ACT out XR poses

Wanhang Lu
Pennsylvania State University
State College, U.S
wvl5336@psu.edu

Mustafa Goktan Gudukbay
Pennsylvania State University
State College, U.S
mfg5472@psu.edu

Yingtian Zhang
Pennsylvania State University
State College, U.S
yjz5396@psu.edu

Yan Kang
Pennsylvania State University
State College, U.S
ybk5166@psu.edu

Kiwan Maeng
Pennsylvania State University
State College, U.S
kvm6242@psu.edu

Mahmut Kandemir
Pennsylvania State University
State College, U.S
mtk2@psu.edu

Anand Sivasubramaniam
Pennsylvania State University
State College, U.S
axs53@psu.edu

Abstract

Accurate position and orientation estimation is extremely important to provide the best immersive experience to users on their eXtended Reality (XR) headsets and mobile devices. XR pipelines use sensory inputs (cameras, IMUs, etc.) to run pose estimations (SLAM) and predictions before rendering each frame. However, even the best SLAM algorithms cannot be perfect and there have been prior efforts to fix inaccuracies through two additional stages - Extrapolation (EX) using additional IMU inputs beyond those available to SLAM, and Back-End Correction (BEC) to account for latencies in frame rendering and communication with a backend machine to perform the rendering in the XR pipeline. Despite these incremental efforts, we find that these silo-ed stages of estimation/correction leave much room for improvement.

To fill this gap, we leverage the analogy with motion estimation/correction in robotics which uses learning-based Action Chunking Transformers (ACT), that we adapt and design for XR pipelines called ACT-XR. We implement ACT-XR using the Monado framework and evaluate it with several off-the-shelf workloads. On average, ACT-XR provides 44.80% (in Euclidean position), 22.37% (in yaw), 78.32% (in pitch) and 35.86% (in roll) lower errors compared to the best performing state-of-the-art prediction/correction mechanisms.

Keywords: XR, pose estimation, viewport prediction, remote rendering.

1 Introduction

XR (eXtended Reality) devices and applications are expected to transform how we interact with each other and the physical and virtual worlds. With strapped headsets and other smart on-person devices, these applications are intended to make such interactions as seamless as we are used to in the current world. Among the numerous challenges to attaining this goal is the need to estimate the position and orientation of the user (of the device) in order to accurately render the view at any time on the device, despite continuous movements and latencies.

Simultaneous Localization and Mapping (SLAM) algorithms have been proposed [10] to use different sensory inputs to make pose estimations. These algorithms take inputs from one or more cameras and IMUs to project an environment map and pose estimation periodically at a certain frequency (e.g., 20 Hz). Since these algorithms cannot be perfect, there has been considerable prior work to improve them for better accuracy [4, 21, 31, 33, 39, 40].

Another source of the discrepancy lies in the frequency mismatch of sensory inputs, which are not coordinated between the sensors (camera at 20 Hz and IMUs at 200 Hz) and the frequency at which feature-based SLAM is executed (5–20 Hz). Recognizing this problem, there have been efforts to extrapolate (EX) the estimates from the SLAM algorithms to incorporate the most recent sensory input [8]. Although such techniques do accommodate more up-to-date sensory input in the estimation, as we will show, they still suffer significant inaccuracies.

A root cause of high errors with the above two estimation stages is the substantial time lag between such estimation and the time the frame is displayed on the headsets. This time includes the cost of rendering the frame (taking 11 ms for a 90hz display). Further, in many settings, including most of today's XR pipelines [1, 16, 28, 30], such rendering is not done on the end-user device/headset, but on a backend desktop/server because of the former's resource constraints and capabilities. This implies that between the extrapolation output and the final frame display, there are additional round-trip network latencies (running to hundreds of milliseconds for large AR assets) between the device and the backend system. To account for this lag, there have been efforts to correct for these effects on the backend system, using techniques such as Regression and LSTM [15, 25, 26, 32]. This backend correction (BEC) also suffers from inaccuracies.

