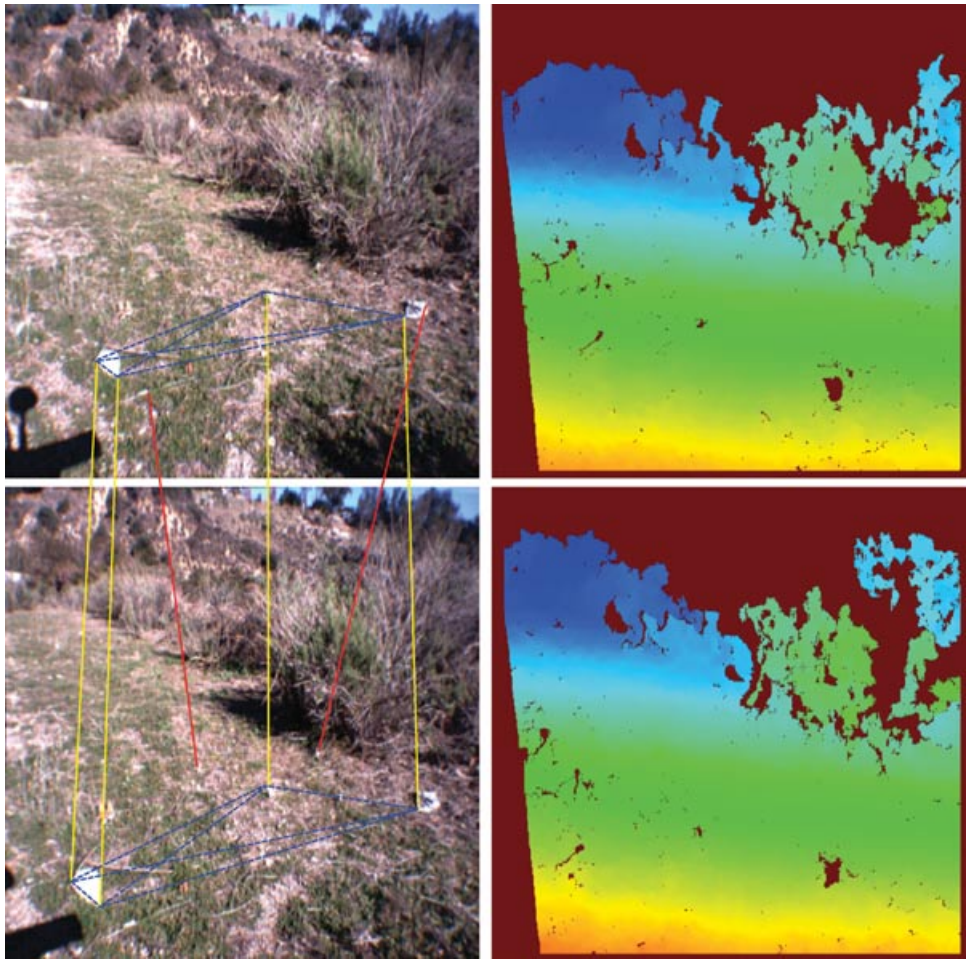

Figure 2. Stereo VO example, showing rectified (left) and disparity (right) images from sequential frames. Features are matched across frames, using the disparity data to reject false matches. The set of features joined by dotted lines (left top and bottom) form a motion clique; i.e., these features move as a single rigid body between successive frames. The remaining feature matches are assumed to be incorrect and are discarded.

Just as importantly, the algorithm does not require an initial motion estimate or inputs of any kind from wheel encoders or IMUs. As a result, it can handle large vehicle rotations and is completely insensitive to wheel/terrain interactions.

Even the best VO algorithms will inevitably suffer from failures; e.g., when the cameras are obscured by close vegetation or the scene contains very little texture. In the LAGR system, we overcome this limitation using a simple two-part filter that fuses the VO estimates with measurements from the IMU and wheel encoders. First, when VO fails, the filter falls back on the wheel encoders and IMU to estimate the frame-to-frame motion; these estimates are less accurate than those from VO but are of course much better than no estimate at all. Second, the vehicle tilt estimate is continuously corrected using the gravity vector sensed by the IMU. The filter takes the raw accelerations measured by the IMU, applies a gate on the magnitude of the acceleration to find constant-velocity segments, and then slowly rotates the vehicle attitude to align it with the gravity vector. This correction has no effect on the vehicle heading but prevents accumulation of error in the roll and pitch axes such that the vehicle always knows up from down.

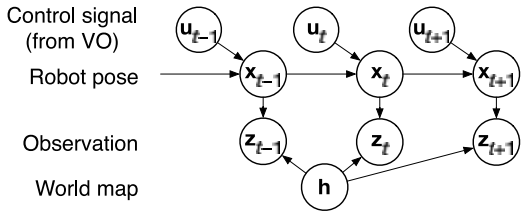

Figure 3. The goal of Gamma-SLAM at time t is to simultaneously infer $\mathbf{h}$ and $\mathbf{x}_{1:t}$ given a sequence of observations $\mathbf{z}_{1:t}$ and a sequence of control signals $\mathbf{u}_{1:t}$ (obtained from VO).

On a Core 2 Duo processor, the typical run time for the VO algorithm is less than 10 ms/frame, tracking approximately 100 features across 512×384 images. Note, however, that the algorithm is dependent on the dense stereo range data produced by the stereo correlator, which requires approximately 100 ms/frame on the same processor. For more information on the JPL VO algorithm, see Howard (2008).

5. GAMMA-SLAM FORMALISM

Our Gamma-SLAM system can be conceived as performing inference on a generative model described by the graphical model in Figure 3. Given the sequence of observations made by the robot from time 1 to t, denoted $\mathbf{z}_{1:t}$, and the sequence of control commands (determined from VO), denoted $\mathbf{u}_{1:t}$, we infer a joint posterior distribution over the map of the environment, $\mathbf{h}$, and the robot's path (pose sequence) through the environment, $\mathbf{x}_{1:t}$.

The world map is a grid of G square cells with side length 0.16 m. We assume that every point observed in cell g (where $g = 1, \ldots, G$) has its height drawn from a Gaussian distribution, with precision (inverse variance) h_g and unknown mean. The entire world map is $\mathbf{h} = \{h_g\}_{g=1}^{G}$, the collection of all grid cells.

5.1. Motion Model

The controller for our robotic vehicle maintains a full six-degree-of-freedom estimate of the robot's pose, but for navigation and planning a lower dimensional representation is sufficient. For the purposes of the algorithm described in this paper, the robot pose $\mathbf{x}_t$ consists of a 2D position $\mathbf{d}_t$ and orientation θ_t. The control signal $\mathbf{u}_t$ (obtained from VO) consists of a robot-centered translation $\Delta\mathbf{d}_t$ and rotation $\Delta\theta_t$. The motion model dictates that $\mathbf{x}_t$ is given by the previous pose incremented by the control signal, plus a small amount of Gaussian noise, $\mathbf{q}_t$:

$$\underset{\mathbf{x}_t}{\begin{bmatrix}\mathbf{d}_t\\ \theta_t\end{bmatrix}} = \underset{\mathbf{x}_{t-1}}{\begin{bmatrix}\mathbf{d}_{t-1}\\ \theta_{t-1}\end{bmatrix}} + \underset{\mathbf{u}_t}{\begin{bmatrix}\Delta\mathbf{d}_t\\ \Delta\theta_t\end{bmatrix}} + \underset{\mathbf{q}_t}{\begin{bmatrix}\mathbf{q}_{dt}\\ q_{\theta t}\end{bmatrix}}. \quad (1)$$

Here the position noise $\mathbf{q}_{dt}$ is drawn from a two-dimensional (2D) Gaussian distribution with mean $\mathbf{0}$ and variance σ_{dt}^2 in the direction of $\Delta\mathbf{d}_t$ (and variance 0 perpendicular to $\Delta\mathbf{d}_t$), where σ_{dt} is proportional to the distance traveled in the current time step:

$$\sigma_{dt} = \alpha_d \, |\Delta\mathbf{d}_t|. \quad (2)$$

The angular noise $q_{\theta t}$ is drawn from a one-dimensional (1D) Gaussian with mean 0 and variance $\sigma_{\theta t}^2$, where $\sigma_{\theta t}$ is a sum of two terms, one proportional to the distance traveled and one proportional to the angular rotation in the current time step:

$$\sigma_{\theta t} = \alpha_{\theta d} \, |\Delta\mathbf{d}_t| + \alpha_\theta |\Delta\theta_t|. \quad (3)$$

The three constant parameters, $\alpha_d, \alpha_{\theta d}$, and α_θ, characterize the VO error and are determined empirically for each robot. Although this model is somewhat crude, it has proven adequate for our purposes. In the LAGR system, VO is much more reliable than wheel odometry and behaves in generally consistent fashion across different environments (with one exception, described in Section 7.3).

5.2. Observation Model

Each observation obtained from dense stereo vision contains a large number of points with locations (x, y, z). For each point observed, we bin the xy locations (horizontal coordinates) into a square grid aligned with the world map. After the observed points are collected into cells according to their (x, y) coordinates, each cell in the grid contains a set of n heights (z coordinates), where n is the number of points observed in that cell. For each grid cell g, we assume that these heights are i.i.d. samples, $\{s_1, \ldots, s_n\}$, from a 1D Gaussian with unknown mean μ_g and precision h_g (where precision $= 1/\text{variance}$). (For simplicity, we almost always omit the subscript g and simply call these μ and h.) Note that observations may have different values of n for different

cells; for example, cells close to the robot tend to have larger n than faraway cells. We can write the likelihood of drawing these n samples as a function of the unknown population mean and precision, μ and h:

$$p(s_1, \ldots, s_n|\mu, h) = \prod_{i=1}^{n} p(s_i|\mu, h) = \prod_{i=1}^{n} f_N(s_i; \mu, h)$$
$$= (2\pi)^{-n/2} h^{n/2} e^{-\frac{1}{2}h\sum_{i=1}^{n}(s_i-\mu)^2}, \quad (4)$$

where $f_N(s_i; \mu, h)$ denotes the 1D Gaussian PDF with mean μ and precision h.

