

A Sliding-Window Filter for Online Continuous-Time Continuum Robot State Estimation

Spencer Teetaert, *Graduate Student Member, IEEE*, Sven Lilge, *Member, IEEE*, Jessica Burgner-Kahrs, *Senior Member, IEEE*, Timothy D. Barfoot, *Fellow, IEEE*

Abstract—Stochastic state estimation methods for continuum robots (CRs) often struggle to balance accuracy and computational efficiency. While several recent works have explored sliding-window formulations for CRs, these methods are limited to simplified, discrete-time approximations and do not provide stochastic representations. In contrast, current stochastic filter methods must run at the speed of measurements, limiting their full potential. Recent works in continuous-time estimation techniques for CRs show a principled approach to addressing this runtime constraint, but are currently restricted to offline operation. In this work, we present a sliding-window filter (SWF) for continuous-time state estimation of CRs that improves upon the accuracy of a filter approach while enabling continuous-time methods to operate online, all while running at faster-than-real-time speeds. This represents the first stochastic SWF specifically designed for CRs, providing a promising direction for future research in this area.

Index Terms—Probability and Statistical Methods, Flexible Robots, Dynamics, State Estimation

I. INTRODUCTION

Continuum robots (CRs) are flexible, small-scale manipulators capable of bending into highly nonlinear shapes and adhere to complex trajectories in confined spaces. This allows them to operate in environments traditional rigid-link robots typically cannot enter, making them suitable for a number of previously inaccessible applications. Examples include minimally invasive surgery [1], industrial inspection and repair [2], and search-and-rescue in disaster areas [3].

Controlling CRs in such applications requires accurate localization within their environment. Significant progress has been made in the physical modeling of CRs, predicting their resulting shape given actuation inputs and interaction forces [4]. Nevertheless, such open-loop methods still suffer from inaccuracies arising from unmodeled effects, approximated material properties, and unknown disturbances or external forces. Consequently, integrated sensing becomes crucial to compensate for these inaccuracies and probabilistic approaches that fuse noisy measurements with suitable prior models have received increased attention.

The vast majority of existing work on probabilistic state estimation employs filtering methods, such as extended Kalman filters (EKF) or particle filters. These methods are computationally efficient and recursive, exploiting the Markov property, which implies that future states only depend on the current state and not the full history [5]. This makes them

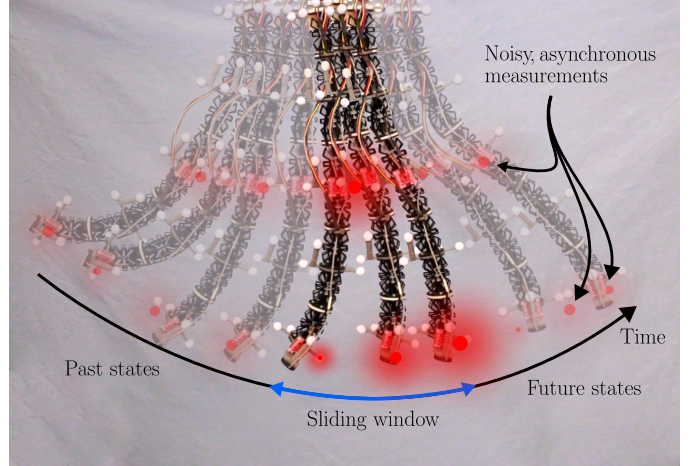


Fig. 1. An example CR state estimation scenario where sensor measurements are produced asynchronously in time. The SWF in this work uses a small window of time to jointly optimize states within the window, improving accuracy over filter-based approaches while enabling online operation, something previous batch methods cannot do.

well suited for online and real-time control scenarios, where estimates must be updated sequentially as new data arrives. However, during the derivation and implementation of most filters, approximations are introduced, e.g., linearizations, such that the approximated posterior may no longer fully capture the Markov structure. This can lead to biased, suboptimal, and consequently inaccurate posterior estimates for nonlinear systems.

Batch estimation methods, such as smoothers, offer a solution to this problem, as they jointly optimize all states using available measurements, retrospectively correcting past approximation errors and producing more accurate posterior estimates. However, such approaches are typically not applicable in online settings, since only past measurements are available at any given time step. Sliding-window filters provide a compromise between filtering and full batch optimization. They perform smoothing over a fixed-length time window, improving estimation accuracy while remaining computationally efficient. To date, such estimators remain largely unexplored in continuum robotics.

This paper directly addresses this gap in the literature and proposes a probabilistic sliding-window filtering approach for CRs. The method is derived from recent batch optimization techniques [6] and enables online estimation of CR states. It is validated on a variety of trajectories using a real-robot prototype, demonstrating improved accuracy over a filtering

This work was supported in part by the National Sciences and Engineering Research Council of Canada (NSERC) and the Ontario Graduate Scholarship.

The authors are with the University of Toronto Robotics Institute, Toronto, ON, Canada. (e-mail: spencer.teetaert@robotics.utoronto.ca)

approach while maintaining real-time operation. The estimation accuracy is shown to be comparable to full batch methods. To the best of the authors' knowledge, this work represents the first probabilistic sliding-window filtering approach specifically designed for CRs. An open-source implementation of the approach is made available to the community at [<link will be added upon publication>](#).

II. RELATED WORK

Most state estimation methods for CRs adopt simplified, non-probabilistic shape representations, such as constant-curvature [7], [8] or polynomial-curvature models [9], [10]. Recent work fuses highly detailed dynamic Cosserat rod models with sensed data to capture fully continuous task-space states [11]–[13], but these methods remain non-stochastic and cannot account for sensor noise or quantify uncertainty.

