Abstract—We propose a system solution to achieve data-efficient, decentralized state estimation for a team of flying robots using thermal images and inertial measurements. Each robot can fly independently, and exchange data when possible to refine its state estimate. Our system front-end applies an online photometric calibration to refine the thermal images so as to enhance feature tracking and place recognition. Our system back-end uses a covariance-intersection fusion strategy to neglect the cross-correlation between agents so as to lower memory usage and computational cost. The communication pipeline uses Vector of Locally Aggregated Descriptors (VLAD) to construct a request-response policy that requires low bandwidth usage. We test our collaborative method on both synthetic and real-world data. Our results show that the proposed method improves by up to 46% trajectory estimation with respect to an individual-agent approach, while reducing up to 89% the communication exchange. Datasets and code are released to the public, extending the already-public JPL xVIO library.

Index Terms—Collaborative localization, drones, space robotics, thermal-inertial odometry.

SUPPLEMENTARY MATERIAL
For code and datasets please visit: https://rpg.ifi.uzh.ch/xctio

I. INTRODUCTION

WITH the successful deployment of the Ingenuity Mars Helicopter and the Perseverance rover on Mars, 1 multi-agent collaborative tasks are now being carried out on another world for the first time. Ingenuity performed 19 flights during its first year of operations, many of which were done in direct support of Perseverance’s exploration of Jezero crater. This continues a new era of exploration for NASA, started by the MarCO mission in 2018 [1], which saw a pair of co-dependent cubesats investigate Mars for the first time. Next, NASA’s CADRE mission plans to send a group of small rovers to cooperatively explore the Lunar surface in 2024 as part of its Commercial Lunar Payload Services program [2].

This trend towards smaller, cooperative agents is reflected by the more dangerous terrain NASA wishes to explore in future missions to the Moon, Mars and beyond. These areas include Recurring Slope Lineae on Mars and lava tubes on the Moon and Mars. Multiple small agents can be released from larger rovers or deployed from landers to explore areas that are too risky for a single agent to explore. It’s also possible for multiple agents to explore large regions quickly, and deviate from the primary mission task without risk of jeopardizing the primary mission timeline.

The ability for agents to localize themselves in an unknown environment with no prior infrastructure (e.g., GPS) is a key capability for autonomous operations of multiple agents (see Fig. 1). Visual-Inertial Odometry (VIO) [3], [4] is one navigation alternative that the Ingenuity Mars Helicopter utilizes to estimate its pose during flights [5]. Additionally, pre-cursors to the CADRE mission, such as the Autonomous-PUFFER research task [6], have demonstrated the ability of multiple agents to exchange information while performing localization to improve their overall estimate by working together while exploring their work area.

At the same time, there has also been growing interest and work to incorporate new sensing inputs to autonomous navigation algorithms that do not rely on day light conditions and can be used to operate these algorithms in extreme locations such as lava tubes that are deprived of external lighting. One of the sensor technologies being investigated for this purpose are thermal cameras, which measure thermal radiation from all objects in their field of view [7]. These types of cameras allow autonomous agents to operate in all types of lighting conditions and require no external lighting source.

This paper aims to incorporate these two separate threads (multi-agent localization and thermal sensing) into a single system based around multiple aerial vehicles to investigate the benefits that such a system could bring to future missions. We present a collaborative VIO system architecture that allows agents to operate in a decentralized way while still making use...
of information that is received from other agents. In addition, we have incorporated recent advancements in thermal image calibration into our system to improve VIO performance on thermal imagery. Our contributions are:

- The development of a scalable, from 1 to N agents, Collaborative Thermal-Inertial Odometry using a photometric calibration algorithm.
- The incorporation of Binary VLAD descriptors in a thermal-inertial odometry solution.
- A scalable and light communication approach over a decentralized flying robot system.
- Code and datasets are released as part of the already public JPL xVIO library\(^2\) [8].

