AstroLoc: An Efficient and Robust Localizer for a Free-flying Robot

Ryan Soussan[1,2], Varsha Kumar[3], Brian Coltin[1,4], Trey Smith[1]

***Abstract*— We present AstroLoc, an efficient and robust monocular visual-inertial graph-based localization system used by the Astrobee free-flying robots onboard the International Space Station (ISS). We provide a novel localization system that limits the traditionally higher computation times for graph-based localization systems and enables the resource constrained Astrobee robots to benefit from their increased accuracy. We also introduce methods for handling cheirality issues for visual odometry and localization factors that further increase localization robustness. We evaluate the performance of AstroLoc on a dataset of ISS activities and show that it greatly improves pose, velocity, and IMU bias estimation accuracy while efficiently running in a limited computation environment. AstroLoc has improved the localization accuracy for the Astrobee robots on the ISS and has led to more successful and longer duration activities. While the AstroLoc system is tuned for the Astrobee robots, it can be configured for any resource constrained platform. The source code for AstroLoc is released to the public.**

I. INTRODUCTION

The Astrobee free-flying robots [1] are a set of three robots currently operating on the International Space Station (ISS) and used for a variety of experiments in microgravity [2] [3] and the ISAAC caretaker project [4]. They use an electric fan for propulsion and an inertial measurement unit (IMU) and front facing camera for navigation as shown in Fig. 1.

The previous localizer for Astrobee [5] was based on the Multi-State Constraint Kalman Filter (MSCKF) [6] and had successfully completed many tasks on the ISS since its introduction but often suffered from large localization drift or lost robot events that forced the robot to be reset or re-docked during an activity. Due to occlusions from objects such as cargo bags, lighting changes, and other environmental discontinuities on the ISS, map landmarks are also not always available, so Astrobee relies heavily on visual-inertial odometry (VIO) for navigation. Additionally, free-flying in microgravity prevents the use of a gravity vector and requires highly accurate localization and velocity estimation to control and station keep the robot.

To improve upon the performance of the MSCKF-based localizer, we use graph-based optimization which performs relinearization during each iteration compared to the single linearization point used by EKF-based systems. Existing graph-based approaches rely on sliding window bundle adjustment for VIO [7] [8] [9] or a parallel mapping process [10] [11] for SLAM that are too expensive to run on Astrobee, whose compute platform [1] is roughly 10 times slower than an Intel i9-9980HK 2.4 GHz CPU and is running a suite of other processes, leaving only a single core for the graph-based localizer.

The authors are with [1]NASA Ames Research Center, Moffett Field, CA, 94035, USA, [2]Aerodyne Industries, [3]Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh, PA, 15213, USA, and [4]KBR, Inc. {ryan.soussan, brian.coltin, trey.smith}@nasa.gov, varshak@andrew.cmu.edu

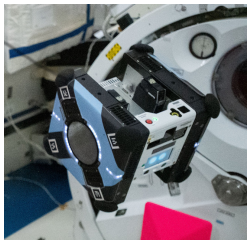
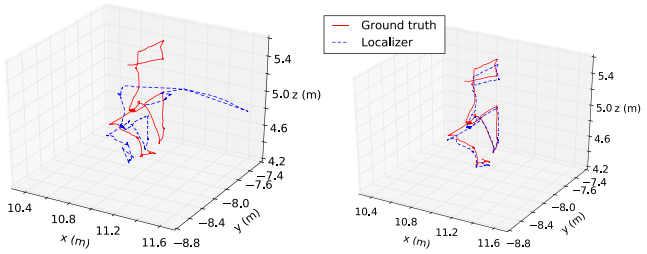


Fig. 1. Astrobee robot flying in the ISS.



(a) Previous EKF-based Localizer (b) AstroLoc

Fig. 2. AstroLoc reduces drift and more accurately tracks Astrobee's position compared to the previous EKF-based localizer.

We therefore present AstroLoc: a monocular visual-inertial graph-based localization system for the Astrobee robots. Our contributions include:

- A novel localization system for a free-flying robot that intelligently incorporates visual odometry measurements, performs sliding window marginalization, and limits factors and optimization times to perform graph-based optimization in a resource constrained environment.
- Methods for handling visual odometry and localization factor cheirality issues to improve robustness.
- Approaches to separately evaluate the accuracy of tracked velocity and IMU bias estimates.
- A modular software framework for state estimation.

