SLAM-Share: Visual Simultaneous Localization and Mapping (SLAM) for Real-time Multi-user Augmented Reality

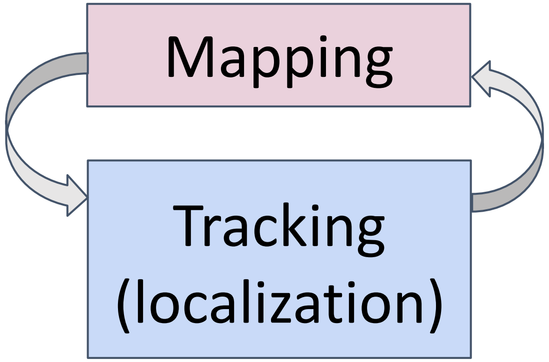
Aditya Dhakal, Xukan Ran, Yunshu Wang, Jiasi Chen and K. K. Ramakrishnan

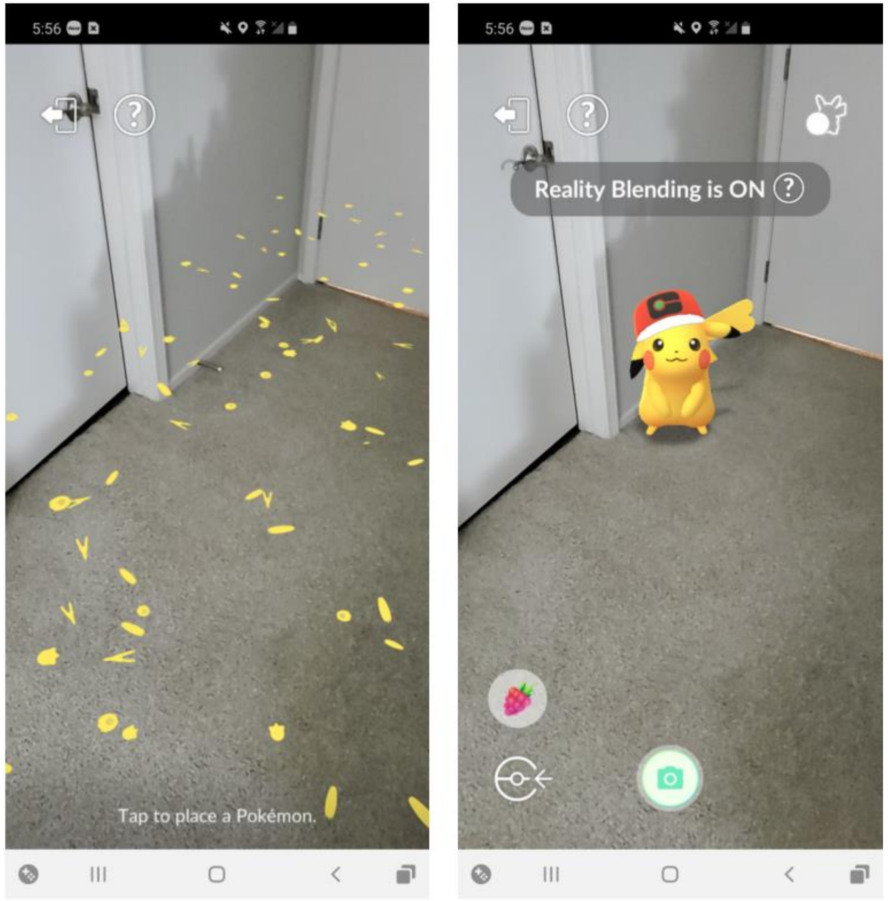
University of California, Riverside

Networked Systems Group
UC RIVERSIDE

Why is SLAM needed for AR?

- Augmented Reality (AR) applications must know the user device's 3D location in the world
- Simultaneous Localization And Mapping (SLAM) is the process for AR app to localize
- SLAM is used when precision greater than GPS is desired





Pokemon Go Buddy Adventure

Networked Systems Group
UC RIVERSIDE

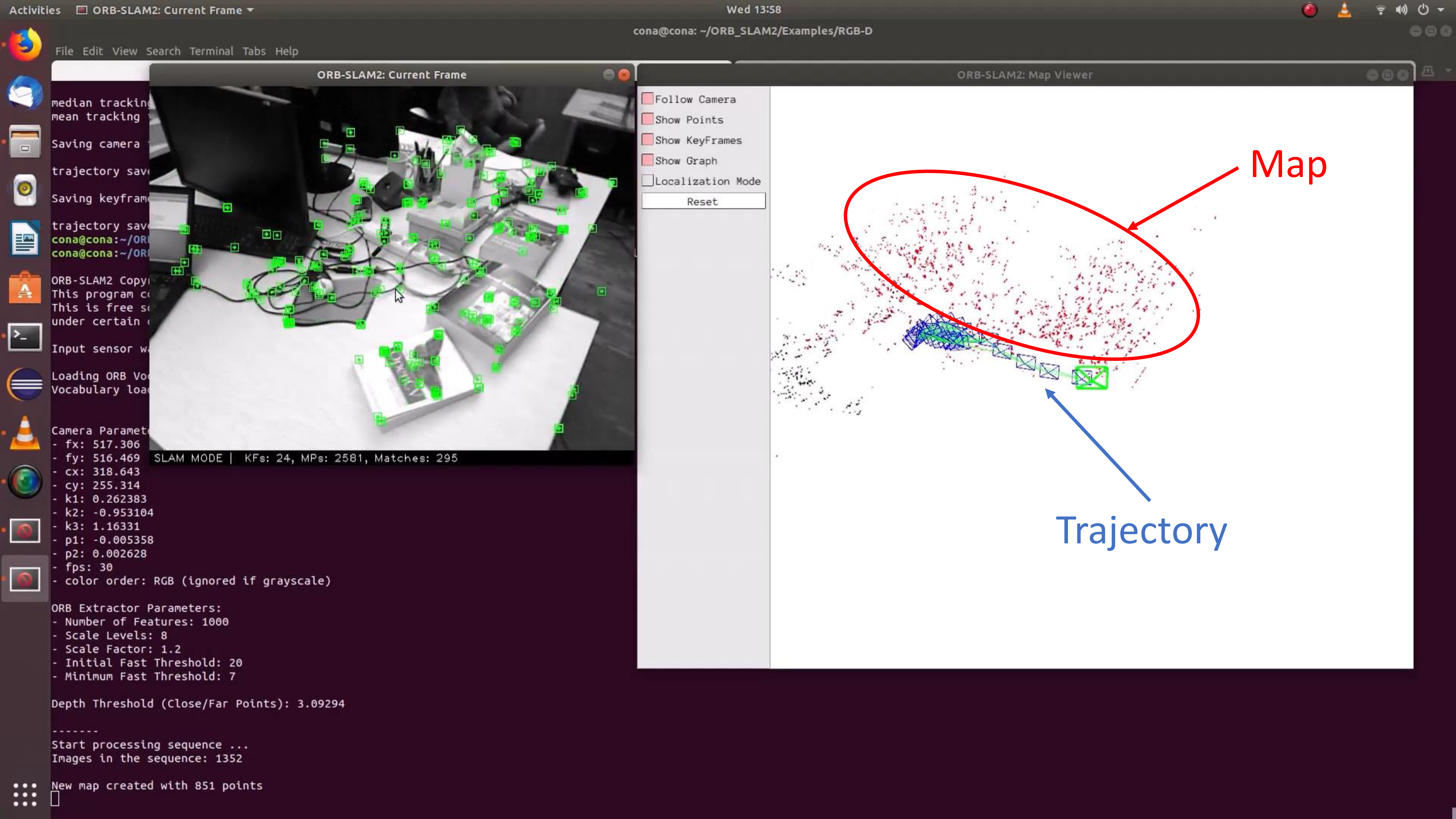
Background: Visual SLAM Execution Steps

- Visual SLAM is based on images of environment

1. **Features are extracted** from image frame
2. **Tracking:** Extracted features are compared to existing map to localize
3. **Mapping:** New features are inserted into the map
    1. **Map-points**: feature points that will go in the map
    2. **Keyframe**: Image frame and its position and orientation
4. Error is minimized in the map