II. RELATED WORK

State estimation for multi-agent systems is a well-known and established problem which has recently gained attention. The solutions can be divided into centralized and decentralized approaches. The former relies on a central unit acting as a server, fusing incoming information from all the agents [9], [10]. The latter directly fuses data onboard requiring efficient fusion processes and low data exchange [11]. While a centralized version is easier to implement, it relies on a central entity to always be reachable, never fail, and scale with the number of agents, limiting the range of exploration. This work focuses on the decentralized approach to be robust and scalable [12].

Decentralized Simultaneous Localization and Mapping (SLAM) faces the problem of exploiting communication. Indirect measurements through the registration of observations are the preferable approach in many scenarios [13]. Place recognition has emerged as a compact and data-efficient solution [14]. The work in [15] uses Vector of Locally Aggregated Descriptors (VLADs) generated by the NetVLAD [16] to reduce the communication bandwidth. Instead of sending information to all the other agents, the agent sends data only to the robot assigned with a specific descriptor. Although this strategy lowers the used bandwidth, it limits the system’s scalability due to the a priori assignment of a centroid of the clustered feature space. We overcome this problem and generate binary VLADs from the feature descriptors extracted from the frames.

The evaluation of graph-based optimization vs. filter-based approaches for structure from motion has been rigorously analyzed in [17]. The conclusion is that optimization approaches outperform filtering in terms of accuracy per unit of computing time. Visual SLAM improves accuracy by increasing the number of features while having minor effects when increasing the number of keyframes. However, the analysis was done with a standard Extended Kalman filter (EKF) formulation, which models the features in the vector state, and the computational time grows quadratically with the number of observations. Other approaches such as the Multi-State Constrained Kalman Filter (MSCKF) [18] express the constraints without including the 3D feature position in the filter state vector, resulting in computational complexity linear in the number of features.

\(^2\)[Online]. Available: https://github.com/jpl-xvio

The work in [19] showed that MSCKF competes with graph-based optimization approaches in terms of accuracy and computational cost. We focus our method on a filter-based approach using the MSCKF formulation of the JPL xVIO library, targeting onboard rotorcraft flight applications. We extend the xVIO back-end to fuse multi-agent data, developing a collaborative formulation of the MSCKF. We use the Covariance Intersection (CI) algorithm [20] in order to avoid the filter being overconfident in the resulting updates. The CI overestimates the resulting covariance after fusing the measurements coming from several agents, ensuring filter consistency. Our back-end is similar to the research in [21] and [22].

Thermal-inertial solutions have recently received considerable attention for odometry systems in challenging scenarios. The technology was not widely used due to its cost and poor resolution. However, recent advances in thermal imaging have made thermal cameras more affordable, with a smaller footprint, higher quality and modest power consumption enabling the portability to robotics applications [7]. The work by Dellaune et al. [23], and Khattak et al. [24], have shown the use of raw thermal data to navigate in dark scenarios where conventional visual cameras fail. Despite the outstanding results obtained by these works, thermal-infrared cameras suffer from photometric inconsistencies due to the thermography generation process. As a result, a standard computer vision program that relies on the photometric consistency assumption might fail to deliver reliable perception data in a sequence of frames. In this direction, the work by Das et al. [25], develops a photometric model of Thermal Infrared (TIR) cameras to attenuate such disturbances, obtaining photometric consistency over time and space. However, place recognition with thermal imaging remains an open challenge, and its application in collaborative systems has not been investigated.

Tab. I summarizes the related work on collaborative visual odometry systems. Our proposal is a fully decentralized thermal inertial odometry (TIO) that uses place recognition with thermal imaging and benefits from a data-efficient collaborative strategy. To the best of our knowledge, this path has not been explored.

