Dynamic Localization and Synchronization of Distributed Camera Networks

Krutik Pandya
Rice University
kp103@rice.edu

***Abstract*—Volumetric capture systems have enabled immersive 4D reconstructions for applications in sports, entertainment, and telepresence. However, these systems require expensive fixed camera rigs, extensive offline calibration, and controlled environments, making them impractical for scalable or everyday use. A major limitation of conventional approaches like Structure-from-Motion (SfM) is their reliance on either static cameras or static targets; both the camera and scene cannot move simultaneously without breaking reconstruction accuracy. Most current systems fail to handle the dynamic nature of free-moving cameras capturing moving subjects making them nonviable for wide-scale adoption.**

This project proposes a foundational step toward building more accessible and dynamic volumetric systems by focusing on the most crucial stage of the pipeline: initializing camera extrinsics using onboard sensor data. By leveraging low-cost SoC sensors such as Ultra-Wideband (UWB) and Inertial Measurement Units (IMUs), an estimated real-time camera pose (position and orientation) to align views across multiple devices. This fusion of sensor data allows us to bypass traditional photometric based calibration approaches, and enabling multi view 4D capture from moving cameras on everyday hardware. Accurate spatial and temporal synchronization is essential—without it, conventional SfM pipelines fail to produce coherent reconstructions. This work shows early results that generating reliable cartesian coordinates and orientations from embedded sensors can unlock scalable, decentralized, and real-time 4D reconstruction systems. On a two-node Raspberry Pi camera network, the system achieves capture with < 60 ms timing jitter, recover UWB baselines within 4–5% (around 1.21 m), and estimates a global target camera baseline of 75.4 ± 2.4 cm, demonstrating that low-cost UWB+IMU sensors can provide stable continuous localization for spatially aware multi-camera capture.

***Index Terms*—Camera calibration, Extended Reality, IMU, UWB, sensor fusion, Volumetric Systems.**

I. Introduction

Multi-camera systems play a critical role in volumetric capture, motion analysis, and immersive media production. Achieving accurate localization and synchronization of cameras in free space, however, presents significant challenges [1]. Traditional approaches rely on fixed rigs or static calibration resources, which restrict flexibility and scalability. As camera networks become increasingly dynamic and distributed, errors in localization and synchronization accumulate, ultimately degrading the quality of 4D reconstructions.

Recent advances in large-scale UWB localization show that jointly leveraging range and angle-of-arrival constraints enables scalable, low-latency, sub-meter spatial accuracy, motivating our use of UWB as the backbone for per-frame pose stamps [**?**]. At capture time, each camera attaches a fused pose stamp and time delta to every color and depth frame. Pose stamps come from UWB and IMU fusion [2]; time deltas are measured relative to device slots and bounded to $[0, 2\tau_{\max}]$ for fixed-size headers. Both are delta-coded to minimize rate. On ingest, a lightweight solver re-aligns frames across cameras using the deltas and short IMU windows, so the saved position data is temporally consistent with the saved color data, enabling artifact-free 4D reconstruction.

This project aims to leverage Ultra-Wideband (UWB) and Inertial Measurement Units (IMU) for robust camera localization, combined with RGB data for enhanced calibration. By addressing bandwidth constraints, sensor drift, and dynamic topology adaptation, this research seeks to establish a framework for scalable, accurate, and efficient multi-camera 4D reconstruction in unconstrained environments. Distributed camera networks generate large volumes of color and depth data, making synchronization and calibration both bandwidth-intensive and error-prone [3]. Most recent transformer based approaches [4] degrades reconstruction quality under extreme viewpoint rotations, and fails on scenes with large non-rigid deformations. The proposed research investigates the following research questions:

II. Proposed Methodology

This project uses a sensor-fusion approach that combines UWB and IMU measurements to estimate camera poses, providing fast, continuous location information that is significantly quicker to obtain than solving for extrinsics from images alone and offloading much of the processing requirements from the camera.

A. Camera Pose Estimation

To enable real-time volumetric capture, it is essential to continuously estimate the extrinsic parameters [5], [6] of each camera: its position $\mathbf{t}$ and orientation $\mathbf{R}$ (or rotation $\mathbf{r}$). The camera projection from 3D world coordinates $\mathbf{X}$ to 2D image coordinates $\mathbf{x}$ is given by:

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{K} \begin{bmatrix} \mathbf{R} \mid \mathbf{t} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{1}$$

where:

- $\mathbf{K}$ is the intrinsic camera matrix
- $\mathbf{R}$ is the rotation matrix obtained from IMU (orientation)

- $\mathbf{t}$ is the translation vector obtained from UWB (camera center)
- (u, v) are image coordinates
- (X, Y, Z) are 4D world coordinates

In the proposed system, $\mathbf{R}$ is estimated from the IMU orientation, while $\mathbf{t}$ is derived from UWB-based localization.

