2013 IEEE International Conference on Robotics and Automation (ICRA)
Karlsruhe, Germany, May 6-10, 2013

Towards Relative Continuous-Time SLAM

Sean Anderson and Timothy D. Barfoot

***Abstract*— Appearance-based batch nonlinear optimization techniques for simultaneous localization and mapping (SLAM) have been highly successful in assisting robot motion estimation. Traditionally, these techniques are applied in a single privileged coordinate frame, which can become computationally expensive over long distances, particularly when a loop closure requires the adjustment of many pose variables. Recent approaches to the problem have shown that a completely relative coordinate framework can be used to incrementally find a close approximation of the full maximum likelihood solution in constant time. However, due to the nature of these discrete-time techniques, the state size becomes intractable when challenged with high-rate sensors. We propose moving the relative coordinate formulation of SLAM into continuous time by estimating the velocity profile of the robot. We derive the relative formulation of the continuous-time robot trajectory and formulate an estimator for the SLAM problem using temporal basis functions. Although we do not yet take advantage of large-scale loop closures, we intentionally use a relative formulation to set the stage for future work that will close loops in constant time. We show how the estimator can be used in a window-style filter to incrementally find the batch solution in constant time. The estimator is validated on a set of appearance-based feature measurements acquired using a two-axis scanning laser rangefinder over a 1.1km trajectory.**

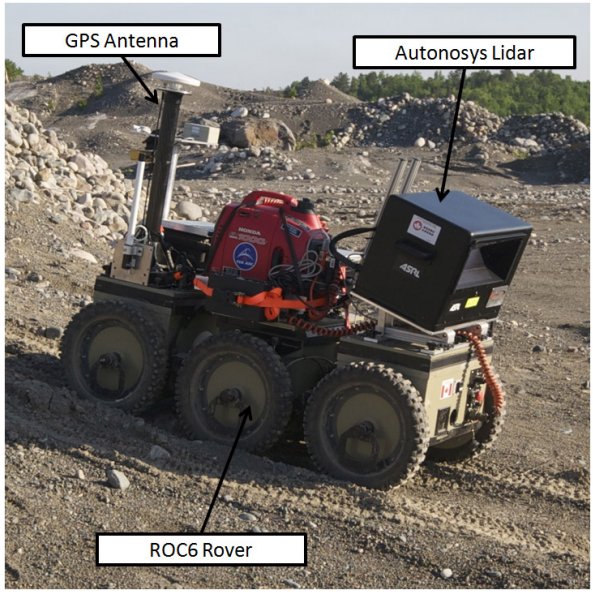

Fig. 1. The ROC6 mobile rover, equipped with an Autonosys LVC0702 lidar and a Thales DG-16 Differential GPS unit.

I. Introduction

State-of-the-art autonomy for mobile robots has favoured passive camera technology due to low cost and the overwhelming success of appearance-based techniques. Specifically, the use of sparse features for visual odometry with a stereo camera has become a leading technique in full six-degree-of-freedom motion estimation. For example, all the Mars rovers use stereo cameras for 3D perception and visual odometry [1].

Due to the effectiveness of these algorithms, much recent effort has been focused on what is known as the full SLAM, or bundle adjustment, problem. The objective of this problem is to minimize the reprojection error of many temporally matched image features by parameterizing both the 3D feature positions and the 6D sensor pose at each image acquisition time. SLAM is considered a core competency for autonomous mobile robots [2] because it is often a prerequisite to path planning, navigation and manipulation tasks. Traditionally, SLAM techniques are applied in a single privileged coordinate frame, which can become computationally expensive over long distances, particularly when a loop closure requires the adjustment of many pose variables.

The authors are members of the Autonomous Space Robotics Lab at the Institute for Aerospace Studies, University of Toronto, 4925 Dufferin Street, Toronto, Ontario, Canada. sean.anderson@mail.utoronto.ca, tim.barfoot@utoronto.ca

A recent approach to SLAM and bundle adjustment advocates that a single privileged coordinate frame is not a requirement to accomplish many common robotic tasks. By reformulating the problem using relative coordinates, an incremental update strategy can be used to perform SLAM in constant time, even at loop closure [3].

Many of these appearance-based techniques have only been validated with the use of passive camera technology, which is reliant on consistent ambient light to generate visual features that are temporally similar in appearance. In a real-world scenario the appearance of unstructured terrain can differ drastically with changes in lighting conditions. For example, a change in the orientation of shadows, or more severely, the total absence of light.

In order to enable robust estimation under varying lighting conditions, active sensors such as lidar are an attractive alternative. Specifically, we investigate the use of an Autonosys LVC0702, a two-axis scanning lidar, as seen on our robot in Figure 1. Although it is possible to use dense lidar data for SLAM, this research is particularly interested in exploiting the accomplished visual techniques that rely on sparse feature measurements. We use the method developed by McManus et al. [4] to form lidar intensity data into images and extract Speeded-Up Robust Features (SURF) [5] that can be robustly matched, independent of lighting conditions. In contrast to a Charge-Coupled Device (CCD) that has a global shutter,

978-1-4673-5643-5/13/$31.00 ©2013 IEEE

the scanning nature of the Autonosys causes the intensity imagery to be susceptible to distortion based on the velocity of the robot and the capture rate of the sensor.

Discrete-time batch estimation techniques typically require a pose variable for each measurement with a unique timestamp. Therefore, sensors that provide high-rate asynchronous measurements cause the state size to become unmanageable. In order to create an estimation framework that is capable of incrementally building the full SLAM solution in constant time, despite the use of high-rate sensors, we propose moving the relative formulation of SLAM into continuous time.

We begin in Section II with a review of the related work. In Section III we formulate the relative continuous-time SLAM problem and derive a novel estimator using the velocity profile of the robot. We find a parametric solution using temporal basis functions and discuss how the estimator can be solved efficiently and in constant time. The estimator is validated experimentally with a 1.1km sequence of lidar intensity imagery in Section IV. Final comments and a discussion of future work are available in Section V.

II. Related Work

Using sparse feature sets to perform bundle adjustment is a problem with foundations from over half a century ago [6]. Throughout the years, estimators have focused on becoming faster and more suitable for mapping over great distances. Formulating the full nonlinear least-squares optimization problem has remained a dominant paradigm because it is both flexible and efficient [7].

Exploiting the use of a graph structure has been fruitful for many recent batch estimation algorithms [3][8][9]. The completely relative formulation of the bundle adjustment problem by Sibley et al. [3] is of particular relevance to this work. In order to run in constant time, the technique incrementally updates an adaptive region of state variables that are anticipated to change in light of new information. More importantly, the undirected graph of relative pose changes allows for constant time operation during loop closure. Similarly to the novel approach of projecting landmark estimates via a kinematic chain of relative pose estimates, we project over a continuous-time estimate of the robot's velocity profile. This profile can be thought of as a kinematic chain with an infinite number of relative pose changes.