### III. METHODOLOGY

We have a set \( U \) of \( U \) Unmanned Aerial Vehicles (UAVs), equipped with a monocular thermal-infrared camera and IMU. Each drone can fly independently by running the JPL xVIO [8]. At start-up, each drone is initialized with the same inertial reference frame. The system extracts Features from accelerated segment test (FAST) corners [26] and uses the pyramidal implementation of the Lucas-Kanade tracker [27] to build tracks out of the extracted features. We associate an Oriented FAST and Rotated BRIEF (ORB) descriptor [28] to
each track to perform track matching between the current UAV tracks and the ones received from the other agents. In xVIO there are three kinds of tracks:

- **Opportunistic**: concatenation of the Lucas-Kanade tracker results over a sequence of frames.
- **SLAM**: tracks corresponding to the feature points saved in the state vector. For these points we know the inverse depth parameterization.
- **MSCKF**: opportunistic tracks that have a size between two and $M$ frames and fulfill the requirements to perform an MSCKF update, that is the baseline between two features in the track is higher than a certain threshold.

An MSCKF track contains a wide baseline between the tracked points, hence a match between opportunistic tracks of two different UAVs fulfill the requirements to define a single MSCKF track. For this reason, hereafter, we will talk only about MSCKF and SLAM tracks. Figure 2 depicts a diagram of our method implemented in the xVIO library. The front-end provides feature point descriptors and generates a VLAD descriptor contained in the request message. The Track Manager unit receives the matches, stores them and forms tracks. The tracks are employed to perform MSCKF and SLAM updates: when a message is received, the back-end finds correspondences between the current tracks and the received ones. The matches generated from the correspondences are then processed as SLAM-SLAM matches updates, or stored and processed afterward as collaborative MSCKF updates.

Details of the the EKF updates, the state and covariance propagation, the thermal front-end and the communication pipeline are described in the following.

### A. Thermal Front-end

Thermal images show low contrast and bring photometric inconsistency over a sequence of frames. To deal with these problems, we employed the work of Das et al. [25] that attenuates the low-frequency non-uniformness and photometric inconsistency. Figure 4 shows that calibrated images return a higher Harris cornerness response than the uncalibrated ones, and hence make it possible to detect and track stronger corners. FAST corner detector compares the intensity value of a central pixel with a ring of pixels around it. A corner is detected if the intensity of $N$ contiguous pixels in the ring is higher or lower than the central one by a threshold $t$. With calibrated thermal frames, the gradient of the image is high, and hence the value of $t$ can be set to detect the most distinctive corners. We counted the number of tracks and evaluated the feature life for both calibrated and uncalibrated frames. As shown in Figure 3, applying spatial and temporal parameters calibration results in approximately four times the number of FAST corners being successfully detected and tracked with the Lucas-Kanade tracker than on uncalibrated images, with average feature track length on calibrated images also being increased. Namely, it is harder to lose the tracked corners since the image has higher contrast than the original thermal frame. This result implies that since four times more tracks are detected, we can potentially have four times more track correspondences between the robots, hence more collaborative updates for the EKF.

### B. State definition

In the xVIO back-end, the state vector $x$ (1), can be divided into the states concerning the IMU $x_I$ and the visual part $x_V$.

The first term, $x_I \in \mathbb{R}^{15}$, defined in (2) includes the position, velocity and orientation quaternion of the IMU frame $\{i\}$ with respect to the world frame $\{w\}$, the gyroscope and the accelerometer biases, $b_g$ and $b_a$ respectively.

The second term, $x_V \in \mathbb{R}^{7M+3N}$, defined in (3) is split into the sliding window $x_S$, that has the positions $\{p_w^c\}$, and the orientations $\{q_w^c\}$ of the camera frame of the last $M$ image time instances, and the feature states $x_F$ (4), that contains the inverse depth parametrization of the landmarks $f_i$ with $i = 1, \ldots, N$.

\[
x = \begin{bmatrix} x_I^T & x_V^T \end{bmatrix}^T \quad (1)
\]

\[
x_I = \begin{bmatrix} p_{w}^T & v_{w}^T & q_{w}^T & b_g^T & b_a^T \end{bmatrix}^T \quad (2)
\]

\[
x_S = \begin{bmatrix} p_{w}^{c_1}^T & \ldots & p_{w}^{c_M}^T & q_{w}^{c_1}^T & \ldots & q_{w}^{c_M}^T \end{bmatrix}^T \quad (3)
\]

\[
x_F = \begin{bmatrix} f_1^T & \ldots & f_N^T \end{bmatrix}^T \quad (4)
\]