We summarize the collection of data samples $\{s_i\}_{i=1}^n$ by its sample mean and variance, m and v:

$$m = \frac{1}{n}\sum_{i=1}^{n} s_i, \quad v = \frac{1}{k}\sum_{i=1}^{n}(s_i - m)^2, \quad \text{where } k = n - 1. \quad (5)$$

Now with some algebraic manipulation, we can rewrite the likelihood equation (4) in terms of m and v:

$$p(s_1, \ldots, s_n|\mu, h) = (2\pi)^{-n/2} e^{-\frac{1}{2}hn(m-\mu)^2} \cdot h^{n/2} e^{-\frac{1}{2}hkv}. \quad (6)$$

Integrating this likelihood over all values of $s_1, \ldots, s_n$ that have the same sufficient statistics m, v gives the likelihood of observing data with mean and variance m, v when we sample n points from a Gaussian distribution whose population mean and precision are μ, h. This likelihood is the product of a normal distribution and a gamma distribution (Raiffa & Schlaifer, 2000):

$$p(m, v|\mu, h; n, k) = f_N(m; \mu, hn) f_{\gamma 2}(v; h, k)$$
$$= (2\pi)^{-\frac{1}{2}} (hn)^{\frac{1}{2}} e^{-\frac{1}{2}hn(m-\mu)^2}$$
$$\cdot \frac{1}{\Gamma\left(\frac{1}{2}k\right)} \left(\tfrac{1}{2}hk\right)^{k/2} v^{\frac{1}{2}k-1} e^{-\frac{1}{2}hkv}. \quad (7)$$

As may be evident from the preceding equation, we use $f_{\gamma 2}$ as a convenient parameterization of the gamma function:

$$f_{\gamma 2}(v; h, k) \stackrel{\text{def}}{=} f_\gamma\left(v; \tfrac{1}{2}k, \tfrac{1}{2}hk\right)$$
$$\stackrel{\text{def}}{=} \frac{1}{\Gamma\left(\frac{1}{2}k\right)} \left(\tfrac{1}{2}hk\right)^{k/2} v^{\frac{1}{2}k-1} e^{-\frac{1}{2}hkv}. \quad (8)$$

Reducing the dimensionality of the hypothesis space for both the robot location and the map makes SLAM algorithms much more efficient. Using the pitch and roll information from the robot's IMU enables us to reduce the robot's pose to four degrees of freedom and to record the stereo range data as height values within cells in a horizontal grid map. For the environments we have tested so far, we have been able to reduce the dimensionality of both the pose space and the map further by choosing to omit the z value (absolute elevation) of the robot and of each map cell. In each observation, we know the relative z values of every point, but the evolution of the z values over time is not strongly constrained by the VO results, so we have chosen not to maintain an accurate estimate of the robot's elevation over time or of the mean elevation of the points in each map cell. This $2\frac{1}{2}$-D representation of the world has been sufficient for navigating a wide variety of off-road environments. For this reason, we do not use the mean height statistic m collected from each observation, except to compute the variance statistic v. We therefore ignore the posterior distribution over μ, computing the marginal posterior distribution over h as explained in Section 5.3. To this end, we eliminate m from the observation likelihood by integrating m out of Eq. (7):

$$p(v|h; k) = \int_m f_N(m; \mu, hn) f_{\gamma 2}(v; h, k) dm = f_{\gamma 2}(v; h, k)$$
$$= \frac{1}{\Gamma\left(\frac{1}{2}k\right)} \left(\tfrac{1}{2}hk\right)^{k/2} v^{\frac{1}{2}k-1} e^{-\frac{1}{2}hkv}. \quad (9)$$

5.3. Bayesian Update of a Map Cell

If we know the precise pose history (path) of the robot $\mathbf{x}_{1:t}$, as well as the sequence of observations $\mathbf{z}_{1:t}$, then we can infer the posterior distribution over the world map. For each grid cell g, the inferred map contains a distribution over the precision (inverse variance) of the Gaussian distribution of heights in that cell. The map can be thought of as combining all of the observations from time 1 to time t, aligned (rotated and translated) according to the pose at each time step, $\mathbf{x}_{1:t}$. Making the simplifying assumption that the precisions of the grid cells are conditionally independent given the robot's path and observations, we express the map at time t as

$$\underbrace{p(\mathbf{h}|\mathbf{x}_{1:t}, \mathbf{z}_{1:t})}_{\text{map at time } t} = \prod_{g=1}^{G} \underbrace{p(h_g|\mathbf{x}_{1:t}, \mathbf{z}_{1:t})}_{\text{one cell of the map}}. \quad (10)$$

A map is made up of cells, each of which contains the sufficient statistics (v, k) of a gamma distribution. This can be interpreted as the distribution of our beliefs about the precision h of the normal distribution of heights from which that cell's points were sampled. When we collect a new observation (a new set of data points for a map cell), we use the previous gamma distribution for the cell as our prior and then use the new data points to update our beliefs, obtaining a new gamma distribution as our posterior for that map cell.

Given the pose at time t, we know which points from the observation at time t correspond to each grid cell in the map. For any particular cell, at time t we observe some number $n = k + 1$ of points that lie in that cell (according to the pose $\mathbf{x}_t$), and we compute their data variance v using Eq. (5). This section explains how to use this observation (summarized by statistics v, k) to update the corresponding cell in the prior map (summarized by statistics v', k'), thus obtaining a posterior map (a posterior distribution over the precision of heights in this cell, summarized by statistics v'', k'').

5.3.1. The Gamma Prior

When estimating the precision h of the Gaussian distribution of heights in a cell, the conjugate prior is the natural conjugate of the likelihood distribution (9), which is a gamma distribution. We parameterize this gamma prior over h with parameters v' and k':

$$p(h|v', k') = f_{\gamma 2}\big(h; v', k'\big) = \frac{1}{\Gamma\big(\frac{1}{2}k'\big)}\big(\tfrac{1}{2}k'v'\big)^{k'/2} h^{\frac{1}{2}k'-1} e^{-\frac{1}{2}hk'v'}. \quad (11)$$

This is the prior distribution before we incorporate our current data samples; the sufficient statistics v' and k' are obtained from the data collected in previous time steps.

5.3.2. The Gamma Posterior

Suppose that for a given cell in the map, the prior distribution over h is defined by Eq. (11), with sufficient statistics (v', k'). Then we take an observation (we observe $n = k + 1$ new points from the cell), and this observation has sufficient statistics (v, k), defined by Eq. (5). By Bayes' rule, the posterior distribution for the cell in the map is proportional to the product of the prior distribution (11) and the likelihood (9):

$$p(h|v', k', v, k) = \frac{p(h|v', k')\, p(v|h; k)}{\int_h p(h|v', k')\, p(v|h; k)\, dh} = \frac{f_{\gamma 2}(h; v', k')\, f_{\gamma 2}(v; h, k)}{\int_h f_{\gamma 2}(h; v', k')\, f_{\gamma 2}(v; h, k)\, dh}. \quad (12)$$

Multiplying Eq. (11) by Eq. (9) and eliminating terms that do not depend on h, we obtain

$$p(h|v', k', v, k) \propto e^{-\frac{1}{2}h(k'v'+kv)}\, h^{\frac{1}{2}(k'+k)-1}. \quad (13)$$

We can write the posterior more simply by defining new posterior parameters k'' and v'': by letting

$$k'' = k' + k\ , \qquad v'' = \frac{k'v' + kv}{k' + k}, \quad (14)$$

we obtain

$$p(h|v', k', v, k) \propto e^{-\frac{1}{2}hk''v''}\, h^{\frac{1}{2}k''-1}. \quad (15)$$

This has the form of a gamma distribution in h. Because the posterior distribution over h is a properly normalized probability distribution, this posterior must be the following gamma distribution:

$$p(h|v', k', v, k) = f_{\gamma 2}(h; v'', k''). \quad (16)$$

If no points have yet been observed in a cell, we use the uninformative prior: $k' = 0$ and $v' = 0$ (or indeed let v' equal any finite value). For this prior, the Bayesian update (14) simply produces $k'' = k$ and $v'' = v$. Thus, if the incoming data are the first data to go into a cell, the sufficient statistics for the posterior distribution will simply be equal to the data statistics.

5.4. Likelihood of an Observation

Suppose we have our map at time $t - 1$, which is $p(\mathbf{h} \mid \mathbf{x}_{1:t-1}, \mathbf{z}_{1:t-1})$. Each cell of this map, $p(h \mid \mathbf{x}_{1:t-1}, \mathbf{z}_{1:t-1})$, is a gamma distribution that is summarized by the sufficient statistics v', k'. Suppose we observe k new points from this cell at time t. What is the likelihood that our observation will have a given data variance, v? (We will need the answer later to determine the relative weights of our particles.)

Suppose a cell of the prior map has statistics v', k', and we now observe $n = k + 1$ new points from the

same cell. The probability that our current observation will have data variance v is

$$p(v \mid v', k'; k) = \int_h p(h, v \mid v', k'; k)\, dh = \int_h p(v \mid h; k)\, p(h \mid v', k')\, dh. \quad (17)$$

By noting that this is the denominator of Eq. (12), we can use Eqs. (12) and (16) to obtain the following:

$$p(v \mid v', k'; k) = \frac{f_{\gamma 2}(v; h, k)\, f_{\gamma 2}(h; v', k')}{f_{\gamma 2}(h; v'', k'')} = \frac{\Gamma\left(\frac{1}{2}k''\right)}{\Gamma\left(\frac{1}{2}k\right)\Gamma\left(\frac{1}{2}k'\right)} \cdot \frac{(kv)^{k/2}(k'v')^{k'/2}}{(k''v'')^{k''/2}} \cdot v^{-1}. \quad (18)$$

5.5. SLAM Using RBPF

The distribution that we estimate at each time step, which we call the target distribution, is the posterior distribution over the path (pose history) of the robot $\mathbf{x}_{1:t}$, and the corresponding map. Our inputs are the history of observations $\mathbf{z}_{1:t}$ and the VO, which we treat as the control signal $\mathbf{u}_{1:t}$. We factorize the target distribution as follows:

$$\underbrace{p(\mathbf{x}_{1:t}, \mathbf{h} \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t})}_{\text{target distribution}} = \underbrace{p(\mathbf{x}_{1:t} \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t})}_{\text{filtering distribution}} \underbrace{p(\mathbf{h} \mid \mathbf{x}_{1:t}, \mathbf{z}_{1:t})}_{\text{map}}. \quad (19)$$

This factorization enables us to estimate the target distribution using RBPF (Doucet, de Freitas et al., 2000), which combines the approximate technique of particle filtering for some variables, with exact filtering (based on the values of the particle-filtered variables) for the remaining variables. We use a particle filter to approximate the first term on the right-hand side of Eq. (19), the filtering distribution for the particle filter, using discrete samples $\mathbf{x}_{1:t}^{[j]}$. For each of these samples (particles), we use exact filtering to obtain the second term in Eq. (19), the map, using the map updates described in Section 5.3.

By Bayes' rule and the probabilistic dependencies implied by the graphical model (Figure 3), we can write the filtering distribution for the particle filter as follows:

$$\underbrace{p(\mathbf{x}_{1:t} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t})}_{\text{filtering distribution}} \propto \underbrace{p(\mathbf{z}_t | \mathbf{x}_{1:t}, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t})}_{\text{importance factor}} \underbrace{p(\mathbf{x}_{1:t} | \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t})}_{\text{proposal distribution}}, \quad (20)$$

where the proportionality constant does not depend on the path $\mathbf{x}_{1:t}$. We further factor the last term:

$$\underbrace{p(\mathbf{x}_{1:t} | \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t})}_{\text{proposal distribution}} = \underbrace{p(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{u}_t)}_{\text{sampling distribution}} \underbrace{p(\mathbf{x}_{1:t-1} | \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t-1})}_{\text{prior filtering distribution}}. \quad (21)$$

Equations (20) and (21) express the filtering distribution at time t as a function of the filtering distribution at $t-1$. The sampling distribution in Eq. (21) is given by the motion model, which is expressed in Eq. (1). To compute the importance factor in Eq. (20), observe that

$$\underbrace{p(\mathbf{z}_t \mid \mathbf{x}_{1:t}, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t})}_{\text{importance factor}} = \int_{\mathbf{h}} p(\mathbf{z}_t, \mathbf{h} \mid \mathbf{x}_{1:t}, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) d\mathbf{h} = \int_{\mathbf{h}} \underbrace{p(\mathbf{z}_t \mid \mathbf{h}, \mathbf{x}_t)}_{\text{likelihood of observation}} \underbrace{p(\mathbf{h} \mid \mathbf{x}_{1:t-1}, \mathbf{z}_{1:t-1})}_{\text{prior map}} d\mathbf{h}. \quad (22)$$

Because we have assumed that each cell of the map is independent, we compute the importance factor separately for each grid cell that is nonempty in both the prior map and the current observation and then take the product over all of these grid cells to obtain the importance factor for the entire observation. To see how to compute the importance factor for a grid cell, note that Eq. (22) for a single map cell is precisely the observation likelihood (17), so we can compute it using Eq. (18).

