DRONES LAB
Distributed Robotics and Networked Embedded Systems

University at Buffalo
The State University of New York

Edge-SLAM: Edge-Assisted Visual Simultaneous Localization and Mapping

Ali J. Ben Ali, Zakieh Sadat Hashemifar, Karthik Dantu

Distributed RObotics and Networked Embedded Sensing (DRONES) LAB

https://droneslab.github.io

ACM MobiSys 2020

University at Buffalo
Department of Computer Science and Engineering
School of Engineering and Applied Sciences

Introduction

- Modern mobile devices
  - Enhanced computing
  - Multiple interaction modalities
  - Multiple connectivity modalities
  - Lots of sensing
- Advanced sensing capabilities enabled richer set of applications
  - Digital manufacturing
  - Mobile interactive games
  - Service robots
  - Collaborative meeting
- Such applications need spatial sensing
  - Place recognition
  - Localization
  - Path estimation

https://www.augmented-minds.com/en/augmented-reality/ar-hardware-devices/

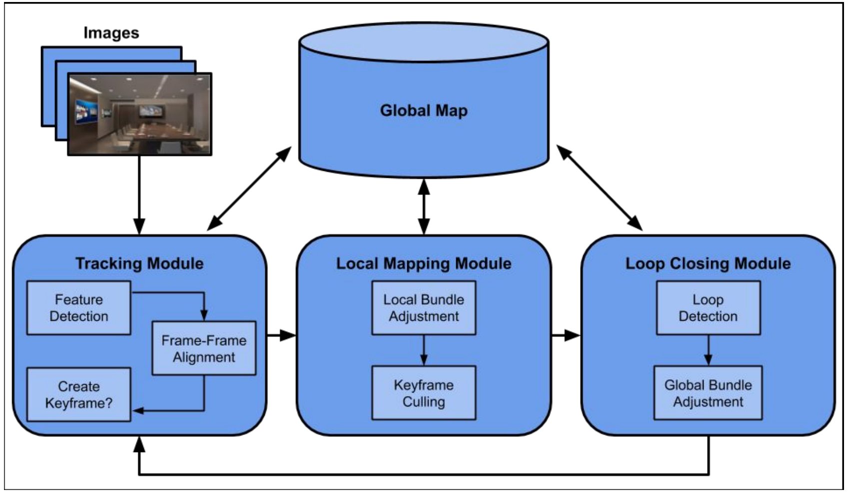
https://www.pcmag.com/news/the-room-service-robots-have-arrived

https://www.microsoft.com/en-us/hololens/buy

University at Buffalo
Department of Computer Science and Engineering
School of Engineering and Applied Sciences

Visual-SLAM Overview (1/2)

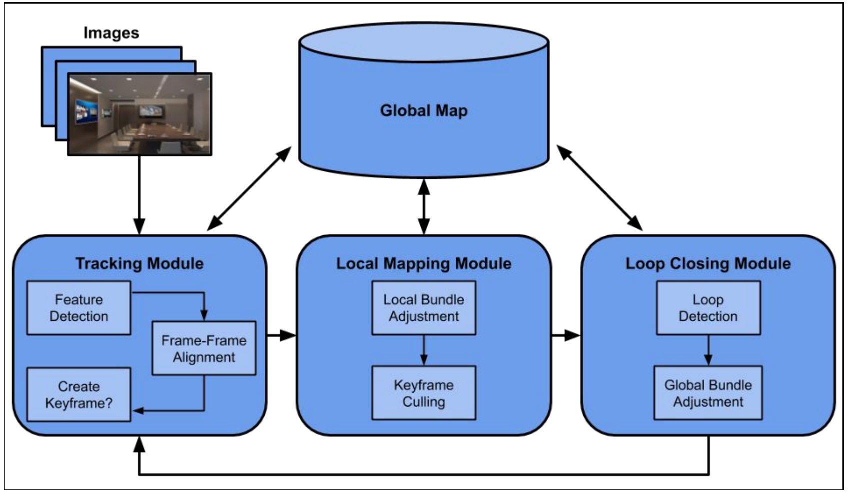
- Visual-SLAM is a spatial sensing algorithm to map the environment and localize the camera with respect to the absolute coordinate system
- Visual-SLAM systems
  - RGBD-SLAM
  - RTAB-Map
  - VINS
  - ORB-SLAM2
- Three main modules
  - Tracking
  - Local mapping
  - Loop closing
- All modules work on a shared global-map



University at Buffalo
Department of Computer Science and Engineering
School of Engineering and Applied Sciences

Visual-SLAM Overview (2/2)