The error state vector $\delta x$ can also be divided as the state vector. The error state is defined for the position vectors as the difference between the true and estimated position, $p_w^c - p_w^{\hat{c}} = \delta p_w^c \in \mathbb{R}^3$, and similarly for the velocity and inertial bias errors. Differently, for the quaternions, the error
projection on the unit focal length camera plane: \( i_z = \frac{1}{c_i z_j} \begin{bmatrix} c_i x_j \newline c_i y_j \end{bmatrix} + n_j \)

where

\[ c_i p_j = \begin{bmatrix} c_i x_j & c_i y_j & c_i z_j \end{bmatrix}^T \]

and \( n_j \) is a zero-mean white Gaussian measurement noise with covariance matrix \( i R_j = \sigma_v^2 I_2 \). We assume the standard deviation \( \sigma_v \) is uniform over the image and it depends on the performance of the visual front-end. \( C(q_d^j) \) is the rotation matrix associated with the quaternion \( q_d^j \) such that \( \dot{\theta} = C(q_d^j) a_x \).

**C. MSCKF-MSCKF update**

We used the same approach proposed by Zhu [21]. Here we want to give a detailed overview of the method, and how this is applied to the xVIO stack.

The xVIO back-end performs MSCKF [18] updates. MSCKF minimizes the residual between tracked feature points and the back-projection on the camera plane of the triangulated landmark. Figure 5 represents how the MSCKF residuals can be interpreted in a collaborative setup. The tracked landmark \( w p_j \) is triangulated by using all the measurements of robots’ cameras, \( i^C \) and \( i^L \). Then the resulting 3D point is projected into the camera planes that observed the feature.

We express the measurement model \( i^C \) and \( i^L \) as in (7) as a nonlinear function of the state \( x \) and the landmark \( w p_j \) (8), such that the measurement innovation \( i^C z_j = i^C z_j - i^L z_j \) can be linearized as in (9).

\[ i^C z_j = h(x^i, w p_j) + n_j \]

where

\[ h(x^i, w p_j) = \begin{bmatrix} i^C z_j - i^L z_j \end{bmatrix} \]

By stacking together the residuals for all the observations of the landmark \( w p_j \), we write:

\[ \begin{bmatrix} \delta z_{j0}^l \\ \delta z_{j0}^r \end{bmatrix} = \begin{bmatrix} i^C z_j \newline i^L z_j \end{bmatrix} \]

The equations (9)-(11), describe the standard procedure to perform an MSCKF update for the agent \( i \). In particular in a single UAV update, we feed the EKF with the bottom part of the system (11). To perform the collaborative MSCKF update, we need to write a residual that depends on the states of all the agents that observed the same landmark \( w p_j \). Writing such a residual is possible by stacking together all the \( \delta z_{j0}^l \), in (11) with \( l \in L \) where \( L = \{ 0, \ldots, L \} \) is the set of the \( L \) robots that observe \( w p_j \) (12).

\[ \begin{bmatrix} \delta z_{j0}^l \\ \delta z_{j0}^r \end{bmatrix} = D(\begin{bmatrix} i^C z_j \\ i^L z_j \end{bmatrix}) \begin{bmatrix} \delta x^0 \\ \delta x^0 \end{bmatrix} + \begin{bmatrix} i^C z_j \\ i^L z_j \end{bmatrix} \]

Where \( D(\cdot) \) represents the matrix having a diagonal that corresponds to the elements of the input vector.

To perform an EKF measurement update with the resulting equation (12), we need to multiply on each side by the left nullspace \( A_j \) of \( i^C z_j \) and by doing so obtain two subsystems as in (11). Finally, we take the part that depends on the state only,

\[ \delta z_j = i^C z_j + i^L z_j \]

and perform the CI-EKF (19)-(22) update to take into account the possible correlations between the agents states.