In Gamma-SLAM (or, indeed, in any grid-based RBPF SLAM algorithm), data association refers to the assignment of the observations to the cells in the global map. Each particle has its own discrete hypothesis about the sequence of poses $\mathbf{x}_{1:t}$, and that sequence of poses specifies exactly the data association of every stereo point observed to a grid cell in the global map. Thus, each particle has its own data association, which contributes to that particle's importance weight.

6. GAMMA-SLAM ALGORITHM

We outline the entire Gamma-SLAM algorithm in Section 6.1. A key part of the algorithm is the computation of the importance factor for each particle, which we explain in detail in Section 6.2. In Section 6.3, we describe our real-time implementation of the algorithm.

6.1. Outline of the Algorithm

At time step t, estimate the effective number of particles (Doucet, Godsill, & Andrieu, 2000) using the particle weights from all J particles at time $t-1$:

$$J_{\text{eff}} = \frac{1}{\sum_{i=1}^{J} \left(w_{t-1}^{[i]}\right)^2}. \tag{23}$$

Then do the following for each particle $j = 1, 2, \ldots, J$:

- If $J_{\text{eff}} \geq J/2$: Let $\tilde{\mathbf{x}}_{1:t-1}^{[j]} = \mathbf{x}_{1:t-1}^{[j]}$, and let $\tilde{w}_{t-1}^{[j]} = w_{t-1}^{[j]}$.
- If $J_{\text{eff}} < J/2$ [Resampling step]: Sample (with replacement) a particle $\tilde{\mathbf{x}}_{1:t-1}^{[j]}$ from the previous time step's particle set, $\{\mathbf{x}_{1:t-1}^{[i]}\}_{i=1}^{J}$, with probability $w_{t-1}^{[i]}$. Set $\tilde{w}_{t-1}^{[j]} = 1/J$.
- **Prediction:** Sample a new pose from the proposal distribution, given by the motion model of Eq. (1):

$$\mathbf{x}_t^{[j]} \sim p\left(\mathbf{x}_t \mid \tilde{\mathbf{x}}_{t-1}^{[j]}, \mathbf{u}_t\right). \tag{24}$$

  Append the new pose onto the particle's path (pose history):

$$\mathbf{x}_{1:t}^{[j]} = \left\{\tilde{\boldsymbol{x}}_{1:t-1}^{[j]}, \mathbf{x}_t^{[j]}\right\}. \tag{25}$$

- **Map update:** Rotate and translate the observation $\mathbf{z}_t$ according to pose $\mathbf{x}_t^{[j]}$, so that the observation aligns with the particle's existing map (so the grid cell boundaries are in the same locations). For each grid cell that is observed, compute the data statistics v, k, and then combine this with the prior map for this cell (statistics v', k') by updating the map using Eq. (14). This gives the statistics v'', k'', which comprise that cell of the particle's map at time t.
- **Importance weight:** For each grid cell g that is nonempty in both the current observation and the prior map, compute an importance factor, $\lambda_{tg}^{[j]}$, using the logarithm of Eq. (18):

$$\begin{aligned}\log \lambda_{tg}^{[j]} = {} & \log \Gamma\left(\tfrac{1}{2}k''\right) - \log \Gamma\left(\tfrac{1}{2}k\right) \\ & - \log \Gamma\left(\tfrac{1}{2}k'\right) + \tfrac{1}{2}\left[k \log(kv)\right. \\ & \left. + k' \log(k'v') - k'' \log(k''v'')\right] - \log v.\end{aligned} \tag{26}$$

  In Section 6.2, we explain how to combine the importance factors $\lambda_{tg}^{[j]}$ across all of the grid cells g in the observation to obtain the overall importance factor for particle j at time t, which we denote $\lambda_t^{[j]}$.

  The new weight of the particle is equal to the product of its old weight and its importance factor:

$$w_t^{[j]} \propto \tilde{w}_{t-1}^{[j]} \cdot \lambda_t^{[j]}, \tag{27}$$

  where at the end of the time step, the particle weights $w_t^{[j]}$ are normalized so $\sum_j w_t^{[j]} = 1$.

6.2. Computing the Importance Factor

Equation (26) enables us to compute, for particle j at time t, the importance factor $\lambda_{tg}^{[j]}$ of each grid cell g that is observed at time t and that exists in the particle's prior map. We now explain how we combine these individual cell's importance factors to obtain the overall importance factor for the particle, $\lambda_t^{[j]}$. In Section 6.2.1, we cover the case in which every grid cell that is observed at time t is already present in the particle's prior map. Then in Section 6.2.2, we discuss how to deal with grid cells that are in the current observation but were not previously observed.

6.2.1. Observations in Which Every Grid Cell Has Been Previously Observed

If every grid cell that is part of the observation at time t was observed previously, then the map from the previous time step gives us the sufficient statistics v' and k' for that cell, and we can use Eq. (26) to compute the importance factor $\lambda_{tg}^{[j]}$ for each grid cell. Next, we need to combine these importance factors from all of the cells to compute $\lambda_t^{[j]}$, the overall importance factor for particle j at time t.

If all grid cells were in fact truly independent of each other as we have assumed, then we could simply calculate the importance factor using

$$\log \lambda_t^{[j]} = \sum_{g \in G_{\text{obs}}} \log \lambda_{tg}^{[j]}, \tag{28}$$

where G_{obs} is the set of grid cells in the observation at time t. In practice, however, this equation results in an observation likelihood that is too steep (the best particle has almost all of the weight, whereas its competitors have almost none), which is probably due (at least in part) to the assumption (10) that the grid cells are independent of each other. This assumption is unrealistic—for example, adjacent cells in a map often have similar height variance. To counteract this effect, we divide by a constant β times the number N_{obs} of grid cells in the current observation:

$$\log \lambda_t^{[j]} = \frac{1}{\beta N_{\text{obs}}} \sum_{g \in G_{\text{obs}}} \log \lambda_{tg}^{[j]}. \tag{29}$$

If the robot's actual environment exactly matched the assumptions of our model, then Eq. (28) would be sufficient, and the constant parameter β introduced in Eq. (29) would not be necessary. Thus, the purpose of β is to compensate for the discrepencies between the generative model and the real-world system. We suspect that the need for β arises due to two inaccurate assumptions of the model. The first, as we have just discussed, is the assumption of independence between grid cells (10). The second is the assumption that the heights of the points observed in each cell are drawn from a Gaussian distribution. A Gaussian distribution over height is not an accurate model for some objects (e.g., the vertical wall of a building would be closer to a bounded uniform distribution than a Gaussian distribution over heights). Although the model is not a perfect representation of the world, introducing the parameter β enables the model to perform well in a variety of difficult real-world outdoor environments, as evidenced by the results in Section 7.

6.2.2. Observations in Which Some Cells Have Not Been Previously Observed

Recall from Section 5.5 that the importance factor for a grid cell is the probability that the statistics observed in the corresponding cell of the current observation were generated by that cell, given our prior experience. If we have already observed that grid cell, then we have prior statistics v', k' for the cell, and we can use Eq. (26) to compute the probability that the cell generated the new observation v, k. In practice, however, it is common for observations to contain many grid cells that have not been previously observed. For such grid cells, it is not obvious how to obtain the importance factor, because we need to know the probability that an unfamiliar grid cell generated an observation.

Because it is typical for some particles to believe that a cell in the current observation has been previously observed while other particles believe that the same cell in the current observation has never been seen before, our problem is essentially equivalent to the common problem in mobile robotic map building of deciding whether a new observation comes from an already known location or a previously unknown location. This dilemma has been referred to as the *revisiting problem* (Stewart, Ko, Fox, & Konolige, 2003) and by the question "loop closure or new place?" (Newman et al., 2007).

Ignoring previously unobserved cells when computing the importance factor. Before discussing how to compute the importance factor for previously unobserved cells, we consider an even simpler approach. The simplest way to handle such cells is to just ignore them when computing the overall importance factor. In this approach, Eq. (29) becomes

$$\log \lambda_t^{[j]} = \frac{1}{\beta N_{\text{prev}}} \sum_{g \in G_{\text{prev}}} \log \lambda_{tg}^{[j]}, \tag{30}$$

where G_{prev} represents the subset of grid cells in the current observation that have previously been observed (that already exist in the particle's prior map) and N_{prev} is the number of such grid cells. Note that this equation reduces to Eq. (29) if every cell in the current observation has been previously observed.

We have found that in some environments, such as Mars Yard courses A and B described in Section 7.1, approximation (30) works quite well. In some other cases, however, this approximation is inadequate. The greater the number of cells in an observation that have not been previously observed, the worse the approximation will be. For instance, a particle that believes that the robot has traveled far since the previous time step could be very far off from the

true pose of the robot, and that particle may believe as a result that there are only a few cells in common between the current observation and the particle's prior map. This particle ought to get a very low importance factor, but if these few cells in the prior map just happen to match well with the few cells they are compared to in the current observation, then the particle's importance factor will be much higher than it ought to be.

This improper handling of previously unknown cells can lead the algorithm to incorrectly identify a loop closure as a new location. This undesirable situation can occur if a particle that incorrectly believes the robot has returned to a parallel path adjacent to its starting path is attributed with too high of an importance factor, causing it to have more weight than another particle that correctly believes the robot has returned to its starting path. To overcome this difficulty, we need to have a model of previously unobserved cells.

Approximating unknown cells by sampling from previously observed cells. Rather than ignoring previously unobserved cells when computing the importance factor, we assume that for the purposes of computing the importance factor, each unknown cell is drawn from a prior distribution over all cells in the environment. We approximate this prior distribution as follows. Let G_{prev} represent the set of all grid cells in the current observation that are in the existing map of particle j, and let G_{new} represent the set of all grid cells in the current observation that according to the particle have not previously been observed. For each grid cell $g \in G_{\text{new}}$, we randomly select a cell from the set of all known cells in the environment (for example, from all of the cells that exist anywhere in the particle's entire map, irrespective of whether they are anywhere near the current observation) and pretend that the previously unobserved cell g has the randomly selected cell's statistics k', v'. We use these statistics from the randomly sampled cell to compute the importance factor of grid cell g using Eq. (26). (Of course, we use these randomly chosen statistics only to compute the importance factor—we do not incorporate them into the particle's new map.)

Now that we know each grid cell's contribution to the importance factor, $\lambda_{tg}^{[j]}$, we must combine these values from all grid cells in the current observation to obtain an overall importance factor for the particle, $\lambda_t^{[j]}$. We do so using the following modified version of Eq. (29):

$$\log \lambda_t^{[j]} = \frac{1}{N_{\text{obs}}} \left[\frac{1}{\beta_{\text{prev}}} \sum_{g \in G_{\text{prev}}} \log \lambda_{tg}^{[j]} + \frac{1}{\beta_{\text{new}}} \sum_{g \in G_{\text{new}}} \log \lambda_{tg}^{[j]} \right]. \tag{31}$$

In this equation, N_{obs} is the total number of grid cells in the current observation, and each of those cells belongs to exactly one of two sets: the set G_{prev} of grid cells in the observation that are already present in the particle's previous map or the set G_{new} of grid cells in the observation that according to the particle have not previously been observed. Note that now there are two separate parameters β, one for previously observed cells and one for unknown cells. Recall that we originally included β in Eq. (29) in part because the world does not fit the model's assumption that cells are independent (e.g., adjacent cells tend to be similar). Now that the previously unobserved cells' statistics are randomly sampled from a large set, these unknown cells more closely match the model's independence assumption than do the previously observed cells. As a result, we use two different constant parameters β, with $\beta_{\text{new}} \leq \beta_{\text{prev}}$.