B. Coordinate Conversion

The UWB module provides measurements in spherical coordinates (d, θ, ϕ), representing range, azimuth, and elevation, respectively. To integrate these measurements with camera-based pipelines and standard pose representations, we convert them to cartesian coordinates (x, y, z) [7]:

$$x = d \cdot \cos(\phi) \cdot \cos(\theta), \tag{2}$$
$$y = d \cdot \cos(\phi) \cdot \sin(\theta), \tag{3}$$
$$z = d \cdot \sin(\phi), \tag{4}$$

where:

- d is the UWB range to the anchor or reference point,
- θ is the azimuth (yaw), and
- ϕ is the elevation (pitch).

These Cartesian coordinates define the camera center in the chosen world or anchor coordinate frame.

C. Sensor Feedback and Error Correction

To move beyond one-shot calibration and fixed thresholds, the system incorporates a feedback mechanism that continuously checks the consistency between UWB and IMU measurements and uses the resulting residuals to refine the sensor models over time. [8] At each time step t, the fusion module maintains an estimate of each node's pose $\mathbf{P}_t = (\mathbf{R}_t, \mathbf{t}_t)$ and uses a simple motion model driven by the IMU to predict the next pose $\tilde{\mathbf{P}}_t = (\tilde{\mathbf{R}}_t, \tilde{\mathbf{t}}_t)$. For a pair of nodes, this prediction yields an expected inter-node distance

$$d_t = \left\| \mathbf{t}_t(1) - \mathbf{t}_t(2) \right\|,$$

which can be directly compared to the UWB range measurement d_t. The difference

$$r_t = d_t - \tilde{d}_t$$

is treated as a residual: small values indicate that IMU-based pose propagation and UWB measurements agree, whereas large values suggest multi-path, bias in the UWB sensor, or drift in the IMU.

This residual stream acts as an error signal in a lightweight feedback loop. In the short term, individual UWB samples with large $|r_t|$ are treated as outliers and down-weighted or rejected in the pose solver, preventing single bad ranges from corrupting the trajectory. Over longer time windows, persistent non-zero mean residuals are interpreted as evidence of systematic UWB bias or scale error and are used to update a simple calibration model. Likewise, the variance of $\{r_t\}$ is tracked during runtime and used to adapt the relative weighting of UWB and IMU in the fusion process: when UWB becomes noisy (e.g., due to multi-path or occlusions), its measurements are automatically down-weighted and the system relies more heavily on IMU and visual constraints, and vice versa.

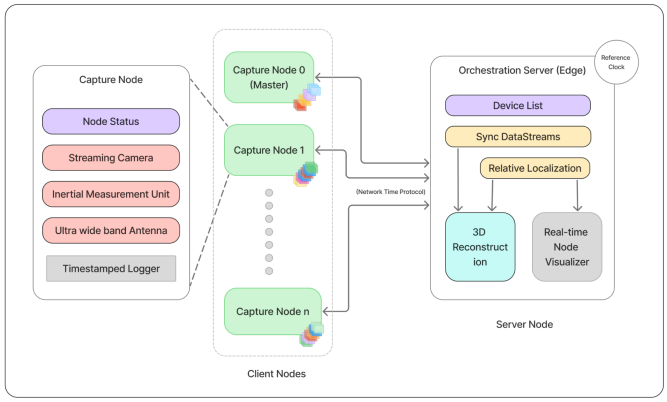

Fig. 1. Dynamic multi-camera capture framework with distributed Raspberry Pi capture nodes (camera+UWB+IMU) coordinated by a central edge orchestration server.

III. System Design

The proposed dynamic multi-camera capture system as shown in Fig.(1) is designed around the principle of distributed capture nodes coordinated by a central orchestration server. The system is optimized for 4D scene reconstruction in environments where cameras are free to move and traditional fixed rig setups are impractical or fail.