It is well known that traditional batch estimation techniques struggle when using ranging or imaging sensors that are effected by motion. In many cases, the effects of motion distortion are usually ignored after reducing platform velocity, increasing capture rate or using a stop-scan-go motion tactic. Concerning the use of sparse appearance-based features, there are few pieces of work that venture to truly tackle this problem. The most recent work by Hedborg et al. [10] demonstrates the use of inexpensive rolling shutter cameras to perform bundle adjustment. Using appearance-based lidar data in an identical fashion to this research, Dong and Barfoot [11] are able to perform motion-compensated visual odometry. Using dense range data from a continuously-spinning 2D lidar, Bosse and Zlot [12] have developed a technique, akin to the iterative closest point (ICP) algorithm, that is capable of iteratively correcting point clouds for motion distortion. Adaptations of their work using surfels have been successful in mapping a large underground mine [13] and estimating the irregular motion of a 2D lidar attached to the end of a spring [14]. In order to keep the state size tractable, these methods all employ some type of interpolation between pose variables.

In our algorithm, we view sparse features extracted from a motion-distorted image as a set of high-rate asynchronous measurements. A more general solution to incorporate high-rate sensors in batch estimation is found by using a continuous-time estimation framework. A continuous-time representation of the robot trajectory trivializes the integrations of high-rate asynchronous measurements. Furgale et al. [15] formally derive the continuous-time SLAM problem and demonstrate the use of a parametric solution for a typical SLAM calibration problem. The use of cubic splines to parameterize the robot trajectory can also be seen in the estimation schemes derived by Bibby and Reid [16], and Fleps et al. [17]. Our approach adopts the parametric state representation due to practicality and effectiveness; however, non-parametric representations using Gaussian processes have also been applied to a similar problem [18].

Both the interpolation and continuous-time estimation schemes used thus far are derived in a single privileged coordinate frame; meaning that large-scale loop closures cannot be calculated in constant time. Our approach adopts the completely relative problem formulation by Sibley et al. [3], but aims to overcome the state size tractability issues associated with high-rate sensors by using the parametric continuous-time estimation framework derived by Furgale et al. [15]. Although the results presented in this paper do not yet formulate large-scale loop closure, we avoid the use of a privileged coordinate frame in preparation for future work that will close large loops in constant time.

III. Methodology

In this section we derive the relative formulation of SLAM in continuous time. We motivate our derivation using the typical visual SLAM problem.

A. Problem Formulation

Similarly to the typical visual SLAM problem, our goal is to minimize error by finding the optimal solution of the robot trajectory and the map parameters. The two major coordinate frames that need to be defined are $\boldsymbol{\mathcal{F}}_{\vec{,}i}$, our inertial reference frame, and $\boldsymbol{\mathcal{F}}_{\vec{,}s}(t)$, the robot's instantaneous sensor frame. In the relative continuous-time formulation of the problem, our algorithm aims to estimate the velocity of the robot,

$$\boldsymbol{\varpi}(t) := \begin{bmatrix} \boldsymbol{\nu}_t^{t,i}(t) \\ \boldsymbol{\omega}_t^{t,i}(t) \end{bmatrix}. \tag{1}$$

This estimate is expressed in the robot's sensor frame over the time interval $t = [0, T]$, where $\boldsymbol{\nu}_t^{t,i}(t)$ and $\boldsymbol{\omega}_t^{t,i}(t)$ are

the linear and angular components. The notation, $\mathbf{v}_c^{b,a}$, refers to a vector from a to b, expressed in frame c. Note that for simplicity, all references to the time-varying sensor frame, $\mathcal{F}_s(t)$, have been replaced with the timestamp that we wish to reference, for example, time t in the velocity $\boldsymbol{\nu}_t^{t,i}$.

In addition to estimating the robot velocity, we also estimate the position of the static landmark parameters, $\mathbf{m}_l$, $l = 1 \dots L$, given a set of N measurements, $\mathbf{y}_{l,n}$, $n = 1 \dots N$, for each landmark. In order to simplify the notation, we denote $k := t_{l,n}$, the time of measurement $\mathbf{y}_{l,n}$. Similarly, we define $j := t_{l,1}$, the first measurement time of the landmark, l.

The sensor model for a single measurement is

$$\mathbf{y}_{l,n} := \mathbf{f}(\mathbf{T}_{k,j}\mathbf{p}_l) + \mathbf{n}_{l,n}, \qquad \mathbf{p}_l := \begin{bmatrix}\mathbf{m}_l \\ 1\end{bmatrix} \tag{2}$$

where $\mathbf{f}(\cdot)$ is the nonlinear camera model and the measurement noise, $\mathbf{n}_{l,n} \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_{l,n})$, is assumed to be normally distributed with covariance $\mathbf{R}_{l,n}$. Note that in relative estimation we express each landmark position, $\mathbf{m}_l = \boldsymbol{\rho}_j^{l,j}$, in frame $\mathcal{F}_s(j)$. The unknown variable, $\mathbf{T}_{k,j}$, is the 4×4 homogeneous transform matrix that specifies the pose change in the sensor frame between measurement times j and k.

In addition to the typical measurement error cost function, we also wish to add a cost term associated with motion. Assuming no knowledge of the control signal, we model the motion as a zero-mean white noise on acceleration. As shown in [15], this can be written as the Gaussian process

$$\dot{\boldsymbol{\varpi}}(t) = \mathbf{w}(t), \qquad \mathbf{w}(t) \sim \mathcal{GP}(0, \mathbf{Q}\delta(t - t')), \tag{3}$$

where the covariance is $\mathbf{Q}\delta(t - t')$ and $\delta(\cdot)$ is Dirac's delta function. We then formulate the following error terms for the measurement and motion models:

$$\mathbf{e}_{l,n} := \mathbf{y}_{l,n} - \hat{\mathbf{y}}_{l,n} = \mathbf{y}_{l,n} - \mathbf{f}(\mathbf{T}_{k,j}\mathbf{p}_l) \tag{4}$$

$$\mathbf{e}_u(t) := \dot{\boldsymbol{\varpi}}(t) \tag{5}$$

where $\hat{\mathbf{y}}_{l,n}$ is the expected sensor measurement based on the current state estimate.

The final objective function we wish to minimize is

$$J(\boldsymbol{\varpi}(t), \mathbf{m}) := J_s + J_u \tag{6}$$

$$J_s := \frac{1}{2}\sum_{l=1}^{L}\sum_{n=1}^{N} \mathbf{e}_{l,n}^T \mathbf{R}_{l,n}^{-1} \mathbf{e}_{l,n} \tag{7}$$

$$J_u := \frac{1}{2}\int_{t=0}^{T} \mathbf{e}_u(\tau)^T \mathbf{Q}^{-1} \mathbf{e}_u(\tau) d\tau, \tag{8}$$

where (8) can be found in [15].

