2018 VLSI TECHNOLOGY SYMPOSIUM HONOLULU

2018 VLSI CIRCUITS SYMPOSIUM HONOLULU

Navion: A Fully Integrated Energy-Efficient Visual-Inertial Odometry Accelerator for Autonomous Navigation of Nano Drones

Amr Suleiman, Zhengdong Zhang, Luca Carlone, Sertac Karaman, and Vivienne Sze

Massachusetts Institute of Technology

http://navion.mit.edu/

Motivation: Autonomous Navigation

Self Driving Cars

UAVs: Unmanned Aerial Vehicles

Robots

[images] Electrek, Amazon, Knightscope, Boston Dynamics

How Does Autonomous Navigation Work?

Perception

Motion Planning

Control

How Does Autonomous Navigation Work?

Perception

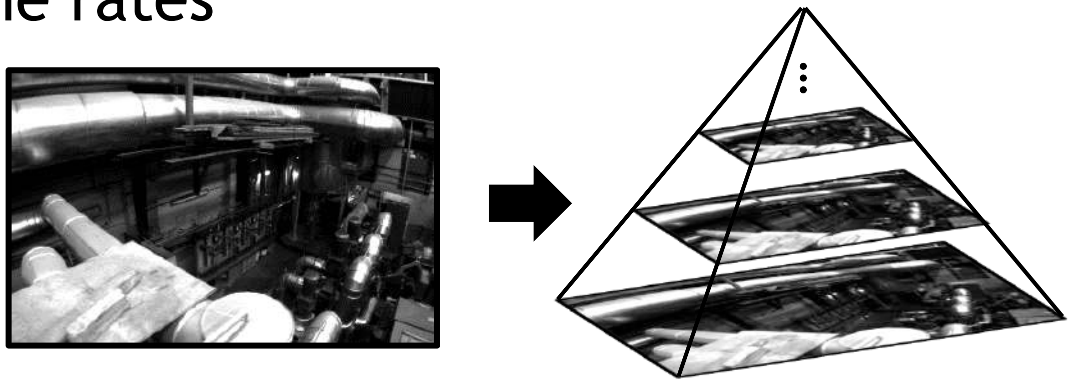
Motion Planning

Control

Perception is the computation bottleneck

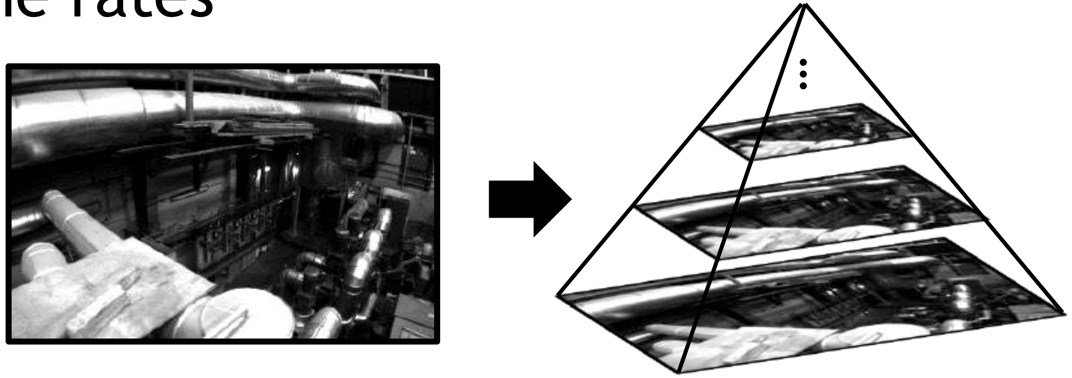
Challenges: High Dimensionality

- **Large amount of data**
  - Sensors data: High resolution & frame rates
  - Data expansion: Image pyramid



Challenges: High Dimensionality

- **Large amount of data**
  - Sensors data: High resolution & frame rates
  - Data expansion: Image pyramid