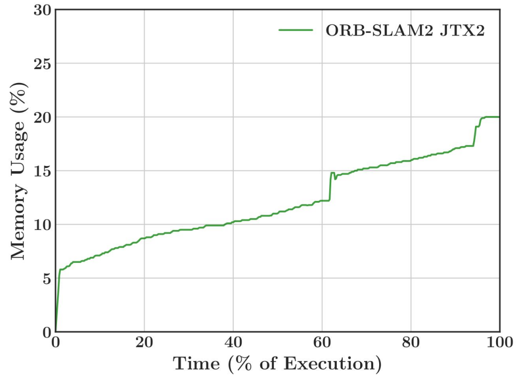
- Tracking module
  - Continuously process incoming images
  - Detect features (SIFT, SURF, ORB)
  - Match features between reference and current frames
  - Create new keyframe
- Local-mapping module
  - Run for every new keyframe
  - Local bundle adjustment
  - Keyframe culling
- Loop-closing module
  - Try to detect loop after every new keyframe
  - Global bundle adjustment

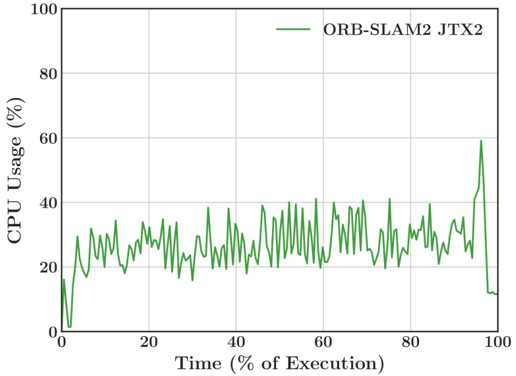


Visual-SLAM on Mobile Devices (1/2)

Challenge: High overhead on mobile device resources

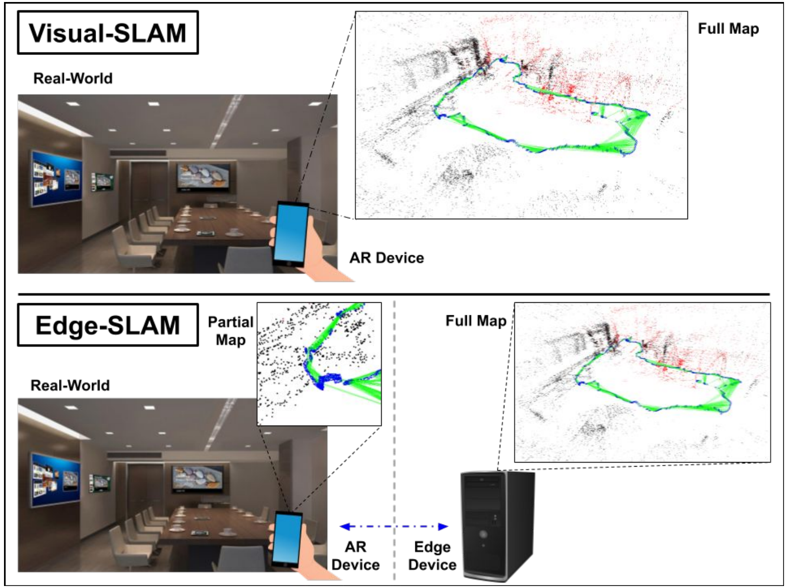
- Running three computational-heavy modules, simultaneously, will limit/prevent
  - Long-term operation
  - Running other applications services
  - Running Visual-SLAM as service to other applications
- When mapping module is active, global-map size grows rapidly
  - Quickly consume the available memory
  - Continuously increase the latency of query, update, and optimize operations





Visual-SLAM on Mobile Devices (2/2)

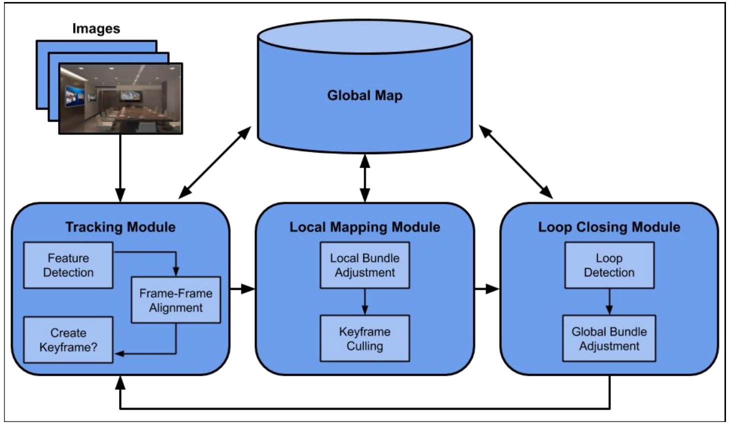
- **Challenge:** High overhead on mobile device resources
- **Idea:** use Edge-Computing architecture
  - Encourages the use of split architecture to deploy applications across mobile-edge
  - Can be as close as one hop away over the local network
  - Has minimal effect on performance and accuracy
  - Challenges?
    - Tight coupling of Visual-SLAM modules