Approximating unknown cells in this way by using samples from a prior distribution, Gamma-SLAM has been successful in a wide range of unstructured outdoor environments, as the results in Section 7 demonstrate.

6.3. Real-Time Implementation

The Gamma-SLAM algorithm has been implemented in both MATLAB and C. The C implementation runs in real time on the Herminator robot and incorporates two key optimizations to reduce CPU and memory requirements.

The first optimization improves run-time performance by fusing image data into a series of intermediate local maps. These robot-centered local maps integrate the range data acquired over a short distance (generally a few meters of travel) and are used as the observations $\mathbf{z}_t$ for the SLAM algorithm. [This simple use of a local map as an intermediate representation may be somewhat reminiscent of SLAM systems that focus on more involved methods for fusing local maps, such as that of Tardos, Neira, Newman,

and Leonard (2002), or systems whose global maps are represented as a collection of local maps, such as that of Bosse, Newman, Leonard, and Teller (2004) or Grisetti et al. (2007).] As a consequence, the relatively expensive global map update and importance weight steps are performed only once every few seconds (if the vehicle is moving) or not at all if the vehicle is stationary.

We justify this approximation by noting that, over distances of a few meters, uncertainties in the VO pose estimate have negligible impact on the local map. On the other hand, we still perform the prediction step (using the VO estimates $\mathbf{u}_t$) at the full frame rate of 5–10 Hz, because this simplifies the motion model and is relatively inexpensive. Typical execution times on a single core of a Core 2 Duo are 4 ms/particle for the map update and importance weight steps and 3 μs/particle for the prediction step. For 200 particles, the average CPU utilization is approximately 50% of a single core.

The second optimization improves memory performance by sharing portions of global maps between particles. In a naïve implementation, in which each particle stores a complete and separate global map, memory usage can quickly become prohibitive. Consider, for example, a 100×100 m map with 0.15-m grid resolution and two floating-point numbers per cell: the total map size is 7 MBytes per particle, which equates to 1.4 GBytes of storage for 200 particles. We reduce these storage requirements by introducing two optimizations. First, the global map is divided into regular *tiles*, each of which represents a small region of the total map (e.g., 10×10 m). These tiles are allocated on an as-needed basis, such that unvisited portions of the map consume little or no memory. Second, we recognize that the resampling process will create duplicate particles in the target set and that these particles can share large portions of their maps (i.e., large numbers of tiles). Sharing is implemented via a copy-on-write scheme: after resampling, all copies of a particle will share the same set of map tiles, but any attempt to update a tile will result in the creation of a new, private copy of that tile. Although this tile-based approach is not as aggressive as some map-sharing algorithms, such as those of Eliazar and Parr (2003) and Grisetti et al. (2007), it has the dual advantages of being very simple and very fast.

The net effect of this memory optimization scheme can be seen in Figure 4, which shows both the memory usage and effective particle count as the robot traverses a loop course in the Arroyo Seco (see Section 7.2). Prior to the first loop closure at around frame 37,000, memory utilization increases monotonically as the robot explores new areas in the environment. After the first loop closure, the effective number of particles J_{eff} drops, the filter is resampled, and the memory utilization drops from a peak of about 160 MBytes to just 70 MBytes (shared among 200 particles). This pattern is repeated for each subsequent resampling, with the total memory oscillating within the range 40–160 MBytes.

7. GAMMA-SLAM RESULTS

The Gamma-SLAM algorithm has been tested in three very different environments: the JPL Mars Yard (open, sandy terrain with rocks), the Arroyo Seco (open, grassy terrain with bushes and trees), and the Small Robotic Vehicle Test Bed at the Southwest Research Institute (SwRI) (maze-like paths through dense scrub). In each case, data were collected by driving the robot around one or more set courses, periodically stopping to survey the ground-truth robot position. Data were processed offline using the real-time SLAM algorithm described in the preceding section.

7.1. Mars Yard

Figures 5 and 6 show the two test courses from the JPL Mars Yard, which consists of sandy terrain littered with orange traffic poles, plastic storage bins, rocks, and a couple of buildings. Figure 5 shows the results for course A, where the robot was driven back and forth over the same terrain (three loops). The first map (lower left) shows the results using VO alone; this VO-only map is created by applying the map update rules from Section 5.3 to the pose taken directly from the VO. The second map (lower right) shows the results for the Gamma-SLAM algorithm. Looking closely at the VO-only map, it is apparent that certain objects are duplicated; these duplications are caused by drift in the VO pose estimate as the robot shuttles back and forth. In addition, a number of narrow gaps have been closed off, making effective planning virtually impossible. In contrast, the Gamma-SLAM map shows no duplication of objects, and the narrow gaps remain open. Figure 6 shows a similar set of results for course B, consisting of three large loops around the Mars Yard. The Gamma-SLAM algorithm has successfully closed the loop on each occasion as the robot returns to the starting point, whereas the VO-only map shows evidence of drift.

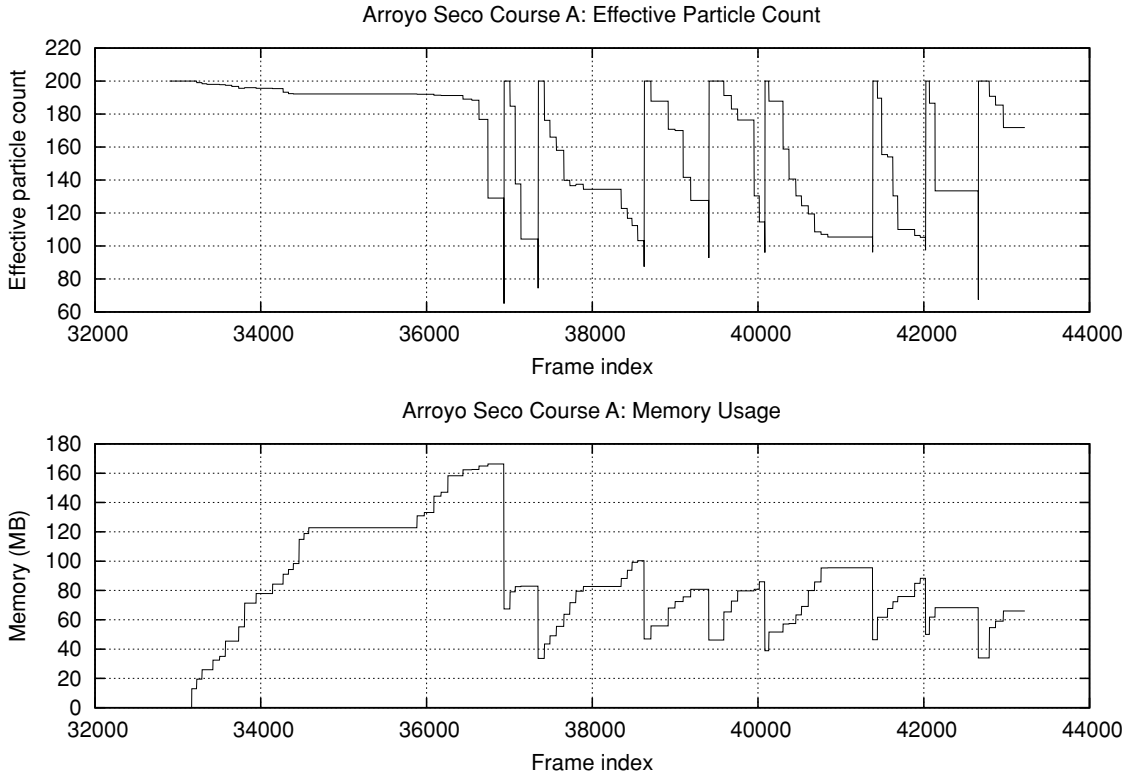

Figure 4. Effective number of particles J_{eff} (top) and memory usage (bottom) for Arroyo Seco course A (described in Section 7.2). The effective number of particles remains roughly constant until the first loop closure (around frame 37,000), at which point the particle set is resampled. Memory usage decreases after each resampling, because duplicated particles share maps. Note that the effective particle count must be less than or equal to the actual particle count of $J = 200$.

Note that the Gamma-SLAM maps capture not only the locations of obstacles (as would an occupancy grid), but also information about the heights of objects. Cells with low variance (flat areas) appear dark, whereas cells with high variance (containing boulders or buildings) appear brighter. These maps can be used directly by the LAGR planner to find optimal paths that may allow the robot to traverse some of the smaller (flatter) obstacles.

The quantitative accuracy of the Gamma-SLAM algorithm is evaluated by comparing the ground-truth robot positions (determined using an electronic TotalStation) with the corresponding SLAM estimates. These two sets of points are related by an unknown 2D rigid-body transform that we estimate via least-squares optimization; the residual error from this fit is used as a measure of accuracy. Table I shows the root-mean-squared (RMS) position errors for each test course, using Gamma-SLAM versus using VO alone. Clearly, Gamma-SLAM offers an improvement over VO, achieving an RMS error of around 0.3 m versus 1.2 m. It should also be noted that, unlike the VO estimate, the SLAM position error does not grow over

Table I. Quantitative results for the Mars Yard, Arroyo Seco, and SwRI data sets. The RMS error denotes the mean distance between the estimated robot position and the ground-truth position determined using an electronic TotalStation. Note that the SwRI results are for a single survey point only.