A. Capture Node Design

Each capture node is an independent modular unit comprising the following components:

- **Streaming Camera:** Captures and streams RGB video data in real time.
- **Inertial Measurement Unit (IMU):** Provides orientation and movement, with crucial pose data.
- **Ultra Wideband (UWB) Module:** Supplies accurate inter-node ranging and angle-of-arrival (AoA) measurements for localization.
- **Timestamped Logger:** Ensures all data streams are chronologically aligned for fusion and analysis.
- **Node Status Monitor:** Maintains health and communication status of the node.

Each node operates asynchronously but can be synchronized through a distributed clock protocol and localization estimation.

B. Edge-Orchestrated Server Node

The orchestration is managed by an Edge Server, which dictates the network operations:

- **Device Coordination:** Maintains a real-time *Device List*, schedules data ingestion, and supervises node synchronization using Network Time Protocol (NTP).
- **Stream Synchronization:** Aligns multi-modal data streams (camera, IMU, UWB) from all capture nodes.
- **Relative Localization:** Calculates relative positions and orientations using UWB and IMU fusion.
- **4D Reconstruction Engine:** Feeds synchronized and localized image data into a Structure-from-Motion or photogrammetry pipeline for volumetric reconstruction.

- **Visualization Module:** Provides a real-time map of capture nodes and a preview of reconstruction fidelity.

C. Scalability and Flexibility

The modular design allows the system to scale from a single pair of nodes in a lab setting to larger constellations of cameras spread across a room or building. Each capture node is exposed to a common network API for registration, status reporting, and data streaming. Adding a new node therefore requires minimal configuration: once connected to the local network, the node announces itself to the edge server, negotiates basic parameters (node ID, sensing capabilities, logging rate), and is inserted into the global device list without requiring manual re-calibration of the entire rig.

Because localization is performed in a shared coordinate frame (via UWB and, when available, global anchors), nodes can be freely re-positioned or repurposed. A node can act as a static reference camera, a mobile "roving" viewpoint, or even a temporary probe that is only active for a subset of the session. The orchestration layer does not assume a fixed physical topology; instead, it continuously tracks which nodes are currently online and where they are estimated to be in space. This flexibility is particularly important in dynamic environments where both cameras and subjects are moving, and where the cost and rigidity of studio-calibrated, fixed rigs are restrictive.

From a systems perspective, scalability is primarily bounded by network bandwidth and the edge server's ability to ingest and synchronize streams. The design anticipates these constraints by allowing different nodes to run at different capture rates and by supporting keyframe-based logging rather than raw video streaming from all devices. In principle, as the number of nodes grows, the system can trade frame rate for spatial coverage, use UWB/IMU-aware culling to reduce redundant frames, or partition nodes across multiple edge servers that share a common time and coordinate frame. This makes the architecture suitable not only for small-scale experiments but also for future deployments with tens of cameras in more complex scenes.

D. Master Node and Time Synchronization

The first node in the network is designated as the Master Node and serves as the local reference clock for all other nodes. In a typical deployment, the master node either runs an NTP service itself or is tightly synchronized to a nearby NTP server on the same network. All other capture nodes periodically query this reference to discipline their local clocks, which provides a coarse global timeline on which sensor measurements and image frames can be indexed.

In the current prototype, NTP-based synchronization [9] is used as a baseline mechanism that is simple, widely supported, and sufficient to bound clock drift over the duration of a capture session. At startup, each node performs an initial synchronization phase, then continues to refine its clock at a lower rate while logging data. The edge server records both the local timestamps and the NTP-offset metadata [10], and uses nearest-timestamp pairing to align frames across nodes during offline analysis. For the static and slowly moving scenes considered in our experiments, this strategy yields inter-frame alignment that is adequate for evaluating UWB distance stability and relative pose estimates.

NTP alone does not guarantee sub-frame alignment, and in practice we observe residual clock offsets on the order of seconds between nodes. For highly dynamic scenes or higher frame rates, this would be insufficient. A key goal of the testbed is therefore to combine the NTP-derived global clock with UWB and IMU signals: NTP provides a coarse temporal grid, while continuous localization supplies a finer-grained notion of when nodes reach specific spatial configurations (for example, a target baseline or orientation). In future versions of the system, the master node will not only act as a time reference but also as a *spatial* coordinator, triggering captures when favorable geometric conditions are met and using UWB/IMU-based residuals to detect and correct synchronization errors over the course of a session.