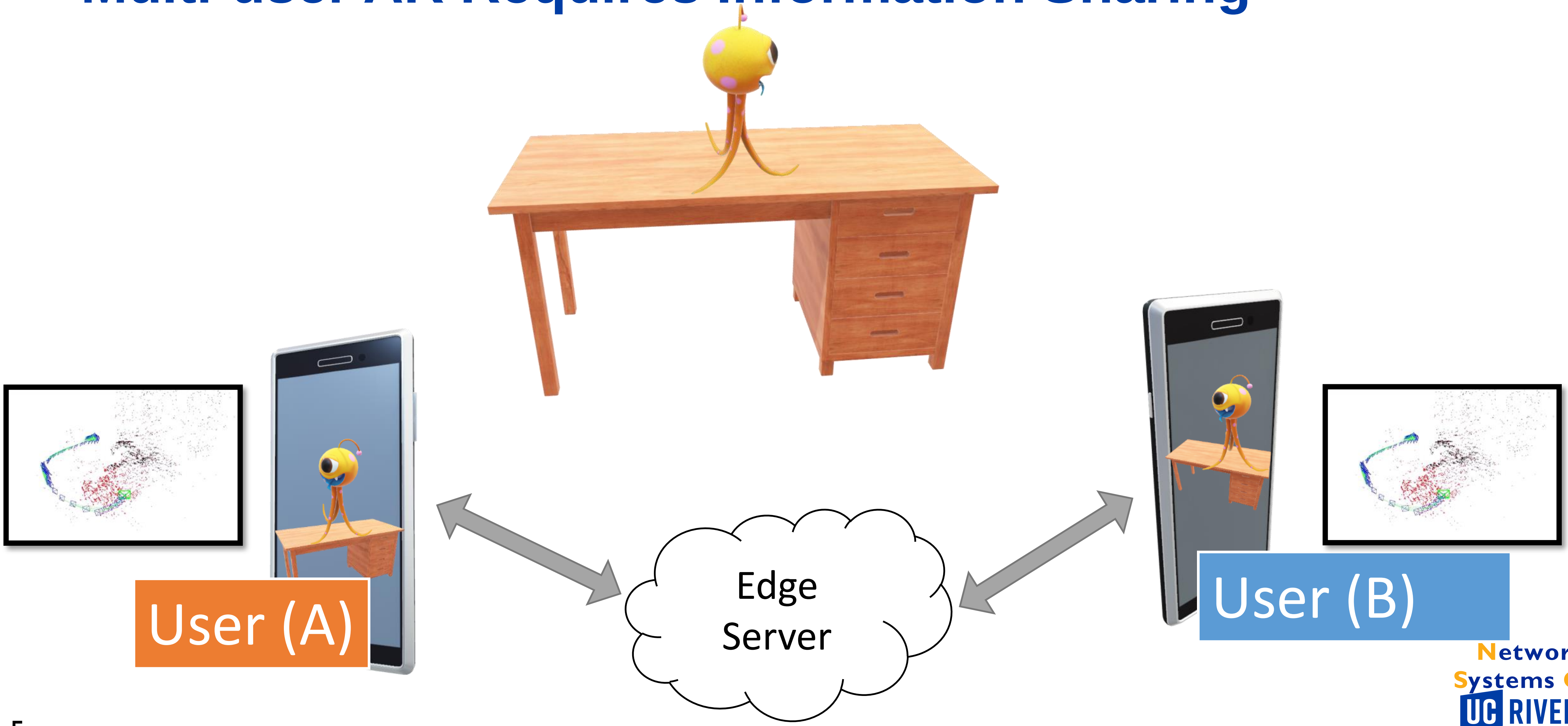
- We base SLAM-Share on ORB-SLAM, a Visual SLAM application

Campos, C., Elvira, R., Rodríguez, J. J. G., Montiel, J. M., & Tardós, J. D. (2021). Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam.


Activities
ORB-SLAM2: Current Frame
Wed 13:58
cona@cona: ~/ORB_SLAM2/Examples/RGB-D
File Edit View Search Terminal Tabs Help
ORB-SLAM2: Current Frame
ORB-SLAM2: Map Viewer
Follow Camera
Show Points
Show KeyFrames
Show Graph
Localization Mode
Reset
Map
Trajectory
SLAM MODE | KFs: 24, MPs: 2581, Matches: 295
median tracking
mean tracking
Saving camera
trajectory sav
Saving keyfram
trajectory sav
cona@cona:~/OR
cona@cona:~/OR
ORB-SLAM2 Copy
This program c
This is free s
under certain
Input sensor w
Loading ORB Vo
Vocabulary loa
Camera Paramet
- fx: 517.306
- fy: 516.469
- cx: 318.643
- cy: 255.314
- k1: 0.262383
- k2: -0.953104
- k3: 1.16331
- p1: -0.005358
- p2: 0.002628
- fps: 30
- color order: RGB (ignored if grayscale)
ORB Extractor Parameters:
- Number of Features: 1000
- Scale Levels: 8
- Scale Factor: 1.2
- Initial Fast Threshold: 20
- Minimum Fast Threshold: 7
Depth Threshold (Close/Far Points): 3.09294

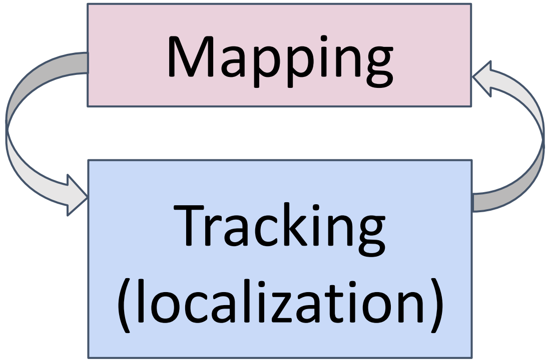
Start processing sequence ...
Images in the sequence: 1352
New map created with 851 points

Multi-user AR Requires Information Sharing



Networked Systems Group
UC RIVERSIDE

How does latency affect the AR display?