		RMS error (m)	
Course	Distance (m)	VO only	Gamma-SLAM
Mars Yard A	164	1.24	0.29
Mars Yard B	204	1.12	0.21
Arroyo Seco A	335	0.42	0.34
Arroyo Seco B	406	1.22	0.32
SwRI maze	543	1.98	0.65

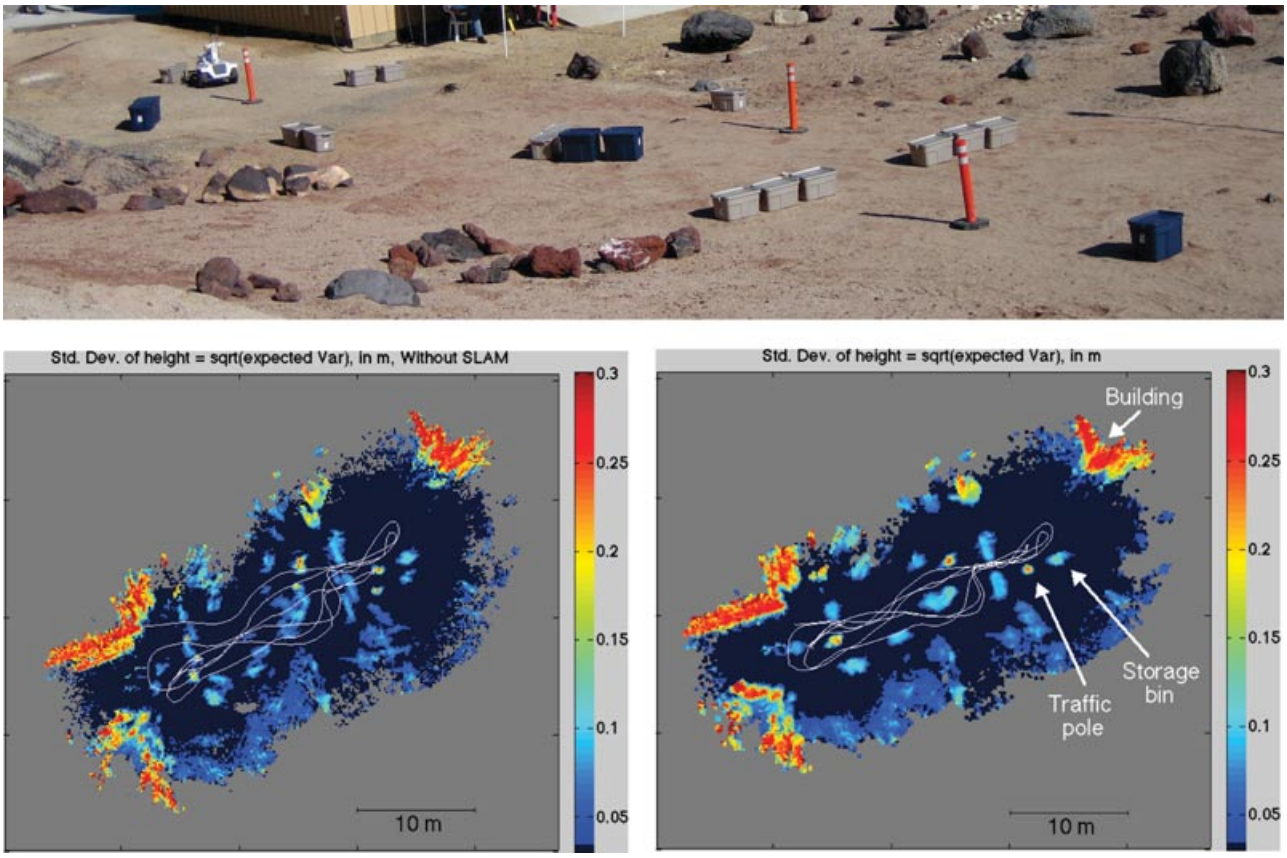

Figure 5. Mars Yard course A. Top: Photo of the course. Lower left: Map made using VO alone, with the robot path shown in white. Lower right: Map made by Gamma-SLAM with 100 particles and 0.16-m grid spacing. The left and right sides of each map correspond to the left and right sides of the photo, respectively. Cell color indicates the expected standard deviation of the heights in each grid cell.

time (so long as the vehicle is traversing previously explored terrain).

7.2. Arroyo Seco

The Gamma-SLAM algorithm was also tested in the vegetated Arroyo Seco environment shown in Figure 7. This environment is significantly less structured than the Mars Yard: the ground underfoot varies from grassy to sandy, and the vegetation ranges from spindly clumps of bushes to extremely dense shrubbery. Figures 8 and 9 show the results for two separate courses in this environment. Course A consists of three loops, totaling 335 m, around a mostly open field. Interestingly, the VO-only and Gamma-SLAM maps for course A are very nearly identical, indicating that the VO was able to achieve a very high degree of accuracy. On course B, however, the maps are noticeably different. This course consists of two loops, covering the same terrain as course A but including diversions at each end into narrow, heavily vegetated areas. Note the area at the lower left of the maps in Figure 9; the VO-only map is cluttered with duplicate obstacles, whereas the Gamma-SLAM map is clean and clearly traversable.

The quantitative comparison of the VO and Gamma-SLAM trajectories is shown in Table I. These results confirm the qualitative comparison of the maps: whereas the difference between VO and Gamma-SLAM is negligible on course A (RMS error of 0.42 m versus 0.34 m), it is very significant (from a mapping perspective) on course B (1.22 m versus 0.32 m). The consistency of the Gamma-SLAM results should also be noted. The Mars Yard and Arroyo Seco

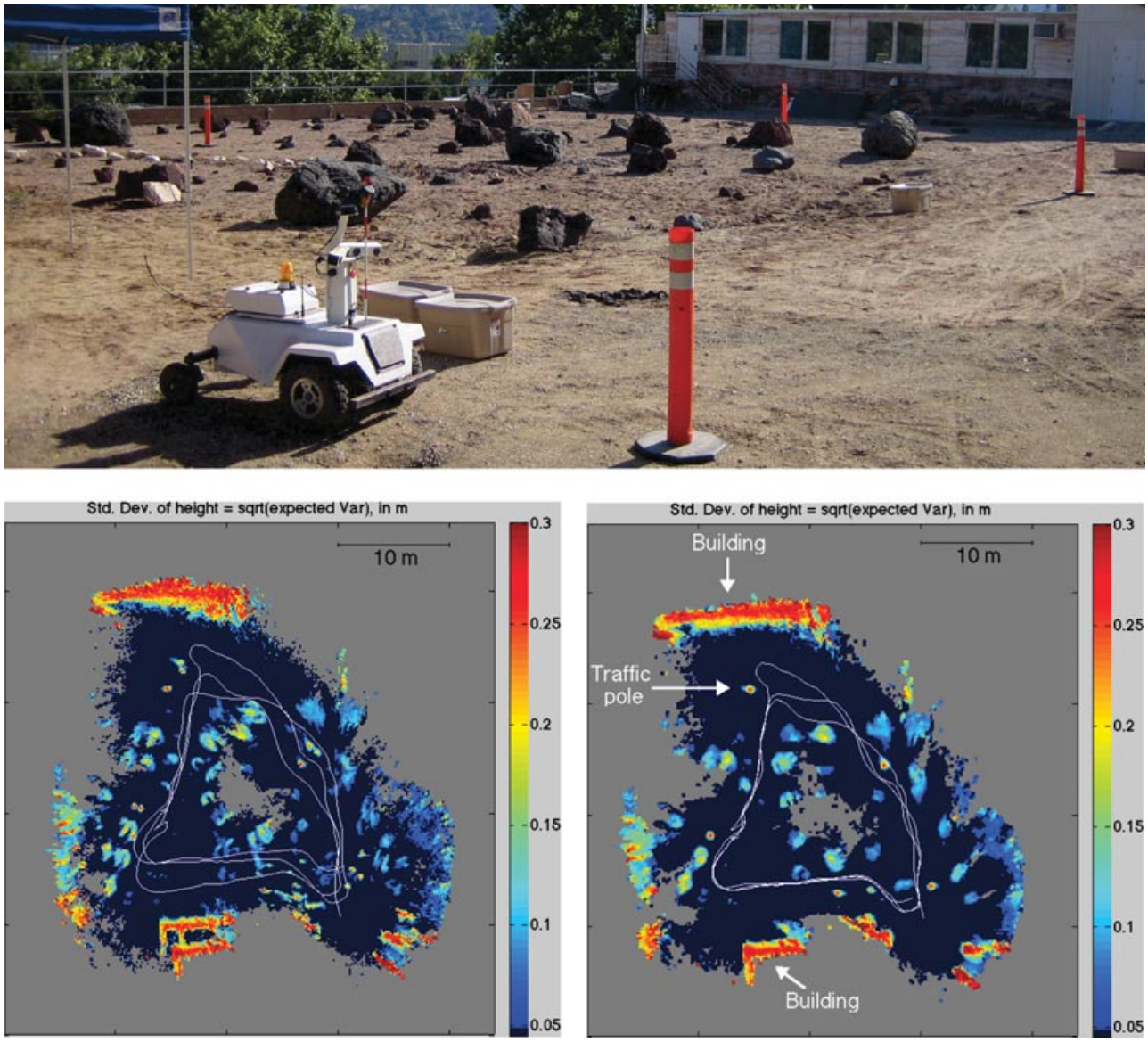

Figure 6. Mars Yard course B. Top: Photo of the course. Lower left: Map made with VO alone, with the robot path shown in white. Lower right: Map made by Gamma-SLAM with 100 particles and 0.16-m grid spacing. The buildings appear at the top and bottom of the Gamma-SLAM map.

data sets were collected 6 months apart, in different environments and using different robots. Nevertheless, the Gamma-SLAM algorithm produces near-identical accuracy.

7.3. SwRI Maze

Figure 10 shows some of the images collected in a maze-like environment at the SwRI Small Robotic Vehicle Test Bed. This maze consists of relatively narrow paths cut into dense scrub, with multiple loops and offshoots. The robot traversed four distinct loops over a total distance of 543 m, eventually returning to its approximate initial location. Results for this course are shown in Figure 11: once again, the Gamma-SLAM map improves on the VO-only map, with clearly navigable paths (blue) and no obvious duplication of obstacles (red).

Figure 7. Images from the Arroyo Seco course, showing varied terrain types and vegetation.

This environment presented by far the most difficult challenge for the algorithm. In addition to the obvious topological complexity, the environment contains specific vegetation types that cause the stereo correlator to fail at close range. This has two consequences: first, the observations for cells bordering the paths may be inconsistent, and second, the VO (which depends on stereo) is comparatively less accurate. Unlike the Mars Yard and Arroyo Seco data sets, it was necessary to tune the algorithm parameters for this environment (including the noise in the motion model and the independence factors β_{prev} and β_{new}).

Although no survey data were collected for this experiment, the initial and final robot positions are known to be approximately equal. On the basis of this one data point, we can compute an RMS position error of 0.65 m for Gamma-SLAM versus 1.98 m for VO.

8. GPS ALIGNMENT

The Gamma-SLAM algorithm described in Section 6 does not incorporate GPS measurements and is not aligned with any global reference frame. For practical purposes, however, some form of global alignment is essential, because navigation goals are usually provided in georeferenced coordinates such as latitude/longitude or universal transverse mercator (UTM). We did not attempt to integrate GPS measurements directly into the Gamma-SLAM algorithm (which is a nontrivial problem) because the benefits would be minimal for our experiments—in the environments we tested, the Gamma-SLAM maps were *much* more precise than the GPS readings. Instead, we have chosen to fit each particle trajectory against the GPS data using an efficient online algorithm. This algorithm does not change the SLAM pose estimates or map (these are still defined in an arbitrary local frame) but does allow us to compute the robot pose and map coordinates in the GPS frame, or equivalently, to transform goals defined in the global frame into goals defined in the SLAM frame. Moreover, by combining the precise relative positions determined by Gamma-SLAM with the imprecise absolute positions determined by GPS, we can obtain a measure of the robot's pose in georeferenced coordinates that is more accurate than that provided by the GPS data alone. Essentially, the robot leverages the precision of its SLAM maps to obtain a superresolution GPS reading, by combining information from all of the GPS readings obtained during the map collection. The algorithm for online GPS alignment is described in detail in the Appendix.