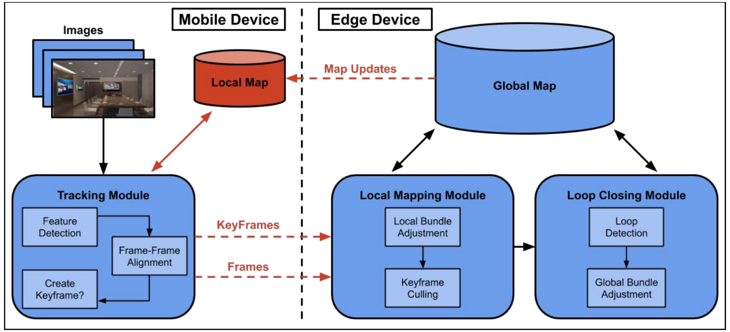
Edge-SLAM Overview (1/2)

- ❖ **Goals**
  - ➢ Reduce usage of mobile device resources
  - ➢ Maintain a low constant rate of resource usage on mobile device
  - ➢ Minimal effect on accuracy
- ❖ **Architecture**
  - ➢ Split modules between mobile device and edge
  - ➢ Use a local-map on mobile device
  - ➢ Maintain the global-map on edge
  - ➢ Introduce two-way communication between tracking and local-mapping modules to share updates

Visual-SLAM

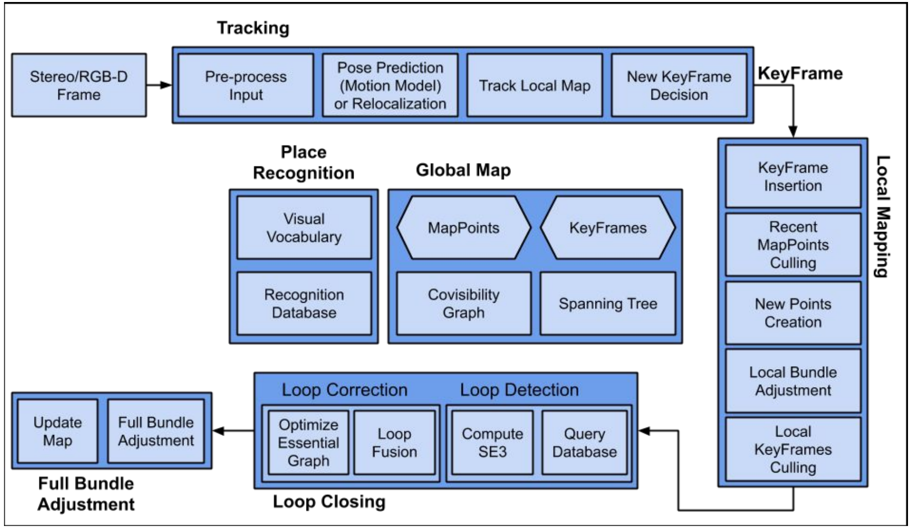


Edge-SLAM



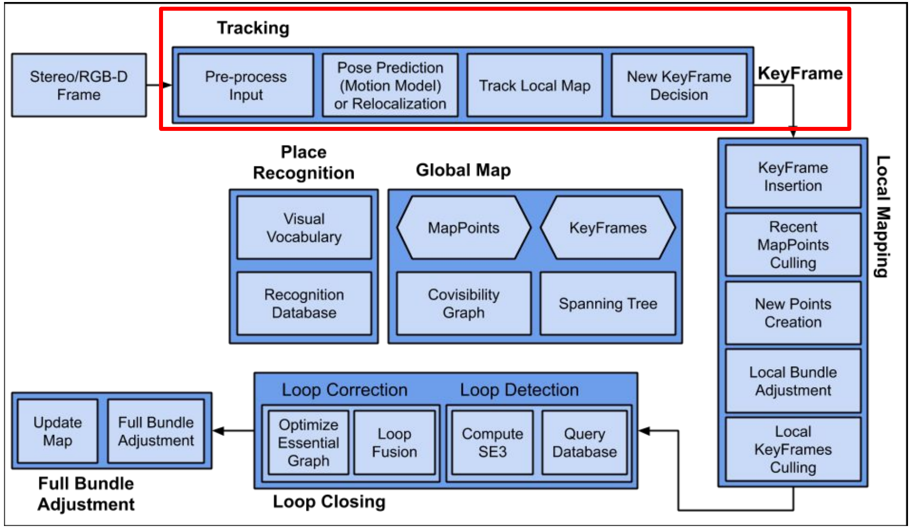
Edge-SLAM Overview (2/2)