Probabilistic approaches typically track simplified models over time using filtering methods. Particle filters have been proposed to track constant-curvature representations of catheters over time [14], [15]. Similarly, various Kalman filter implementations have been proposed to fuse noisy measurements with constant-curvature representations of CRs. For example, extended Kalman filters have been applied to tendon-driven CRs in [16], [17]. The state estimation of a multi-backbone CR using an unscented Kalman filter is presented in [18]. Filtering approaches for pneumatically actuated soft robots include both extended Kalman filters [19], [20] and unscented Kalman filters [21].

Beyond filtering, probabilistic smoothing has been applied to state estimation problems for CRs. Early work employs Rauch-Tung-Striebel (RTS) smoothing applied along the robot's arc length to recover quasi-static continuous shape estimates featuring variable curvature [22], [23]. However, these approaches lack temporal smoothing, as they only consider the robot's shape along the spatial domain at a single instant. Analogously, recent work introduces Gaussian process (GP) regression over CR arc length to estimate quasi-static states [24], [25]. Lately, these approaches have been extended to include both temporal and spatial smoothing within a batch optimization framework [6], [26].

As discussed earlier, temporal filtering and batch smoothing represent two extremes in estimation approaches. Filtering methods are computationally efficient and can be run online, but they typically offer limited accuracy. In contrast, smoothing methods provide superior estimation accuracy, yet they are not suitable for online settings and generally scale poorly. A sliding-window approach offers a practical trade-off, aiming to combine the advantages of both methods by balancing computational efficiency and estimation accuracy. Sliding-window filtering methods have been studied in other robotics domains, such as mobile robot simultaneous localization and mapping (SLAM) [27] or planetary surface estimation for autonomous landing [28], but remain underexplored in continuum robotics.

First advances toward sliding-window estimation for CRs are presented in [29] and [30], although both approaches exhibit certain limitations. In [29], a simplified constant-curvature model is used for the sliding-window implementa-

tion, which significantly restricts the range of shape deformations that can be captured. While this approach may perform well in free-space scenarios, it is likely to fall short under environmental interactions. In contrast, [30] approximates the CR shape using a rigid-link model and limits the evaluation of the sliding-window estimator to simulations. Moreover, neither approach provides a probability distribution of the estimated posterior, limiting the ability to quantify its uncertainty.

In conclusion, a gap exists in the current literature, which this paper addresses by introducing the first probabilistic sliding-window estimator for CRs, enabling smooth estimation of their variable-curvature shape over both space and time.

III. METHODOLOGY

A. Estimation Framework

We construct a factor-graph estimator for estimating the state $\mathbf{x}$ of a continuum robot. This state includes the pose $\mathbf{T}(s, t) \in SE(3)$, velocity $\boldsymbol{\omega}(s, t) \in \mathbb{R}^6$, and strain $\boldsymbol{\epsilon}(s, t) \in \mathbb{R}^6$ of the robot along its entire arc length $0 \leq s \leq L$ continuously in time for a given period $0 \leq t \leq T$. We make use of the prior and measurement factors introduced in previous work on batch smoothing [6] to model the robot and measurement relationships of the system. Structuring the estimation problem beginning from these factors will allow us to adopt the continuous-time methodology from previous work into a sliding-window filter (SWF). The factors are used to construct a maximum a posteriori (MAP) estimation problem, which can be solved using nonlinear optimization techniques. Specifically, given measurements $\mathbf{y}$, and control inputs $\mathbf{v}$, we are finding the state $\mathbf{x}^*$,

$$\mathbf{x}^* = \arg \max_{\mathbf{x}} p(\mathbf{x} | \mathbf{y}, \mathbf{v}) \quad (1)$$

$$\equiv \arg \min_{\mathbf{x}} - \sum_i \log(\phi_i(\mathbf{x})). \quad (2)$$

Each factor $\phi_i(\mathbf{x})$ is of the form

$$\phi_i(\mathbf{x}) \propto \exp \left(-\frac{1}{2} \mathbf{e}_i(\mathbf{x})^T \boldsymbol{\Sigma}_i^{-1} \mathbf{e}_i(\mathbf{x}) \right), \quad (3)$$

where $\mathbf{e}_i$ represents an error term and $\boldsymbol{\Sigma}_i^{-1}$ represent an inverse covariance for some factor $\phi_i(\mathbf{x})$. The prior factors used are formulated using an approximate Cosserat rod model, derived from a 'white-noise-on-acceleration' motion prior shown in past works to be effective for both mobile robotics [31] and continuum robotics [6], [24], [25]. In practice, this means our control inputs $\mathbf{v}$ are zero, as the prior factors encapsulate the dynamics of the system. The measurement factors used include tip pose measurements and gyroscope measurements. This optimization problem is solved by linearizing each cost term into a linear least-squares problem and iterating using the Gauss-Newton method until convergence. Specifically,

$$\mathbf{H}^T \mathbf{W}^{-1} \mathbf{H} \delta \mathbf{x} = -\mathbf{H}^T \mathbf{W}^{-1} \mathbf{e}(\mathbf{x}) \quad (4)$$

is iterated until convergence, where $\mathbf{H}$ is the Jacobian of stacked errors $\mathbf{e}(\mathbf{x})$, and $\mathbf{W}$ is the block diagonal matrix