We evaluate the performance of AstroLoc on a dataset of 12 ISS activities and show that it exhibits significant decreases in pose, velocity, and IMU bias RMSE in localization and VIO tests while maintaining a fast runtime suitable for the Astrobee robots. AstroLoc has been deployed to the Astrobee robots on the ISS and has enabled them to perform longer duration activities with improved accuracy compared to the

previous localizer. The AstroLoc system is tuned for the Astrobee robots but many of the parameters in §IV can be adjusted for other computationally limited platforms. We release the code[1] to the public.

II. Related Work

A. EKF-based Approaches

The MSCKF [6] uses marginalized visual odometry errors along with map-based features in an augmented-state EKF for localization, while ROVIO [12] performs monocular VIO using an EKF with a direct photometric error.

B. Graph-based VIO Approaches

OKVIS [9] performs monocular or stereo VIO using BRISK features with a graph-based optimizer. SVO [13] and DSO [7] are both direct graph-based visual odometry methods with support for monocular and stereo systems and VI-DSO [14] adds IMU support. Each of these methods require costly sliding window bundle adjustment and direct methods often rely on camera frame rates that are much faster than the Astrobee rate of 15 Hz. Smart factors [15] marginalize out 3D feature points to more efficiently perform graph-based visual odometry but correlate state parameters and require an expensive SVD process as described further in §IV.

C. Graph-based SLAM Approaches

PTAM [16] introduced the concept of parallel tracking and mapping that many modern SLAM methods rely on. ORB-SLAM [10] uses ORB features along with the DBoW2 bag-of-words library [17] to perform feature-based loop-closures. Basalt [8] and Kimera [18] perform VIO and SLAM, where Basalt uses bidirectional patch tracking with LSSD costs and ORB feature loop-closures and Kimera adds semantics to produce a 3D mesh construction of an environment with optional semantic segmentation, but each of these require a stereo camera. VINS-Mono [11] uses a combination of direct and indirect features for monocular visual odometry and adds loop-closures for 4 DoF map optimization, but relies on a gravity vector to reduce the dimensionality of its map optimization and requires bundle adjustment for 3D feature point estimation.

D. Free-flyer Localizers

The SPHERES [19] robots require ultrasonic beacons for localization while the JAXA Int-Ball [20] needs stereoscopic markers placed throughout the ISS. The previous localizer for Astrobee [5] used map-based image features but relied on the MSCKF for sensor fusion.

E. Other Aerospace Localizers

VISINAV [21] performs localization for planetary entry, descent, and landing (EDL) using natural terrain and an MSCKF. Relative pose estimation is calculated for autonomous orbital rendezvous and proximity operations (RPO) in [22] but requires an EKF and either fiducials or an existing 3D CAD model for tracked objects.

To take advantage of graph-based optimization while also using our prebuilt image feature map, we design a localization system for Astrobee's limited compute platform that we present in the following sections.

III. System Overview

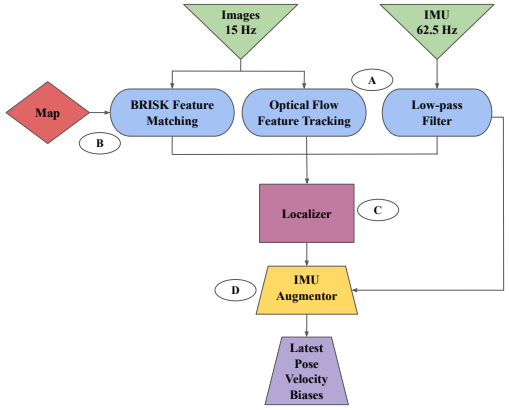


Fig. 3. Localization pipeline showing map-dependent and independent data used by the localizer along with the IMU Augmentor which extrapolates the navigation state produced by the localizer with the latest IMU measurements.

The AstroLoc localization pipeline processes map-dependent and map-independent data to generate a navigation state S_i consisting of the pose, velocity, and IMU biases of the robot. Fig. 3 shows the pipeline in more detail.

A. Map-Independent Sensor Measurements

The pipeline processes images and IMU measurements at 15 Hz and 62.5 Hz respectively as shown in Part A of Fig. 3. Vibrational noise from the impeller fan is removed by passing the IMU measurements through low-pass and notch filters. Optical flow feature tracks are generated using the pyramidal Lucas-Kanade algorithm [23], where forward and backward passes are conducted on a sequence of images to limit outliers.

B. Map-Dependent Sensor Measurements

The pipeline uses mapped 3D BRISK features stored in a DBoW2 library [17] and sends measurements and their associated map features from each image to the localizer. The mapping procedure is described in more detail in [5].

C. Graph-Based Localizer

The graph-based localizer in Part C processes the measurement inputs and outputs a navigation state S_i. Its design is discussed further in §IV.