- **Implementation**
  - Use **ORB-SLAM2** as a prototype
    - State-of-the-art Visual-SLAM system for Monocular, RGB-D, and Stereo cameras
    - Open-source system
    - 20 classes and 18,000 LOC
    - Three threads (one per module) run simultaneously
    - Fourth thread runs on-demand for full bundle adjustment (FBA)
    - All threads work on shared global-map structure that is managed through a complex locking mechanism which increases the coupling level of the threads



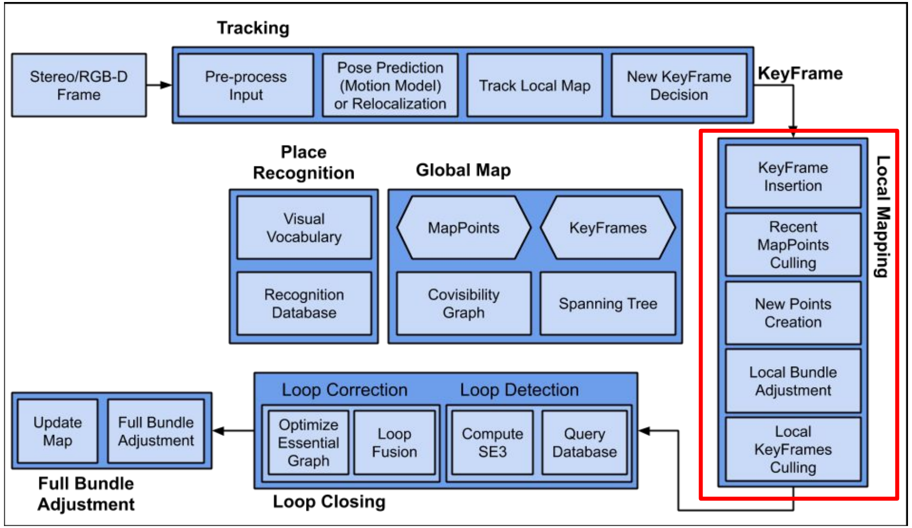
ORB-SLAM2: Tracking Thread

- Extract ORB features
- Use features to estimate and optimize camera pose
  - If tracking is lost, query the recognition database for keyframe candidates for global relocalization
- Track local map
  - Project the map into the frame and search more map-point correspondences
- New keyframe decision
  - Decide whether a frame should be added to map as keyframe or not by evaluating 5 conditions



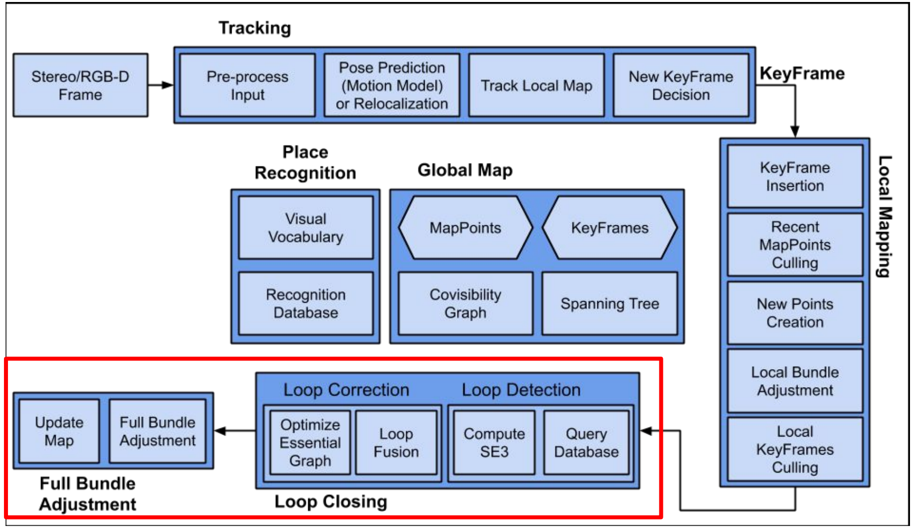
ORB-SLAM2: Local-Mapping Thread

- Insert new keyframe into global-map
- Recent map-points culling
  - Ensure that map-points are trackable and not wrongly triangulated
- New map-point creation
  - Create new map-points by triangulating ORB from connected keyframes
- Local bundle adjustment
  - Optimize current keyframe, all connected keyframes, and all map-points seen by those keyframes
- Local keyframes culling
  - Detect and delete redundant keyframes