B. Kinematics

In order to calculate the expected measurements, $\hat{\mathbf{y}}_{l,n}$, we derive the relationship between the robot velocity, $\boldsymbol{\varpi}(t)$, and the transform $\mathbf{T}_{k,j}$. We begin with our transformation matrix definition,

$$\mathbf{T}_{t,i} := \begin{bmatrix}\mathbf{C}_{t,i} & \mathbf{r}_t^{i,t} \\ \mathbf{0} & 1\end{bmatrix} \equiv \begin{bmatrix}\mathbf{C}_{t,i} & -\mathbf{C}_{t,i}\mathbf{r}_i^{t,i} \\ \mathbf{0} & 1\end{bmatrix}, \tag{9}$$

where $\mathbf{C}_{t,i}$ is a 3×3 matrix describing the rotation of the sensor frame with respect to the inertial frame and $\mathbf{r}_i^{t,i}$ is the 3×1 translation of the sensor frame with respect to the inertial frame, expressed in the inertial frame. The kinematic equation relating angular velocity to rotation is

$$\dot{\mathbf{C}}_{t,i} = -\boldsymbol{\omega}_t^{t,i\times}\mathbf{C}_{t,i}, \tag{10}$$

where $(\cdot)^\times$ is the skew-symmetric operator:

$$\begin{bmatrix}u_1 \\ u_2 \\ u_3\end{bmatrix}^\times := \begin{bmatrix}0 & -u_3 & u_2 \\ u_3 & 0 & -u_1 \\ -u_2 & u_1 & 0\end{bmatrix}.$$

Expanding the time derivative of the translational component, $\dot{\mathbf{r}}_t^{i,t}$, we find that

$$\dot{\mathbf{r}}_t^{i,t} = -\boldsymbol{\omega}_t^{t,i\times}\mathbf{r}_t^{i,t} + \boldsymbol{\nu}_t^{t,i}, \tag{11}$$

where we define

$$\boldsymbol{\nu}_t^{t,i} := -\mathbf{C}_{t,i}\dot{\mathbf{r}}_i^{t,i}. \tag{12}$$

Substituting our kinematic equations, (10) and (11), into the time derivative of the transformation matrix,

$$\dot{\mathbf{T}}_{t,i} = \begin{bmatrix}-\boldsymbol{\omega}_t^{t,i\times}\mathbf{C}_{t,i} & -\boldsymbol{\omega}_t^{t,i\times}\mathbf{r}_t^{i,t} + \boldsymbol{\nu}_t^{t,i} \\ \mathbf{0} & 0\end{bmatrix} \tag{13}$$

$$= -\begin{bmatrix}\boldsymbol{\omega}_t^{t,i\times} & -\boldsymbol{\nu}_t^{t,i} \\ \mathbf{0} & 0\end{bmatrix}\begin{bmatrix}\mathbf{C}_{t,i} & \mathbf{r}_t^{i,t} \\ \mathbf{0} & 1\end{bmatrix} \tag{14}$$

$$= -\begin{bmatrix}\boldsymbol{\nu}_t^{t,i} \\ \boldsymbol{\omega}_t^{t,i}\end{bmatrix}^\boxplus \mathbf{T}_{t,i}, \tag{15}$$

$$\dot{\mathbf{T}}_{t,i} = -\boldsymbol{\varpi}(t)^\boxplus \mathbf{T}_{t,i}, \tag{16}$$

where $(\cdot)^\boxplus$ is the SE(3) operator,

$$\mathbf{w}^\boxplus = \begin{bmatrix}\mathbf{u} \\ \mathbf{v}\end{bmatrix}^\boxplus := \begin{bmatrix}\mathbf{v}^\times & -\mathbf{u} \\ \mathbf{0}^T & 0\end{bmatrix}. \tag{17}$$

Through some manipulation, it is simple to show from the relationship in (16) that

$$\dot{\mathbf{T}}_{t,j} = -\boldsymbol{\varpi}(t)^\boxplus \mathbf{T}_{t,j}. \tag{18}$$

The time integral over the period j to k gives us

$$\mathbf{T}_{k,j} = \mathbf{1} - \int_j^k \boldsymbol{\varpi}(t)^\boxplus \mathbf{T}_{t,j} dt. \tag{19}$$

Note that this integration needs to be calculated using a method that preserves the properties of the transformation matrix. One way to do this is using the exponential map,

$$\mathbf{T} = e^{-\boldsymbol{\pi}^\boxplus}, \quad \boldsymbol{\pi} := \begin{bmatrix}\boldsymbol{\rho} \\ \boldsymbol{\phi}\end{bmatrix}, \tag{20}$$

where $\boldsymbol{\rho}$ is a translational component and $\boldsymbol{\phi}$ uses the axis-angle parameterization, $\boldsymbol{\phi} := \phi\mathbf{a}$, such that ϕ is the angle and $\mathbf{a}$ is a unit-length vector. For a constant velocity, $\boldsymbol{\varpi}$, we note the relationship $\boldsymbol{\pi} = \Delta t \cdot \boldsymbol{\varpi}$. In implementation, we approximate the solution to (19) numerically by sampling

$\boldsymbol{\varpi}(t)$ with a small timestep, Δt, and creating a chain of transformation matrices that span from time j to k.

$$\mathbf{T}_{k,j} \approx e^{-\Delta t \cdot \boldsymbol{\varpi}(k)^{\boxplus}} \dots e^{-\Delta t \cdot \boldsymbol{\varpi}(j+2\Delta t)^{\boxplus}} e^{-\Delta t \cdot \boldsymbol{\varpi}(j+\Delta t)^{\boxplus}} \tag{21}$$

Alternatively, we note that for very small pose changes, the exponential map can be approximated using

$$e^{-\boldsymbol{\pi}^{\boxplus}} \approx \mathbf{1} - \boldsymbol{\pi}^{\boxplus}. \tag{22}$$

C. Perturbing the Kinematics

In order to linearize our measurement error term, we look to solve for the effect of a small change in the state. Consider the following perturbation to the kinematics,

$$\mathbf{T} = e^{-\boldsymbol{\pi}^{\boxplus}} = e^{-\delta\boldsymbol{\xi}^{\boxplus}} e^{-\bar{\boldsymbol{\pi}}^{\boxplus}} \approx (\mathbf{1} - \delta\boldsymbol{\xi}^{\boxplus})\bar{\mathbf{T}}. \tag{23}$$

The kinematic equation, $\dot{\mathbf{T}} = -\boldsymbol{\varpi}^{\boxplus}\mathbf{T}$, becomes

$$\frac{d}{dt}((\mathbf{1} - \delta\boldsymbol{\xi}^{\boxplus})\bar{\mathbf{T}}) \approx -(\bar{\boldsymbol{\varpi}} + \delta\boldsymbol{\varpi})^{\boxplus}(\mathbf{1} - \delta\boldsymbol{\xi}^{\boxplus})\bar{\mathbf{T}}. \tag{24}$$

By expanding and dropping small terms, we find

$$\dot{\bar{\mathbf{T}}} - \delta\boldsymbol{\xi}^{\boxplus}\dot{\bar{\mathbf{T}}} - \delta\dot{\boldsymbol{\xi}}^{\boxplus}\bar{\mathbf{T}} \approx -\bar{\boldsymbol{\varpi}}^{\boxplus}\bar{\mathbf{T}} + \bar{\boldsymbol{\varpi}}^{\boxplus}\delta\boldsymbol{\xi}^{\boxplus}\bar{\mathbf{T}} - \delta\boldsymbol{\varpi}^{\boxplus}\bar{\mathbf{T}}. \tag{25}$$