D. IMU Augmentor

Since the localizer is used with a controller that expects frequent navigation state updates at a rate similar to that of the IMU, S_i is subsequently passed to the IMU Augmentor in Part D. This is also discussed in more detail in §IV.

[1] https://github.com/nasa/astrobee

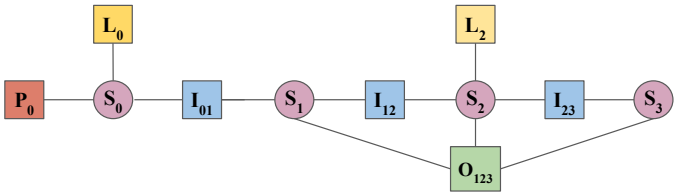


Fig. 4. Factor graph representation for the AstroLoc localizer. Navigation states are represented by nodes S_i. IMU preintegration factors I_{ij} are used to connect neighboring navigation states. Factors for map-based landmarks are represented by L_i and for visual odometry smart factors by O_i. Priors for the oldest navigation state are shown using P_i.

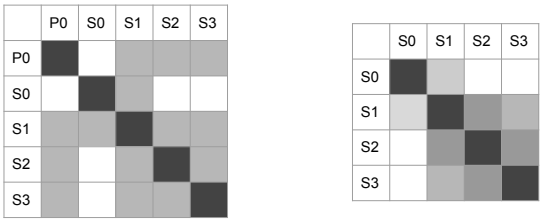
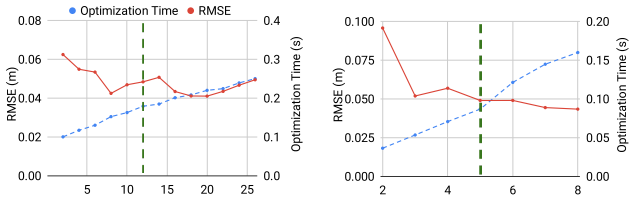


Fig. 5. Example Hessian before and after smart factor marginalization for a graph containing feature track measurements at states S_1, S_2, and S_3 and the feature point P_0.



(a) Maximum number of smart factors (b) Max measurements per smart factor

Fig. 6. Trade-offs in accuracy versus computation time for the number of smart factors included in the graph and the number of measurements added per smart factor. AstroLoc uses a maximum of 12 smart factors and five measurements per factor.

IV. Efficient Localization

AstroLoc performs graph-based nonlinear optimization with the factor graph representation shown in Fig. 4. It adds IMU measurements as preintegration factors [24] to avoid reintegrating IMU data during each optimization iteration and uses smart factors [15] to incorporate optical-flow feature tracks as visual odometry constraints and remove the costly estimation of 3D features that most graph-based visual odometry and SLAM approaches require. AstroLoc inserts map-based feature measurements using projection factors with the error function in (1) where the map feature is not included in the optimization problem to avoid increasing graph solve times. Optimization is performed using the GTSAM library [25].

A. Limiting Cost of Visual Odometry Smart Factors

Visual odometry smart factors are the most expensive component of the AstroLoc graph optimization process. The factor error is formed by projecting a triangulated feature point f into a set of images containing feature point measurements per (1).

$$\mathbf{p}_i = \mathbf{u}_i - \pi({}^{\mathbf{C}}_{\mathbf{B}}\mathbf{T}\,{}^{\mathbf{B}}_{\mathbf{W}}\mathbf{T}\mathbf{f}) \tag{1}$$

The projection function π for a camera model represents the mapping $\pi\colon \mathbb{R}^3 \mapsto \mathbb{R}^2$ where $\mathbb{R}^2$ is image space, transform ${}^{\mathbf{C}}_{\mathbf{B}}\mathbf{T}$ is the extrinsic calibration of the camera, transform ${}^{\mathbf{B}}_{\mathbf{W}}\mathbf{T}$ is the inverse of the robot body pose in the the world frame, and $\mathbf{u}_i$ is the feature measurement in image i. The smart factor forms the error vector $\mathbf{e}$ by stacking the errors per (1) for each feature measurement in the measurement set. The error function is the same as a bundle-adjustment approach but unlike bundle-adjustment procedures the 3D feature points are not included in the optimization problem. These are removed with a marginalization procedure mirroring that of [6] that requires the singular value decomposition of the feature Jacobian matrix.

While the smart factor marginalization procedure reduces the dimensionality of the factor as shown in Fig. 5, many off-diagonal components of the Hessian may still be present that increase the optimization time for the graph, indicated by the non-empty blocks of the Hessian matrix in the same figure. Additionally, each smart factor in the graph performs the singular value decomposition marginalization during each optimization iteration, greatly reducing the speed of the localization procedure. AstroLoc therefore limits both the overall number of smart factors included in the graph and the measurements included in each smart factor to reduce their computational cost. The impact of limiting the total number of smart factors on the graph solve time is shown in Fig. 6a where the data is generated using a parameter sweep with the evaluation procedure described in §VI.