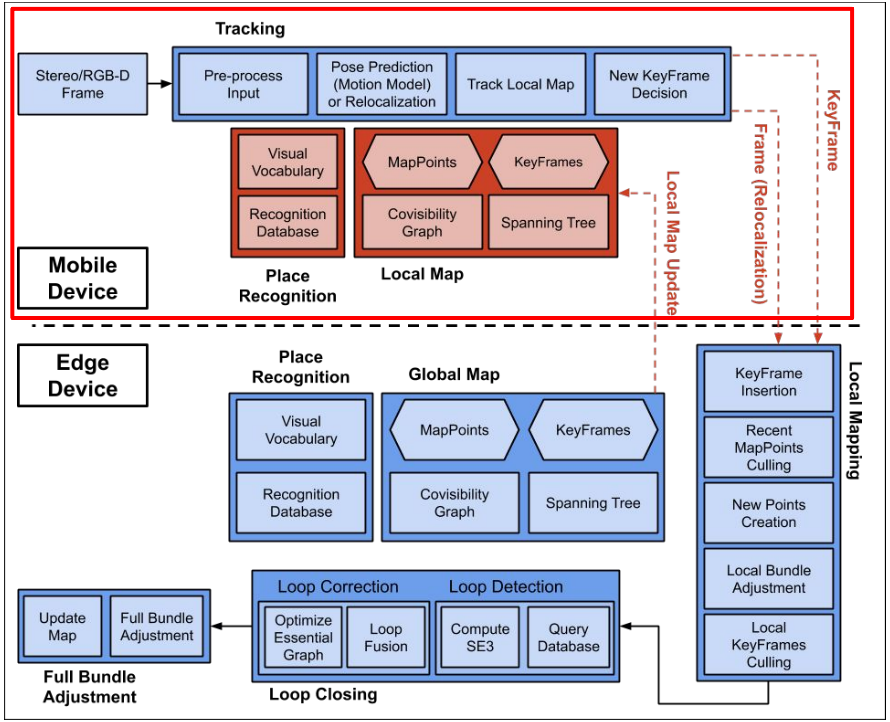
ORB-SLAM2: Loop-Closing & Bundle Adjustment

- Loop detection
  - Detects loop candidates
  - Compute similarity transformation
    - Ensure the loop is geometrically valid
- Loop correction
  - Loop fusion
    - Fuse duplicate map-points
    - Insert new edges to attach the loop closure
  - Optimize essential graph
    - Perform a pose graph optimization to distribute the loop-closing error along the graph
- Full bundle adjustment (FBA)
  - Runs on separate thread in ORB-SLAM2
  - Optimizes the global-map