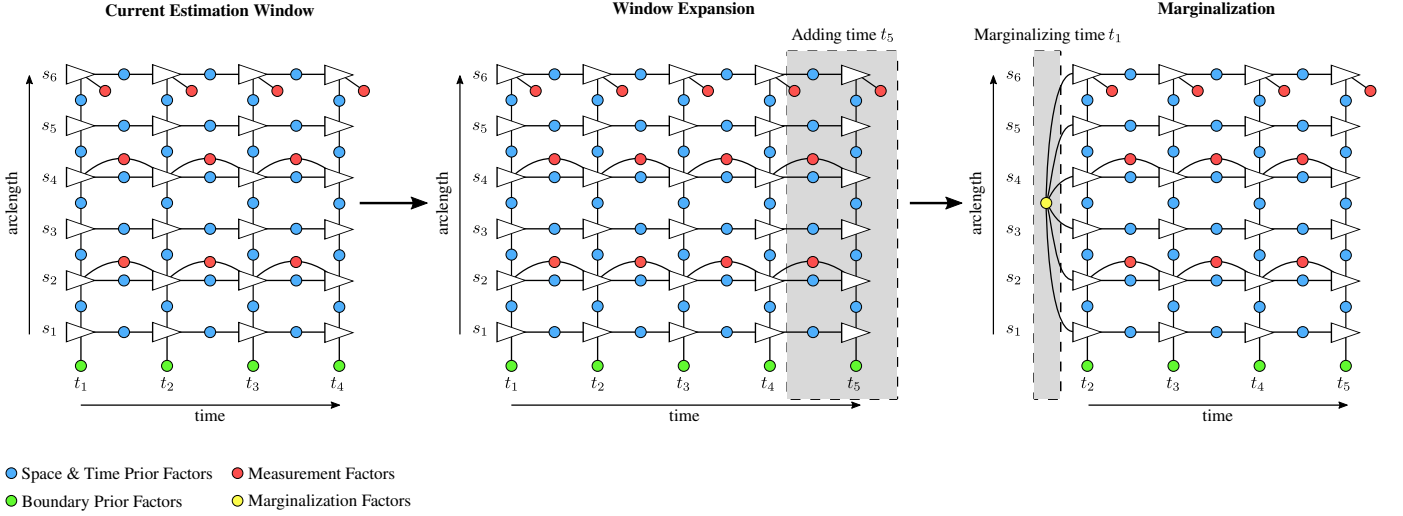


Fig. 2. Factor graph representation of the sliding-window filter: The window is initially expanded to incorporate the next discrete time step. Once the new state is included, the oldest time step is marginalized out, maintaining a fixed window size while propagating information forward.

of stacked covariances Σ_i . Upon convergence, we extract the state covariance using the Laplace approximation as

$$\Sigma = (H^T W^{-1} H)^{-1}. \quad (5)$$

We refer the reader to [6] for additional details and a more in-depth introduction of the estimation framework.

B. Sliding-Window Filter Formulation

Let us define a window with w time steps that include states $\mathbf{x}_{a:b}$, with $b = a + w$. We assume that the window itself has the Markov property and is only dependent on the preceding state and window measurements. The window contains several classes of factors:

- $\phi_p(\mathbf{x}_{a-1})$ prior factors preceding the window,
- $\phi_m(\mathbf{x}_{i-1}^j, \mathbf{x}_i^j)$ motion factors,
- $\phi_s(\mathbf{x}_{i-1}^j, \mathbf{x}_i^j)$ spatial factors,
- $\phi_b(\mathbf{x}_i^0)$ boundary prior factors,
- $\phi_y(\mathbf{x}_{i-1}^j, \mathbf{x}_i^j)$ time-interpolated measurement factors. (6)

For brevity, we will denote all spatial factors for the entire robot state at time i as $\phi_s(\mathbf{x}_i) = \prod_{j=1}^N \phi_s(\mathbf{x}_i^{j-1}, \mathbf{x}_i^j)$. Similarly, $\phi_m(\mathbf{x}_i)$, $\phi_b(\mathbf{x}_{i-1}, (\mathbf{x}_i))$ and $\phi_y(\mathbf{x}_i)$ are defined. The joint factorization over the sliding-window is then

$$p(\mathbf{x}_{a:k} | \mathbf{y}_{1:k}, \mathbf{x}_{a-1}) \propto \phi_p(\mathbf{x}_{a-1}) \phi_b(\mathbf{x}_{a-1}^0) \phi_s(\mathbf{x}_{a-1}) \times \prod_{i=a}^k \phi_b(\mathbf{x}_i^0) \phi_m(\mathbf{x}_{i-1}, \mathbf{x}_i) \phi_s(\mathbf{x}_i) \phi_y(\mathbf{x}_{i-1}, \mathbf{x}_i),$$

where $p(\mathbf{x}_{a:k} | \mathbf{y}_{1:k}, \mathbf{x}_{a-1})$ is the posterior distribution over the window states given all measurements in the window and the preceding state $\mathbf{x}_{a-1}$. Since $\mathbf{x}_{a-1}$ lies outside the window,

we marginalize it to form an effective prior factor on the first in-window state:

$$\psi_a(\mathbf{x}_a) := \int \phi_m(\mathbf{x}_{a-1}, \mathbf{x}_a) \phi_p(\mathbf{x}_{a-1}) \phi_b(\mathbf{x}_{a-1}^0) \times \phi_s(\mathbf{x}_{a-1}) \phi_y(\mathbf{x}_{a-1}, \mathbf{x}_a) d\mathbf{x}_{a-1}. \quad (7)$$

This marginalized term $\psi_a(\mathbf{x}_a)$ acts as a single prior factor that encapsulates all information from before the active window, allowing the optimization to depend only on variables within $\mathbf{x}_{a:k}$. Substituting this factor back into the joint expression yields

$$p(\mathbf{x}_{a:k} | \mathbf{y}_{1:k}, \mathbf{x}_{a-1}) \propto \psi_a(\mathbf{x}_a) \left(\prod_{i=a}^k \phi_b(\mathbf{x}_i^0) \phi_s(\mathbf{x}_i) \right) \times \left(\prod_{i=a+1}^k \phi_m(\mathbf{x}_{i-1}, \mathbf{x}_i) \phi_y(\mathbf{x}_{i-1}, \mathbf{x}_i) \right). \quad (8)$$

This sliding-window factor graph formulation is visually depicted in Fig. 2. Note that in this form we can see three categories of factors emerge: the marginalized prior factor $\psi_a(\mathbf{x}_a)$, the factors that are only a function of states that vary down the arc length of the robot, and the motion factors connecting consecutive time steps within the window. The separation of all of these factors in Eq. (8) highlights the conditional independence of the formulation also visible in Fig. 2. That is, states at a later time step are conditionally independent of states at an earlier time step given the state at the current time step. This is the Markov property once again visible in the formulation. By iterating over a window of time steps and marginalizing out old states, we can mitigate the error introduced by the Markov assumption that is embedded in the structure of the factor graph itself.