AstroLoc also performs measurement selection to reduce the number of off-diagonal terms in the graph's Hessian matrix. Each feature track includes measurements from the same subset of timestamps which are evenly spaced over the duration of the graph. This limits the correlation between navigation state nodes in the graph and increases the sparsity of the optimization problem while also increasing the portion of the graph constrained by visual odometry factors. Fig. 6b shows the tradeoff in optimization time and accuracy versus the number of measurements included per smart factor. In practice, AstroLoc uses five measurements per smart factor and includes a maximum of 12 smart factors in its factor graph.

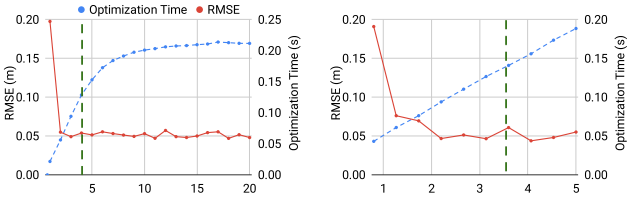
B. Sliding Window Marginalization

While AstroLoc supports adding marginalization factors when sliding the graph window using the linearization of each removed factor [9], it uses a more efficient method instead that inserts priors for the oldest state parameter nodes remaining in the graph. Marginalization with linearized factors correlates state parameters that shared a factor with marginalized parameters and introduces many off-diagonal elements in the resulting Hessian matrix,increasing optimization times. Additionally, when using smart factors adding marginalization factors becomes difficult as only a subset each smart factor's measurements are removed when the window is slid rather than the entire smart factor. AstroLoc

therefore adds priors to the oldest remaining state parameters in the graph using their current estimated values and the covariances generated by the most recent optimization iteration's inverted Hessian matrix. This limits the density of the optimization problem and yields faster solve times.

C. Limiting Optimization Runtime

AstroLoc's convergence criteria include both a threshold for the minimum change in relative error while optimizing and a limit on the maximum number of optimization iterations performed. As Fig. 7a shows, error reduction



(a) Maximum optimization iterations (b) Sliding window duration

Fig. 7. Trade-offs in accuracy versus computation time for the maximum optimization iterations and sliding window duration of the graph. AstroLoc uses a maximum of four optimization iterations and a sliding window duration of 3.5 seconds.

quickly saturates while the iteration threshold and resulting computation time continue to increase. To prevent outlier optimization times that use many iterations, AstroLoc limits the maximum optimization iterations to four.

In addition to limiting the total number of smart factors, AstroLoc limits the total factor count for map-based localization factors to around 40. The localization factors are much more efficient to compute as they are image projection factors that only depend on one state parameter node. AstroLoc limits the sliding window graph duration to 3.5 seconds which provides some extra buffer time for outlier map-based feature measurement delays. The trade-off in accuracy versus runtime for graph duration is depicted in Fig. 7b. These constraints help limit the maximum runtime for AstroLoc and enable it to provide localization updates at a frequency of around 5 Hz which are then fed into the IMU Augmentor described in the following section to provide even higher frequency updates to the controller.

D. High Frequency Updates using the IMU Augmentor

The IMU Augmentor delivers high frequency 62.5Hz localization updates that the controller requires by extrapolating the latest localization estimates provided by the localizer with the latest IMU data. It performs extrapolation by maintaining a queue of IMU measurements and integrating them on top of the most recent navigation state using its IMU bias estimates. Without the IMU Augmentor, feedback to the controller occurs at 5Hz instead of 62.5Hz and Astrobee over and undershoots its desired poses and velocities while path planning.

V. Robust Localization

AstroLoc introduces several techniques along with using a robust loss function to prevent lost robot and high drift events during localization.

A. Robust Cost using Huber Loss

To avoid drift due to outlier measurements AstroLoc feeds each factor through a Huber loss function per (2).

$$h(n) = \begin{cases} n & \text{if } n < k \\ n\sqrt{k/n} & \text{otherwise} \end{cases} \tag{2}$$

Here $n = \mathbf{e}^T\mathbf{e}$ is the error norm for a factor. AstroLoc uses the standard threshold $k = 1.345$. The Huber loss assumes that large factor errors are often correlated with outlier data.