E. UWB/IMU-Aware Keyframe Selection

Beyond simple time-based sampling (e.g., capturing at a fixed frame rate), the system leverages UWB and IMU signals to decide which frames are most informative for 4D reconstruction. Algorithm 1 summarizes our UWB/IMU-aware keyframe selection strategy. At each time step t, the node reads the current UWB distance d_t, IMU orientation R_t, and image frame I_t. The first frame is always promoted to a keyframe and initializes the reference pose $(d_{\text{last}}, R_{\text{last}})$. For subsequent frames, we compute the change in baseline Δd_t relative to the last keyframe and the change in orientation $\Delta\theta_t$ between R_t and R_{last}. If either the baseline change exceeds a distance threshold Δd or the orientation change exceeds a rotation threshold $\Delta\theta$, the frame is marked as a new keyframe and the reference pose is updated. Otherwise, the frame is treated as redundant and can be discarded or stored at lower priority. This policy uses continuous UWB and IMU measurements to select spatially informative "lucky" frames and cull near-duplicate views, reducing data volume while preserving the geometric diversity needed for downstream 4D reconstruction.

IV. Experiment

We evaluate the prototype on a small Stream4D-pi-array deployment consisting of two Raspberry Pi capture nodes equipped with RGB cameras, UWB modules, and IMUs, coordinated by an edge server. The goal of these experiments is not to exhaustively benchmark reconstruction quality, but to validate that low-cost sensors can provide stable, continuous pose estimates suitable for spatially aware capture. Therefore the focus is on (i) basic capture stability (frame rate and timing jitter), (ii) UWB ranging behavior in static and slowly varying configurations, (iii) relative and global localization accuracy derived from UWB+IMU fusion, and (iv) the behavior of a simple UWB/IMU-aware keyframe selection policy. All results are computed directly from the logged .csv traces produced by the system.

Algorithm 1 UWB/IMU-Aware frame capture and stream.

Require: UWB distance stream d_t, IMU orientation stream R_t, image stream I_t, distance threshold Δd, rotation threshold $\Delta\theta$

Initialize keyframe set $\mathcal{K} \leftarrow \emptyset$
Initialize last keyframe pose $(d_{\text{last}}, R_{\text{last}}) \leftarrow$ None
for each time step t **do**
  Read current UWB distance d_t
  Read current IMU orientation R_t
  Read current image frame I_t
  if d_{last} is **None then**
    Mark I_t as a keyframe
    $\mathcal{K} \leftarrow \mathcal{K} \cup \{(t, I_t, d_t, R_t)\}$
    $(d_{\text{last}}, R_{\text{last}}) \leftarrow (d_t, R_t)$
  else
    $\Delta d_t \leftarrow |d_t - d_{\text{last}}|$
    $\Delta\theta_t \leftarrow \text{AngleBetween}(R_t, R_{\text{last}})$
    if $\Delta d_t > \Delta d$ **or** $\Delta\theta_t > \Delta\theta$ **then** ▷ Spatially informative change
      Mark I_t as a keyframe
      $\mathcal{K} \leftarrow \mathcal{K} \cup \{(t, I_t, d_t, R_t)\}$
      $(d_{\text{last}}, R_{\text{last}}) \leftarrow (d_t, R_t)$
    else ▷ Frame is likely redundant
      Optionally discard I_t or store as low-priority
    end if
  end if
end for
return keyframe set $\mathcal{K}$

A. Hardware Setup

We evaluate our prototype on a two-node distributed camera network as shown in Fig.(2), built from Raspberry Pi 5 boards, each equipped with a Pi Camera Module 3, a BNO055 IMU, and an NXP UWB module. Each Pi runs our Python logging stack, which records RGB frames, IMU readings, and UWB ranges with local timestamps. A separate host machine collects the logs for offline synchronization and analysis.

Node 1 and Node 2 are placed on a table at approximately head height and pointed towards a static scene. For the relative localization experiment, we keep one node roughly fixed and move the other along a short trajectory on the table to induce small but measurable changes in baseline and height. For the global coordinate experiment, we additionally place a UWB-tagged target object in the scene and solve for its global 4D position together with the Pi node using multiple anchors.

B. Data Collection Protocol

During the two-node experiment, each Pi continuously captures RGB frames at approximately 1 Hz, that corresponds to 1 fps and for every frame, logs the most recent IMU and UWB measurements into a per-node CSV file. Each frame record includes: (i) the camera timestamp, (ii) the IMU sample (orientation, temperature, and calibration status), and (iii) the nearest UWB measurement (distance, azimuth, elevation, and NLOS flag). From these logs, we construct a set of samples across nodes by matching frames whose timestamps lie within a small temporal window.