The Jacobian \( i^C H_i \) can be rewritten according to the state the various columns refer to as:

\[ \delta z_j = \begin{bmatrix} i^C z_j \\ \ldots \\ i^L z_j \end{bmatrix} \delta x + n_j \]
Fig. 6. Collaborative SLAM representation. $M^{-2}c^1$ and $0_c^1$ are the anchor poses of two different agents 0 (green) and 1 (red) respectively. These anchors observed the landmark $w_p_j$ first with respect to the other frames, $w_p_j$ is parametrized according to $M^{-2}c^1$ for UAV 0 and to $0_c^1$ for UAV 1.

D. SLAM-SLAM update

To perform the collaborative update between two SLAM features, we adopted the approach proposed by Zhu [22] for its efficiency. The method constrains the EKF, that is, when two SLAM features of two different agents 0 and 1 match, it implies that they refer to the same 3D point in the world frame and so the difference between the landmarks in the world frame must be zero:

$$w_p_j^0 - w_p_j^1 = 0 \quad (15)$$

as Figure 6 shows.

In $xVIO$ the SLAM landmarks are stored in the EKF state in terms of an inverse depth parametrization. The parametrization is calculated with respect to the anchor pose $w_p_{aj}$, also part of the state, that is the pose of the camera frame that observed the feature point first. Hence, we can write (15) in terms of the states of the two robots:

$$w_p_{aj}^0 + \frac{1}{\rho_j^0} C(w_p_{aj})^T \begin{bmatrix} \alpha_j^0 \\ \beta_j^0 \\ 1 \end{bmatrix} - \frac{1}{\rho_j} C(w_p_{aj})^T \begin{bmatrix} \alpha_j \\ \beta_j \\ 1 \end{bmatrix} = 0 \quad (16)$$

where $\alpha_j^0, \beta_j^0, \rho_j^0$ are the inverse depth parametrization of the landmark $w_p_j$ observed by the drone $l$. Linearizing (16), we can write the residual obtaining a model that depends on the states of the two UAVs involved in the update as in (14), that is then used to perform the CI-EKF (19)-(22) update.

E. EKF prediction step

The prediction step is the same as in $xVIO$, that is the IMU measurements are propagated using the inertial integration model described in [29]. For sake of completeness we report the IMU measurement model but we refer the reader to Delaune et al. [8] work for more details:

$$\omega_{IMU} = \omega_a^0 + b_g + n_g \quad (17)$$

$$a_{IMU} = a_{g}^0 + (a_{g}^0 - w_g) + b_a + n_a \quad (18)$$

where $n_g$ and $n_a$ are zero-mean Gaussian white noises, the biases $b_g$ and $b_a$ are modeled as random walk with zero-mean Gaussian white noise. The gravity vector $g$ is considered constant, the Coriolis forces and the planet curvature are neglected.

F. EKF measurements update step

To fuse the data coming from different agents, we should keep track of the cross-correlation between the various states. However, doing so is expensive in terms of memory since the number of elements in each drone’s covariance will increase quadratically with the number of agents sharing data. The Covariance-Intersection (CI) algorithm [20] allows us to neglect the correlation information between the robots, and perform data fusion with consistent covariance information. Namely, the resulting covariance out of the CI algorithm is larger than the real estimate one could obtain considering the cross-correlation. By doing so, the EKF is not overconfident, and it is kept consistent. The CI algorithm consists of a convex minimization problem that finds the best weights $\omega$ to scale the covariances of each agent $l \in \{l_0, \ldots, l_L\}$ involved in the update to obtain a covariance with the minimum eigen values. We use the CI-EKF to update the drone state in the same form Zhu proposed, where $l_0$ denotes the drone where the update is performed:

$$S = R + \sum_{l \in \{l_0, \ldots, l_L\}} \frac{1}{\omega_l} H_l^T P_{k-1}^l H_l \quad (19)$$

$$K = \frac{1}{\omega_0} P_{k-1}^0 \omega_0^{-1} \quad (20)$$

$$P_k^l = (I - \omega_0^{-1} K H_l^T) \frac{1}{\omega_0} P_{k-1}^0 \quad (21)$$

$$\delta x_{l}^0 = K \delta z_{l}^0 \quad (22)$$

Where $R$ is the covariance matrix of the Gaussian noise described in Section III-B, $P_{k-1}^l$ is the covariance term of the drone $l$, whereas $P_{k}^l$ is the new covariance of the agent that is performing the update. Notice that $I_{\omega_0}$ is a diagonal matrix where each element $a_{i}^{\omega_0}$ of the diagonal is defined as:

$$a_{i}^{\omega_0} = \begin{cases} 1 & \text{if } i \in \mathbb{J} \\ \omega_0 & \text{otherwise} \end{cases} \quad (23)$$

$\mathbb{J}$ is the set of ids in the state vector $x$ that are directly involved in the update (i.e., the parts of the state that will be updated). This formulation is crucial to perform the CI update without making the EKF underconfident on the part of the state not involved in the collaborative update.

Before propagating the state and the covariance to remove outliers, the innovation of each collaborative update undergoes a $\chi^2$ test with 95% confidence. We compute the Mahalanobis distance with the real covariance of each agent and not with the one scaled by the CI algorithm. The reasons behind this choice are twofold:

- Being conservative: the original covariance is smaller than the scaled one and so the resulting distance is larger, hence is more difficult to pass the $\chi^2$ test.
- Avoid useless computation: we compute the CI covariance only if the test is passed.
G. Communication Pipeline

To perform the collaborative MSCKF and SLAM updates, the agents need the tracks, the state, and the covariance of the other robots. We define a message called MessageUAV which contains all the information needed to perform the collaborative updates. A naive approach to share data between agents would be one in which each robot sends a MessageUAV to all the others at the camera frame rate. The result would be an extremely high usage of the communication channel considering that an estimate of the message size is around 190.7 kB. We computed this estimate by averaging the weights of the MessageUAV sent by the robots in simulation.

To avoid this significant usage of the bandwidth, we developed a keyframe-based request/response data exchange system to lower the data sent over the network. To perform the collaborative updates, the agents need to find correspondences between the features they are tracking and the tracks they receive. Hence, this is the same as performing place recognition and then loop-closure among a series of keyframes.

Each drone sends to all the others a message named RequestUAV at the camera frame rate. This message contains a Binary VLAD descriptor of the frame the agent is viewing. We use a modified version of the BVVLAD [30], to perform place recognition between the UAVs. The ORB descriptor is binary and so we create a VLAD by using binary operations only, in particular, we use the OR (\(\lor\)) and the AND (\(\land\)) operations, instead of sums and subtractions. Doing so we do not need to perform normalization nor exponential decay law to reduce the impact of recurring features. VLADs are of a fixed length, and they can be shortened applying dimensionality reduction through PCA [31]. The resulting request message has a fixed size that is much smaller than a MessageUAV. We created a coarse vocabulary of ORB visual words of 64 centroids. Hence, without applying any dimensionality reduction sharing a VLAD is the same as sharing 64 ORB descriptors. The result is that the RequestUAV message weighs 2.05 kB, including the timestamp and the id of the sender, resulting in lower data consumption. When a robot receives the request message from another agent, it looks among its keyframes to find a loop-closure. A keyframe contains the information needed by the MessageUAV and a VLAD. The score between the received descriptor \(r_j\) and the keyframe one \(r_k\) is given by:

\[
s = \frac{\text{Hamming}(r_j \land r_k)}{D_{\text{max}}}
\]  

(24)

Where \(D_{\text{max}}\) is the maximum distance that can be computed between two descriptors and \(\text{Hamming}(\cdot)\) is the Hamming distance of the resulting vector. If the score is over a certain threshold, the keyframe is wrapped into a message and shipped to the agent who sent the request.