B. Preventing Cheirality Errors for Map-based Projection Factors using Pose Factor Fallback

AstroLoc inserts map association pairs p_i consisting of an image measurement $\mathbf{u}_i$ and its associated 3D map features $\mathbf{f}_i$ using the error function in (1). If every feature point in the set of pairs P for an image suffers a cheirality error, the localizer instead adds a pose prior factor so the measurements can still be included in the optimization problem. The pose ${}^{\mathbf{W}}_{\mathbf{C_e}}\mathbf{T}$, where $\mathbf{C_e}$ is the estimated camera frame, is found using the perspective-three-point algorithm [26] with a RANSAC selection procedure described in [5]. Four randomly selected pairs from P are used to estimate ${}^{\mathbf{W}}_{\mathbf{C_e}}\mathbf{T}$ and the rest of the associations are checked for consistency. This is repeated for several iterations and the pose with the most inliers is used for ${}^{\mathbf{W}}_{\mathbf{C_e}}\mathbf{T}$. The error function for the factor is shown in (3), where ${}^{\mathbf{W}}_{\mathbf{B}}\mathbf{T}$ is the pose estimate in the graph at the image timestamp and ${}^{\mathbf{C}}_{\mathbf{B}}\mathbf{T}$ is the transform from the body to camera frame.

$$\mathbf{e} = \log({}^{\mathbf{W}}_{\mathbf{B}}\mathbf{T}^{-1}\,{}^{\mathbf{W}}_{\mathbf{C_e}}\mathbf{T}\,{}^{\mathbf{C}}_{\mathbf{B}}\mathbf{T}) \tag{3}$$

This fallback procedure helps avoid localization drift as more map-based measurements can be included in the graph even with poor current position and orientation estimates that often lead to cheirality errors.

C. Preventing Cheirality Issues in Visual Odometry Smart Factors using Measurement Pruning

In addition to providing a fallback for cheirality issues for map-based localization factors, AstroLoc also introduces a measurement pruning procedure to help eliminate cheirality issues for visual odometry smart factors. Smart factors are more susceptible to cheirality issues since if one measurement in the smart factor suffers a cheirality issue, the entire factor is ignored. Recognizing that these are often the result of a faulty pose estimate or sequence of faulty pose estimates that have been recently added to the localizer, AstroLoc tests each smart factor before an optimization cycle and fixes smart factors with cheirality errors using the following approach. It first removes the most recent feature measurements and checks the resulting smart factor for errors. The measurements are checked individually, and if the cheirality error is removed by discarding a measurement the factor is added without the faulty measurement. Otherwise, measurement sequences are checked, starting again with the most recent measurements and additionally removing previous measurements until the error is removed. If the cheirality error still remains the factor is discarded.

D. Sanity Checking

The resulting navigation state from the localizer is checked using a covariance based sanity checker and pose history sanity checker. If covariances for S_i exceed a set threshold, the localizer is reset. Additionally, if a set length of poses from the localizer drift too far from pose estimates derived using map-based associations as described in §V-B, the localizer is also reset.

VI. Experiments

We evaluate AstroLoc on a dataset of 12 ISS activities and compare results with the previous Astrobee localizer and a base GTSAM localizer described in §VI-A. Ground truth is created using our mapping pipeline described in [5]. AstroLoc runs on Astrobee's Inforce 6501 Micro SoM featuring a Qualcom Snapdragon SoC mobile processor and utilizes a single processing core which as mentioned in §I runs $\sim$10 times slower than an Intel i9-9980HK 2.4 GHz CPU.

A. Comparisons

The reliance on a gravity vector and lack of a prebuilt feature-based map prevent the comparison with many of the methods from §II, in addition to their increased computational cost that prohibits their use on Astrobee's limited compute platform. We therefore compare AstroLoc to both the previous Astrobee localizer [5] and a base GTSAM implementation. The base GTSAM localizer provides a graph-based localization reference that still incorporates the IMU preintegration and visual odometry smart factors used by AstroLoc but does not include the efficiency improvements discussed in §IV or the robust techniques of §V.

B. Evaluation Metrics

We evaluate the localizer in two modes: (1) VIO mode, using only VIO information, with map-based measurements suppressed. (2) Localization mode, using both VIO and map-based measurements. VIO mode is useful for simulating the localizer behavior when prior map features are not recognized, as often occurs due to environment changes. As an additional performance metric we measure the number of lost events that occur on each dataset defined as when the robot has drifted by more than 1.5 meters. Lost events can require crew intervention during activities and avoiding them is therefore a high priority for AstroLoc.