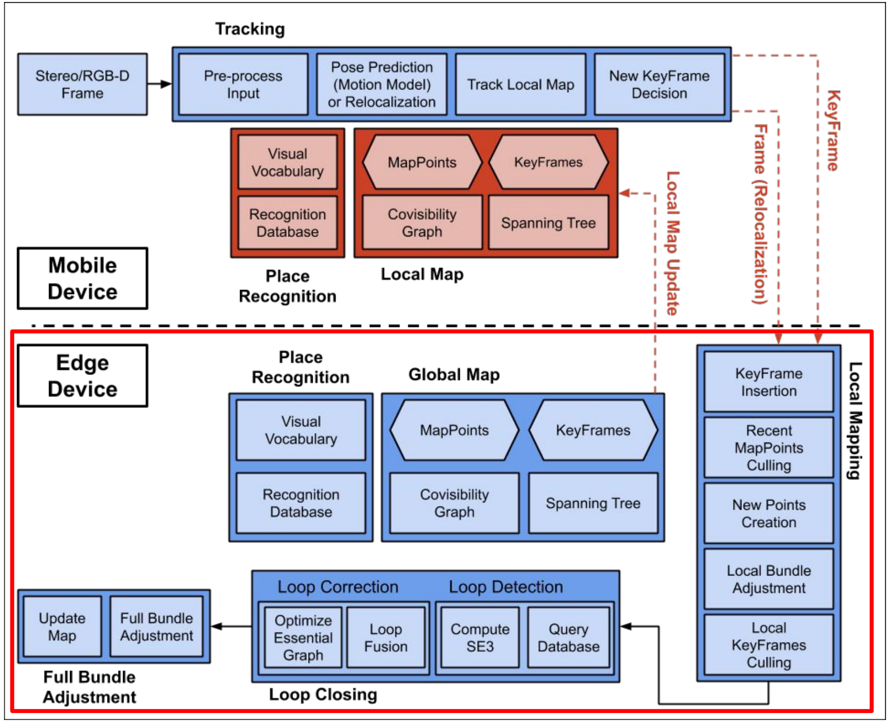
Edge-SLAM: Mobile Device Operation

- Maintain a local-map
- Run tracking thread
  - New keyframe decision
    - Process partially (cond. 3-5)
    - If created, send to edge immediately
  - Fully replace local-map update received from edge with current local-map
  - If tracking lost, then
    - Try relocalization using current local-map
    - Send a frame every 0.5s to edge to receive a relocalization-specific local-map update



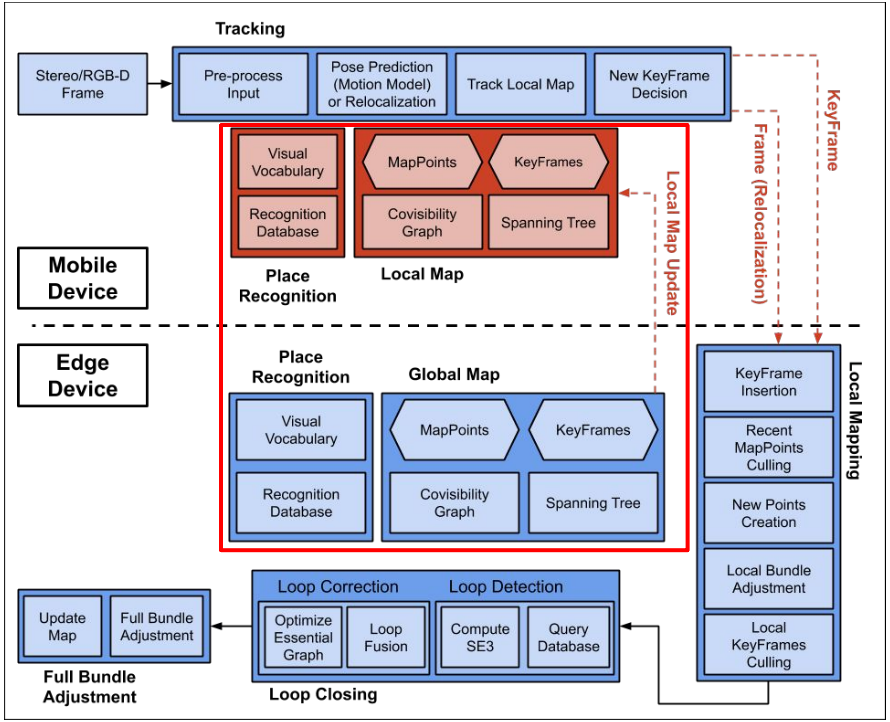
Edge-SLAM: Edge Operation

- Maintain the global-map
- Run local-mapping thread
  - New keyframe decision
    - Finalize decision (cond. 1, 2)
  - Added new module: local-map update
- Run loop-closing thread
- Run full bundle adjustment thread
- Communicate with mobile device through asynchronous TCP connections
  - All connections are initiated between the tracking thread on mobile device and the local-mapping thread on the edge



Edge-SLAM: Local-Map Update

- **Objectives**
  - Minimize mobile device drift
  - Minimize local-map reconstruction overhead on mobile device
  - Limit network usage
- **Operation**
  - Run as part of local-mapping thread on edge
  - Timer-based updates
    - If global-map has changed, send a local-map update every 5s
  - A local-map update consists of the most recent 6 keyframes along with their map-points



Edge-SLAM: Experiment Setup (1/3)

- **Two distinct mobile devices**
  - NVIDIA JETSON-TX2
    - 64-bit NVIDIA Denver and ARM Cortex-A57 CPUs
    - NVIDIA Pascal GPU with 256 CUDA-cores
    - 8 GB Memory
    - Comparable to Magic Leap One
  - Dell Latitude laptop
    - Intel Core i5-520M (2.4GHz, 3M cache, Dual-Core)
    - Intel HD Graphics with dynamic frequency
    - 8 GB Memory
    - Loosely comparable to Microsoft Hololens