C. Filter Implementation

We now face the challenge of implementing a practical solution for the aforementioned factor graph optimization

problem. During estimation, we are constantly expanding our estimation window and marginalizing out old states as we go. How we extract states and choose a window size also requires attention. The details of these four items are described below.

1) *Window Expansion*: At each new time step, we initialize N new states in our factor graph, corresponding to points along the robot at the new time. They are connected to each other via ‘space binary factors’ and to the previous time step via ‘time binary factors’ as described in previous work [6]. Measurement factors are also added to the new states as appropriate. A principled way to initialize the mean of these states is to use the mean produced by taking a step with the prior model. As this setup has each new state connected to two separate factors, it is not necessarily possible to initialize the new states this way. To resolve this, we find two different initializations for the new states: one that is consistent with the prior factor in space, and one that is consistent with the prior factor in time. We then average these two initializations to find a practical compromise for the initial mean.

2) *Marginalization*: To keep the size of the window bounded, and to maintain information from before the window, we marginalize out old states. Simultaneously, these states are removed from the estimation problem, locking them in place. We start with a joint Gaussian distribution over the variable we wish to marginalize, $\mathbf{x}_m$, and those we wish to keep, $\mathbf{x}_r$,

$$p\left(\begin{bmatrix} \mathbf{x}_r \\ \mathbf{x}_m \end{bmatrix}\right) \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_r \\ \boldsymbol{\mu}_m \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_{rr} & \boldsymbol{\Sigma}_{rm} \\ \boldsymbol{\Sigma}_{mr} & \boldsymbol{\Sigma}_{mm} \end{bmatrix}\right). \quad (9)$$

The marginal distribution over $\mathbf{x}_r$ is given as

$$p(\mathbf{x}_r) \sim \mathcal{N}(\boldsymbol{\mu}_r, \boldsymbol{\Sigma}_{rr}). \quad (10)$$

In practice, we will perform this marginalization in information form, where the information matrix $\mathbf{H} = \boldsymbol{\Sigma}^{-1}$ and information vector $\mathbf{h} = \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}$. The marginalization is then performed via the Schur complement,

$$\mathbf{H}_r = \mathbf{H}_{rr} - \mathbf{H}_{rm}\mathbf{H}_{mm}^{-1}\mathbf{H}_{mr}, \quad (11)$$

$$\mathbf{h}_r = \mathbf{h}_r - \mathbf{H}_{rm}\mathbf{H}_{mm}^{-1}\mathbf{h}_m. \quad (12)$$

The joint information matrix is constructed during optimization using the Laplace approximation, and the marginalization information is iterated with the remainder of the problem. As the window slides, the information associated with only locked states is accumulated and carried forward to future time steps without the need for iteration.

3) *State Extraction*: At each time step, we wish to extract a mean and covariance estimate. While we could extract the state mean and covariance from any point in the window, the best estimate will come from the state at the back of the window, as it has been updated with all measurements in the window. However, this introduces latency in the estimate by the period of the window. Conversely, extracting from the front of the window provides a live update at the expense of losing state updates from future measurements, reducing estimation accuracy. In this work, we present results from the former approach, as the latency introduced for all practical window choices remains small (≤ 0.1 s) for the data set used.

One quirk of the continuous-time formulation arises when considering how to extract covariance estimates from the

filter. The continuous-time covariance interpolation expressions in [6] depend on the joint covariance between each neighboring state in time. As we lock states and slide the window, the joint covariance estimates are no longer consistent between window locations as the states are relinearized. To address this, we store an estimate for the joint covariance between each pair of neighboring times as they are locked and use these during interpolation. This results in having two sources of variance estimates for the robot at each time step, which do not necessarily align. In practice, this discrepancy is typically small, but it can result in some discontinuities in the covariance estimate over time.

4) *Window Size*: In this formulation, the choice of window size is critical. It should be clear that setting the window size to include all time steps results in the original batch method. Conversely, setting the window size to only include a single time step results in a filter method, which, given the setup in this paper, results in an iterated filter algorithm. One can strike a balance between accuracy and computational efficiency by selecting an appropriate window size for the given application. In the filter case, only one time step is explicitly represented in the state. As such, the covariance extracted at each time step only reflects uncertainty at that time step (as opposed to a joint between two times), and does not support continuous-time querying of the state covariance after the fact. In contrast, larger window sizes enable after-the-fact continuous-time interpolation. In both cases, measurements within the window/step are still used in continuous measurement factors during estimation.

D. Experimental Setup

1) *Estimator Configurations*: We use the batch method from [6] as a baseline and an estimate for the upper-bound on achievable accuracy through a filter-based approach. The baseline is compared against the proposed SWF with sizes varying from 0 seconds (e.g. filter) to 0.2 seconds. For each estimator, the robot is discretized into $N = 5$ states along its length, and time steps between estimation nodes are set at a frequency of 30Hz. The number of temporal states included in the estimator, K , varies with window size. The noise parameters for the factors in each estimator are kept consistent across all methods for fair comparison. All experiments are run on an Intel i7-13850HX CPU @ 3.80 GHz with 64GB of RAM.