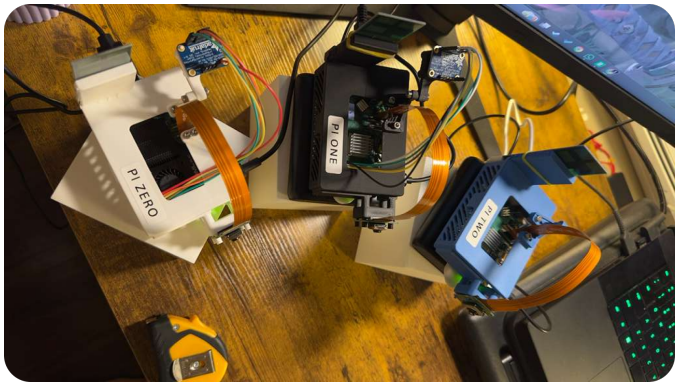

Fig. 2. Two-node Raspberry Pi capture setup for the relative localization and global coordinate experiments. Each node consists of a Raspberry Pi 5 with Pi Camera Module 3, BNO055 IMU, and NXP UWB module; a UWB-tagged target object and anchors are placed in the scene for baseline estimation.

C. Time Synchronization

Before data collection, we perform network time synchronization using an NTP-like protocol so that each node maintains its own local clock with bounded drift. However, due to network latency and startup offsets, the clocks are not perfectly aligned.

We quantify the residual offset using the paired frame timestamps. In the current prototype, we correct for this by aligning frames post hoc based on nearest timestamps. In future work, we aim to use the continuous UWB+IMU trajectories to define synchronization in terms of spatial events (e.g., reaching a target baseline or pose) rather than purely raw timestamps.

D. Derived Coordinate Datasets

From the raw logs, we generate two additional datasets used in analysis. First, we fuse UWB distance and IMU orientation to estimate per-frame 4D positions for each node in a shared coordinate frame. This file contains the estimated (x, y, z) position of Node 1 and Node 2, as well as the Euclidean distance between them for each frame where the solver succeeds.

Second, for the global coordinate experiment, we use multiple UWB anchors to jointly solve for the global 4D coordinates of a tagged target object and the Pi node, producing. This dataset contains the global position of the target and the Pi node for each valid measurement, along with the inferred target Pi baseline.

V. Results

In this section, we report experimental results obtained with our prototype system, focusing on timing stability, sensor-fusion–based pose estimation, and how these factors influence downstream 4D reconstruction quality.

TABLE I
CAPTURE STATISTICS FOR NODE 1 AND NODE 2. DURATIONS AND INTER-FRAME INTERVALS ARE COMPUTED FROM THE IMAGE TIMESTAMPS.

Node	# Frames	Duration [s]	Mean Δt [s]	std(Δt) [s]	FPS
Node 1	103	112.27	1.10	0.047	0.91
Node 2	104	111.97	1.09	0.055	0.92

A. Frame Timing and Capture Stability

We first analyze the temporal properties of the capture pipeline using the per-frame timestamps from the nodes. For Node 1, the 103 recorded frames span 112.27 s, with a mean inter-frame interval of 1.10 s and a standard deviation of 0.047 s. For Node 2, the 104 frames span 111.97 s, with a mean inter-frame interval of 1.09 s and a standard deviation of 0.055 s. The corresponding effective frame rates are 0.91 fps and 0.92 fps, respectively.

B. UWB Ranging Stability

Next, we study the behavior of the UWB distance measurements recorded per frame. Considering all valid UWB samples, Node 1 reports an average distance of 111.25 cm with a standard deviation of 21.97 cm (100 samples), while Node 2 reports 111.28 cm with a standard deviation of 21.60 cm (102 samples). These statistics include a small number of shorter distances and outliers due to motion and occasional multipath.

To focus on the static regime where the true inter-node distance is approximately constant, we restrict attention to frames with UWB distances of at least 100 cm. In this subset, Node 1 reports a mean distance of 121.0 cm with a standard deviation of 5.38 cm across 81 samples, and Node 2 reports 121.21 cm with a standard deviation of 5.72 cm across 81 samples. This corresponds to roughly 4–5% relative variation around the nominal baseline, which is sufficient for initializing camera baselines in downstream multi-view reconstruction pipelines.