User B's View

User A's View (Ground Truth)

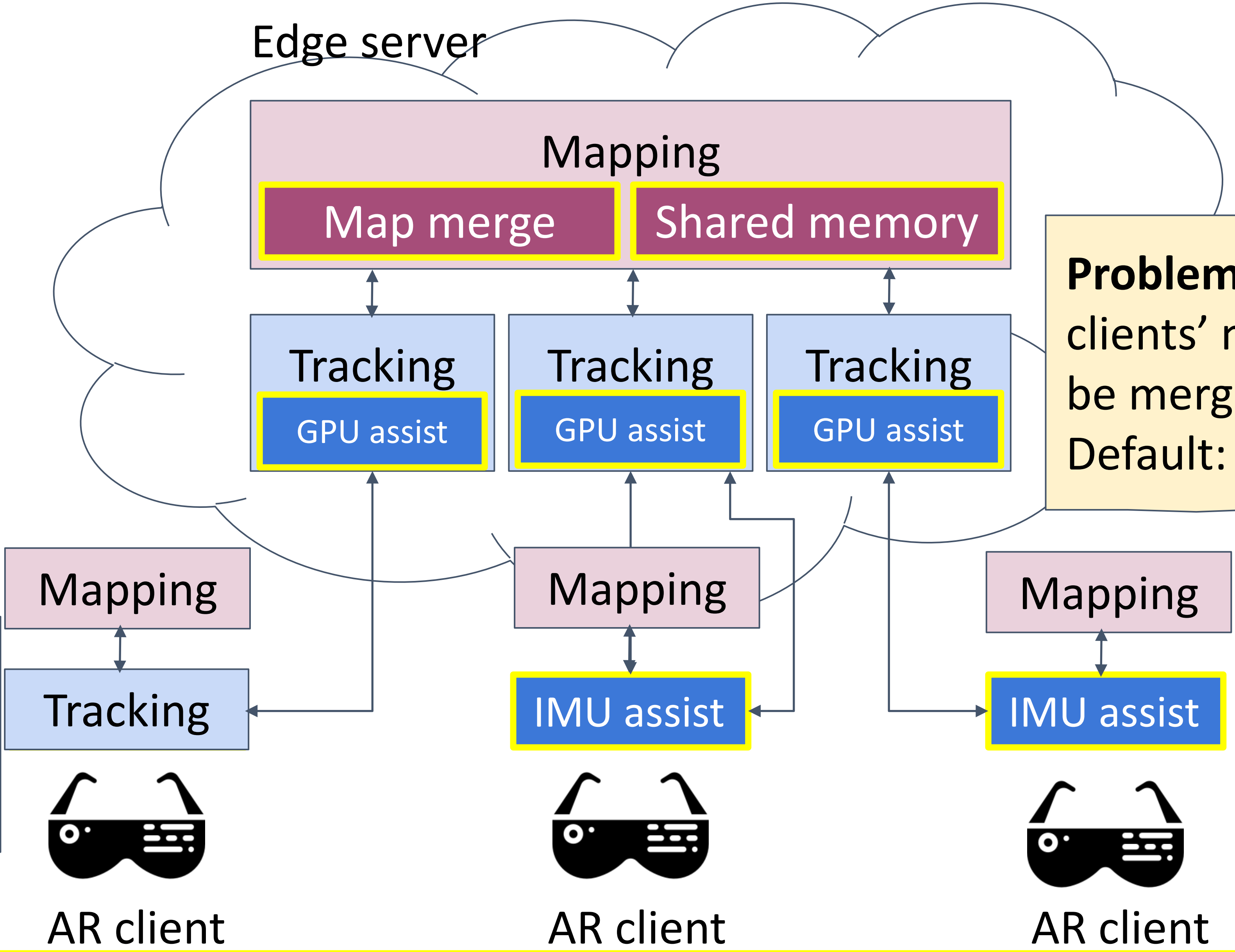
Case (a): Without information sharing, no holograms appear

Case (b): With slow tracking, holograms may appear later

Case (c): With slow map merging, holograms may appear inaccurately placed

Networked Systems Group
UC RIVERSIDE

SLAM-Share Overview

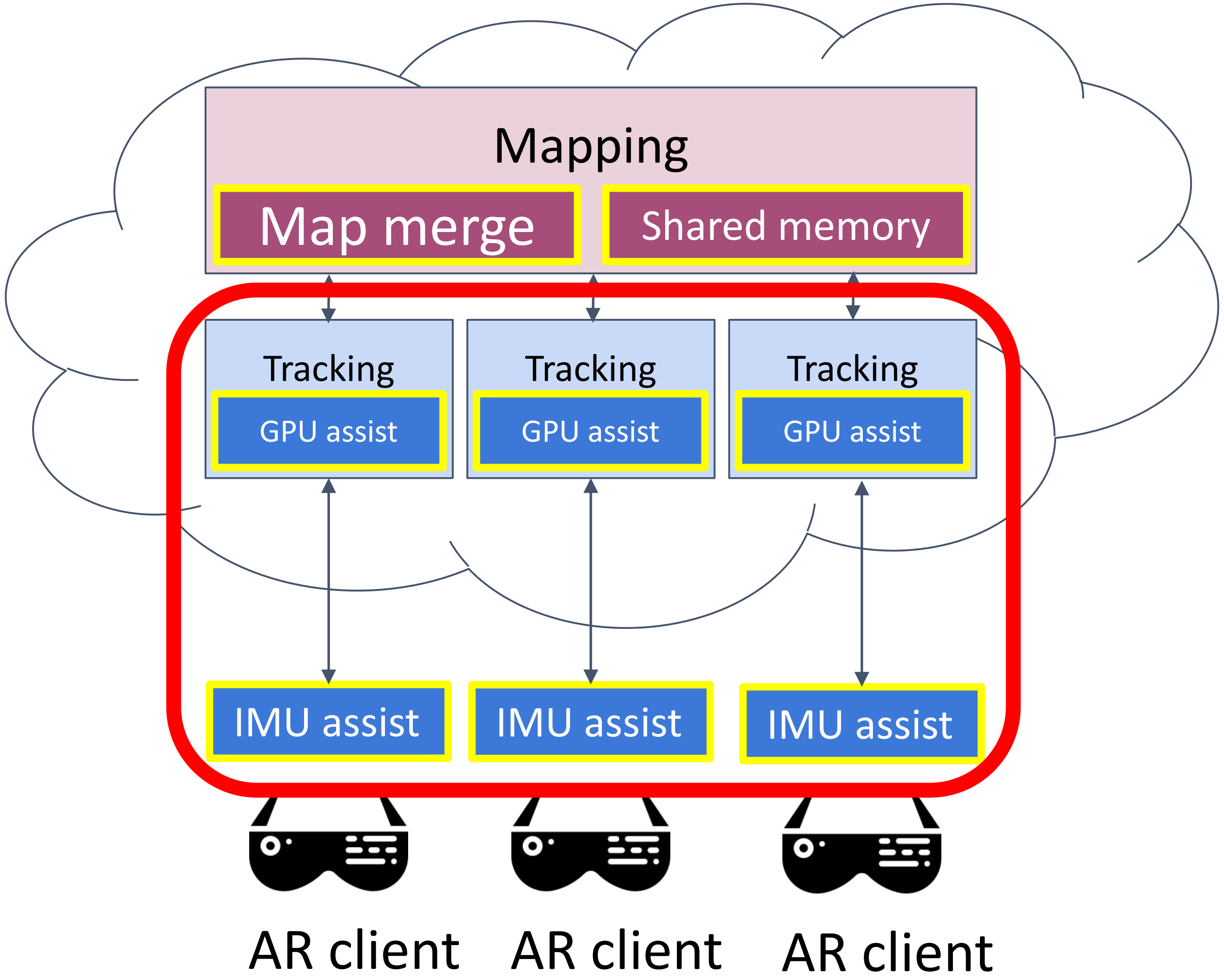


Problem: Multiple clients' maps need to be merged quickly! Default: takes 3 sec

Problem: Tracking is slow on mobile clients! < 30 FPS

Our contributions: New offloading architecture with IMU assist, GPU assist, map merging, and shared memory for high-throughput, multi-user visual SLAM for AR