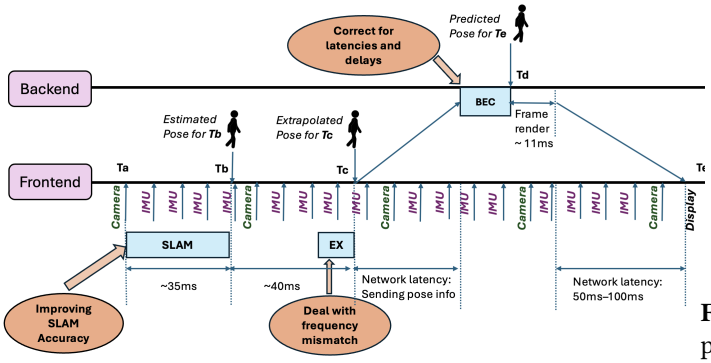
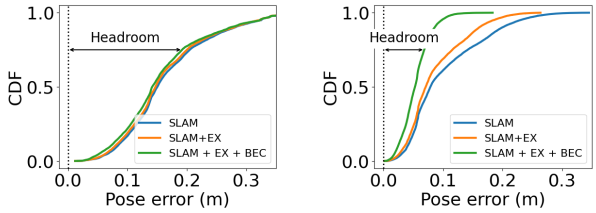


Figure 1. Pose estimation and rendering pipeline of current XR systems, split between a client (headset) and backend (desktop/server). Prior works on pose estimation focus individually on one of (i) SLAM algorithms, (ii) Extrapolation (EX) based on subsequent IMU inputs, which operates at a higher frequency than the camera/SLAM, or (iii) Correction at the Backend (BEC) to account for subsequent network latencies and rendering delays. Each such stage operates independently (silo-ed) on the outputs of the prior stage.



(a) MIPP05 **(b)** MIPB06

Figure 2. CDF of pose estimation errors in the current XR pipeline. This shows three stages in the pipeline: (i) Error of Basalt [38] at time T_b; (ii) Error of the extrapolated pose at time T_c (SLAM+EX); and (iii) Error despite the BEC done at the server at T_d (SLAM+EX+BEC). As shown, MIPP05 has less predictability and higher motion variability compared to MIPB06, making the three-stage pipeline even less effective.

Our goal, in this work, is not to improve the accuracy of (or propose new) SLAM algorithms. Rather, working under the assumption that even the most sophisticated SLAM algorithm can generate inaccurate estimations [18] [17] based on the latencies and associated variability between estimation and the eventual display, we intend to fix the inaccuracies from whatever SLAM algorithm is employed, as close to the final display stage as possible so as to benefit from up-to-date information.

2 Motivation and Problem Statement

Figure 1 shows the pipeline of representative functionalities in current XR systems that are of concern in this study, from estimating the position/pose of the user (client) to subsequently rendering the frame/image on the user's headset for that pose at time T_e. These functionalities happen at the client (user's headset) and the backend (more resource-rich desktop or server), as is depicted in the figure. There are 3 stages to this estimation, as explained below.

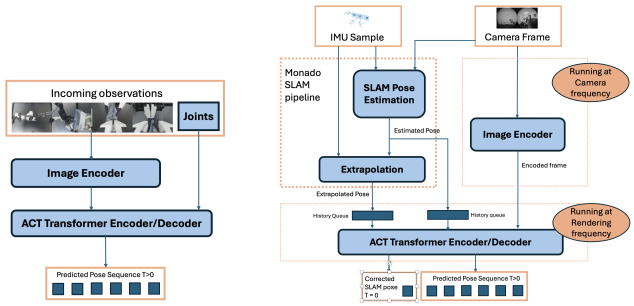
Stage 1: SLAM: SLAM algorithms (operating at the camera sensing frequency) use up-to-date sensory inputs (at T_a in the figure) and take around 35–40 ms on current headsets to produce the pose estimate at time T_b. These algorithms are obviously not perfect, and can have inaccuracies and jitteriness.