1) Velocity and IMU Bias Integration: To more precisely track performance of velocity and IMU bias estimates we integrate graph estimates for each of these and compare the results with ground truth data. Integrating velocities utilizes the time difference between successive velocity estimates to add position updates to the starting position of the robot per (4).

$$\mathbf{p} = \mathbf{p}_0 + \sum_{i=1}^{N} \Delta T \mathbf{v}_i \tag{4}$$

Here ΔT is the time difference between successive velocity estimates $\mathbf{v}_i$ and $\mathbf{v}_{i-1}$ and $\mathbf{p}_0$ is the initial position of the robot. Accurate velocity estimation is critical for the controller to station keep and execute precise maneuvers in microgravity.

To evaluate the IMU biases estimated by the localizer, we integrate each IMU measurement in a recording using the latest IMU bias estimate occurring before or at the measurement's timestamp and generate a set of relative integrated IMU pose increments ${}^{i-1}_{i}\mathbf{T}$. These are then added to the initial robot pose per (5), much like to the velocity integration procedure.

$${}^{w}_{i}\mathbf{T} = {}^{w}_{i-1}\mathbf{T}\,{}^{i-1}_{i}\mathbf{T} \tag{5}$$

Here ${}^{w}_{0}\mathbf{T}$ is the initial pose of the robot in the world frame. While some degree of noise will still be present in the integration, accurately tracked biases will produce less overall drift. Tracking IMU biases is especially important as we use these to extrapolate localization estimates as described in §III.

C. VIO Results

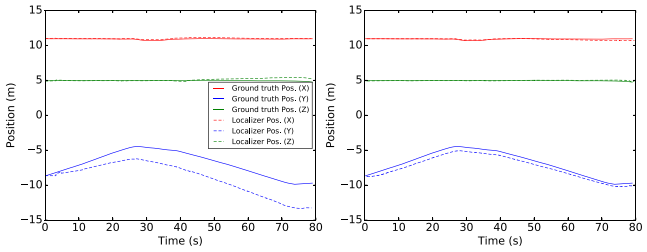
AstroLoc exhibits large improvements in VIO mode compared to the previous localizer and base GTSAM localizer as displayed in Table I. RMSEs are reported for all activities but the two lost events suffered by the previous localizer and base GTSAM implementation, while the last row shows the position RMSE including these outliers.

TABLE I. VIO Results

RMSE	Prev Loc	BaseGT	AstroLoc
Pos. (m)	0.6212	0.5328	0.2777
Orientation (rad)	0.0730	0.0783	0.0696
Rel Pos. (m)	0.2788	0.2080	0.1440
Rel Orientation (rad)	0.0662	0.0632	0.0659
Integrated Vel. Pos. (m)	0.7953	0.9614	0.2931
Rel Integrated Vel. Pos. (m)	0.2624	0.2179	0.1470
IMU Bias Pos. (m)	83.9615	4.0982	2.1535
Rel IMU Bias Pos. (m)	12.3075	0.7425	0.4033
Lost Events	2	2	0
Pos. w/ Outliers (m)	2.4179	0.7508	0.3721

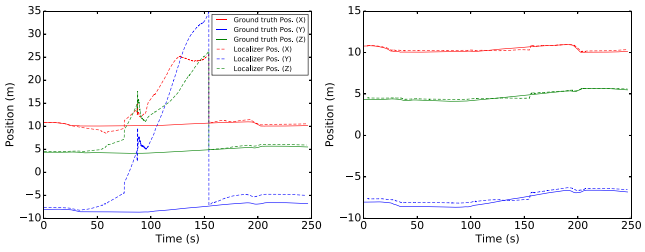
1) Robustness: Both the base GTSAM implementation and AstroLoc outperform the previous localizer in most VIO categories, demonstrating the improved accuracy of graph-based approaches. AstroLoc though further improves upon the base GTSAM method in position, velocity, and IMU bias estimation. The base GTSAM method naively includes every visual odometry measurement and without the AstroLoc methods for smart factor spacing and handling cheirality errors it accrues more drift and gets lost twice akin to the previous localizer. It additionally suffers larger errors in IMU bias estimation and velocity estimation as shown in Table I, which can result in erroneous velocity and position corrections performed by the live controller during an activity. Fig. 8 shows an example of the reduced drift for AstroLoc in VIO mode compared to the previous localizer.

2) Efficiency: AstroLoc greatly improves upon the base GTSAM implementation's runtimes, running roughly six times faster as shown in Table II. The combination of intelligent visual odometry measurement selection, factor and



(a) Previous Localizer (b) AstroLoc

Fig. 8. AstroLoc reduces drift in VIO mode by utilizing an efficient graph-based localization system while the previous localizer steadily veers off course.



(a) Previous Localizer (b) AstroLoc

Fig. 9. Previous EKF-based localizer accruing position errors and ultimately getting lost during an activity whereas AstroLoc accurately tracks the robot position.