2) *Dataset*: We evaluate each method on the dataset collected in [6] from the 3D printed tendon-driven continuum robot shown in Fig. 1 [32]. The dataset consists of five different trajectories, each 10 seconds long, with varying motion characteristics and contact interactions. The individual trajectories are labelled as “Out-of-Bounds”, “Fast Contact”, “Impulse 1”, “Impulse 2”, and “Slow Free Space”. A full description of each trajectory can be found in [6]. The robot is 46.6cm long and has an outer diameter of 3.6cm. It is resistant to elongation and shear and can be reasonably approximated by a Kirchhoff rod, though we note we do not constrain the estimator to such. The robot is equipped with two 6-degree-of-freedom (DoF) electromagnetic pose sensors

TABLE I
PROPOSED ESTIMATOR AND BASELINE PERFORMANCE SUMMARY

Trial	Tip RMSE		Average NEES	Average Runtime (ms)
	Pos (%)	Rot (rad)		
Out-of-Bounds				
Filter (0s)	2.38	0.045	N/A	4.4
SWF (0.1s)	1.88	0.042	6.93	10.2
Batch (10s)	2.04	0.041	7.60	1359
Fast Contact				
Filter (0s)	1.75	0.048	N/A	4.0
SWF (0.1s)	1.29	0.038	3.56	10.2
Batch (10s)	1.29	0.038	3.80	1379
Impulse 1				
Filter (0s)	1.80	0.053	N/A	3.9
SWF (0.1s)	1.70	0.042	5.96	8.7
Batch (10s)	1.70	0.042	6.54	1377
Impulse 2				
Filter (0s)	2.11	0.075	N/A	4.0
SWF (0.1s)	1.44	0.050	4.99	8.9
Batch (10s)	1.43	0.050	5.27	1380
Slow Free Space				
Filter (0s)	1.16	0.043	N/A	4.0
SWF (0.1s)	1.16	0.042	5.26	9.1
Batch (10s)	1.16	0.042	5.64	1380

For each presented metrics except for NEES, the lowest value is optimal. NEES has an optimal value of 6, the number of DoFs in the pose states being evaluated.

(Aurora v3, Northern Digital Inc., Canada) at its tip and base, and two gyroscope units (ISM330DHCX, STMicroelectronics NV, Netherlands) mounted at the midpoint and tip of the robot. Ground truth pose measurements are collected at five points along the robot using an external motion capture system (Vicon Motion Systems Ltd., UK). All data is collected asynchronously, resulting in varying timestamps for each of the sensor modalities.

E. Metrics

We evaluate each method using the root mean square error (RMSE) between estimated and ground truth tip poses. For position values, we normalize the result with respect to the length of the robot. As the estimated nodes and the ground truth measurements are typically asynchronous, all metrics are evaluated using the continuous-time interpolation method provided by the estimation framework. The average normalized estimation error squared (NEES) is computed to evaluate the consistency of each estimator. As the filter method does not support continuous-time interpolation of covariances, presenting the NEES is not possible. Finally, the average runtime per time step is measured for each method to assess computational efficiency.

IV. RESULTS

To select the optimal sliding-window size for our estimator, we evaluate its performance across the five experimental trajectories using various window sizes ranging from 0s (the filter baseline) to 10s (equivalent to the batch optimization baseline). The results, summarized in Fig. 3, indicate that increasing the window size generally leads to improved tip position and rotation RMSE. However, the improvements

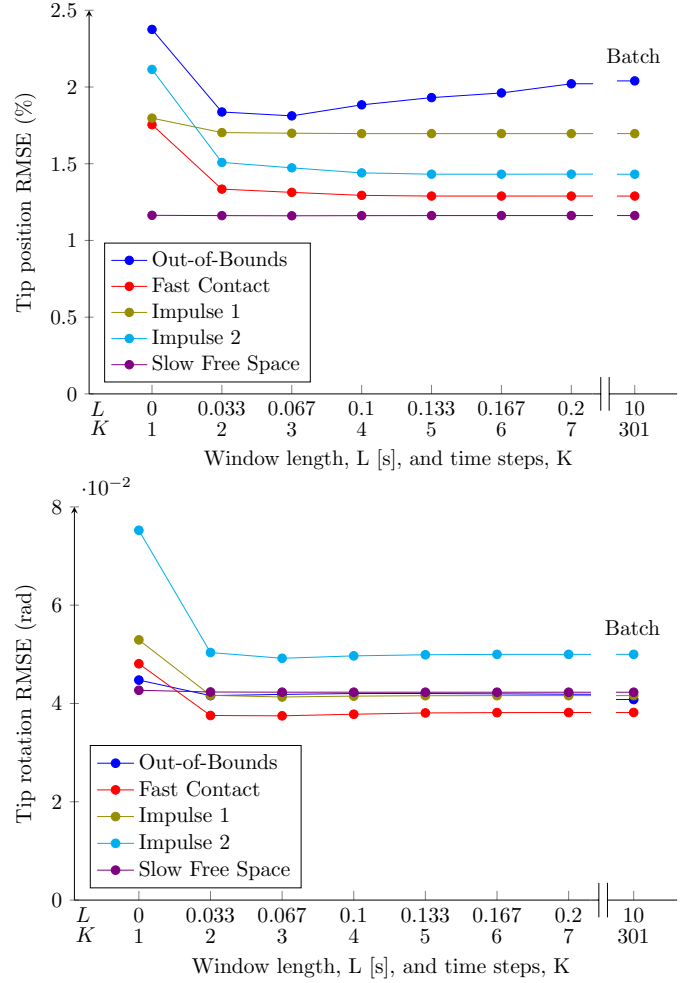


Fig. 3. Tip position and rotation RMSE across each of the five experimental trajectories for different window sizes. A window size of 0s corresponds to the filter baseline, while a window size of 10s corresponds to the batch optimization baseline. In general, larger window sizes lead to lower RMSE values, with diminishing returns as the window size grows past 0.1s. Notably, even small window sizes (e.g., 0.033s) provide significant improvements over the filter baseline, indicating that incorporating a short history of states can substantially enhance estimation accuracy.

exhibit diminishing returns beyond a window size of approximately 0.1s. Notably, even small window sizes (e.g., 0.033s) yield significant enhancements over the filter baseline. During the Out-of-Bounds trajectory, we do observe an increase in position RMSE when increasing the window size beyond a certain point, which is unexpected. Theories for this outlier behavior are discussed in the following section. Based on these findings, we select a window size of 0.1s to present as our primary result in the remainder of this section.