Stage 2: Extrapolation (EX): To address this mismatch between IMU sensing frequencies and the heavy weight SLAM algorithm, previous work [8] has used the additional IMU inputs that arrived subsequent to the ones used by SLAM, to fine-tune pose estimates. Such techniques are broadly referred to as "Extrapolation", and in Figure 2, we use Linear Combination (the most optimized technique used in Monado [8]) to account for these additional IMU inputs, as shown in the (SLAM+EX) line. While such an extrapolation does improve accuracy, the magnitude of improvement is not as significant—for instance, in the MIPP05 dataset 2a, we still have over 50% frames having higher than 14cm Euclidean errors.

Stage 3: Backend Correction (BEC): To address these high network latencies, there has been prior work [15, 20, 26, 27, 32] which takes the pose estimations from the headset and refines them further in the backend before rendering the frame, as is shown in Figure 1. These try to account for the round-trip latency of network transmissions in the pose estimations, to provide a better estimate at time T_d for the ground truth at time T_e, apart from trying to correct for any errors in the previous two stages. We have again run experiments using Linear Regression (LR) (as suggested in [25, 26]) to perform these corrections/adjustments at the Backend (BE) subsequent to the estimates from the headset. In Figure 2, we plot the estimation errors against the ground truth for different frames as a CDF graph shown by the line SLAM+EX+BEC. As we see, while errors do reduce compared to using the previous two approaches (to some extent for the MIPB06 dataset in the Monado SLAM dataset [19]), there is still considerable scope for improvement, especially in MIPP05, where the improvement is negligible.

3 Our Proposal – Holistic Correction

We draw inspiration for such a multi-parameter model from the domain of robotics (SLAM is again widely used in robotics to estimate pose/positions [23, 24, 40]), where there are several similarities to what we are trying to achieve.



(a) ACT for robotic arm control **(b)** ACT-XR

Figure 3. Original ACT model proposed in [41] for robotic arm control and our ACT-XR adaptation.

For instance, **continuous error mitigation** approaches are used to ensure proper control as the robot performs different tasks [2, 35, 36]. Robotic movements are expected to be continuous, smooth, and not jerky. XR systems similarly need **non-jerky and seamless rendering** of frames despite sudden shifts in the head position.

Robotic pose estimation also goes through multiple compute stages. It has been well recognized in the robotic community that individually correcting errors in these stages does not suffice, and a **holistic end-to-end consideration** is more fruitful [5, 11, 12, 22, 34, 37]. We believe a similar strategy is also mandated in our context.

Based on these similarities, we adopt techniques from robotics to the XR context in order to perform a holistic error mitigation mechanism at the last (BEC) stage of the pipeline, and borrow the state-of-the-art - Action Chunking with Transformers (ACT) [41] - from that domain for our purpose.

3.1 ACT-XR for BEC Stage

Figure 3a shows the structure of ACT [13] that takes input from (1) four mounted cameras and (2) the current position of robotic arms. These are fed to a transformer-based encoder to predict the future position of the arms. Rather than a single position in the immediate future, ACT predicts several poses for the next k time steps to capture the temporal correlations among joint movements better.

We have modified ACT for our purposes, as shown in Figure 3b, which we call ACT-XR. ACT-XR's transformer layers and hyper-parameters remain the same as ACT. We make three major changes to adapt ACT for XR:

- We use both historical SLAM-generated 6DoF (6 degrees-of-freedom) pose estimation and IMU-based extrapolated poses as the input positions for Encoder/Decoder layers.
- In addition to predicting poses for the future (as is done in ACT), we also predict the pose for Time=0 in ACT-XR, i.e., for time T_c. This helps train the model, recognizing inherent inaccuracies present even before the model starts predicting.
- We utilize the frequency mismatch between camera and render to reduce compute cost. we run the Image Encoder and Transformer layers separately at different frequencies – the first at 54Hz (MSD Camera Sensing Frequency) and the latter at 90 Hz.

3.2 Implementing ACT-XR

We have implemented ACT-XR on a desktop (backend) with an Intel i7-12700k processor, 32 GB memory, and an RTX 4090 GPU (later, we also conduct experiments with an implementation on a Jetson AGX Orin board [29] to explore possibilities of running this on a headset-like platform). Training of ACT-XR is done once, offline, and takes around 12 hours on the desktop. This is regardless of the domain/video that is eventually used in the XR application; hence, it does not affect the runtime performance studies.