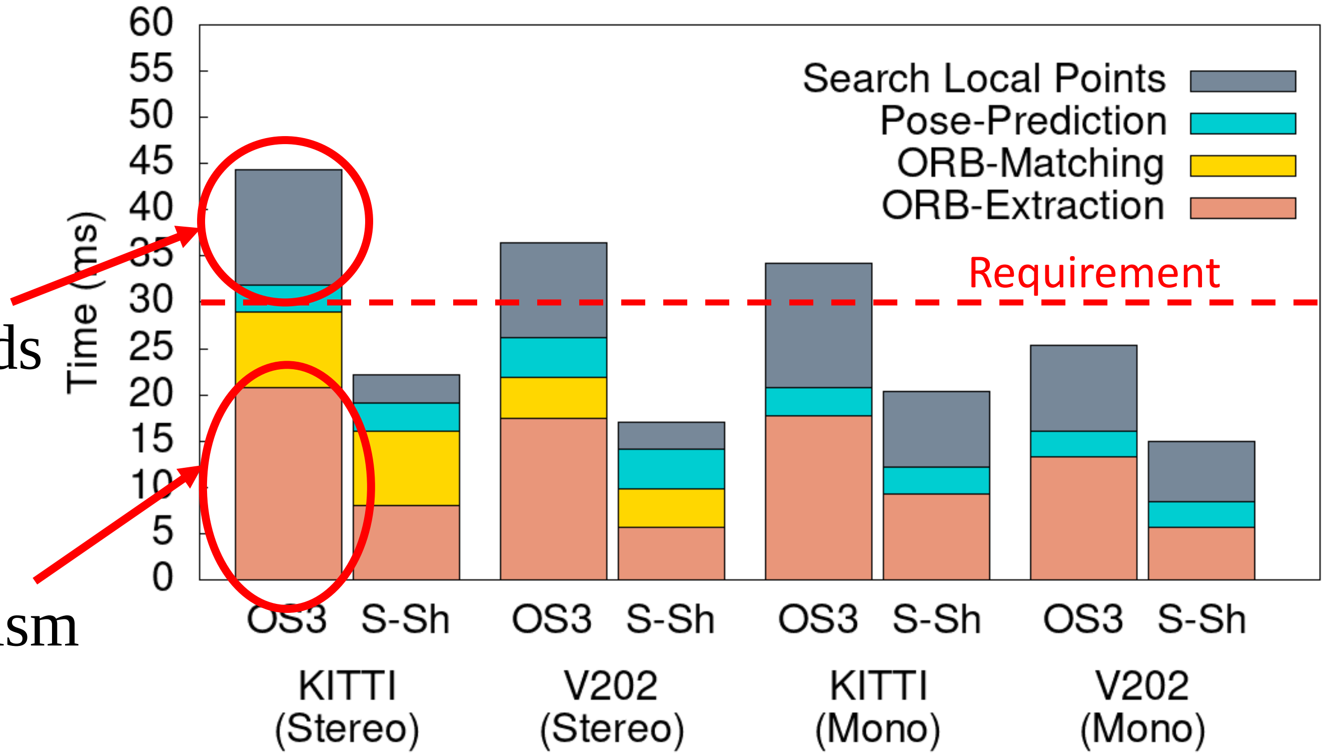
Tracking



Mapping
Map merge
Shared memory
Tracking
GPU assist
Tracking
GPU assist
Tracking
GPU assist
IMU assist
IMU assist
IMU assist
AR client
AR client
AR client

GPU assist:
How does the GPU help?

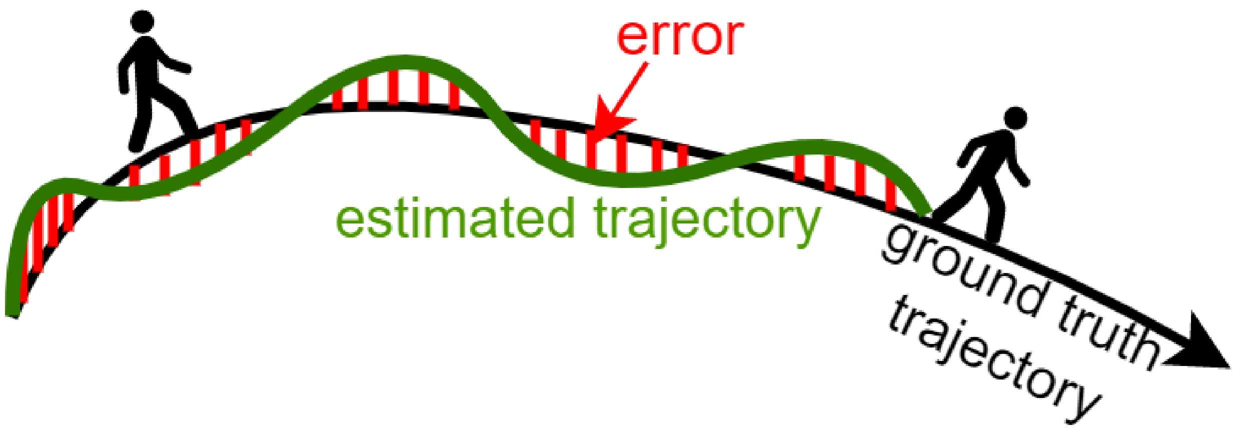
- Search Local Points is time-consuming
  → SLAM-Share exploits parallel threads
- ORB-Extraction is time-consuming
  → SLAM-Share exploits GPU parallelism
- Overall, SLAM-Share reduces tracking time by more than 40% compared to ORB-SLAM3 run in CPU only