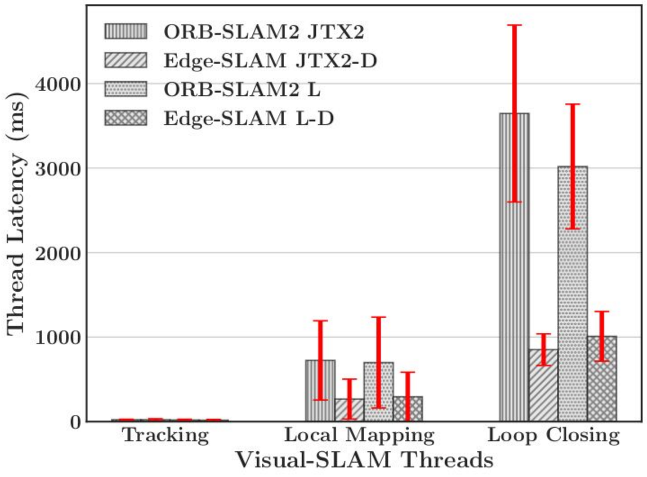
Edge-SLAM: Experiment Setup (2/3)

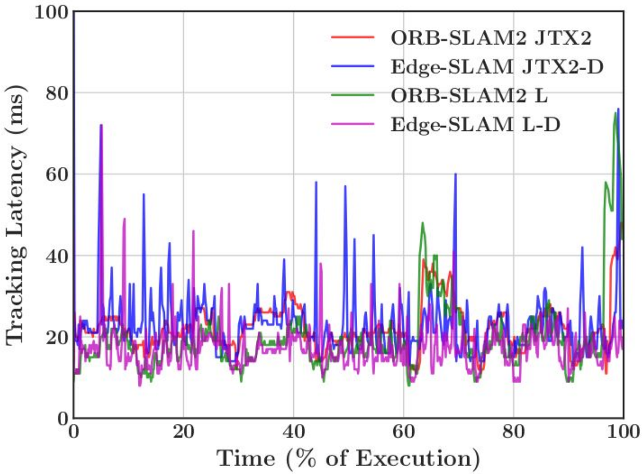
- **Edge machine**
  - Dell XPS desktop
    - Intel Core i7 9700K (8-Core/8-Thread, 12MB Cache, Overclocked up to 4.6GHz)
    - NVIDIA GeForce GTX 1080
    - 32 GB Memory
- **Network**
  - JETSON-TX2 connected to lab private Wi-Fi network
  - Dell laptop connected to campus public Wi-Fi network
  - Dell desktop connected to campus wired network

Edge-SLAM: Experiment Setup (3/3)

- **Datasets**
  - Pre-collected RGB-D dataset of campus building floor (52,427 frames, 1,774 seconds) for long-run experiments
  - TUM RGB-D dataset for short-run experiments
- **Experiments**
  - Run ORB-SLAM2 on JETSON-TX2 (**ORB-SLAM2 JTX2**)
  - Run ORB-SLAM2 on Dell laptop (**ORB-SLAM2 L**)
  - Run Edge-SLAM on JETSON-TX2 and Dell desktop (**Edge-SLAM JTX2-D**)
  - Run Edge-SLAM on Dell laptop and Dell desktop (**Edge-SLAM L-D**)