We emulated the headset using Monado [8], an industry-level OpenXR runtime that runs SLAM algorithms along with extrapolation mechanisms described earlier.

4 Experimental Setup

4.1 Workloads

We also use the Monado SLAM Dataset (MSD), which comes with Monado and is also used in related SLAM studies [38]. MSD is provided by Collabora Consorium [7] which has been captured using HTC Vive [16] with users playing three different XR games [3], [6] [14]) in a fixed indoor environment. The camera input frequency is 54 Hz, and the IMU sensing frequency is 1000 Hz for these MSD datasets.

In this work, we use Basalt as the SLAM baseline, as reported in [9] it meets latency constraints while still having a relatively small average translation error (ATE) when running with Monado.

The target frame rendering frequency is set to 90Hz, a typical refresh rate for commercial XR headsets and mobile phones. The SLAM ground truth provided by MSD is at 1000 Hz, which makes it safe to construct training sets with fine-grained prediction windows without losing precision.

4.2 Metrics

Our main target is the accuracy of predicting the user's pose at time T_e (Figure 1) before the rendering starts at time T_d. We use the error between this estimation at T_d (in different schemes against which we compare) compared to the ground truth at T_e. There are six degrees of freedom in describing the pose/position (the three-dimensional coordinates and the angle of view in these three dimensions). In the interest of space, we show the errors in these degrees of freedom as four separate metrics: one each in the error of the Euler angle for each of the three dimensions (Yaw, Pitch, and Roll) which

depicts the orientation, and the fourth depicting the error in the Euclidean distance in the three-dimensional space.

4.3 Baselines

We implement the following baseline XR pipelines and compare our approach to them: **NoBEC**: The default pipeline in Monado [8] where only the first two stages of pose estimation/correction (i.e., SLAM + EX) are present, and no subsequent effort is made to correct any residual errors. **LR**:Use Linear Regression model at the backend for correction, as suggested in [25][26]. **DTR**: [26] also proposed a Decision Tree Regression model for the backend correction. **LSTM**: [32] [15] suggest using LSTM models for 360 video viewport prediction.

5 Evaluation Results

Figure 5a shows the errors in the different baselines of position estimates with respect to the ground truth at time T_e averaged over all frames across all datasets.

We note that NoBEC has considerable errors that amount to 10.71%, 6.54%, 5.65%, and 2.93%, on average, for position, yaw, pitch, and roll, respectively. For baseline models trained with possibly erroneous SLAM inputs and outputs, these models can compound their own prediction errors on top of SLAM errors, thus causing built-in *"error amplification"* for the multi-stage XR pose pipeline.

The important point to take away from these results is that ACT-XR consistently gives lower errors across all these attributes. On average, it has 46.20%, 43.47%, 93.31% and 30.89% lower errors, compared to not performing any BEC. More importantly, in each of these attributes, it yields 44.80% (compared to LR in position), 22.37% (compared to DTR in yaw), 78.32% (compared to DTR in pitch), and 35.86% (compared to DTR in roll) lower errors compared to the best-performing BEC strategy for each attribute. This clearly demonstrates the merit of ACT-XR and its ability to closely track the user's movements in the virtual space.

To study the resilience to network latency, we also conduct experiments varying this as a parameter in our experiments, and the resulting position errors are shown in Figure 4.

As expected, higher network latencies imply higher error rates, and we see this across all the BEC models and in the baseline, which does not make any corrections (NoBEC). Of the prior models, the simple linear regression (LR) in the BEC stage works as well as – if not better than – more sophisticated DTR or LSTM. However, across the spectrum of these network latencies, ranging from as low as 50ms to as high as 1s, we see that ACT-XR consistently delivers the lowest errors of all these choices, making it quite robust to network latencies. This also suggests that our solution would still produce the best estimates under such conditions in real settings, where network latencies can dynamically vary.