OS3 = ORB-SLAM3
S-Sh = SLAM-Share

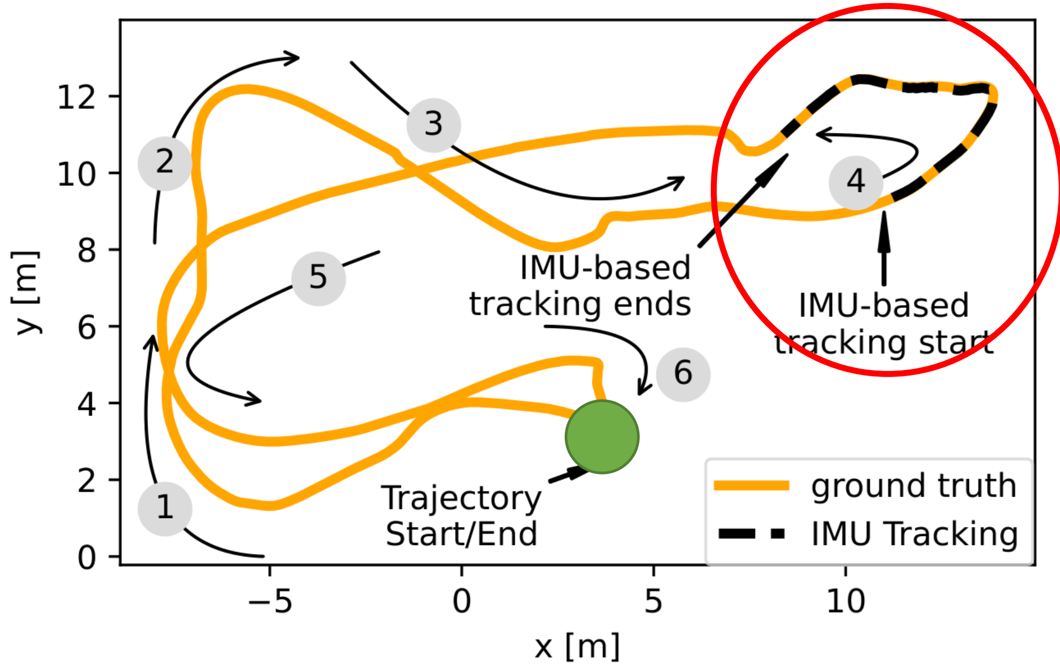
Networked Systems Group
UC RIVERSIDE

IMU assist



to

Server Contact Lost!

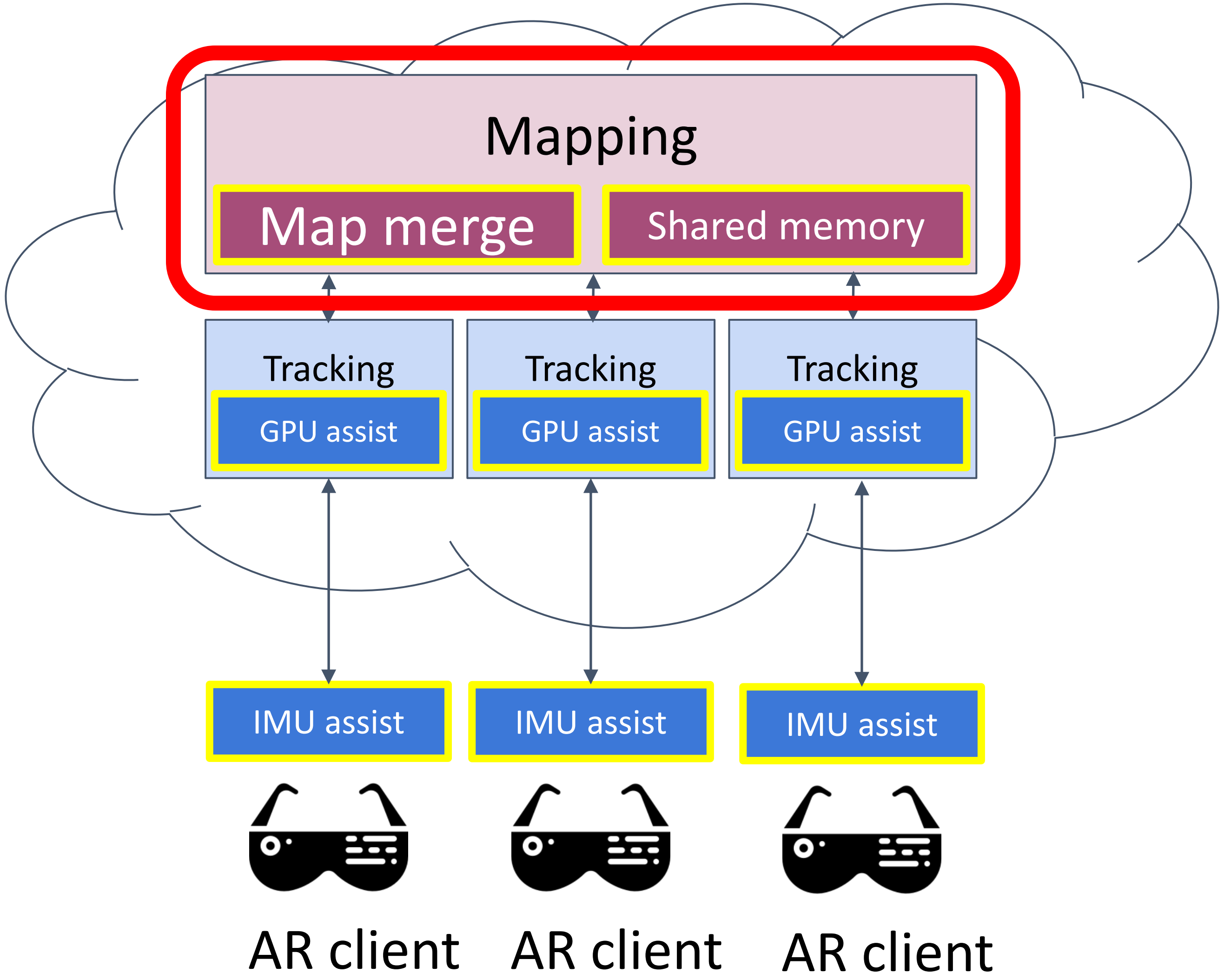


- Once server contact restored
  - Client merges IMU + SLAM pose

- Evaluation: IMU-based tracking is accurate for a short time
  - But long term IMU-based tracking accumulates errors

RTT (ms)	IMU-Tracking region ATE RMSE (cm)
0 (Baseline)	2.41
90	2.45
200	2.67
300	2.71
10000	300

Mapping

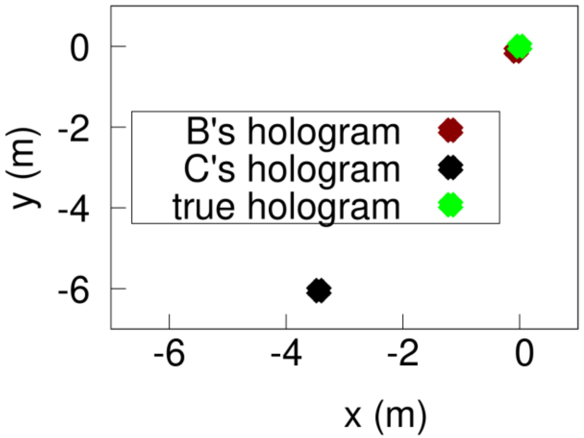


Networked Systems Group
UC RIVERSIDE

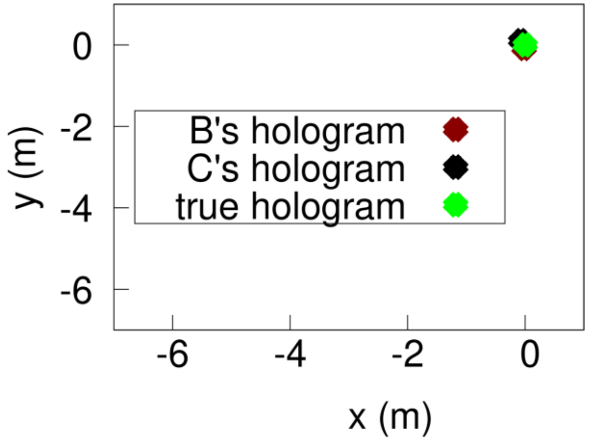
Why is map merging needed?