- **Growing map size**

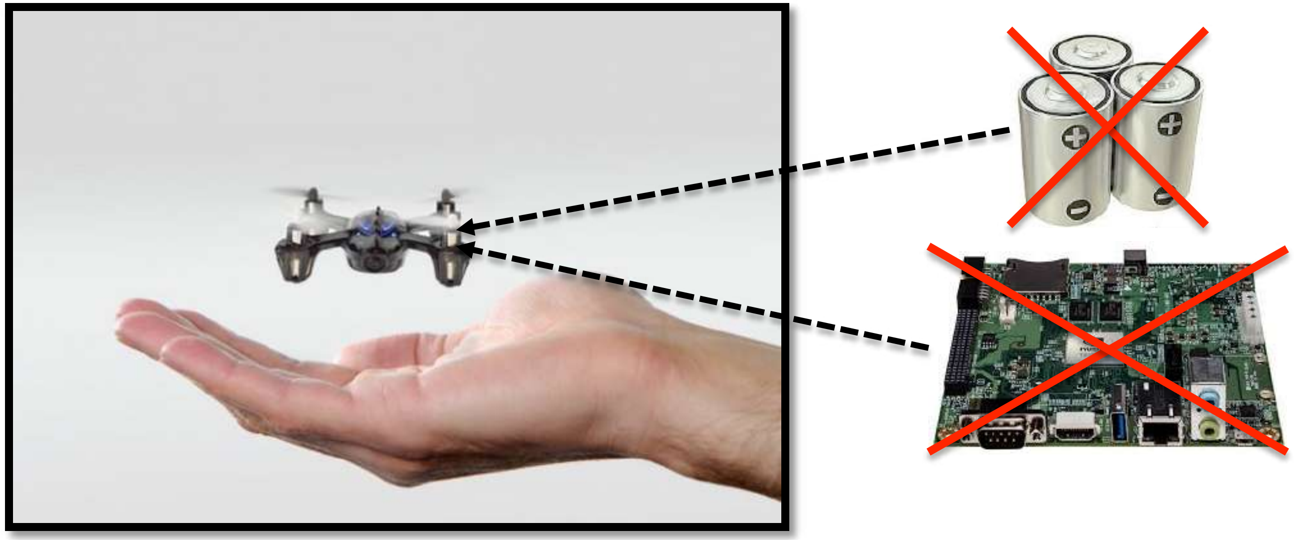
[T. Pire et al., 2017]

Challenges: Low Power Budget

Big battery

Mobile CPU, GPU

Challenges: Low Power Budget



Big battery

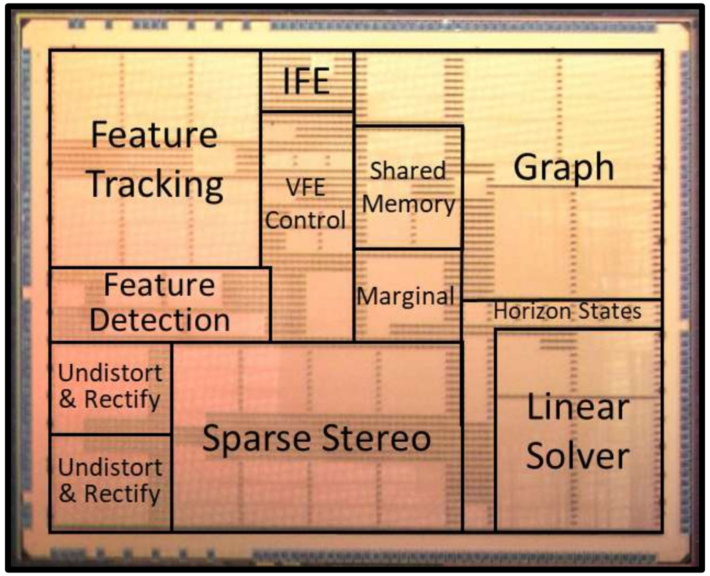
Mobile CPU, GPU

For example:

Insect-scale UAV (100mg)

Lifting	Cameras	CPU, GPU
100 mW	100 mW	10 – 100 W

Navion: Energy-Efficient Visual-Inertial Odometry



- Energy-efficient & real-time localization and mapping
- Process stereo images at up to 171 fps
- 24 mW average power consumption

Outline

- Localization & Mapping: Visual-Inertial Odometry (VIO)
- Chip Architecture
- Main Contributions
- Chip Specifications and Comparisons
- Summary