Each agent maintains a database of keyframes. To populate the database, we follow the same approach proposed for PTAM [32]. Namely, every time the ratio between the baseline \(b\), that is the distance between the previous keyframe and the current camera pose, and the average depth \(\bar{z}\) of the tracked features is higher than a threshold that is usually around 10 – 20\%, a new keyframe is defined.

### Table II

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Agent</th>
<th>Collaborative</th>
<th>Ours</th>
<th>xVIO [8]</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Castle</td>
<td>UAV0</td>
<td>0.4512 ± 0.2632</td>
<td>0.6158 ± 0.4136</td>
<td>30.26%</td>
<td></td>
</tr>
<tr>
<td>Parallel</td>
<td>UAV1</td>
<td>0.3871 ± 0.1680</td>
<td>0.4817 ± 0.3039</td>
<td>19.64%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>UAV2</td>
<td>0.6108 ± 0.3730</td>
<td>0.5169 ± 0.3533</td>
<td>-18.17%</td>
<td></td>
</tr>
<tr>
<td>Castle</td>
<td>UAV0</td>
<td>0.3247 ± 0.1678</td>
<td>0.3798 ± 0.1973</td>
<td>9.26%</td>
<td></td>
</tr>
<tr>
<td>Around</td>
<td>UAV1</td>
<td>0.5639 ± 0.1811</td>
<td>0.7186 ± 0.3547</td>
<td>49.02%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>UAV2</td>
<td>0.5679 ± 0.2851</td>
<td>0.5955 ± 0.3276</td>
<td>5.42%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>UAV3</td>
<td>0.8717 ± 0.4094</td>
<td>0.7240 ± 0.3524</td>
<td>-20.40%</td>
<td></td>
</tr>
</tbody>
</table>

### Table III

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Agent</th>
<th>Collaborative Calibrated Ours</th>
<th>Independent Calibrated Ours</th>
<th>xVIO [8] Baseline</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mars</td>
<td>UAV0</td>
<td>0.1610 ± 0.0731</td>
<td>0.1860 ± 0.0928</td>
<td>0.1906 ± 0.0882</td>
<td>0.1835 ± 0.0926</td>
</tr>
<tr>
<td>Yard</td>
<td>UAV1</td>
<td>0.0438 ± 0.0189</td>
<td>0.0812 ± 0.0381</td>
<td>0.0621 ± 0.0248</td>
<td>0.0813 ± 0.0382</td>
</tr>
</tbody>
</table>

IV. Experiments

Figure 8 shows the 3D representation of the environments we recorded the data for testing our system. To validate our system, we used real thermal recordings from the JPL Mars Yard. The data was recorded from the industrial version of the FLIR Boson camera with a lens providing a 95-deg field of view (FOV). It generates 640 x 512 images in the 7.5 \(\mu\)m - 13.5 \(\mu\)m longwave infrared spectral range, with a thermal sensitivity under 40 \(\mu\)K and an 8-ms time constant. The camera was calibrated using as a target a circle grid pattern laminated over a Gatorfoam board left for 5 minutes in the sun, then the Kalibr toolbox [33] was used to find the calibration parameters. The IMU used was the ICM20608 on-board the Pixhawk mini. This dataset is the same as the collection used for the work by Delaune et al. [23]. We merged two Robot Operating System (ROS) bag files where two UAVs fly two squared trajectories with different altitudes, over the JPL Mars Yard for almost 10m. Figure 7 illustrates the two drones trajectories with their estimates. The data is collected with a five-minute distance in time. Therefore, we can rely on the assumption that the thermal fingerprint of the objects does not drastically change over time. The results of our system compared against the baseline that is the standard xVIO implementation are reported in Table III.