TABLE II. VIO RUNTIMES

Localizer	Avg. Runtime (s)	Avg. Opt. Time (s)
BaseGT	1.0588	0.8525
AstroLoc	0.1785	0.1259
Rel Decrease	83%	85%

iteration limiting, and efficient sliding window marginalization prevent excessive computation that would lead to large delays for Astrobee. The previous localizer is not included as it runs at a fixed 62.5Hz.

D. Localization (VIO+Map) Results

Table III displays AstroLoc's significant improvements in position, orientation, velocity, and IMU bias estimation compared to the previous localizer and base GTSAM implementation.

TABLE III. LOCALIZATION (VIO+MAP) RESULTS

RMSE	Prev Loc	BaseGT	AstroLoc
Pos. (m)	0.0948	0.2417	0.0491
Orientation (rad)	0.0439	0.0505	0.0275
Integrated Vel. Pos. (m)	0.2681	0.3228	0.1417
Rel Integrated Vel. Pos. (m)	0.1200	0.1534	0.0656
IMU Bias Pos. (m)	83.5386	2.4744	2.4173
Rel IMU Bias Pos. (m)	12.2643	0.4682	0.4784
Lost Events	2	0	0
Pos. w/ Outliers (m)	1.0277	0.3105	0.2636

1) Robustness: The AstroLoc and the base GTSAM localizer suffer no lost events compared to the two the previous localizer experiences, however AstroLoc yields a further 80% and 15% decrease in position RMSE with and without outlier events compared to the base GTSAM localizer. The combination of AstroLoc's fallback approach for map-based measurement factor cheirality errors and improved VIO performance when map-based features are sparse or unavailable help prevent it from getting lost and accruing large drift on the dataset. Results from one activity where AstroLoc accurately tracks localization position while the previous localizer gets lost are displayed in Fig. 9.

2) Efficiency: Table IV displays the improved runtime of AstroLoc compared to the base GTSAM localizer. Similar to the VIO results, AstroLoc demonstrates significant runtime improvements. Runtimes are only slightly increased compared to VIO as localization map-based factors do not yield as significant of a performance hit as the smart factors.

TABLE IV. LOCALIZATION (VIO+MAP) RUNTIMES

Localizer	Avg. Runtime (s)	Avg. Opt. Time (s)
BaseGT	1.1335	0.8919
AstroLoc	0.2050	0.1389
Rel Decrease	82%	84%

VII. CONCLUSION

We have presented AstroLoc, a localization system for the Astrobee robots that efficiently runs graph-based monocular visual-inertial localization on their limited compute platform and introduces methods to recover from cheirality issues for visual odometry and localization factors to improve localization accuracy and robustness. While the AstroLoc system is carefully tuned for the Astrobee robots, the parameters in §IV including number of smart factors, measurements per smart factor, sliding window duration, and maximum optimization iterations can be adjusted for other resource limited platforms. We have validated the system on data from 12 ISS activities and shown that it helps prevent Astrobee from getting lost while improving pose, velocity, and IMU bias estimation accuracy.

Since AstroLoc was deployed on the ISS, Astrobee's improved navigation when map-based updates are unavailable has dramatically improved operational reliability. Astrobee has executed multiple ISS activities with over two hours of flying time. Reduced reliance on the prior map has allowed the Astrobee team to invest much less effort in running ISS activities specifically to collect new map imagery (∼40% fewer activities and 30% fewer images per activity).

While this work primarily addresses limiting localization and VIO drift for Astrobee and has enabled longer duration and more robust experiments on the ISS, we additionally investigate using semantics for mapping and localization in [27] and in future work we wish to explore using more lifelong mapping approaches to further reduce the impact of environmental changes on localization performance.

VIII. ACKNOWLEDGEMENTS

We would like to thank Marina Moreira, Kathryn Hamilton, Oleg Alexandrov, and the rest of the Astrobee and Astrobee Facilities team for supporting this work.

REFERENCES

[1] T. Smith, J. Barlow, M. Bualat, T. Fong, C. Provencher, H. Sanchez, E. Smith, *et al.*, "Astrobee: A new platform for free-flying robotics on the International Space Station," in *Int. Symp. on Artificial Intelligence, Robotics and Automation in Space*, 2016.

[2] K. Albee, C. Oestreich, C. Specht, A. Teran Espinoza, J. Todd, I. Hokaj, R. Lampariello, and R. Linares, "A robust observation, planning, and control pipeline for autonomous rendezvous with tumbling targets," *Frontiers in Robotics and AI*, p. 234, 2021.