Using the nominal solution, $\dot{\bar{\mathbf{T}}} = -\bar{\boldsymbol{\varpi}}^{\boxplus}\bar{\mathbf{T}}$, (25) can be manipulated to find the perturbed solution,

$$\delta\dot{\boldsymbol{\xi}}(t) = -\bar{\boldsymbol{\varpi}}(t)^{\boxtimes}\delta\boldsymbol{\xi}(t) + \delta\boldsymbol{\varpi}(t), \tag{26}$$

where $(\cdot)^{\boxtimes}$ is the SE(3) operator [19],

$$\mathbf{w}^{\boxtimes} = \begin{bmatrix}\mathbf{u}\\ \mathbf{v}\end{bmatrix}^{\boxtimes} := \begin{bmatrix}\mathbf{v}^{\times} & \mathbf{u}^{\times}\\ \mathbf{0} & \mathbf{v}^{\times}\end{bmatrix}. \tag{27}$$

We have shown in (21) how the nonlinear nominal solution can be integrated numerically. The perturbed solution in (26) is a *linear time-varying* equation of the form

$$\dot{\mathbf{x}}(t) = \mathbf{A}(t)\mathbf{x}(t) + \mathbf{B}(t)\mathbf{u}(t). \tag{28}$$

The general solution to the initial value problem is given by

$$\mathbf{x}(t) = \boldsymbol{\Phi}(t,0)\mathbf{x}(0) + \int_0^t \boldsymbol{\Phi}(t,\tau)\mathbf{B}(\tau)\mathbf{u}(\tau)d\tau, \tag{29}$$

where $\boldsymbol{\Phi}(t,\tau)$ is called the *state transition matrix*. The state transition matrix always exists and is unique. For our particular perturbed equation, it can be expressed as

$$\boldsymbol{\Phi}(t,\tau) = \bar{\boldsymbol{\mathcal{T}}}(t)\bar{\boldsymbol{\mathcal{T}}}(\tau)^{-1}, \tag{30}$$

where $(\cdot)^{\boxdot}$, is the SE(3) operator given by

$$\boldsymbol{\mathcal{T}} := \mathbf{T}^{\boxdot} = \begin{bmatrix}\mathbf{C} & \mathbf{r}\\ \mathbf{0}^T & 1\end{bmatrix}^{\boxdot} = \begin{bmatrix}\mathbf{C} & -\mathbf{r}^{\times}\mathbf{C}\\ \mathbf{0} & \mathbf{C}\end{bmatrix}. \tag{31}$$

Using (29), we find that

$$\delta\boldsymbol{\xi}(t) = \bar{\boldsymbol{\mathcal{T}}}(t)\bar{\boldsymbol{\mathcal{T}}}(0)^{-1}\delta\boldsymbol{\xi}(0) + \bar{\boldsymbol{\mathcal{T}}}(t)\int_0^t \bar{\boldsymbol{\mathcal{T}}}(\tau)^{-1}\delta\boldsymbol{\varpi}(\tau)d\tau. \tag{32}$$

D. Parametric Solution

Moving forward in this derivation, we wish to formally define a parametric representation for the continuous-time state. We choose to do this by using a weighted sum of $6B$ temporal basis functions,

$$\boldsymbol{\varpi}(t) := \boldsymbol{\Psi}(t)\mathbf{c}, \tag{33}$$

$$\boldsymbol{\Psi}(t) := \begin{bmatrix}\boldsymbol{\psi}_1(t) & \boldsymbol{\psi}_2(t) & \cdots & \boldsymbol{\psi}_{6B}(t)\end{bmatrix}, \tag{34}$$

$$\mathbf{c} := \begin{bmatrix}c_1 & c_2 & \cdots & c_{6B}\end{bmatrix}^T, \tag{35}$$

where we have B basis functions in each of the 6 problem dimensions, and each basis function, $\boldsymbol{\psi}_b(t)$, is a 6×1 column vector. With this parametric representation, we can now vary our velocity estimate, $\boldsymbol{\varpi}(t)$, using only the coefficients, $\mathbf{c}$.

With respect to the static map parameters, we define $\mathbf{m}$ to be a stacked state vector of all the landmark positions. Additionally, we define the following projection and dilation matrices, $\mathbf{P}_l$ and $\mathbf{D}$, to help with the derivation:

$$\mathbf{m}_l := \mathbf{P}_l\mathbf{m}, \qquad \mathbf{D} := \begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\\ 0 & 0 & 0\end{bmatrix}. \tag{36}$$

The joint state vector for which we wish to solve is defined as $\mathbf{z} := \begin{bmatrix}\mathbf{c}^T & \mathbf{m}^T\end{bmatrix}^T$. In order to minimize our objective function, J, we aim make it quadratic in a small perturbation of the state, $\delta\mathbf{z}$, and find the optimal state update $\delta\mathbf{z}^*$ by solving $\frac{\partial^T J}{\partial \delta\mathbf{z}} = \mathbf{0}$. In order to do this, we must first linearize the error terms with respect to $\delta\mathbf{z}$ and then find an initial guess for the nominal solution, $\bar{\mathbf{z}}$. We note that the linearization of J_u, is very similar to the one shown in [15], and therefore this paper will focus on the linearization of J_s.

Noting that (4) has a composition of two nonlinearities, one for the sensor model and one for transforming the landmark into the sensor frame, we begin by defining

$$\mathbf{g}_{l,n}(\mathbf{z}) := \mathbf{T}_{k,j}\mathbf{p}_l = \mathbf{T}(k)\mathbf{T}(j)^{-1}\mathbf{p}_l. \tag{37}$$

Using the usual perturbation scheme,

$$\mathbf{g}_{l,n}(\bar{\mathbf{z}} + \delta\mathbf{z}) \approx \left(\mathbf{1} - \delta\boldsymbol{\xi}(k)^{\boxplus}\right)\bar{\mathbf{T}}(k)\bar{\mathbf{T}}(j)^{-1} \times \left(\mathbf{1} + \delta\boldsymbol{\xi}(j)^{\boxplus}\right)(\bar{\mathbf{p}}_l + \mathbf{D}\,\delta\mathbf{m}_l). \tag{38}$$

After expanding and dropping the product of small terms,

$$\mathbf{g}_{l,n}(\bar{\mathbf{z}} + \delta\mathbf{z}) \approx \bar{\mathbf{T}}_{k,j}\bar{\mathbf{p}}_l + \begin{bmatrix}\bar{\mathbf{T}}_{k,j}\bar{\mathbf{p}}_l^{\boxminus} & \bar{\mathbf{T}}_{k,j}\end{bmatrix} \times \begin{bmatrix}\bar{\boldsymbol{\mathcal{T}}}(j)\bar{\boldsymbol{\mathcal{T}}}(k)^{-1}\delta\boldsymbol{\xi}(k) - \delta\boldsymbol{\xi}(j)\\ \mathbf{D}\,\delta\mathbf{m}_l\end{bmatrix}, \tag{39}$$

where $(\cdot)^{\boxminus}$ is the homogeneous coordinate operator,

$$\mathbf{y}^{\boxminus} := \begin{bmatrix}\boldsymbol{\epsilon}\\ \eta\end{bmatrix}^{\boxminus} = \begin{bmatrix}\eta\mathbf{1} & \boldsymbol{\epsilon}^{\times}\\ \mathbf{0}^T & \mathbf{0}^T\end{bmatrix}. \tag{40}$$

To simplify the term $\bar{\boldsymbol{\mathcal{T}}}(j)\bar{\boldsymbol{\mathcal{T}}}(k)^{-1}\delta\boldsymbol{\xi}(k) - \delta\boldsymbol{\xi}(j)$, we substitute the solution for $\delta\boldsymbol{\xi}(t)$ found in (32).