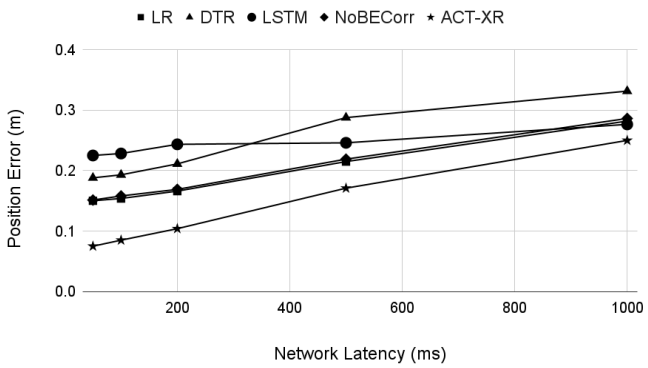
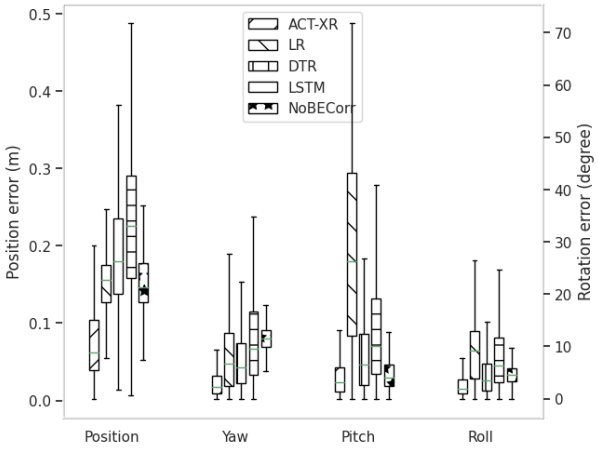


Figure 4. Error in Position prediction for different network latencies (and consequently prediction windows). Errors will, in general, increase for larger windows due to higher uncertainties.



(a) Box plot of errors for the four attributes characterizing pose/position across all workloads and frames. Traditional models of BEC may compound any errors in the early stages and show higher error rates.

BEC. Model	position	yaw	pitch	roll
NoBEC	0.158	11.795	5.121	5.260
LR	0.154	9.752	27.206	10.238
DTR	0.198	8.589	8.394	5.668
LSTM	0.228	9.4559	11.167	6.5537
ACT-XR	0.085	6.667	1.819	3.635

(b) Average errors

Figure 5. Accuracy of ACT-XR when compared with traditional models used in BEC, under a prediction window (network latency) of 100ms.

6 Concluding Remarks and Future Work

We have tackled the important problem of enhancing user experience on XR devices, which critically depends on the prediction of the position/orientation of the user at any instant. Despite three stages (SLAM, EX, and BEC) of estimation/correction in current XR pipelines, this study shows that these do not suffice, leaving considerable headroom for improvement. Instead, we have presented ACT-XR to holistically address this problem and achieves great improvement.

In the future, we intend to explore the ability to create fully on-edge XR pipelines that do not require a backend machine, and avail of the latest sensory inputs at the device to make better predictions. We will also explore and collect a wider spectrum of datasets/workloads from very diverse usages (beyond indoor gaming).

References

[1] ALVR Team. ALVR. https://github.com/alvr-org/ALVR. Accessed: 2025-09-08.

[2] J.-J. Bai, H.-X. Song, and S.-M. Hu. Multi-dimensional and message-guided fuzzing for robotic programs in robot operating system. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, ASPLOS '24, page 763–778, New York, NY, USA, 2024. ACM.

[3] Beat Games. Beat Saber. https://beatsaber.com/, 2018. Accessed: 2025-09-08.

[4] C. Campos, R. Elvira, J. J. Gomez, J. M. M. Montiel, and J. D. Tardos. ORB-SLAM3: An accurate open-source library for visual, visual-inertial and multi-map SLAM. *IEEE Transactions on Robotics*, 37(6):1874–1890, 2021.

[5] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, In press.

[6] Cloud Games Ltd. pistolwhip. https://www.cloudheadgames.com/pistol-whip. Accessed: 2025-09-08.