[3] C. Oestreich, A. T. Espinoza, J. Todd, K. Albee, and R. Linares, "On-orbit inspection of an unknown, tumbling target using NASA's Astrobee robotic free-flyers," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 2039–2047.

[4] T. Smith, M. Bualat, A. Akanni, O. Alexandrov, L. Barron, J. Benton, B. Coltin, T. Fong, J. Garcia, K. Hamilton, L. Hill, M. Moreira, R. Morris, N. Ortega, J. Pea, J. Rogers, M. Savchenko, K. Sharif, and R. Soussan, "ISAAC: An integrated system for autonomous and adaptive caretaking," in *ISS R&D Conference*, 2021.

[5] B. Coltin, J. Fusco, Z. Moratto, O. Alexandrov, and R. Nakamura, "Localization from visual landmarks on a free-flying robot," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 4377–4382.

[6] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint Kalman filter for vision-aided inertial navigation," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*. IEEE, 2007, pp. 3565–3572.

[7] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 3, pp. 611–625, 2017.

[8] V. Usenko, N. Demmel, D. Schubert, J. Stueckler, and D. Cremers, "Visual-inertial mapping with non-linear factor recovery," *IEEE Robotics and Automation Letters (RA-L) & Int. Conference on Intelligent Robotics and Automation (ICRA)*, vol. 5, no. 2, pp. 422–429, 2020.

[9] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual–inertial odometry using nonlinear optimization," *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 314–334, 2015.

[10] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: a versatile and accurate monocular SLAM system," *IEEE transactions on robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[11] T. Qin, P. Li, and S. Shen, "VINS-Mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.

[12] M. Bloesch, S. Omari, M. Hutter, and R. Siegwart, "Robust visual inertial odometry using a direct EKF-based approach," in *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2015, pp. 298–304.

[13] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast semi-direct monocular visual odometry," in *2014 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2014, pp. 15–22.

[14] L. Von Stumberg, V. Usenko, and D. Cremers, "Direct sparse visual-inertial odometry using dynamic marginalization," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 2510–2517.

[15] L. Carlone, Z. Kira, C. Beall, V. Indelman, and F. Dellaert, "Eliminating conditionally independent sets in factor graphs: A unifying perspective based on smart factors," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 4290–4297.

[16] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," in *2007 6th IEEE and ACM international symposium on mixed and augmented reality*. IEEE, 2007, pp. 225–234.

[17] D. Gálvez-López and J. D. Tardos, "Bags of binary words for fast place recognition in image sequences," *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1188–1197, 2012.

[18] A. Rosinol, M. Abate, Y. Chang, and L. Carlone, "Kimera: an open-source library for real-time metric-semantic localization and mapping," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 1689–1696.

[19] S. Nolet, "The spheres navigation system: from early development to on-orbit testing," in *AIAA Guidance, Navigation and Control Conference and Exhibit*, 2007, p. 6354.

[20] S. Mitani, M. Goto, R. Konomura, Y. Shoji, K. Hagiwara, S. Shigeto, and N. Tanishima, "Int-ball: Crew-supportive autonomous mobile camera robot on iss/jem," in *2019 IEEE Aerospace Conference*. IEEE, 2019, pp. 1–15.

[21] A. I. Mourikis, N. Trawny, S. I. Roumeliotis, A. E. Johnson, A. Ansar, and L. Matthies, "Vision-aided inertial navigation for spacecraft entry, descent, and landing," *IEEE Transactions on Robotics*, vol. 25, no. 2, pp. 264–280, 2009.

[22] J. Kelsey, J. Byrne, M. Cosgrove, S. Seereeram, and R. Mehra, "Vision-based relative pose estimation for autonomous rendezvous and docking," in *2006 IEEE Aerospace Conference*, 2006, pp. 20 pp.–.

[23] J.-Y. Bouguet *et al.*, "Pyramidal implementation of the affine Lucas Kanade feature tracker description of the algorithm," *Intel corporation*, vol. 5, no. 1-10, p. 4, 2001.

[24] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "IMU preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation," in *Proc. RSS*. Georgia Institute of Technology, 2015.

[25] F. Dellaert, "Factor graphs and GTSAM: A hands-on introduction," Georgia Institute of Technology, Tech. Rep., 2012.

[26] X.-S. Gao, X.-R. Hou, J. Tang, and H.-F. Cheng, "Complete solution classification for the perspective-three-point problem," *IEEE transactions on pattern analysis and machine intelligence*, vol. 25, no. 8, pp. 930–943, 2003.

[27] I. Miller, R. Soussan, B. Coltin, T. Smith, and V. Kumar, "Robust semantic mapping and localization on a free-flying robot in microgravity," in *2022 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2022.