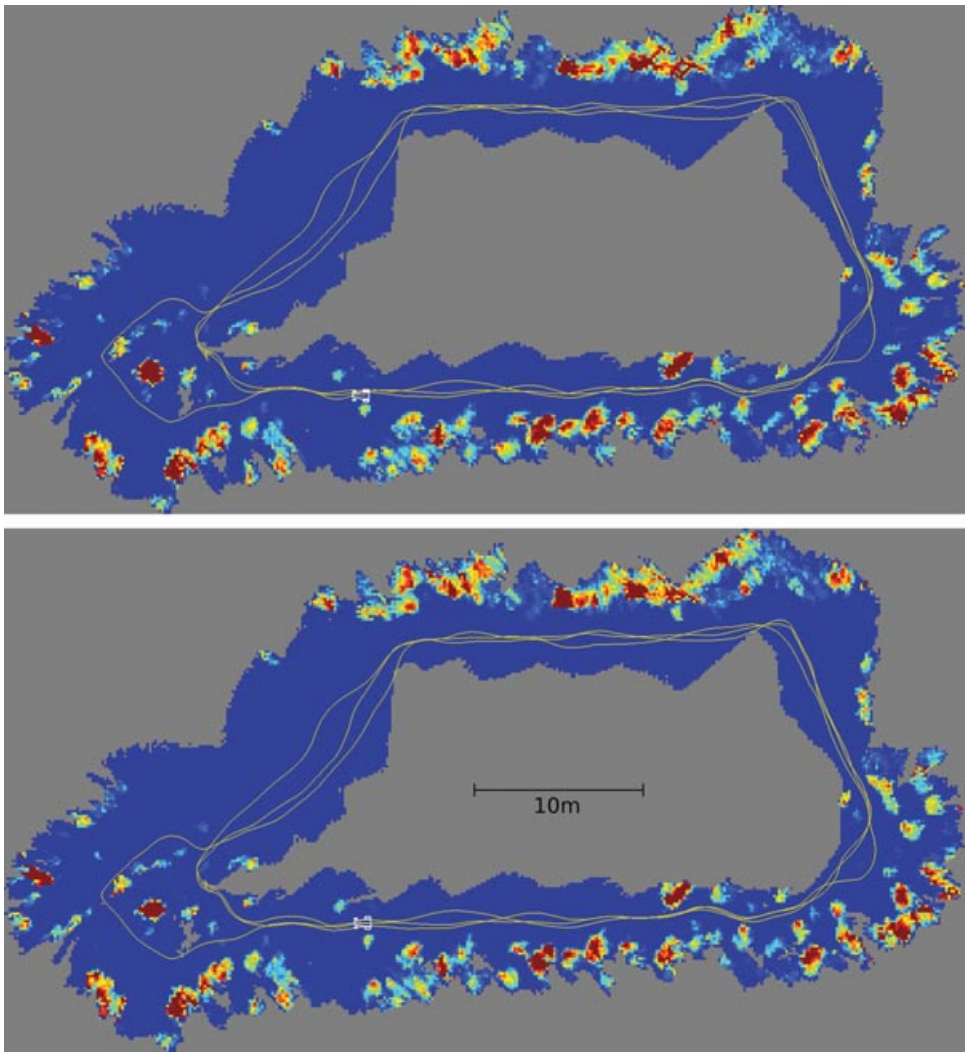

Figure 8. Maps from the Arroyo Seco course A (three loops totaling 335 m) with 200 particles and 0.15-m grid spacing. Top: Map from VO, with the robot path shown in white. Bottom: Map from Gamma-SLAM.

Figure 12 shows the GPS-aligned trajectory for the Arroyo Seco course B. For convenience, the UTM coordinates have been translated by a constant offset that defines the UTM site frame. Also shown in Figure 12 is a close-up of the SLAM maps produced on courses A and B after GPS alignment (rotation and translation) into the UTM site frame. The maps are consistent at the submeter scale, despite the fact that the two data sets were collected approximately 30 min apart, and the WAAS-enabled GPS reports a nominal accuracy of 4 m.

9. DISCUSSION

The Gamma-SLAM algorithm described in this paper is designed for easy integration with the path planner. On each planning cycle, the planner takes the current best map (augmented with any local data that have not yet been fused into the global map), computes the traversability costs, and constructs the optimal path. Unfortunately, we were not able to field an integrated SLAM and autonomous navigation system within the time contraints of the LAGR program.

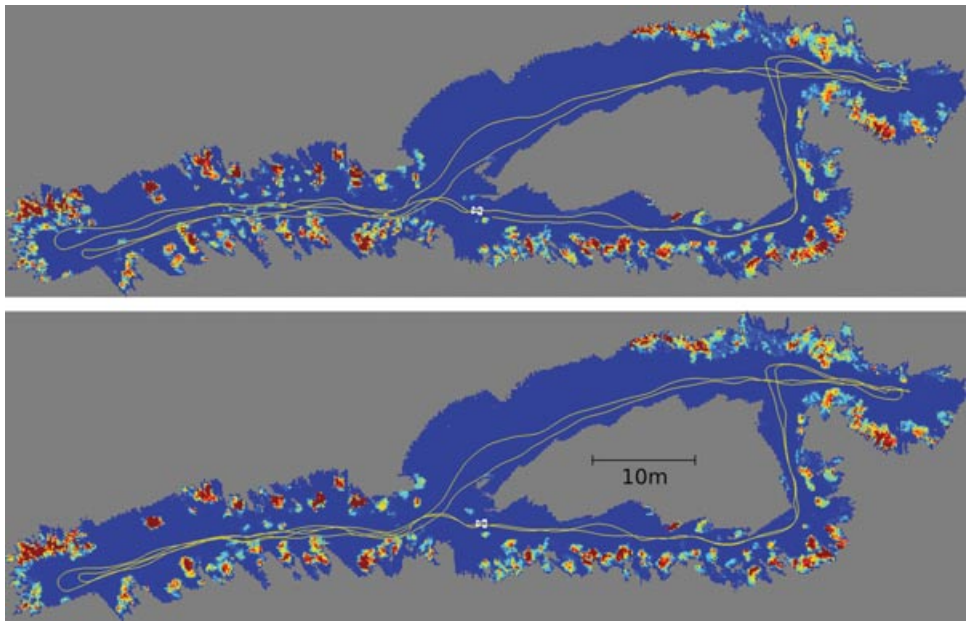

Figure 9. Maps from the Arroyo Seco course B (two loops totaling 406 m) with 200 particles and 0.15-m grid spacing. Top: Map from VO, with the robot path shown in white. Bottom: Map from Gamma-SLAM. Note the lower left region, which has duplicated features in the VO-only map but is "clean" in the Gamma-SLAM map.

Figure 10. Images from the SwRI maze course (grass-covered paths cut through dense scrub).

Consequently, all of the results presented in this paper were generated by manually driving the robot while collecting ground-truth measurements.

The fact that maps are immediately useful for planning is one of the key advantages of grid-based RBPF SLAM approaches in general and of the Gamma-SLAM algorithm in particular. In the Gamma-SLAM algorithm, elevation variance is a good proxy for cell traversability, and once Gamma-SLAM has been used to determine the correct data association (to assign each observed point to a grid cell in the global map), one can easily augment cells with other relevant statistics such as local slope, point density, and so on. This is in marked contrast to feature-based SLAM algorithms, which generally require an additional process for updating and maintaining traversability maps.

Grid-based SLAM algorithms, such as those based on occupancy grids, often involve a smoothing step as part of the map-matching process. This is particularly important in structured environments, in which objects have sharp boundaries, in order to make the objective function smooth (e.g., particles that are nearly correct will get a

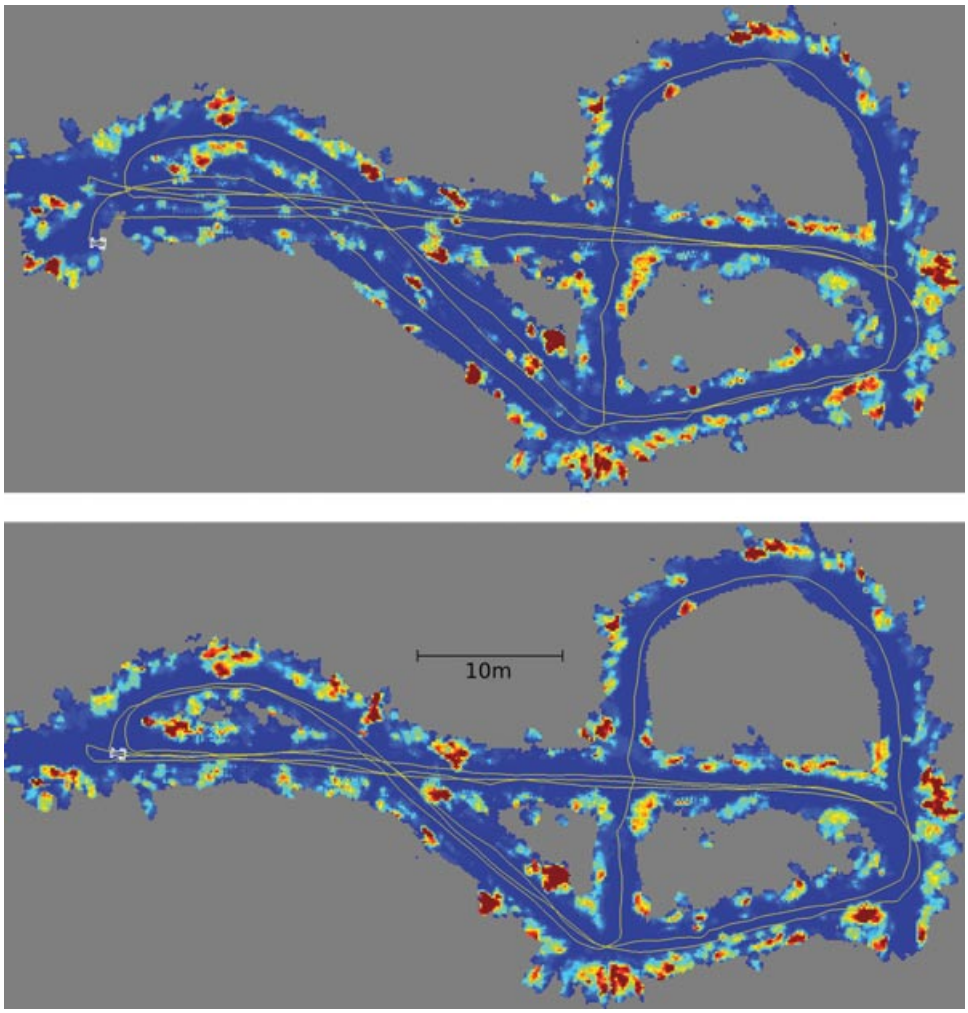

Figure 11. Maps from the SwRI maze course (multiple loops totaling 543 m) with 200 particles and 0.15-m grid spacing. Top: Map from VO, with the robot path shown in white. Bottom: Map from Gamma-SLAM.

higher importance weight than particles that are farther off the mark). In natural outdoor environments, however, smoothing is less necessary because many natural objects (such as bushes, scrub, and boulders) are inherently smooth—they extend over multiple grid squares and have gradual boundaries. For this reason, we do not do any explicit smoothing of the grid maps in our Gamma-SLAM implementation. Even without explicit smoothing, however, Gamma-SLAM is also effective with discrete obstacles that have sharp boundaries, such as the traffic poles, storage bins, and buildings shown in Figure 5. The reason is that *implicit* smoothing was still taking place. Some of this implicit smoothing is due to the fact that when an object lies across the boundary of multiple grid cells, it contributes observation points to all of the cells in which it lies. More smoothing is due to inherent uncertainty in the stereo measurements. Because this stereo error is greater for points that are farther from the robot, we restrict the observations to points within a limited range: we discard stereo points that are farther than a prespecified distance (5 or 10 m) away from the robot. The stereo measurements for the points that remain are fairly accurate, but the errors that still remain provide implicit spatial smoothing.