We present qualitative results of the proposed SWF on the Out-of-Bounds trajectory. Fig. 4 provides a side-by-side visual comparison of the SWF's estimated robot shape against the actual robot while Fig. 5 shows the tip pose estimates and uncertainties compared to the ground truth measurements. See the supplementary video for a full demonstration of the SWF on the evaluated dataset.

The proposed estimator and each baseline are evaluated on all five experimental trajectories. Their evaluation metrics are

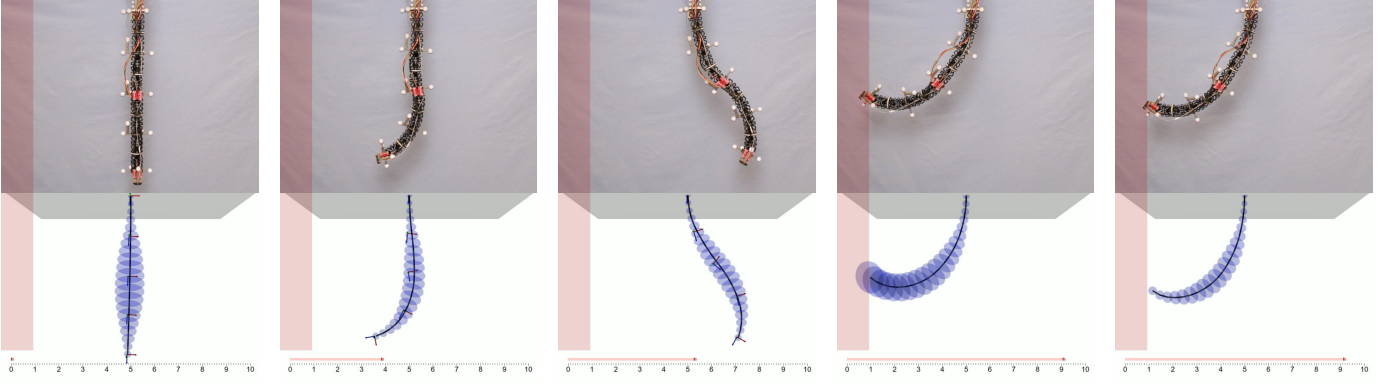


Fig. 4. Side-by-side comparison of the proposed SWF with a window size of 0.1s (bottom) and the Out-of-Bounds experimental trajectory (top). The region where pose data is lost is highlighted in red on the left of the frames. The current time of each frame is provided on a visualized timeline. The SWF is able to accurately track the tip pose of the robot in real-time, even during fast motions and in the presence of occasional pose measurement dropouts. Sudden increases in uncertainty are observable when the pose measurements are lost, but the filter quickly recovers once measurements resume (see rightmost frames).

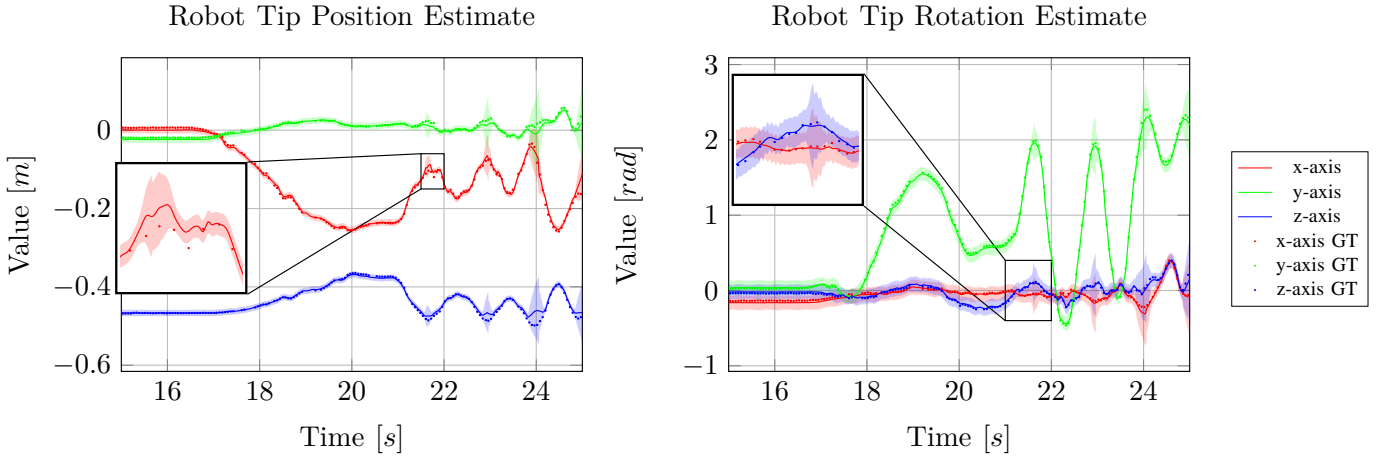


Fig. 5. Tip estimation results of a SWF with a window size of 0.1s on the Out-of-Bounds trajectory. The mean, 3σ uncertainty bounds, and ground truth values collected from the motion capture system are shown for both position and orientation. The SWF demonstrates comparable performance to the batch optimization method, though has less smooth estimates. Notably, when the pose measurements drop out at 21.5s, the estimate contains a sudden increase in uncertainty and displays the expected discontinuity in uncertainty growth when the next measurement is received.

presented in Table I.

Lastly, for several window sizes we evaluate the runtime performance of the proposed SWF over all estimation windows. We include the runtimes for each window estimated across all five trajectories. Window sizes with average runtimes that are real-time capable (i.e., below 33.3 ms) are shown in Fig. 6. The distributions exhibit multiple modes, stemming from the varying number of iterations required for convergence in each window (mostly 3, sometimes 4 or 2). These results will vary given the nature of the trajectory, as more complex motions may require more iterations to converge. We find that for our system, window sizes up to 0.3s consistently achieve real-time performance throughout operation.