After some simplification,

$$\bar{\boldsymbol{\mathcal{T}}}(j)\bar{\boldsymbol{\mathcal{T}}}(k)^{-1}\delta\boldsymbol{\xi}(k) - \delta\boldsymbol{\xi}(j) = \bar{\boldsymbol{\mathcal{T}}}(j)\Bigg(\int_0^k \bar{\boldsymbol{\mathcal{T}}}(\tau)^{-1} \times \delta\boldsymbol{\varpi}(\tau)d\tau - \int_0^j \bar{\boldsymbol{\mathcal{T}}}(\tau)^{-1}\delta\boldsymbol{\varpi}(\tau)d\tau\Bigg) \tag{41}$$

$$= \int_j^k \bar{\boldsymbol{\mathcal{T}}}_{\tau,j}^{-1}\delta\boldsymbol{\varpi}(\tau)d\tau. \tag{42}$$

From the temporal basis function definition, $\boldsymbol{\varpi}(t) = \boldsymbol{\Psi}(t)\mathbf{c}$, it is clear that $\bar{\boldsymbol{\varpi}}(t) + \delta\boldsymbol{\varpi}(t) = \boldsymbol{\Psi}(t)(\bar{\mathbf{c}} + \delta\mathbf{c})$ and therefore $\delta\boldsymbol{\varpi}(t) = \boldsymbol{\Psi}(t)\delta\mathbf{c}$. Substituting this relationship into (42),

$$\bar{\boldsymbol{\mathcal{T}}}(j)\bar{\boldsymbol{\mathcal{T}}}(k)^{-1}\delta\boldsymbol{\xi}(k) - \delta\boldsymbol{\xi}(j) = \int_j^k \bar{\boldsymbol{\mathcal{T}}}_{\tau,j}^{-1}\boldsymbol{\Psi}(\tau)d\tau\ \delta\mathbf{c}. \tag{43}$$

Substituting (43) back into (39),

$$\mathbf{g}_{l,n}(\bar{\mathbf{z}} + \delta\mathbf{z}) \approx \bar{\mathbf{g}}_{l,n} + \mathbf{G}_{l,n}\delta\mathbf{z}, \tag{44}$$

correct to first order, where

$$\bar{\mathbf{g}}_{l,n} := \bar{\mathbf{T}}_{k,j}\bar{\mathbf{p}}_l, \tag{45}$$

$$\mathbf{G}_{l,n} := \left[\bar{\mathbf{T}}_{k,j}\bar{\mathbf{p}}_l^{\boxminus}\int_j^k \bar{\boldsymbol{\mathcal{T}}}_{\tau,j}^{-1}\boldsymbol{\Psi}(\tau)d\tau \quad \bar{\mathbf{T}}_{k,j}\mathbf{D}\,\mathbf{P}_l\right]. \tag{46}$$

Returning to our measurement error term,

$$\mathbf{e}_{l,n} \approx \mathbf{y}_{l,n} - \mathbf{f}(\bar{\mathbf{g}}_{l,n} + \mathbf{G}_{l,n}\delta\mathbf{z}) \tag{47}$$

$$\approx \bar{\mathbf{e}}_{l,n} - \mathbf{H}_{l,n}\delta\mathbf{z}, \tag{48}$$

correct to first order, where

$$\bar{\mathbf{e}}_{l,n} := \mathbf{y}_{l,n} - \mathbf{f}(\bar{\mathbf{g}}_{l,n}), \tag{49}$$

$$\mathbf{H}_{l,n} := \mathbf{F}_{l,n}\mathbf{G}_{l,n}, \qquad \mathbf{F}_{l,n} := \left.\frac{\partial\mathbf{f}}{\partial\mathbf{g}}\right|_{\bar{\mathbf{g}}_{l,n}}. \tag{50}$$

Setting $\frac{\partial^T J}{\partial\delta\mathbf{z}} = \mathbf{0}$, we find the optimal state update equation for a single iteration of Gauss-Newton:

$$(\mathbf{A}_s + \mathbf{A}_u)\delta\mathbf{z}^* = \mathbf{b}_s + \mathbf{b}_u. \tag{51}$$

With respect to J_s, $\frac{\partial^T J_s}{\partial\delta\mathbf{z}} = \mathbf{A}_s\delta\mathbf{z} - \mathbf{b}_s$, where,

$$\mathbf{A}_s := \sum_{l,n}\mathbf{H}_{l,n}^T\mathbf{R}_{l,n}^{-1}\mathbf{H}_{l,n}, \tag{52}$$

$$\mathbf{b}_s := \sum_{l,n}\mathbf{H}_{l,n}^T\mathbf{R}_{l,n}^{-1}\bar{\mathbf{e}}_{l,n}. \tag{53}$$

As in [15], we find that $\frac{\partial^T J_u}{\partial\delta\mathbf{z}} = \mathbf{A}_u\delta\mathbf{z} - \mathbf{b}_u$, where,

$$\mathbf{A}_u := \begin{bmatrix}\int_0^T \dot{\boldsymbol{\Psi}}(\tau)^T\mathbf{Q}^{-1}\dot{\boldsymbol{\Psi}}(\tau)\ d\tau & \mathbf{0} \\ \mathbf{0} & \mathbf{0}\end{bmatrix}, \tag{54}$$

$$\mathbf{b}_u := -\mathbf{A}_u\bar{\mathbf{z}}. \tag{55}$$

Note that the term $\mathbf{A}_u$ only needs to be calculated once, as it does not depend on the state parameters. Following the normal Gauss-Newton approach, the solution to $\delta\mathbf{z}^*$ can be used to update the nominal solution $\bar{\mathbf{z}}$ in an iterative fashion.

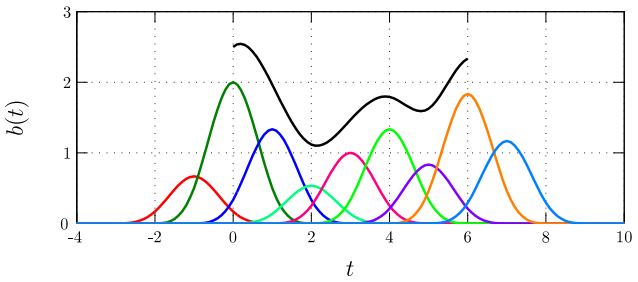

Fig. 2. This figure illustrates our use of cubic B-spline basis functions in one dimension. A single cubic B-spline is a smooth piecewise polynomial defined over 4 *time segments*. The points in time that separate two time segments are known as *knots*. We use a uniform knot spacing, such that only 4 cubic B-splines are nonzero at any time. The example shows how 9 uniformly spaced cubic B-splines can be used to define the weighted sum (black) over 6 time segments.