Fig.(3) plots the per-frame UWB distances for both nodes. The bulk of the measurements cluster tightly around 121 cm, with a few outliers that correlate with either camera motion or suspected non-line-of-sight (NLOS) conditions. These statistics confirm that our capture loop achieves near-uniform 1 Hz sampling with sub-50 ms jitter on both nodes. Fig.(3) visualizes the inter-frame intervals over time and shows that occasional outliers are rare and bounded.

C. Relative 4D Localization Between Nodes and Global Coordinate Stability

Using the fused 4D coordinates, we evaluate how well the system recovers relative geometry between the two nodes. For each frame where both node poses are available, we compute the Euclidean distance between Node 1 and Node 2 in the estimated coordinate frame. We calculate the positions of nodes by assigning master node as the global origin in the world frame, and with respect to that we calculate the distance between nodes.

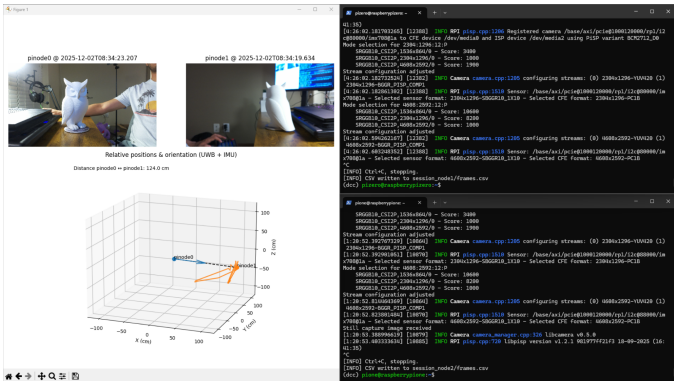

Fig. 3. Per-frame UWB distance between Node 1 and Node 2. Most samples cluster around 121 cm, with 4–5% variation in the static regime and a few outliers due to motion and suspected NLOS conditions.

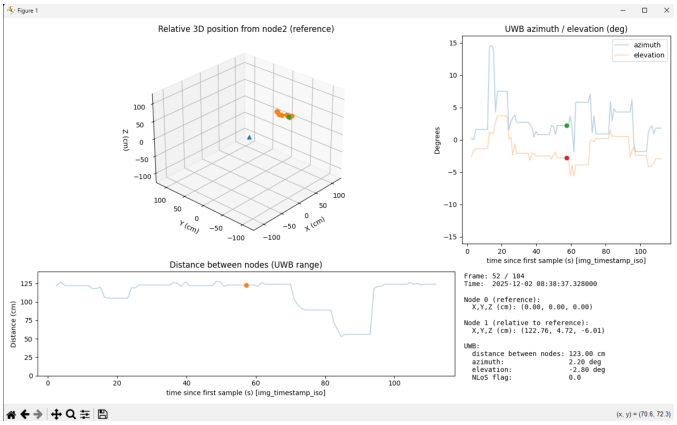

Fig. 4. Inter-frame intervals for Node 1 and Node 2 over the two-node experiment. Both nodes operate near 1 Hz with low timing jitter (standard deviation < 60 ms), and occasional outliers are rare and bounded.

Across valid measurements, the distance between the global target position and the Pi node position ranges from 71.0 cm to 80.0 cm, while keeping master static, and the subordinate moving, with a mean of 75.4 cm and a standard deviation of 2.36 cm. The per-axis variation of the target position is modest: the global target x coordinate has a standard deviation of 1.36 cm, y has 1.50 cm, and z has 3.22 cm. These results suggest that, under static-moving conditions, the global solver can recover a stable baseline between the target and the Pi node, with only a few centimeters of jitter. Fig.(4) shows the cloud of global target positions and Pi node positions across all measurements, along with the mean baseline vector between their centroids.

D. Synchronization Error and Future Spatial Synchronization

The continuous UWB+IMU trajectories, however, provide a richer signal that can support more advanced notions of synchronization. In this experiment, we use timestamp exchange as a baseline and rely on nearest-neighbor timestamp matching to synchronize frames offline. Fig.(5) shows a fast 4D reconstruction that fails when trying traditional reconstruction using Reality-Scan, but works with VGGT, like approaches.