V. DISCUSSION

The results clearly demonstrate the usefulness of this filter, achieving comparable accuracy to the batch solver while operating online. Even with small window sizes, significant improvements can be seen in the accuracy of the estimator. The runtimes achieved in both Fig. 6 and Table I show that even while keeping a complex, continuous shape estimate,

continuum robot state estimation in real time is possible with compute to spare. This additional compute will be useful for downstream applications, such as controllers, planners, and any other processing an end user may need to perform.

Our method maintains the continuous-time properties of the original batch framework at the expense of smoothness in the estimate. Measurements can still be fed into the framework asynchronously, removing the need for any sort of pre-integration or running at extremely high frequencies. If we are comfortable losing a continuous-time covariance representation, a byproduct of this proposed SWF is a version of the filter algorithm applied to CR state estimation that runs in under 5ms (over 200Hz). This filter still provides access to a continuous representation of the state mean, which is not dependent on the joint covariance between times. We hope this formulation will enable state estimation on more dynamic systems than what has traditionally been studied in the field.

In Fig. 3 we see for four of the five trajectories, the behavior of the window filter is as expected, with larger window sizes leading to better accuracy. However, in the Out-of-Bounds trajectory we see an unexpected increase in position RMSE

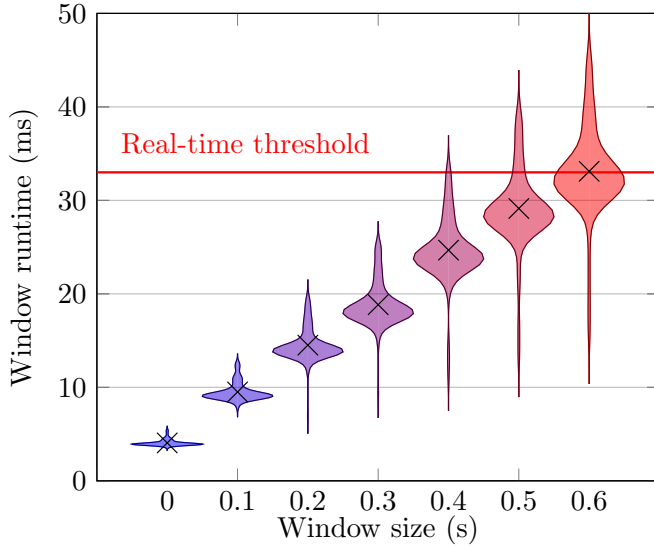


Fig. 6. Window runtime distributions for different window sizes. Larger window sizes lead to higher runtimes due to their larger state size. The long tails on each distribution corresponds to different numbers of iterations required for convergence. Each window's state size is selected such that the estimated nodes are generated at a frequency of 30 Hz, as such, each window should ideally have a runtime below 33.3 ms to maintain real-time performance.

when increasing the window size beyond a certain point. Our best hypothesis for this behavior is that when the pose sensor leaves the workspace, the measurements near the boundaries severely degrade. This could result in the batch solution overfitting to poor measurements while a shorter window filter could better handle the pose sensor dropout by relying on the gyroscopes fully. With a window method we expect the results would vary given different sensor configurations, noise profiles, and data rates. The data in this work was collected at frequencies between 30–50Hz, with low noise. We expect that with lower frequency and higher noise sensors, larger window sizes would be required to achieve similar performance.

The proposed method does have some drawbacks, most notably the latency introduced by extracting states from the back of the window. In practice, practitioners may wish to extract states further forward in the window based on their use case. We find that when extracting the state from the front of the window, no noticeable improvement is seen compared to the filter method. This result is surprising given the Markov property justification provided earlier. Our best hypothesis for why this occurs comes from the fact that we are not running estimation once per measurement time (as in a traditional IEKF) but rather adding multiple measurement time steps to each new window. We expect this behavior to return to expectations in cases with noisier and lower frequency data. Despite this unexpected behavior, the proposed method still provides a principled way to perform continuous-time state estimation of a continuum robot online, while maintaining uncertain estimates.

VI. CONCLUSION

The SWF proposed in this work strikes a balance between estimation accuracy, computational efficiency, and online operation capacity at the expense of introducing a small latency and reduced smoothing compared to the batch approach. As in [6], we maintain a factor-graph formulation of the estimation problem, a choice we hope will lead to further adoption and a more versatile framework for others down the line. An open-source implementation for this estimator is provided for the community at [<link will be added upon publication>](#).