E. Computational Efficiency

There are three major computational costs associated with running this estimator. The first major cost is the numerical integrations of (19) and (46). We note from (41) that the integral term in $\mathbf{G}_{l,n}$ can be written as

$$\int_j^k \bar{\boldsymbol{\mathcal{T}}}_{\tau,j}^{-1}\boldsymbol{\Psi}(\tau)d\tau = \bar{\mathbf{T}}(j)^{\boxdot}(\mathbf{I}(k) - \mathbf{I}(j)), \tag{56}$$

where,

$$\mathbf{I}(t) := \int_0^t \bar{\mathbf{T}}(\tau)^{-\boxdot}\boldsymbol{\Psi}(\tau)d\tau. \tag{57}$$

Given that many of the required integral calculations will overlap, we aim to compute all the integral terms efficiently by conducting a single pass over the entire time period we are currently optimizing. In the full batch optimization, the following equations are used iteratively for $t = [0, T]$,

$$\bar{\mathbf{T}}(t + \Delta t) \leftarrow e^{-(\Delta t\cdot\boldsymbol{\Psi}(t+\Delta t)\bar{\mathbf{c}})^{\boxplus}}\bar{\mathbf{T}}(t), \tag{58}$$

$$\mathbf{I}(t + \Delta t) \leftarrow \mathbf{I}(t) + \Delta t\cdot\bar{\mathbf{T}}(t + \Delta t)^{-\boxdot}\boldsymbol{\Psi}(t + \Delta t), \tag{59}$$

where $\bar{\mathbf{T}}(0) = \mathbf{1}$ and $\mathbf{I}(0) = \mathbf{0}$. The integration step, Δt, is chosen so that we have intermediate terms for $\bar{\mathbf{T}}$ and $\mathbf{I}$ at each measurement time, $t_{l,n}$. By storing each of these intermediate steps, we can then compute $\bar{\mathbf{T}}_{k,j}$ and $\mathbf{G}_{l,n}$ for each measurement.

The second major cost in this estimator is building the matrices $\mathbf{A}_{11}$, $\mathbf{A}_{12}$ and $\mathbf{A}_{22}$. This step can be exceptionally expensive depending on the choice of basis function used to parameterize $\boldsymbol{\Psi}(t)$. Specifically, it is highly beneficial to use a basis function type that has *local support*. At any given time, we want only a subset of the basis functions, $\psi_b(t)$, to be non-zero. Using local support introduces an exploitable sparsity to the matrix $\boldsymbol{\Psi}(t)$. Our implementation of the estimator uses cubic B-spline basis functions, as seen in Figure 2, because they have local support in addition to simple analytical integrals and derivatives.

The third cost is related to solving the linear system of equations in (51). To do this efficiently, we note that it can

be written using the 2×2 block structure,

$$\begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{12}^T & \mathbf{A}_{22} \end{bmatrix} \begin{bmatrix} \delta\mathbf{c} \\ \delta\mathbf{m} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{bmatrix}. \tag{60}$$

The relative formulation of the problem and addition of the motion model change the traditional block-diagonal sparsity pattern of $\mathbf{A}_{11}$. However, since $\mathbf{A}_{22}$ is still block-diagonal and is typically much larger than $\mathbf{A}_{11}$, we may apply the Schur complement in the typical bundle adjustment fashion with no additional cost. It is simple to solve for $\delta\mathbf{c}$ while exploiting the block-diagonal sparsity of $\mathbf{A}_{22}$ to find $\mathbf{A}_{22}^{-1}$ efficiently. Back substituting the solution for $\delta\mathbf{c}$, we can then quickly calculate the landmark updates, $\delta\mathbf{m}$.

F. Constant Time Estimation

Although using the Schur complement provides large computational savings, the cost of finding $\delta\mathbf{c}$ is still cubic in the number of basis functions used to approximate $\boldsymbol{\varpi}(t)$. In order to run the estimator online we adopt a window-style filtering approach similar to [20].

Similar to how the traditional discrete-time approach optimizes a subset of the most recent pose variables, we optimize over a subset of the basis functions used in the time segments leading up to the most recent measurement. However, in order for this technique to achieve constant time performance, the subset of basis functions must have temporally local effects on the robot trajectory, $\boldsymbol{\varpi}(t)$. In order for this to be true, it is a requirement that the basis functions used in the estimator have local support.

In order to set up the local batch optimization problem, we begin by identifying the period of time affected by the active subset of basis functions. We then add all landmarks measured during the active time period to a list of active parameters that will be updated in the optimization. All measurements of these active landmarks are included when building the terms $\mathbf{A}_s$ and $\mathbf{b}_s$. This may require the full nominal solution, $\bar{\mathbf{c}}$, as some of the static basis function coefficients may be required to evaluate $\bar{\boldsymbol{\varpi}}(t)$ near and outside the bounds of the active time period.

IV. Experimental Results

In this section we test and report results for our relative continuous-time SLAM technique. The estimator is tested using a set of real data that highlight our algorithm's ability to handle high-rate asynchronous measurements, while the robot undergoes motion in unstructured terrain.

A. Dataset

The dataset we use was collected at the Ethier Sand and Gravel pit in Sudbury, Ontario, Canada. During the *Visual Teach and Repeat* experiment described in [21], the ROC6 field robot, seen in Figure 1, was taught a 1154m route. Our algorithm is tested on the initial teach pass of this route, which took roughly 51 minutes and began at 7:45 pm. For ground truth, a Thales DG-16 Differential GPS unit was used to record the robot position with a Circular Error Probability (CEP) of 40cm. An Autonosys LVC0702 lidar was used to capture intensity imagery at 2Hz with a maximum range of 53.5m. The raw laser-rangefinder data were packaged into a stack of intensity, azimuth, elevation, range and time images. The traverse that we will be using for testing contains a total of 6880 *image stacks* at a resolution of 480×360[1].

B. Processing

Using a GPU-accelerated implementation of the SURF algorithm, we extract features from the intensity imagery and generate initial feature matches using only the descriptors and a similarity threshold. The sparse feature measurements are then corrected using our Autonosys calibration model [22]. In order to filter outliers, we then apply a variant of the typical RANdom SAmple Consensus (RANSAC) algorithm [23] used in the visual odometry pipeline. We compensate for motion in the RANSAC process by applying a simplified version of our estimator with a constant velocity model.

C. Estimation

From the 6880 intensity images, we found 1,703,254 temporally matched SURF feature measurements, each with a unique timestamp. A brute-force approach using a discrete-time batch estimator would require a pose estimate at each measurement time in the system. Meaning that for the 1154m dataset, it would require $6 \times$ 1,703,254 = 10,219,524 state variables, excluding the 606,849 different landmark positions. Using basis functions to parameterize our continuous-time state, we show how the number of required state variables can be drastically reduced.