The Gamma-SLAM algorithm has proven its ability to generate both accurate pose estimates and

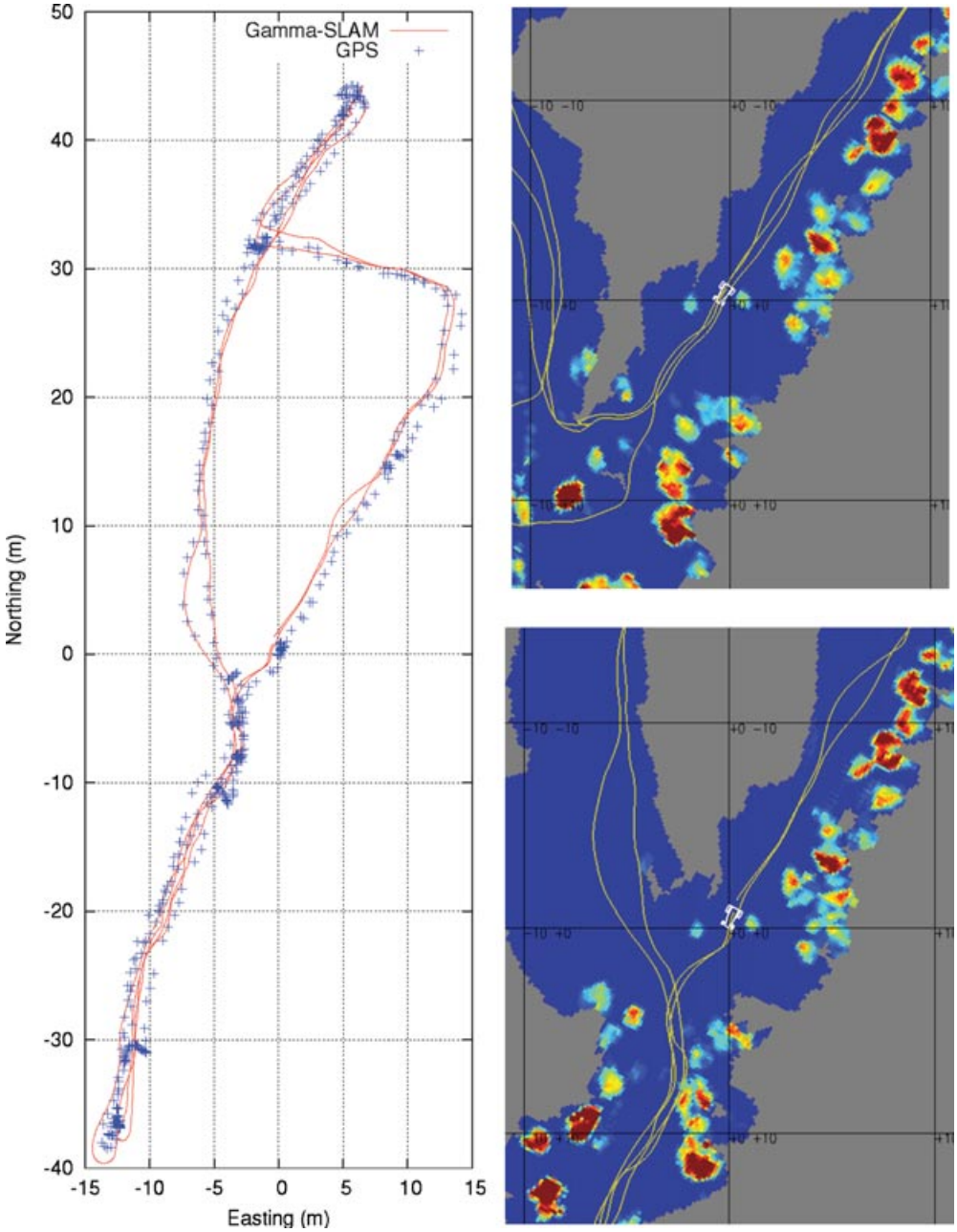

Figure 12. GPS alignment from the Arroyo Seco experiment. Left: Gamma-SLAM trajectory aligned in the UTM site frame (course B). Right: maps from courses A and B after rotation and translation into the UTM site frame.

self-consistent maps in a wide variety of environments. There are, however, some key limitations and unsolved challenges. First, the algorithm is designed to operate in environments that are effectively 2D, such as open fields and trails. Individual trees and forested areas can be handled by detecting and removing overhanging structures from the data, but the approach will not generalize to environments containing arbitrary 3D structures, such as overpasses, tunnels, or bridges. Second, the algorithm can handle only loops of a certain size, beyond which the particle set becomes very sparse and loop closure

becomes highly improbable. The current implementation can reliably close loops of over 100 m using 200 particles, but the practical upper bound is not known. Third and finally, more work is required to address the problem of map reuse and map merging; that is, we desire the ability to merge two maps, collected at different times or by different robots, into a single self-consistent representation. The GPS alignment algorithm described in Section 8 provides only a partial solution to this problem; although it can align maps with submeter accuracy, the residual uncertainty can be enough to blur key features. Recent work on distributed mapping and map merging in indoor environments (Carpin, Birk, & Jucikas, 2005; Konolige, Fox, Limketkai, Ko, & Stewart, 2003; Makarenko, Williams, & Durrant-Whyte, 2003) may be relevant here.

In Gamma-SLAM, we maintain a posterior distribution over the precision (inverse variance) of the heights in each grid cell, but for our SLAM algorithm, we discard the information about the mean height by integrating it out of the probability distributions in Section 5. We have chosen not to maintain an accurate estimate of the robot's elevation over time or of the mean elevation of the points in each map cell, because the evolution of the elevation of the robot over time is not strongly constrained by the VO results or by the IMU and because it was not necessary for the environments we tested. If we had chosen not to marginalize over the mean in our algorithm, an uncorrected drift in the mean elevation values would lead to an artificially inflated measurement of the elevation variance when the robot returns to the same cell after a long delay (e.g., the second time around a large loop). However, in other experimental setups, it might be advantageous to estimate the posterior distribution of the mean height of each cell as well as its precision. In that case, rather than a gamma distribution, the natural conjugate distribution would be a normal-gamma distribution (Raiffa & Schlaifer, 2000).

10. CONCLUSION

This paper describes Gamma-SLAM, a stereo vision-based algorithm for SLAM that uses a grid-based world representation. Rather than occupancy grids or elevation maps, Gamma-SLAM is based on *variance grid maps*, which maintain an estimate of the variance of heights of the points that are observed in each cell of a 2D grid. The update rules and observation likelihood for variance grid maps are derived and utilized in a RBPF-based SLAM framework. The system is implemented in real time, and experiments demonstrate that Gamma-SLAM produces accurate, consistent maps in a variety of challenging unstructured outdoor environments.

APPENDIX: GPS ALIGNMENT ALGORITHM

We begin this Appendix by describing a batch algorithm for GPS alignment. Then we derive an efficient online version of the algorithm. This algorithm was used to obtain the results described in Section 8 (and shown in Figure 12).

A.1. Batch Algorithm for GPS Alignment

A WAAS-enabled GPS receiver on the robot provides estimates of latitude, longitude, and horizontal estimated position error (EPE), with typical errors of 3–10 m (depending on how much sky is visible). For simplicity, we assume that this horizontal error estimate σ is proportional to the standard deviation of the error distribution of that GPS reading.

Define the $2 \times t$ matrix $\mathbf{B}$ so that its ith column, $\mathbf{b}_i$, contains the 2D position of the robot inferred by Gamma-SLAM at time step i (where $i = 1, \ldots, t$). Note that the values in $\mathbf{B}$ (in meters) are in a *local* coordinate system. Let the $2 \times t$ matrix $\mathbf{A}$ contain the GPS reading of the 2D global position of the robot at each of the same time steps, $\mathbf{a}_i$, in the *global* (UTM) coordinate system (also in meters). Finally, let σ_i denote the GPS receiver's horizontal EPE for the reading at time i.

Our goal is to find the 2×1 translation vector $\boldsymbol{l}$ (in meters) and 2×2 rotation matrix $\mathbf{R}$ that transform the SLAM map from local coordinates to global coordinates with the least error from the GPS signal. Furthermore, the measure of how egregious a particular error is should be divided by the corresponding horizontal EPE: a difference of 10 m is more worrisome when the EPE reading is 5 m than when the EPE reading is 20 m. Thus, our goal is to find $\mathbf{R}$ and $\boldsymbol{l}$ that minimize the following error function:

$$\epsilon = \sum_{i=1}^{t} \left\| \frac{\mathbf{a}_i - (\mathbf{R}\mathbf{b}_i + \boldsymbol{l})}{\sigma_i} \right\|_2^2. \tag{A.1}$$

Let $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$ represent the following weighted averages of the GPS positions and the SLAM positions,

respectively:

$$\boldsymbol{\mu} = \frac{1}{\sum_{i=1}^{t} \omega_i^2} \sum_{i=1}^{t} \omega_i^2 \mathbf{a}_i\,, \qquad \boldsymbol{\nu} = \frac{1}{\sum_{i=1}^{t} \omega_i^2} \sum_{i=1}^{t} \omega_i^2 \mathbf{b}_i\,, \quad \text{where } \omega_i = \frac{1}{\sigma_i}. \tag{A.2}$$

It can be shown that the rotation and translation that minimize the error function (A.1) map from $\boldsymbol{\nu}$ to $\boldsymbol{\mu}$. In other words, if we subtract the weighted mean $\boldsymbol{\mu}$ from each $\mathbf{a}_i$ and subtract the weighted mean $\boldsymbol{\nu}$ from each $\mathbf{b}_i$, then the translation from one to the other that minimizes the error function is the zero translation. Denote these mean-subtracted GPS positions and mean-subtracted SLAM positions, respectively, by

$$\tilde{\mathbf{a}}_i = \mathbf{a}_i - \boldsymbol{\mu}, \qquad \tilde{\mathbf{b}}_i = \mathbf{b}_i - \boldsymbol{\nu}. \tag{A.3}$$

Define the $2 \times t$ matrix $\tilde{\mathbf{A}}$ so that the ith column is the vector $\tilde{\mathbf{a}}_i$, and similarly define the $2 \times t$ matrix $\tilde{\mathbf{B}}$ so that its ith column is $\tilde{\mathbf{b}}_i$. Then we can rewrite the error function (A.1) as

$$\epsilon = \sum_{i=1}^{t} \left\| \omega_i \left(\tilde{\mathbf{a}}_i - \mathbf{R}\tilde{\mathbf{b}}_i \right) \right\|_2^2. \tag{A.4}$$

Letting $\boldsymbol{\Omega}$ be the diagonal matrix of the weights, $\boldsymbol{\Omega} = \mathrm{diag}(\omega_1, \ldots, \omega_t)$, we can rewrite the error in matrix form:

$$\epsilon = \left\| \tilde{\mathbf{A}}\boldsymbol{\Omega} - \mathbf{R}\tilde{\mathbf{B}}\boldsymbol{\Omega} \right\|_F^2, \tag{A.5}$$

where $\|\cdot\|_F^2$ represents the square of the Frobenius norm (the sum of the squares of all of the elements of a matrix).