- Map merging fuses the shared information between users
  - Map merging brings together users' maps and puts them in same "perspective"

Without map-merging:



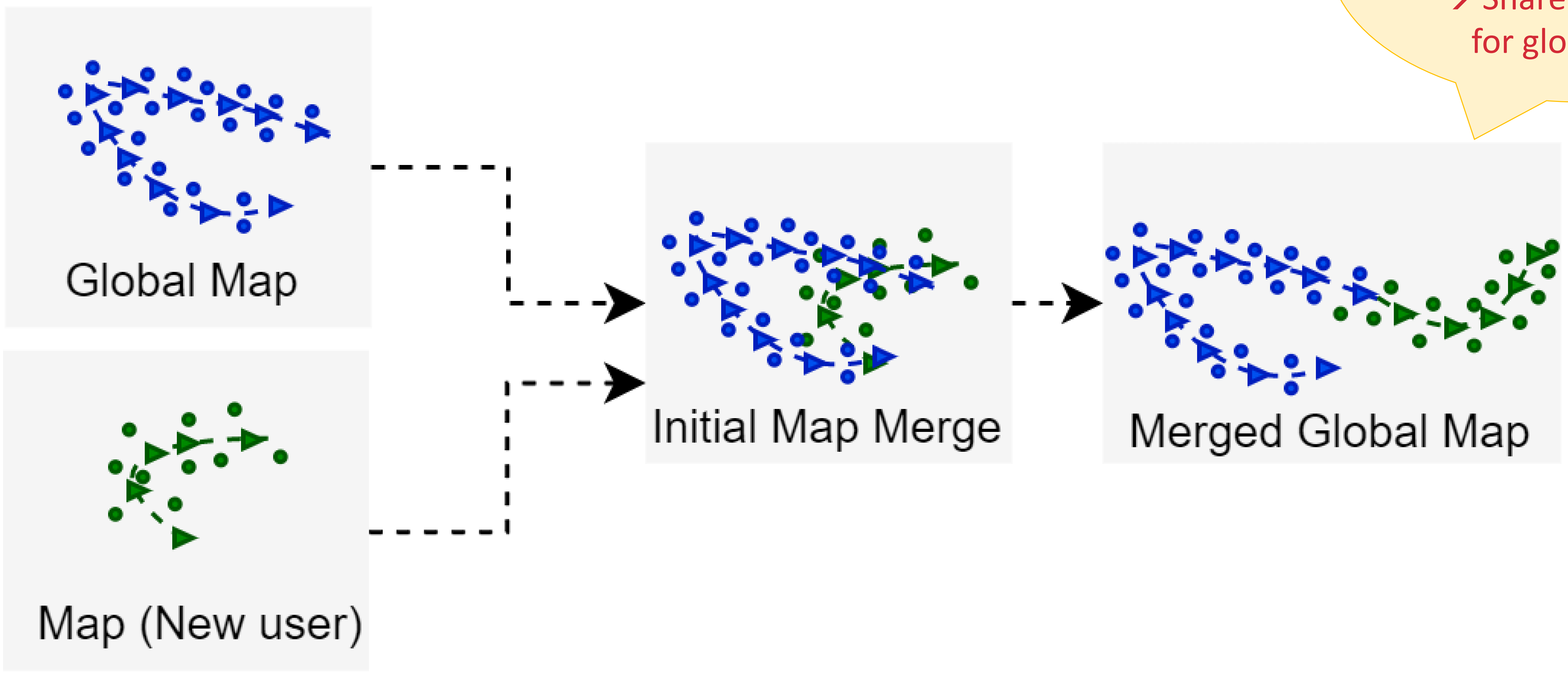
With map-merging:



- Without map merging, the virtual objects will be misplaced for some users
- With map merging, the virtual objects are at the same place for all users

Networked Systems Group
UC RIVERSIDE

Map Merge Example



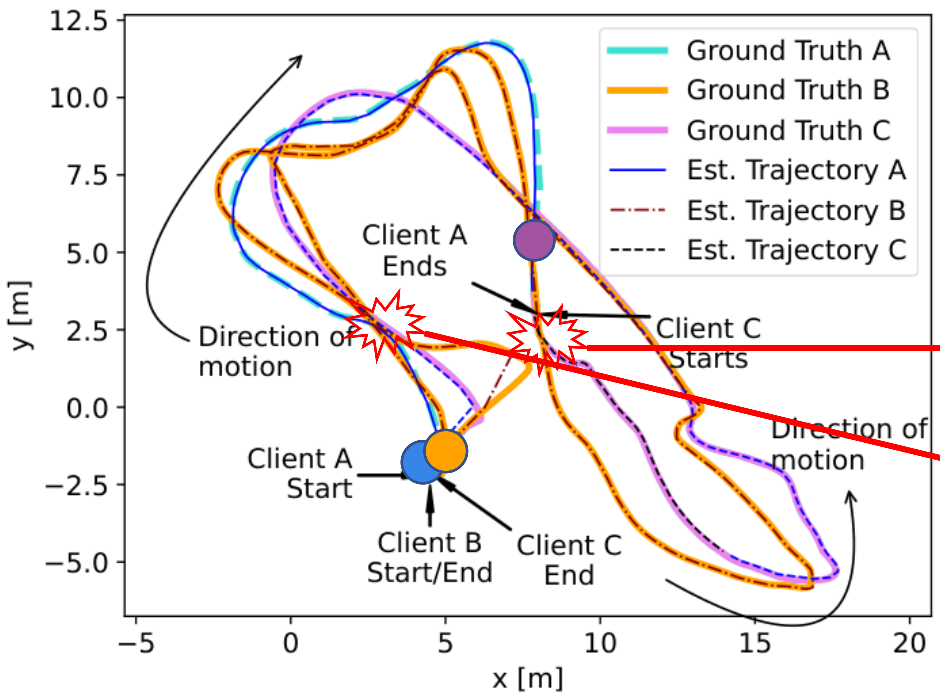
Each client keeps local copies of shared map → inefficient!
→ Shared memory for global map