REFERENCES

- [1] J. Burgner-Kahrs, D. C. Rucker, and H. Choset, "Continuum robots for medical applications: A survey," *IEEE Transactions on Robotics*, vol. 31, no. 6, pp. 1261–1280, 2015.
- [2] X. Dong, D. Axinte, D. Palmer, S. Cobos, M. Raffles, A. Rabani, and J. Kell, "Development of a slender continuum robotic system for on-wing inspection/repair of gas turbine engines," *Robotics and Computer-Integrated Manufacturing*, vol. 44, pp. 218–229, 2017.
- [3] E. W. Hawkes, L. H. Blumenschein, J. D. Greer, and A. M. Okamura, "A soft robot that navigates its environment through growth," *Science Robotics*, vol. 2, no. 8, p. eaan3028, 2017.
- [4] C. Armanini, F. Boyer, A. T. Mathew, C. Duriez, and F. Renda, "Soft robots modeling: A structured overview," *IEEE Transactions on Robotics*, vol. 39, no. 3, pp. 1728–1748, 2023.
- [5] T. D. Barfoot, *State estimation for robotics*. Cambridge University Press, 2024.
- [6] S. Teetaert, S. Lilge, J. Burgner-Kahrs, and T. D. Barfoot, "A stochastic framework for continuous-time state estimation of continuum robots," 2025, *arXiv:2510.01381*.
- [7] R. J. Roesthuis, M. Kemp, J. J. van den Dobbelsteen, and S. Misra, "Three-dimensional needle shape reconstruction using an array of fiber bragg grating sensors," *IEEE/ASME Transactions on Mechatronics*, vol. 19, no. 4, pp. 1115–1126, 2014.
- [8] F. Stella, C. Della Santina, and J. Hughes, "Soft robot shape estimation with imus leveraging pcc kinematics for drift filtering," *IEEE Robotics and Automation Letters*, vol. 9, no. 2, pp. 1945–1952, 2023.
- [9] B. Kim, J. Ha, F. C. Park, and P. E. Dupont, "Optimizing curvature sensor placement for fast, accurate shape sensing of continuum robots," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 5374–5379.
- [10] G. Pei and J. Hughes, "Polynomial curvature model for imu-based proprioception in soft robotic manipulators under external forces," in *IEEE 8th International Conference on Soft Robotics*, 2025, pp. 1–7.
- [11] T. Zheng, Q. Han, and H. Lin, "Full state estimation of continuum robots from tip velocities: A cosserat-theoretic boundary observer," *IEEE Transactions on Automatic Control*, vol. 70, no. 5, pp. 2859–2872, 2025.
- [12] T. Zheng, C. McFarland, M. Coad, and H. Lin, "Estimating infinite-dimensional continuum robot states from the tip," in *IEEE 7th International Conference on Soft Robotics*. IEEE, 2024, pp. 572–578.
- [13] T. Zheng and J. Burgner-Kahrs, "Estimating dynamic soft continuum robot states from boundaries," 2025, *arXiv:2505.04491*.
- [14] A. Brij Koolwal, F. Barbagli, C. Carlson, and D. Liang, "An ultrasound-based localization algorithm for catheter ablation guidance in the left atrium," *International Journal of Robotics Research*, vol. 29, no. 6, 2010.
- [15] J. A. Borgstadt, M. R. Zinn, and N. J. Ferrier, "Multi-modal localization algorithm for catheter interventions," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 5350–5357.
- [16] A. Ataka, P. Qi, A. Shiva, A. Shafti, H. Wurdemann, H. Liu, and K. Althoefer, "Real-time pose estimation and obstacle avoidance for multi-segment continuum manipulator in dynamic environments," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 2827–2832.
- [17] R. Peng, Y. Wang, and P. Lu, "A tendon-driven continuum manipulator with robust shape estimation by multiple imus," *IEEE Robotics and Automation Letters*, vol. 9, no. 4, pp. 3084–3091, 2024.
- [18] Y. Chen, S. Zhang, L. Zeng, X. Zhu, and K. Xu, "Model-based estimation of the gravity-loaded shape and scene depth for a slim 3-actuator continuum robot with monocular visual feedback," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 4416–4421.

- [19] J. Y. Loo, K. C. Kong, C. P. Tan, and S. G. Nurzaman, "Non-linear system identification and state estimation in a pneumatic based soft continuum robot," in *2019 IEEE Conference on Control Technology and Applications (CCTA)*, 2019, pp. 39–46.
- [20] D. Kim, M. Park, and Y.-L. Park, "Probabilistic modeling and bayesian filtering for improved state estimation for soft robots," *IEEE Transactions on Robotics*, vol. 37, no. 5, pp. 1728–1741, 2021.
- [21] M. Mehl, M. Bartholdt, S. F. Ehlers, T. Seel, and M. Schappler, "Adaptive state estimation with constant-curvature dynamics using force-torque sensors with application to a soft pneumatic actuator," in *IEEE International Conference on Robotics and Automation*, 2024, pp. 14 939–14 945.
- [22] A. W. Mahoney, T. L. Bruns, P. J. Swaney, and R. J. Webster, "On the inseparable nature of sensor selection, sensor placement, and state estimation for continuum robots or "where to put your sensors and how to use them"," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 4472–4478.
- [23] P. L. Anderson, A. W. Mahoney, and R. J. Webster, "Continuum Reconfigurable Parallel Robots for Surgery: Shape Sensing and State Estimation With Uncertainty," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1617–1624, 2017.
- [24] S. Lilge, T. D. Barfoot, and J. Burgner-Kahrs, "Continuum robot state estimation using gaussian process regression on $SE(3)$," *The International Journal of Robotics Research*, vol. 41, no. 13-14, pp. 1099–1120, 2022.
- [25] S. Lilge, T. Barfoot, and J. Burgner-Kahrs, "State estimation for continuum multirobot systems on $se(3)$," *IEEE Transactions on Robotics*, vol. 41, pp. 905–925, 2025.
- [26] S. Teetaert, S. Lilge, J. Burgner-Kahrs, and T. D. Barfoot, "Space-time continuum: Continuous shape and time state estimation for flexible robots," 2024, *arXiv:2409.12302*.
- [27] G. P. Huang, A. I. Mourikis, and S. I. Roumeliotis, "An observability-constrained sliding window filter for slam," in *2011 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2011, pp. 65–72.
- [28] 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.
- [29] H. Abdelaziz, A. Nada, H. Ishii, and H. El-Hussieny, "State estimation of continuum robots: A nonlinear constrained moving horizon approach," 2023, *arXiv:2308.03931*.
- [30] G. Bastos, Jr., "A moving horizon estimation for a class of soft continuum manipulators," *Journal of Computational and Nonlinear Dynamics*, vol. 19, no. 81004, 2024.
- [31] S. Anderson and T. D. Barfoot, "Full steam ahead: Exactly sparse gaussian process regression for batch continuous-time trajectory estimation on $se(3)$," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 157–164.
- [32] P. T. Dewi, P. Rao, and J. Burgner-Kahrs, "A lightweight modular segment design for tendon-driven continuum robots with pre-programmable stiffness," in *IEEE 7th International Conference on Soft Robotics*, 2024, pp. 531–536.