Before running the constant-time estimator on the entire 1154m trajectory, we must decide on a spacing for the temporal basis functions and an appropriate window size over which to optimize. In order to train these two parameters, we use a 250m subsection of the full trajectory. This subsection has 1400 image stacks, from which we extracted SURF features and found 459,863 temporally matched measurements of 158,468 different landmarks. In order to compare the results of the estimator to the GPS ground truth, we consider the integral of our velocity estimate, as seen in Figure 3.

In order to find an appropriate knot spacing, we ran the full batch estimator on the training section a number of times, varying the number of knots from 6 to 3000. We found the root-mean-square (RMS) Euclidean error by comparing the integrated position of our velocity estimate to the DGPS tracks. The estimate and DGPS tracks were put into a common reference frame by aligning the first 15 meters of each trajectory.

The result of varying the number of knots can be seen in Figure 4 (timing was broken down into the three major steps described in Section III-E). Using the efficient methods described earlier, we see that the integration and build steps of the algorithm are relatively constant in cost as the number of knots increases. As expected, the solution to $\mathbf{A}\delta\mathbf{z}^* = \mathbf{b}$ is cubic in complexity and begins to dominate the other costs as the number of basis functions is increased. As the knot

[1]The lighting-invariant appearance-based lidar dataset used in this work is available upon request.

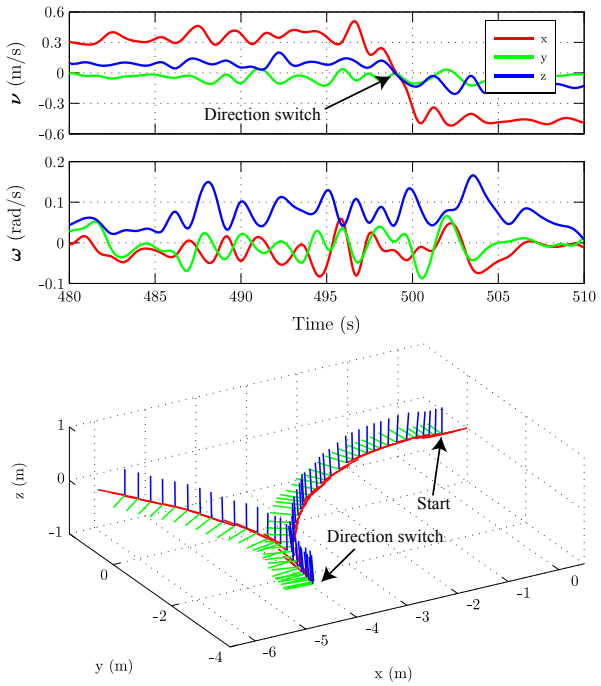

Fig. 3. Example output from our Relative Continuous-Time SLAM estimator. The sample linear and angular velocity estimates directly correspond to the integrated 3D trajectory. An animation of this process can be seen in the supplementary video.

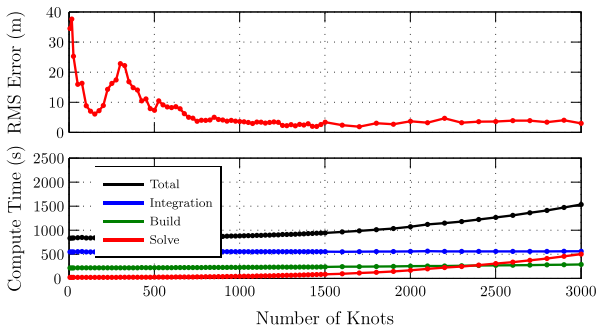

Fig. 4. The full batch estimator was run on a 250m subsection of the full dataset. By varying the number of knots from 6 to 3000, we aimed to find an appropriate knot spacing to use over the entire trajectory. For this subsection of the dataset, it was decided that 1000 knots (a spacing of approximately 0.75 seconds) was acceptable to represent the robot trajectory.

spacing is reduced, the parameterization is given enough degrees of freedom to properly represent the velocity profile of the robot; however, there are diminishing returns with respect to reducing error. For this subsection of the dataset, it was decided that 1000 knots (a spacing of approximately 0.75 seconds) was acceptable to represent the robot trajectory.

In order to limit the growth rate in solve time, we applied our window-style filter to the 250m trajectory. In a similar fashion to the first experiment, we then varied the number of active time segments from 1 to 22. The results in Figure 5 indicate that with a window size of only 2 time segments the solution has nearly converged. This result is understandable given that the average feature track length is 2.8 sequential frames. The compute time shown in this experiment is the average amount of time taken to solve a single window

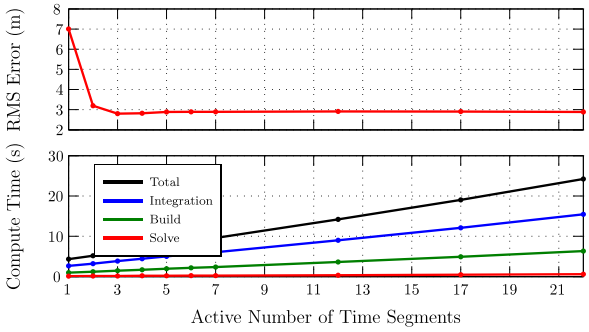

Fig. 5. Using 1000 knots over the 250m training segment, we now vary the number of active time segments from 1 to 22. With only 2 active time segments, we see that the solution has nearly converged. Here, the compute time is the average time it took to compute a single window optimization.

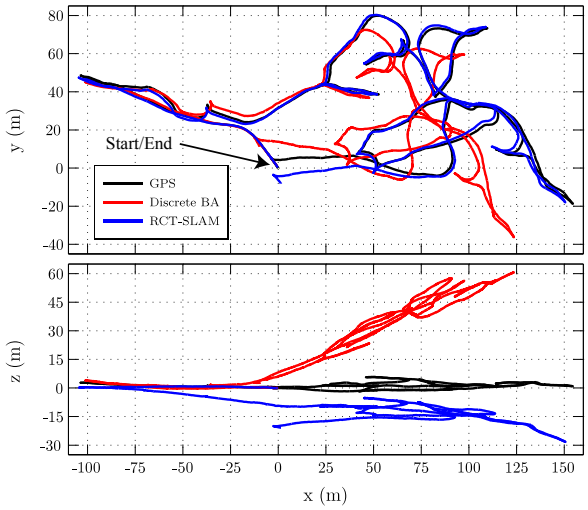

Fig. 6. Top and side views of the robot trajectory over the whole 1154m dataset. The estimator used a knot spacing of 0.75 seconds, and an active window size of 3 time segments. DGPS tracks are in black, the discrete relative bundle adjustment solution is in red and the integral of our velocity profile estimate is in blue.

optimization. It is easy to see that increasing window size has a negative effect on the amount of time spent integrating and building the terms needed in the solution.

The final result for the full 1154m trajectory, seen in Figure 6, is compared to a discrete-time relative bundle adjustment implementation. To keep the discrete implementation's state size tractable, a single relative pose estimate is used for each of the image stacks and the laser scans are assumed to be taken instantaneously (i.e. no motion distortion). The integral of our velocity estimate, discrete relative bundle adjustment solution, and DGPS track are put into a common reference frame by aligning the first 30 meters of each trajectory. The accuracy of our algorithm is clearly superior to the discrete case; error over the entire trajectory can be seen in Figure 7. We note that direction switches in the robot trajectory tended to cancel some existing estimation error, especially in elevation.