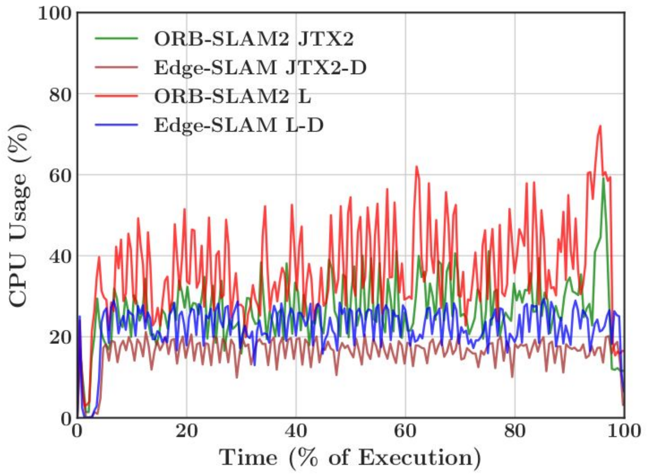
Edge-SLAM Evaluation: Threads Latencies

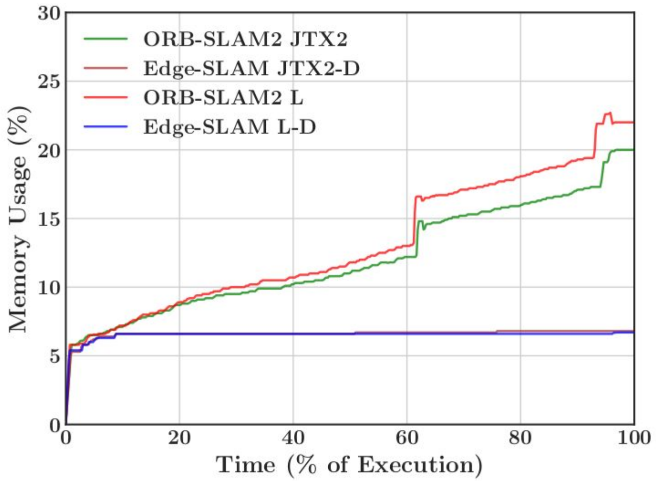




- ❖ Overall latency of ORB-SLAM2 and Edge-SLAM. The average latency per-module shows that Edge-SLAM offloads the two CPU-intensive tasks
- ❖ Tracking module latency on the mobile device over time better shows the latency for that module in each configuration

Edge-SLAM Evaluation: Resource Usage





- ❖ CPU usage of ORB-SLAM2 and Edge-SLAM on the mobile device
- ❖ Memory usage of ORB-SLAM2 and Edge-SLAM on the mobile device. The jumps in memory use at 60% time and 95% time are due to loop closures in ORB-SLAM2

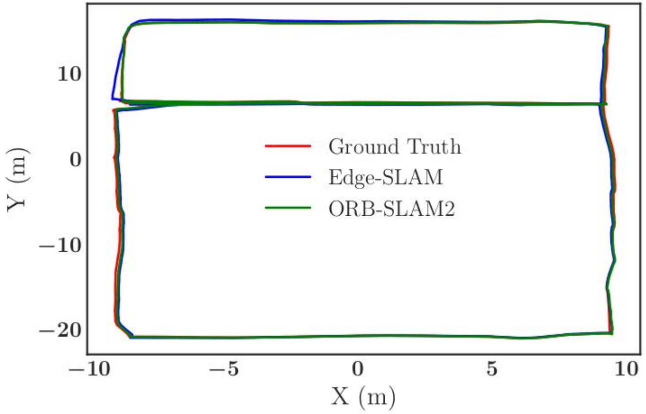
Edge-SLAM Evaluation: Local-Map Update Latency

Map Update \ Edge-SLAM	Edge-SLAM JTX2-D (ms)	Edge-SLAM L-D (ms)
Construct Map Update on Edge	57.09 ±0.69	58.30 ±0.66
Re-Construct Map Update on Mobile Device	411.43 ±4.84	285.68 ±3.18

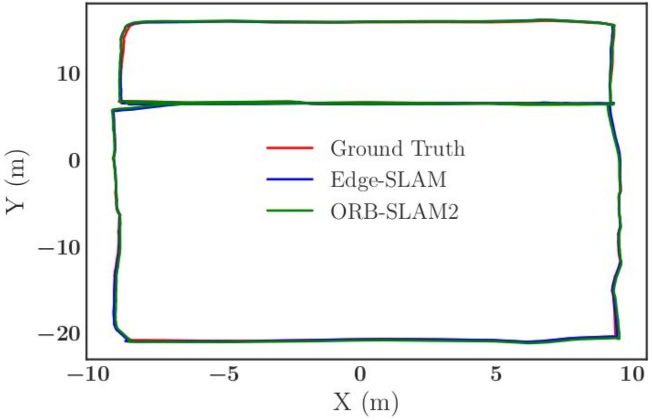
Keyframe Update \ Edge-SLAM	Edge-SLAM JTX2-D (ms)	Edge-SLAM L-D (ms)
Transmit Keyframe from Mobile Device to Edge	162.43 ±2.90	142.53 ±6.38

Edge-SLAM Evaluation: Mapping Accuracy

Accuracy Measure \ Visual-SLAM	ORB-SLAM2 JTX2	Edge-SLAM JTX2-D	ORB-SLAM2 L	Edge-SLAM L-D
Mean Localization Error (cm)	20.59 ±10.92	19.23 ±11.32	20.90 ±12.77	21.39 ±9.16



- ORB-SLAM2 and Edge-SLAM trajectories running on JETSON-TX2 compared to the ground-truth



- ORB-SLAM2 and Edge-SLAM trajectories running on laptop compared to the ground-truth

Summary

- ❖ Edge-SLAM adapts edge computing architecture into Visual-SLAM on mobile devices with minimal loss of accuracy
- ❖ Edge-SLAM offloads the computation-intensive modules of Visual-SLAM to the edge.
- ❖ Edge-SLAM reduces resources used on the mobile device
- ❖ Edge-SLAM enables long operation of Visual-SLAM on mobile devices by maintaining a low constant rate of used resources
- ❖ We open-source our Edge-SLAM implementation which can be found at http://droneslab.github.io/edgeslam