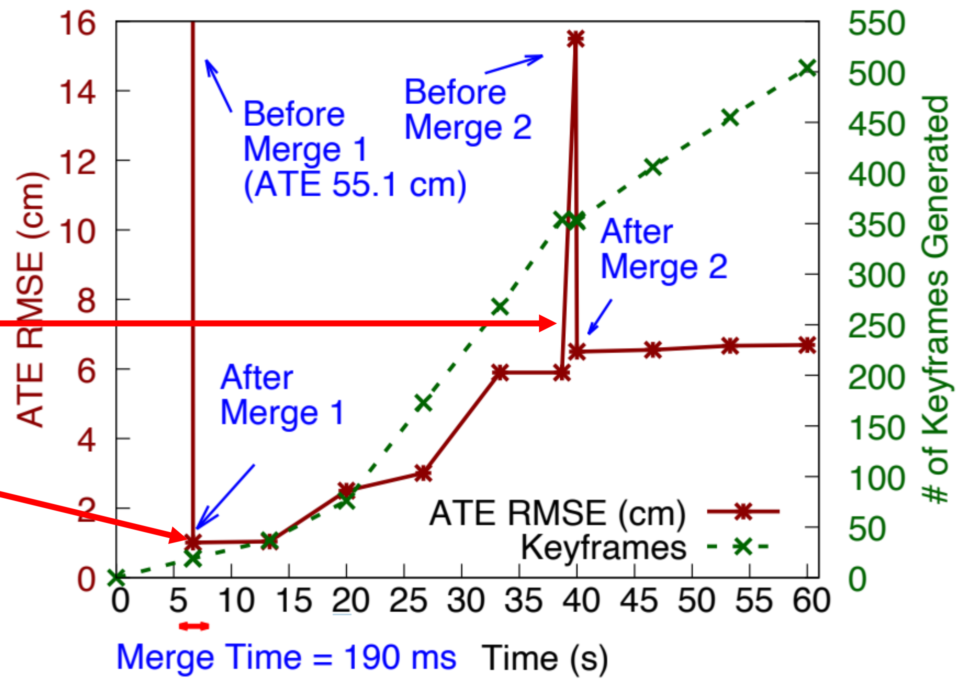
Networked Systems Group
UC RIVERSIDE

Does ATE remain low throughout?

- We show a scenario of merging 3 clients' maps with SLAM-Share

- Need low ATE for accurate virtual object placement





How Fast Does SLAM-Share Merge Maps?

- Baseline: multi-user implementation of Edge-SLAM
- Baseline map transfer from client to Edge server adds latency
- SLAM-Share's use of shared memory lowers overheads
- Merging new map to global map is time consuming
  - SLAM-Share incrementally updates the map

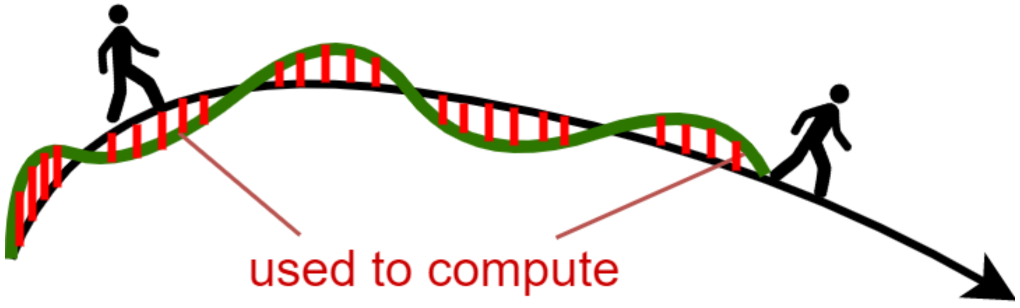
Latency breakdown of map update of SLAM-Share and Baseline when performing one Map-Merge between two maps

Component	Baseline (ms)	SLAM-Share (ms)
Serialization (app)	78.1	N/A
Encoding	N/A	3
Map transfer (to server)	66	0.11
Deserialization (app)	390.8	0
Map Merging	**2339**	**190**
Map transfer (to client)	6.4	0.1
Load Map (in client)	19.8	N/A
Total	**2900.1**	**193.21**

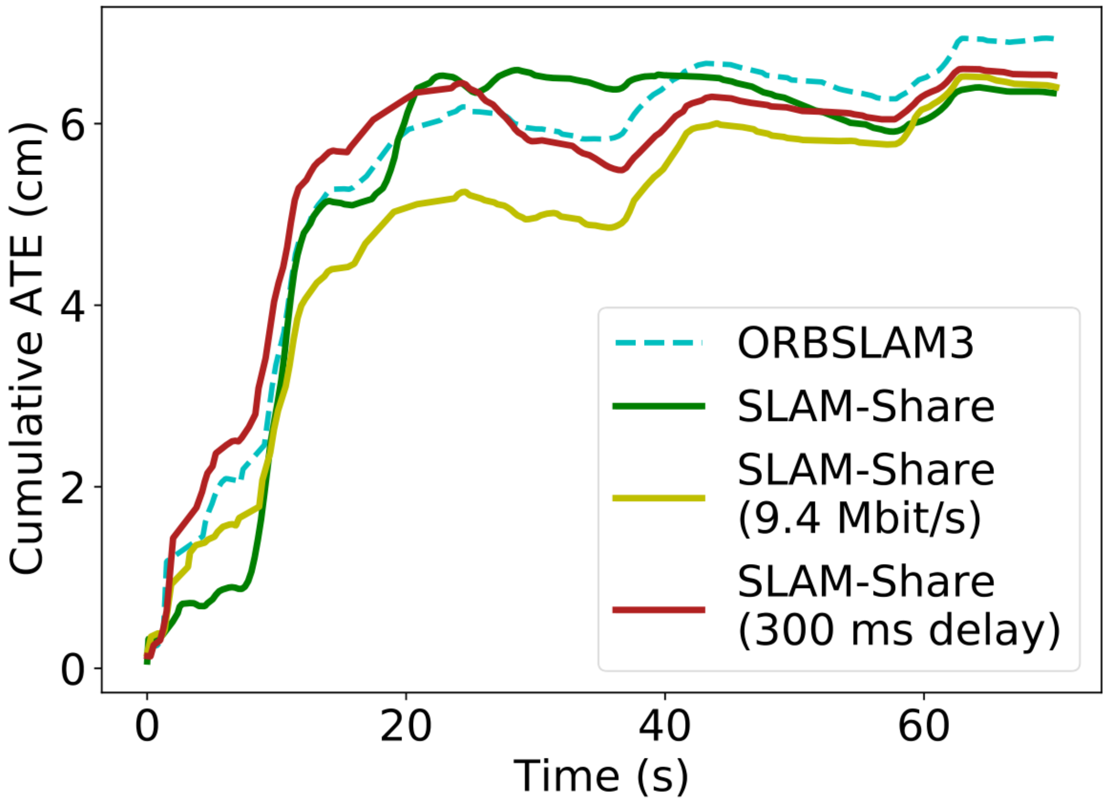
SLAM-Share Map Merge is an order of magnitude faster

Ali AJ, Kouroshli M, Semenova S, Hashemifar ZS, Ko SY, Dantu K. Edge-SLAM: edge-assisted visual simultaneous localization and mapping. ACM Transactions on Embedded Computing Systems. 2022 Oct 29;22(1):1-31.

Is Multi-User SLAM-Share as good as Single User ORB-SLAM3?