We find the rotation matrix, $\mathbf{R}$, that minimizes the error (A.5), using the orthogonal Procrustes method (Golub & Van Loan, 1989): Take the singular value decomposition (SVD) of $\tilde{\mathbf{A}}\boldsymbol{\Omega}(\tilde{\mathbf{B}}\boldsymbol{\Omega})^T = \tilde{\mathbf{A}}\boldsymbol{\Omega}^2\tilde{\mathbf{B}}^T$:

$$\tilde{\mathbf{A}}\boldsymbol{\Omega}^2\tilde{\mathbf{B}}^T = \mathbf{U}\mathbf{S}\mathbf{V}^T, \tag{A.6}$$

where $\mathbf{S}$ is diagonal and the matrices $\mathbf{U}$ and $\mathbf{V}$ are unitary (orthonormal) matrices. Then $\mathbf{U}\mathbf{V}^T$ is the unitary transformation matrix that minimizes the error. In this case, we require a pure rotation (determinant = 1). If $|\mathbf{U}\mathbf{V}^T| = -1$ (which can happen if the robot's path is fairly linear and/or the GPS error is high), there is a simple modification (ten Berge, 2006): The rotation matrix that minimizes the error, ϵ, is

$$\mathbf{R} = \mathbf{U}\mathbf{V}^T, \qquad \text{if } \left|\mathbf{U}\mathbf{V}^T\right| = 1; \tag{A.7}$$

$$\mathbf{R} = \mathbf{U}\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}\mathbf{V}^T, \qquad \text{if } \left|\mathbf{U}\mathbf{V}^T\right| = -1. \tag{A.8}$$

A.2. Online Algorithm for GPS Alignment

An online (incremental) version of the algorithm described above requires the running updates of a few values:

$$\boldsymbol{\alpha}_t \stackrel{\text{def}}{=} \sum_{i=1}^{t} \omega_i^2 \mathbf{a}_i\,, \quad \boldsymbol{\beta}_t \stackrel{\text{def}}{=} \sum_{i=1}^{t} \omega_i^2 \mathbf{b}_i\,, \quad \gamma_t \stackrel{\text{def}}{=} \sum_{i=1}^{t} \omega_i^2. \tag{A.9}$$

The update equations from time $t-1$ to time t for these sums, as well as for the weighted averages $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$ from Eq. (A.2), are

$$\boldsymbol{\alpha}_t = \boldsymbol{\alpha}_{t-1} + \omega_t^2 \mathbf{a}_t\,, \quad \boldsymbol{\beta}_t = \boldsymbol{\beta}_{t-1} + \omega_t^2 \mathbf{b}_t\,,$$
$$\gamma_t = \gamma_{t-1} + \omega_t^2\,, \quad \boldsymbol{\mu}_t = \frac{\boldsymbol{\alpha}_t}{\gamma_t}\,, \quad \boldsymbol{\nu}_t = \frac{\boldsymbol{\beta}_t}{\gamma_t}, \tag{A.10}$$

Using these values, we can find the value at time t of the 2×2 matrix $\tilde{\mathbf{A}}\boldsymbol{\Omega}^2\tilde{\mathbf{B}}^T$ by updating from its value at time $t-1$:

$$\left[\tilde{\mathbf{A}}\boldsymbol{\Omega}^2\tilde{\mathbf{B}}^T\right]_t = \left[\tilde{\mathbf{A}}\boldsymbol{\Omega}^2\tilde{\mathbf{B}}^T\right]_{t-1} + \gamma_{t-1}(\boldsymbol{\mu}_{t-1} - \boldsymbol{\mu}_t)(\boldsymbol{\nu}_{t-1} - \boldsymbol{\nu}_t)^T + \omega_t^2 \tilde{\mathbf{a}}_t \tilde{\mathbf{b}}_t^T.$$

These update equations provide the 2×2 matrix, $\tilde{\mathbf{A}}\boldsymbol{\Omega}^2\tilde{\mathbf{B}}^T$, on which to perform SVD. The optimal transformation from local (SLAM) coordinates to global (georeferenced) coordinates is then simply obtained using Eqs. (A.6)–(A.8).

Because the update equations require us to maintain and perform only a few calculations on a fixed number of 2-vectors and 2×2 matrices, this GPS alignment requires extremely low overhead in terms of both memory and computation time, making it easy to incorporate into the real-time SLAM system.

ACKNOWLEDGMENTS

This work was performed for the Jet Propulsion Laboratory, California Institute of Technology, and was sponsored by the DARPA LAGR program through an agreement with the National Aeronautics and

Space Administration. G.W.C. is supported in part by National Science Foundation grant SBE-0542013.

REFERENCES

Angelova, A., Matthies, L., Helmick, D., Sibley, G., & Perona, P. (2006). Learning to predict slip for ground robots. In IEEE International Conference on Robotics and Automation (ICRA), Orlando, FL (pp. 3324–3331).

Bailey, T., & Durrant-Whyte, H. (2006). Simultaneous localization and mapping (SLAM): Part ii. IEEE Robotics and Automation Magazine, 13(3), 108–117.

Bosse, M., Newman, P., Leonard, J., & Teller, S. (2004). Simultaneous localization and map building in large-scale cyclic environments using the Atlas framework. International Journal of Robotics Research, 23(12), 1113–1139.

Carpin, S., Birk, A., & Jucikas, V. (2005). On map merging. Robotics and Autonomous Systems, 53(1), 1–14.

Dailey, M., & Parnichkun, M. (2006). Simultaneous localization and mapping with stereo vision. In Proceedings International Conference on Control, Automation, Robotics, and Vision (ICARCV), Singapore (pp. 1–6).

Doucet, A., de Freitas, N., Murphy, K., & Russell, S. (2000). Rao–Blackwellised particle filtering for dynamic Bayesian networks. In 16th Conference on Uncertainty in AI (UAI), Stanford, CA (pp. 176–183).

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

Durrant-Whyte, H., & Bailey, T. (2006). Simultaneous localization and mapping: Part I. IEEE Robotics and Automation Magazine, 13(2), 99–110.

Eliazar, A., & Parr, R. (2003). DP-SLAM: Fast, robust simultaneous localization and mapping without predetermined landmarks. In Proceedings 18th International Joint Conference on Artificial Intelligence (IJCAI-03), Acapulco, Mexico (pp. 1135–1142).

Elinas, P., Sim, R., & Little, J. J. (2006). σSLAM: Stereo vision SLAM using the Rao–Blackwellised particle filter and a novel mixture proposal distribution. In Proceedings IEEE International Conference on Robotics and Automation (ICRA), Orlando, FL (pp. 1564–1570).

Fischler, M. A., & Bolles, R. C. (1981). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM, 24(6), 381–395.

Golub, G. H., & Van Loan, C. F. (1989). Matrix computations. Baltimore, MD: Johns Hopkins University Press.

Grisetti, G., Tipaldi, G. D., Stachniss, C., Burgard, W., & Nardi, D. (2007). Fast and accurate SLAM with Rao–Blackwellized particle filters. Robotics and Autonomous Systems, 55(1), 30–38.

Haehnel, D., Burgard, W., Fox, D., & Thrun, S. (2003). An efficient FastSLAM algorithm for generating maps of large-scale cyclic environments from raw laser range measurements. In Proceedings of the IEEE/RSJ International Conference on Robots and Systems (IROS), Las Vegas, NV (pp. 206–211).

Hirschmuller, H., Innocent, P., & Garibaldi, J. (2002). Fast, unconstrained camera motion estimation from stereo without tracking and robust statistics. In Proceedings International Conference on Control, Automation, Robotics, and Vision (ICARCV), Singapore (pp. 1099–1104).

Howard, A. (2008). Real-time stereo VO for autonomous ground vehicles. In Proceedings of the IEEE/RSJ International Conference on Robots and Systems (IROS), Nice, France (pp. 3946–3952).

Howard, A., Turmon, M., Matthies, L., Tang, B., Angelova, A., & Mjolsness, E. (2007). Towards learned traversability for robot navigation: From underfoot to the far field. Journal of Field Robotics, 23(11/12), 1005–1017.

Jackel, L. D., Krotkov, E., Perschbacher, M., Pippine, J., & Sullivan, C. (2007). The DARPA LAGR program: Goals, challenges, methodology, and phase I results. Journal of Field Robotics, 24, 945–973.

Konolige, K., Fox, D., Limketkai, B., Ko, J., & Stewart, B. (2003). Map merging for distributed robot navigation. In Proceedings of the IEEE/RSJ International Conference on Robots and Systems (IROS), Las Vegas, NV (pp. 212–217).

Kümmerle, R., Triebel, R., Pfaff, P., & Burgard, W. (2007). Monte Carlo localization in outdoor terrains using multi-level surface maps. In International Conference Field and Service Robotics (FSR), Chamonix, France (pp. 213–222).

Makarenko, A. A., Williams, S. B., & Durrant-Whyte, H. F. (2003). Map merging for distributed robot navigation. In Proceedings of the IEEE/RSJ International Conference on Robots and Systems (IROS), Las Vegas, NV (pp. 3258–3263).

Montemerlo, M., & Thrun, S. (2003). Simultaneous localization and mapping with unknown data association using FastSLAM. In Proceedings IEEE International Conference on Robotics and Automation (ICRA), Taipei, Taiwan (pp. 1985–1991).

Newman, P., Chandran-Ramesh, M., Cole, D., Harrison, A., Posner, I., & Schröter, D. (2007). Describing, navigating and recognising urban spaces—Building an end-to-end SLAM system. In Proceedings 13th International Symposium of Robotics Research (ISRR), Hiroshima, Japan.

Pfaff, P., Triebel, R., & Burgard, W. (2007). An efficient extension to elevation maps for outdoor terrain mapping and loop closing. International Journal of Robotics Research, 26(2), 217–230.

Raiffa, H., & Schlaifer, R. (2000). Applied statistical decision theory. Wiley Classics Library.

Se, S., Barfoot, T., & Jasiobedzki, P. (2005). Visual motion estimation and terrain modeling for planetary rovers. In Proceedings International Symposium on Artificial Intelligence, Robotics and Automation in Space (iSAIRAS), Munich, Germany.

Sim, R., Elinas, P., Griffin, M., Shyr, A., & Little, J. J. (2006). Design and analysis of a framework for real-time vision-based SLAM using Rao–Blackwellised particle filters. In Proceedings of the 3rd Canadian Conference on Computer and Robotic Vision (CRV), Quebec City, Canada (pp. 21–27).

Sim, R., Elinas, P., & Little, J. J. (2007). A study of the Rao-Blackwellised particle filter for efficient and accurate vision-based SLAM. International Journal of Computer Vision, Special Issue on Vision in Robotics, 74(3), 303–318.

Sim, R., & Little, J. J. (2006). Autonomous vision-based exploration and mapping using hybrid maps and Rao–Blackwellised particle filters. In Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS), Beijing, China (pp. 2082–2089).

Stewart, B., Ko, J., Fox, D., & Konolige, K. (2003). The revisiting problem in mobile robot map building: A hierarchical Bayesian approach. In Proceedings Conference on Uncertainty in AI (UAI) (pp. 551–558).

Tardos, J. D., Neira, J., Newman, P. M., & Leonard, J. J. (2002). Robust mapping and localization in indoor environments using sonar data. International Journal of Robotics Research, 21(4), 311–330.

ten Berge, J. M. (2006). The rigid orthogonal Procrustes rotation problem. Psychometrika, 71(1), 201–205.

Triebel, R., Pfaff, P., & Burgard, W. (2006). Multi level surface maps for outdoor terrain mapping and loop closing. In Proceedings of the IEEE/RSJ International Conference on Robots and Systems (IROS) (pp. 2276–2282).