Localization and Mapping Using VIO

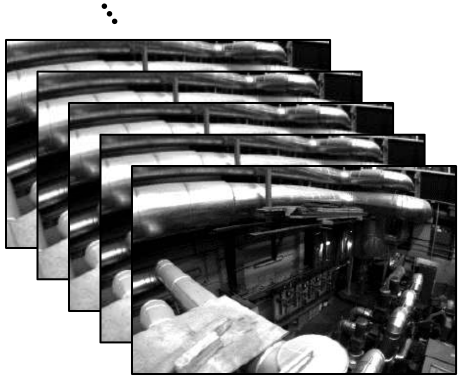
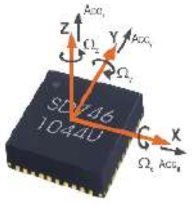


Image sequence

IMU

Inertial Measurement Unit



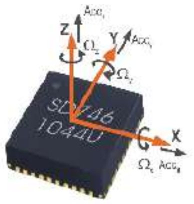
Visual-Inertial Odometry (VIO)

Localization and Mapping Using VIO

Image sequence

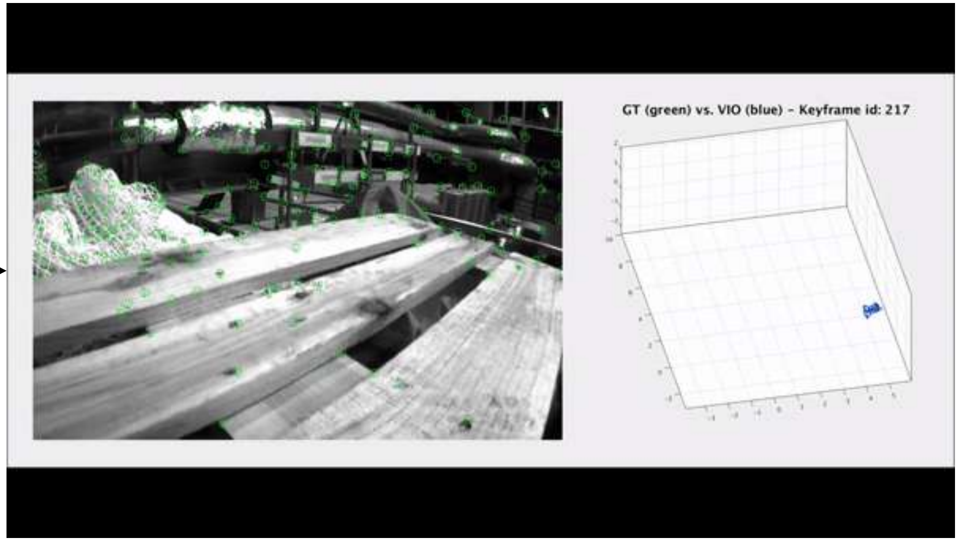
IMU

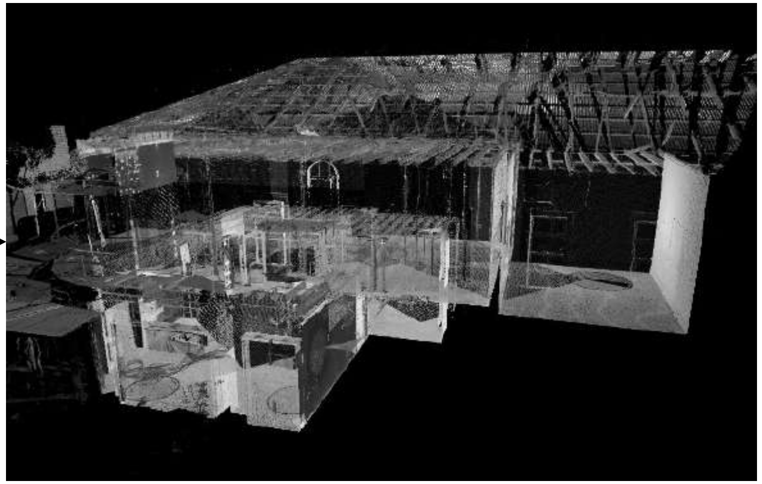
Inertial Measurement Unit



Visual-Inertial Odometry (VIO)