In particular, rather than insisting that all cameras fire at exactly the same global time, future versions of our system

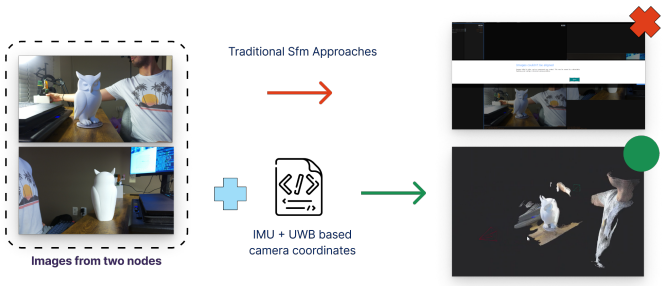

Fig. 5. Two-view 4D reconstruction using sensor-derived poses.

can define synchronization in terms of spatial events for example, triggering captures when the nodes reach a target baseline, or resampling the estimated continuous trajectories and synthesizing virtual frames at common canonical poses. Such spatially aware synchronization would allow the system to (i) culling redundant frames in static scenes, (ii) prioritize "lucky" frames that offer good parallax and low motion blur for 4D methods such as MoGE [11] or VGGT [4], and (iii) stabilize dynamic scenes by compensating for camera motion before multi-view fusion. We leave a full exploration of these ideas to future work, but our initial results demonstrate that the underlying UWB and global coordinate signals are sufficiently stable to support this direction.

VI. Conclusion

In this work, we built and evaluated a low-cost, distributed multi-camera capture system using Raspberry Pi nodes equipped with RGB cameras, UWB ranging, and IMU sensors coordinated by an edge server. Our timing analysis shows that simple NTP-based synchronization and local scheduling achieve 1 Hz capture across nodes with sub–50–60 ms jitter, providing sufficiently aligned timestamps for multi-node sensing. Static experiments demonstrate that UWB delivers a stable geometric baseline, with only 4–5% variation in range and per-axis noise on the order of a few centimeters, making it a practical prior for initializing multi-view geometry. By using UWB and IMU thresholds to trigger keyframes, our system avoids redundant frames and emphasizes views that reflect meaningful motion, better matching the needs of downstream SfM and transformer-based 4D reconstruction models. Together, these results validate the feasibility of sensor-assisted initialization for dynamic, free-moving camera networks and lay the groundwork for future integration with full 4D reconstruction pipelines at larger scales and higher frame rates. Future work will integrate this front-end with full 4D reconstruction pipelines (e.g., VGGT/MoGE/Dust3r), extend to more nodes and higher frame rates, and explore tighter fusion of UWB, IMU, and visual features for robust pose estimation in cluttered, non-static scenes.

References

[1] S. Wang, V. Leroy, Y. Cabon, B. Chidlovskii, and J. Revaud, "DUSt3R: Geometric 3D vision made easy," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 20630–20640, 2024.

[2] S. Ansari, N. Wadhwa, R. Garg, and J. Chen, "Wireless software synchronization of multiple distributed cameras," *arXiv preprint arXiv:1812.09366*, 2018.

[3] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. M. Montiel, and J. D. Tardós, "Orb-slam3: An accurate open-source library for visual, visual-inertial and multi-map slam," *arXiv preprint arXiv:2007.11898*, 2020.

[4] J. Wang, M. Chen, N. Karaev, A. Vedaldi, C. Rupprecht, and D. Novotny, "Vggt: Visual geometry grounded transformer," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 5294–5306, 2025.

[5] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.

[6] J. L. Schönberger and J.-M. Frahm, "Structure-from-motion revisited," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[7] J. D. Jackson, *Classical Electrodynamics*. Wiley, 3rd ed., 1999.

[8] V. Goudar and A. P. Schoellig, "Online spatio-temporal calibration of tightly-coupled ultrawideband-aided inertial localization," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5568–5575, IEEE, 2021.

[9] J. Martin, J. Burbank, W. Kasch, and D. L. Mills, "Network Time Protocol Version 4: Protocol and Algorithms Specification." RFC 5905, June 2010.

[10] "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems," 2020.

[11] Y. Wei, J. Zhang, *et al.*, "Moge: Learning monocular geometry estimation with global consistency," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Project Code

The code is available at GitHub: rice-exp-lab/Stream4D-pi-array. To accompany this prototype, we also provide a short video walkthrough of the setup and capture pipeline, available at: https://www.youtube.com/watch?v=9BnnOTp1prw.