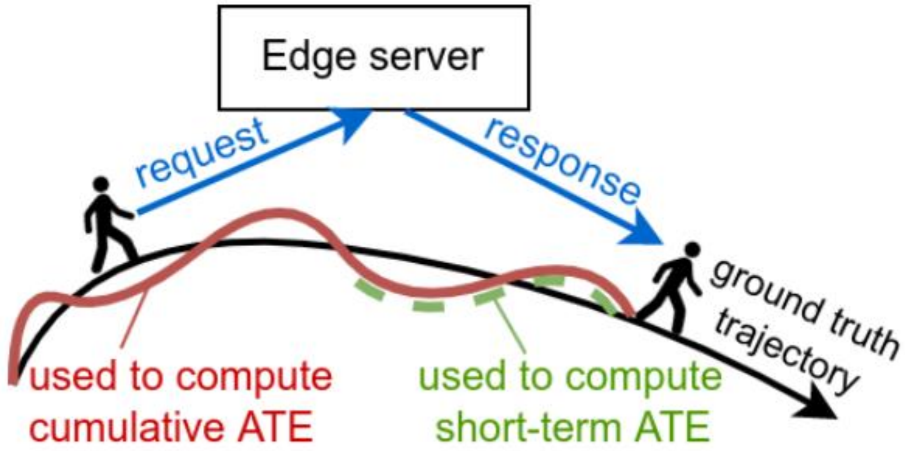
- Evaluation
  - ATE of map created by SLAM-Share with 9.4 Mbit/second bandwidth between client and Edge
  - ATE of the map created by SLAM-Share with 300ms delay added for each packet

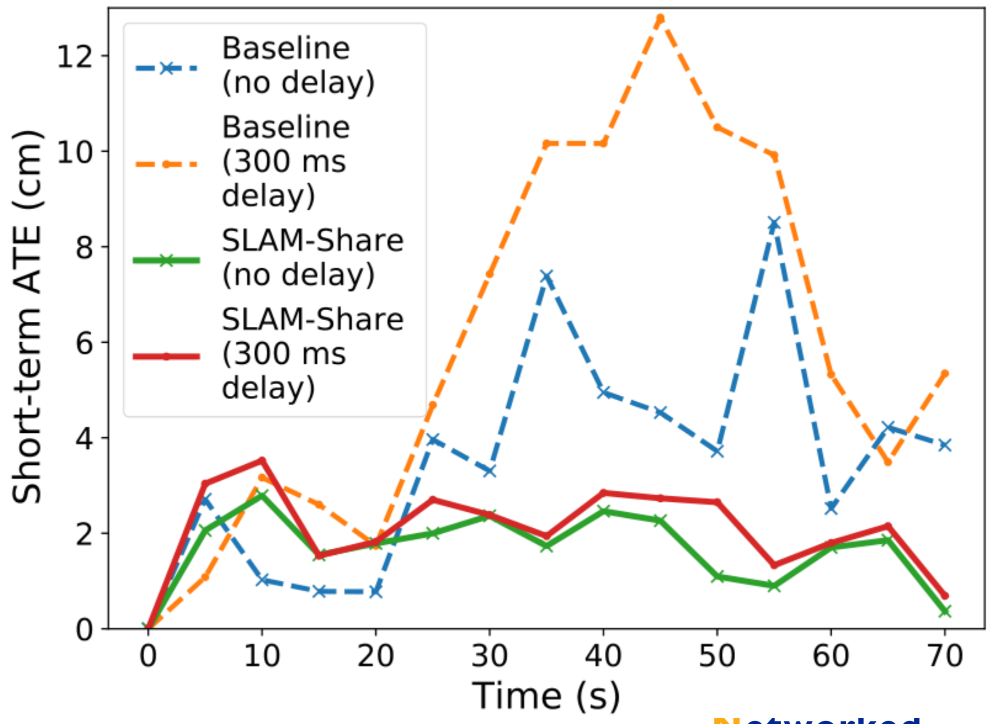


→ SLAM-Share multi-user maps are as accurate as those of single-user ORB-SLAM3

Is SLAM-Share Accurate When There is Network Delay?

- Comparisons
  - SLAM-Share and baseline
  - With and without added delay





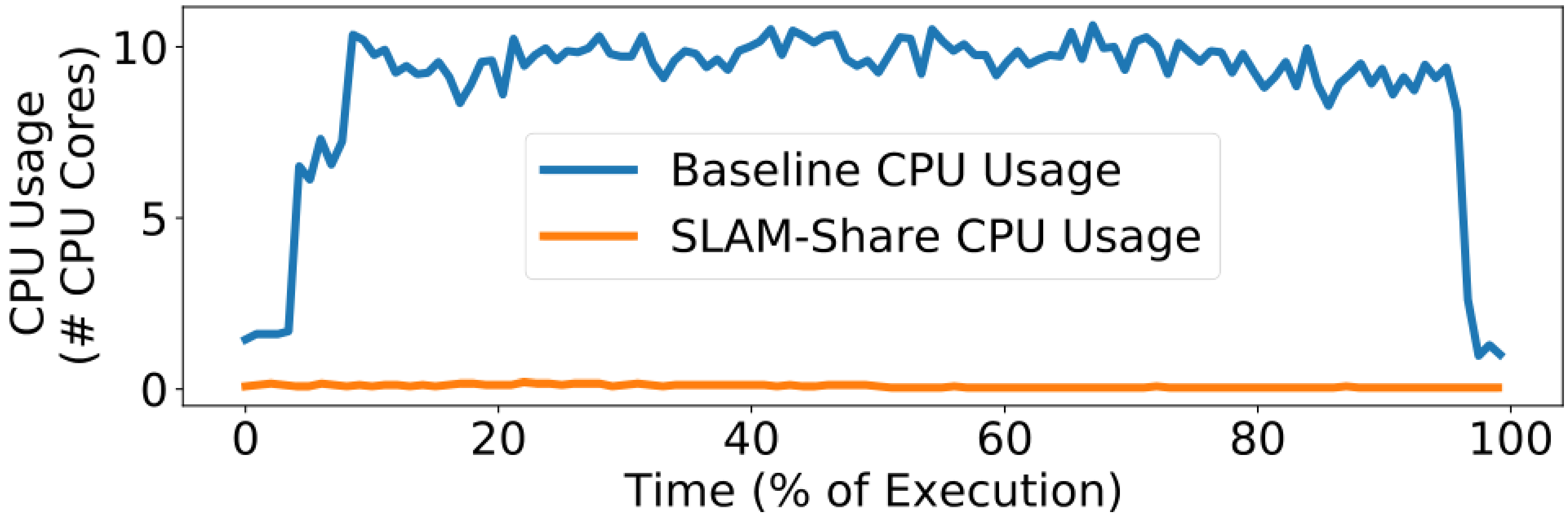
→ SLAM-Share has almost same accuracy despite 300 ms delay

→ Baseline suffers from higher short-term inaccuracies with increased delay

Networked Systems Group
UC RIVERSIDE

CPU Overhead of SLAM-Share vs. Baseline Clients

- We evaluated the overall CPU use in SLAM-Share and Baseline clients



- SLAM-Share uses less than 1% of single CPU Core

Conclusion

- SLAM-Share improves key components of Visual SLAM: tracking and mapping
  - Intelligently re-thinks partitioning of SLAM tasks between mobile client and the Edge Cloud
- SLAM-Share exploits GPU-based tracking on the edge cloud
  - Speed up of tracking by more than 40%
- SLAM-Share uses shared-memory on edge cloud to rapidly merge client maps
  - SLAM-Share's Map Merging is an order of magnitude faster
- SLAM-Share achieves high-throughput **multi-user** visual SLAM-Share
  - Very resource/power efficient on client - very small CPU and memory consumption
- Open-source code available: https://github.com/network-lab2/slam-share