As suggested by the results of the first experiment, the estimator used a knot spacing of 0.75 seconds. This correlated to the use of 4,091 knots and only $6 \times 4{,}090 = 24{,}540$

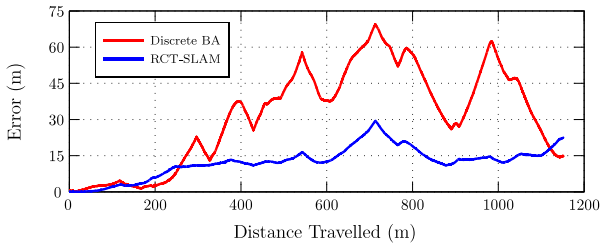

Fig. 7. Translational error growth over the entire 1154m trajectory. The integral of our velocity estimate, discrete relative bundle adjustment solution, and DGPS track are put into a common reference frame by aligning the first 30 meters of each trajectory.

state variables in the parameterization of $\varpi(t)$. In relation to the second experiment, it was decided that the estimator should use a window size of 3 active time segments; this cost approximately 6 seconds per local batch optimization and led to a total run time of 6.9 hours[2]. In contrast, the discrete-time implementation ran in 1.3 hours. The disparity in the computational requirement of the two systems arises due to the addition of the expensive integration phase.

V. Conclusion and Future Work

This paper has developed a novel estimation scheme by deriving the completely relative formulation of SLAM in continuous time. Although still lacking large-scale loop closure, we estimate the velocity profile of the robot and avoid the use of a single privileged coordinate frame, thus setting the stage for future work that will close loops in constant time. A window-style filter was applied to the estimator by using cubic B-splines to parameterize the velocity profile. The approach was validated using lidar intensity imagery captured over a 1.1km trajectory. The next step is to extend this work to include large loop closures. There is also work to be done on the required computational time of our implementation. We hope to gain some performance by migrating our implementation from Matlab to C++. Furthermore, we believe the cost of the integration phase can be drastically reduced with the addition of some heuristics.

Acknowledgment

We would like to extend our deepest thanks to the staff of the Ethier Sand and Gravel in Sudbury, Ontario, Canada for allowing us to conduct our field tests on their grounds. We also wish to thank Dr. James O'Neill from Autonosys for his help in preparing the lidar sensor for our field tests. In addition, we would also like to acknowledge (from the Autonomous Space Robotics Laboratory) Colin McManus for being instrumental in gathering the data used in this paper, Andrew Lambert for his help in preparing the GPS payload, Paul Furgale and Chi Hay Tong for their work on the GPU SURF algorithm and Hang Dong for his work on the Autonosys calibration routine. Lastly, we also wish to thank DRDC Suffield, MDA Space Missions, the NSERC, the Canada Foundation for Innovation and the Canadian Space Agency for providing us with the financial and in-kind support necessary to conduct this research.

[2]All experiments were implemented in Matlab and timed with a 3.2GHz processor and 12GB of 1333MHz DDR3 RAM

References

[1] M. Maimone, Y. Cheng, and L. Matthies, "Two years of visual odometry on the mars exploration rovers," *Journal of Field Robotics*, vol. 24, no. 3, pp. 169–186, 2007.

[2] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

[3] G. Sibley, C. Mei, I. Reid, and P. Newman, "Adaptive relative bundle adjustment," in *Proceedings of Robotics: Science and Systems*, Seattle, USA, June 2009.

[4] C. McManus, P. T. Furgale, and T. D. Barfoot, "Towards appearance-based methods for lidar sensors," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, 9-13 May 2011, pp. 1930–1935.

[5] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," *Computer Vision–ECCV 2006*, pp. 404–417, 2006.

[6] D. Brown, "A solution to the general problem of multiple station analytical stereo-triangulation. rca data reduction," Tech. Rep., 1958.

[7] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon, "Bundle adjustment - a modern synthesis," *Vision algorithms: theory and practice*, pp. 153–177, 2000.

[8] F. Dellaert and M. Kaess, "Square root sam: Simultaneous localization and mapping via square root information smoothing," *The Intl. Journal of Robotics Research*, vol. 25, no. 12, pp. 1181–1203, 2006.

[9] K. Konolige, J. Bowman, J. Chen, P. Mihelich, M. Calonder, V. Lepetit, and P. Fua, "View-based maps," *The International Journal of Robotics Research*, vol. 29, no. 8, p. 941, 2010.

[10] J. Hedborg, P. Forssén, M. Felsberg, and E. Ringaby, "Rolling shutter bundle adjustment," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 1434–1441.

[11] H. Dong and T. D. Barfoot, "Lighting-invariant visual odometry using lidar intensity imagery and pose interpolation," in *Proc. of the Intl. Conf. on Field and Service Robotics*, Matsushima, Japan, July 2012.

[12] M. Bosse and R. Zlot, "Continuous 3d scan-matching with a spinning 2d laser," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2009, pp. 4312–4319.

[13] R. Zlot and M. Bosse, "Efficient large-scale 3d mobile mapping and surface reconstruction of an underground mine," in *Proc. of the Intl. Conf. on Field and Service Robotics*, Matsushima, Japan, July 2012.

[14] M. Bosse, R. Zlot, and P. Flick, "Zebedee: Design of a spring-mounted 3-d range sensor with application to mobile mapping," *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1104–1119, 2012.

[15] P. T. Furgale, T. D. Barfoot, and G. Sibley, "Continuous-time batch estimation using temporal basis functions," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, St. Paul, USA, 14-18 May 2012, pp. 2088–2095.

[16] C. Bibby and I. Reid, "A hybrid slam representation for dynamic marine environments," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2010, pp. 257–264.

[17] M. Fleps, E. Mair, O. Ruepp, M. Suppa, and D. Burschka, "Optimization based imu camera calibration," in *In Proc. of the IEEE Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2011, pp. 3297–3304.

[18] C. H. Tong and T. D. Barfoot, "Gaussian process gauss-newton for 3d laser-based visual odometry," in *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)*, Karlsruhe, Germany, May 2013.

[19] G. M. T. D'Eleuterio, "Dynamics of an elastic multibody chain: Part c – recursive dynamics," *Dynamics and Stability of Systems*, vol. 7, no. 2, pp. 61–89, 1992.

[20] G. Sibley, L. Matthies, and G. Sukhatme, "Sliding window filter with application to planetary landing," *Journal of Field Robotics*, vol. 27, no. 5, pp. 587–608, 2010.

[21] C. McManus, P. T. Furgale, B. E. Stenning, and T. D. Barfoot, "Visual teach and repeat using appearance-based lidar," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, St. Paul, USA, 14-18 May 2012, pp. 389–396.

[22] H. Dong, S. Anderson, and T. D. Barfoot, "Two-axis scanning lidar geometric calibration using intensity imagery and distortion mapping," in *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)*, Karlsruhe, Germany, May 2013.

[23] M. Fischler and R. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Comm. of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.