[7] Collabora Ltd. Collabora. https://www.collabora.com. Accessed: 2025-09-08.

[8] Collabora Ltd. Monado. https://monado.dev/. Accessed: 2025-09-08.

[9] M. de Mayo, D. Cremers, and T. Pire. The Monado SLAM Dataset for Egocentric Visual-Inertial Tracking. In *Proceedings of the International Conference on Intelligent Robots and Systems*, IROS '25, 2025. Accessed: 2025-09-08.

[10] H. Durrant-Whyte and T. Bailey. Simultaneous Localization and Mapping: Part I. *IEEE Robotics & Automation Magazine*, 13(2):99–110, 2006.

[11] W. Fan, X. Guo, E. Feng, J. Lin, Y. Wang, J. Liang, M. Garrad, J. Rossiter, Z. Zhang, N. Lepora, L. Wei, and D. Zhang. Digital twin-driven mixed reality framework for immersive teleoperation with haptic rendering. *IEEE Robotics and Automation Letters*, 8(12):8494–8501, 2023.

[12] H. Fang, H.-S. Fang, Y. Wang, J. Ren, J. Chen, R. Zhang, W. Wang, and C. Lu. AirExo: Low-cost exoskeletons for learning whole-arm manipulation in the wild. In *Proceedings of the IEEE International Conference on Robotics and Automation*, ICRA '24, pages 15031–15038. IEEE, 2024.

[13] Z. Fu, T. Z. Zhao, and C. Finn. Mobile ALOHA: Learning bimanual mobile manipulation with low-cost whole-body teleoperation. https://arxiv.org/abs/2401.02117, 2024. Accessed: 2025-09-08.

[14] Halfbrick Studios. ThrillofFight. https://www.thethrillofthefight2.com/. Accessed: 2025-09-08.

[15] X. Hou and S. Dey. Motion prediction and pre-rendering at the edge to enable ultra-low latency mobile 6DoF experiences. *IEEE Open Journal of the Communications Society*, 1:1674–1690, 2020.

[16] HTC Co. HTC-Vive Headset. https://www.vive.com/us/. Accessed: 2025-09-08.

[17] T. Hu, T. Du, Z. Qu, and M. Gorlatova. XR reality check: What commercial devices deliver for spatial tracking. https://arxiv.org/abs/2508.08642, 2025. Accessed: 2025-09-08.

[18] T. Hu, T. Scargill, F. Yang, Y. Chen, G. Lan, and M. Gorlatova. SEESys: Online pose error estimation system for visual SLAM. In *Proceedings of the 22nd ACM Conference on Embedded Networked Sensor Systems*, SENSYS '24, pages 322–335, 2024.

[19] HuggingFace Co. Monado SLAM Datasets. https://huggingface.co/datasets/collabora/monado-slam-datasets. Accessed: 2025-09-08.

[20] G. K. Illahi, A. Vaishnav, T. Kämäräinen, M. Siekkinen, and M. Di Francesco. Learning to predict head pose in remotely-rendered virtual reality. In *Proceedings of the 14th ACM Multimedia Systems Conference*, MMSys '23, page 27–38, New York, NY, USA, 2023. ACM.

[21] H. Jiang, Y. Xu, K. Li, J. Feng, and L. Zhang. RoDyn-SLAM: Robust dynamic dense RGB-D SLAM with neural radiance fields. https://arxiv.org/abs/2407.01303, 2024. Accessed: 2025-09-08.

[22] L. Ke, J. Wang, T. Bhattacharjee, B. Boots, and S. Srinivasa. Grasping with chopsticks: Combating covariate shift in model-free imitation learning for fine manipulation. https://arxiv.org/abs/2011.06719, 2020. Accessed: 2025-09-08.

[23] P.-Y. Lajoie and G. Beltrame. Swarm-SLAM: Sparse decentralized collaborative simultaneous localization and mapping framework for multi-robot systems. *IEEE Robotics and Automation Letters*, 9(1):475–482, Jan. 2024.