We used the RPG trajectory evaluator tool [34] to compute the Absolute Trajectory Error between the ground-truth and the estimates. The collaborative setup with photometric calibrated thermal images outperforms the standard xVIO by 12.26%
We also created two synthetic datasets with visual data to stress different aspects of our method. We modified the VI-Sensor Simulator [35] to generate ground-truth visual, inertial and landmark data to test the back-end of the system. The drones simulated by the VI-Sensor Simulator mount a visual camera that generates 752 x 480 images with a focal length of 455 px at 30 Hz. The IMU sensor simulated is the ADIS16448 that generates data at 200 Hz. The datasets Castle Around and Castle Parallel are recorded using a 3D reconstruction of the Scottish Inveraray Castle. The former simulates four drones flying 220m squared trajectories around the castle with different altitudes and rotations. The latter consists of 3 drones flying 30m parallel trajectories at the same altitude which stresses the communication pipeline. Table II shows the results using the collaborative approach.

Table IV reports the different data usage between the proposed communication pipeline and the naive approach in the different datasets. Notably, our approach shows a decrease of data exchanged by 89% in one of the scenarios. In the Castle and 46.12%.

We also created two synthetic datasets with visual data to stress different aspects of our method. We modified the VI-Sensor Simulator [35] to generate ground-truth visual, inertial and landmark data to test the back-end of the system. The drones simulated by the VI-Sensor Simulator mount a visual camera that generates 752 x 480 images with a focal length of 455 px at 30 Hz. The IMU sensor simulated is the ADIS16448 that generates data at 200 Hz. The datasets Castle Around and Castle Parallel are recorded using a 3D reconstruction of the Scottish Inveraray Castle. The former simulates four drones flying 220m squared trajectories around the castle with different altitudes and rotations. The latter consists of 3 drones flying 30m parallel trajectories at the same altitude which stresses the communication pipeline. Table II shows the results using the collaborative approach.

Table IV reports the different data usage between the proposed communication pipeline and the naive approach in the different datasets. Notably, our approach shows a decrease of data exchanged by 89% in one of the scenarios. In the Castle and 46.12%.

We also created two synthetic datasets with visual data to stress different aspects of our method. We modified the VI-Sensor Simulator [35] to generate ground-truth visual, inertial and landmark data to test the back-end of the system. The drones simulated by the VI-Sensor Simulator mount a visual camera that generates 752 x 480 images with a focal length of 455 px at 30 Hz. The IMU sensor simulated is the ADIS16448 that generates data at 200 Hz. The datasets Castle Around and Castle Parallel are recorded using a 3D reconstruction of the Scottish Inveraray Castle. The former simulates four drones flying 220m squared trajectories around the castle with different altitudes and rotations. The latter consists of 3 drones flying 30m parallel trajectories at the same altitude which stresses the communication pipeline. Table II shows the results using the collaborative approach.

Table IV reports the different data usage between the proposed communication pipeline and the naive approach in the different datasets. Notably, our approach shows a decrease of data exchanged by 89% in one of the scenarios. In the Castle and 46.12%.

We also created two synthetic datasets with visual data to stress different aspects of our method. We modified the VI-Sensor Simulator [35] to generate ground-truth visual, inertial and landmark data to test the back-end of the system. The drones simulated by the VI-Sensor Simulator mount a visual camera that generates 752 x 480 images with a focal length of 455 px at 30 Hz. The IMU sensor simulated is the ADIS16448 that generates data at 200 Hz. The datasets Castle Around and Castle Parallel are recorded using a 3D reconstruction of the Scottish Inveraray Castle. The former simulates four drones flying 220m squared trajectories around the castle with different altitudes and rotations. The latter consists of 3 drones flying 30m parallel trajectories at the same altitude which stresses the communication pipeline. Table II shows the results using the collaborative approach.
ACKNOWLEDGMENTS

The research was funded by the Combat Capabilities Development Command Soldier Center and Army Research Laboratory. This research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, and was sponsored by the JPL Visiting Student Research Program (JVSRP) and the National Aeronautics and Space Administration (80NM0018D0004).

REFERENCES