[24] W. Liu, B. Yu, Y. Gan, Q. Liu, J. Tang, S. Liu, and Y. Zhu. Archytas: A framework for synthesizing and dynamically optimizing accelerators for robotic localization. In *Proceedings of the 54th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-54, page 479–493, New York, NY, USA, 2021. ACM.

[25] Y. Liu, B. Han, F. Qian, A. Narayanan, and Z.-L. Zhang. Vues: Practical mobile volumetric video streaming through multiview transcoding. In *Proceedings of the 28th Annual International Conference on Mobile Computing And Networking*, MobiCom '22, page 514–527, New York, NY, USA, 2022. ACM.

[26] Y. Liu, P. Zhou, Z. Zhang, A. Zhang, B. Han, Z. Li, and F. Qian. MuV2: Scaling up multi-user mobile volumetric video streaming via content hybridization and sharing. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*, ACM MobiCom '24, page 327–341, New York, NY, USA, 2024. ACM.

[27] J. Meng, S. Paul, and Y. C. Hu. Coterie: Exploiting frame similarity to enable high-quality multiplayer VR on commodity mobile devices. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '20, page 923–937, New York, NY, USA, 2020. ACM.

[28] Meta, Inc. Meta Quest. https://www.meta.com/quest/quest-3/. Accessed: 2025-09-08.

[29] NVIDIA Co. Jetson AGX Orin. https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-orin/. Accessed: 2025-09-08.

[30] NVIDIA Co. NVIDIA GeforceNow. https://www.nvidia.com/en-us/geforce-now/. Accessed: 2025-09-08.

[31] T. Qin, J. Pan, S. Cao, and S. Shen. A general optimization-based framework for local odometry estimation with multiple sensors. https://arxiv.org/abs/1901.03638, 2019. Accessed: 2025-09-08.

[32] M. F. R. Rondón, L. Sassatelli, R. Aparicio-Pardo, and F. Precioso. TRACK: A new method from a re-examination of deep architectures for head motion prediction in 360° videos. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(9):5681–5699, 2022.

[33] A. Rosinol, A. Violette, M. Abate, N. Hughes, Y. Chang, J. Shi, A. Gupta, and L. Carlone. Kimera: from SLAM to spatial perception with 3D dynamic scene graphs. https://arxiv.org/abs/2101.06894, 2021. Accessed: 2025-09-08.

[34] S. Ross, G. J. Gordon, and J. A. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. https://arxiv.org/abs/1011.0686, 2011. Accessed: 2025-09-08.

[35] D. Shah and T. M. Aamodt. Collision prediction for robotics accelerators. In *Proceedings of the ACM/IEEE 51st Annual International Symposium on Computer Architecture*, ISCA '24, pages 566–581, 2024.

[36] D. Shah, N. Yang, and T. M. Aamodt. Energy-efficient realtime motion planning. In *Proceedings of the 50th Annual International Symposium on Computer Architecture*, ISCA '23, New York, NY, USA, 2023. ACM.

[37] A. Souissi, H. Fradi, and P. Papadakis. Leveraging event streams with deep reinforcement learning for end-to-end UAV tracking. https://arxiv.org/abs/2410.14685, 2024. Accessed: 2025-09-08.

[38] 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)*, 5(2):422–429, 2020.

[39] L. Wu, H. Zhu, S. He, J. Zheng, C. Chen, and X. Zeng. GauSPU: 3D Gaussian Splatting processor for real-time SLAM systems. In *Proceedings of the 57th IEEE/ACM International Symposium on Microarchitecture*, MICRO-57, pages 1562–1573, 2024.

[40] X. Zhang, H. Zhu, Y. Duan, W. Zhang, L. Shangguan, Y. Zhang, J. Ji, and Y. Zhang. Map++: Towards user-participatory visual SLAM systems with efficient map expansion and sharing. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*, ACM MobiCom '24, page 633–647, New York, NY, USA, 2024. ACM.

[41] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn. Learning fine-grained bimanual manipulation with low-cost hardware. https://arxiv.org/abs/2304.13705, 2023. Accessed: 2025-09-08.