Localization





Mapping

Localization and Mapping Using VIO

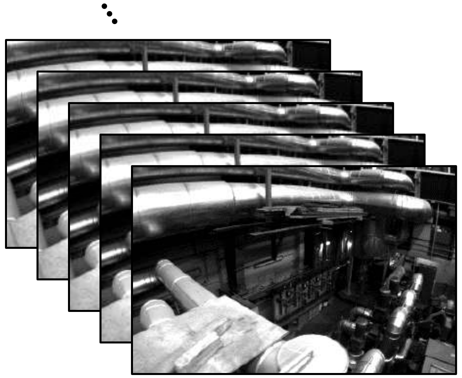
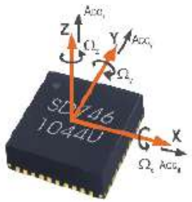


Image sequence

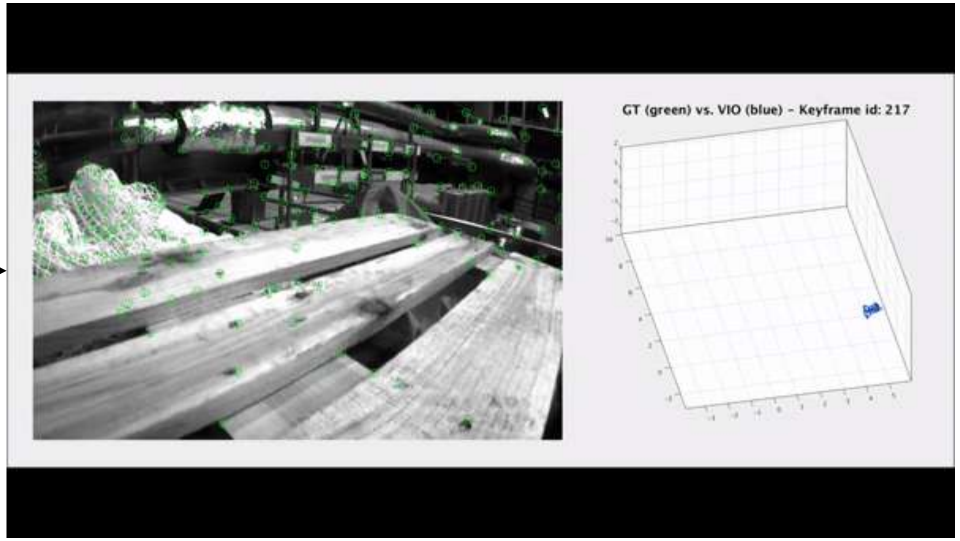
IMU

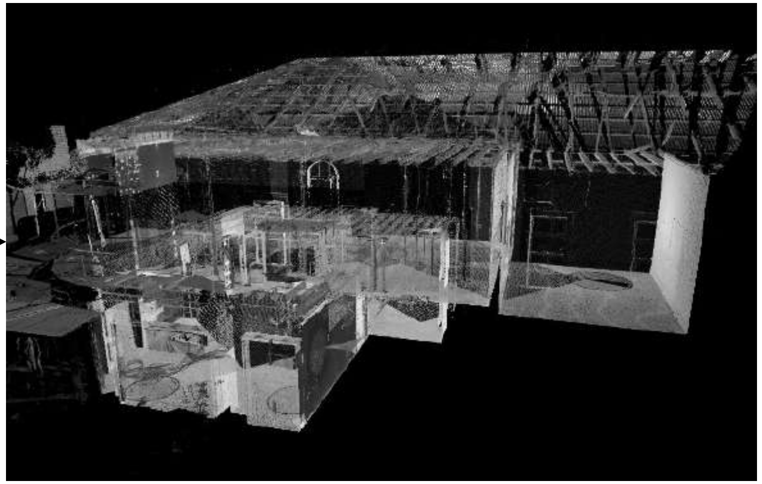
Inertial Measurement Unit



Visual-Inertial Odometry (VIO)

Localization





Mapping

Subset of SLAM algorithms

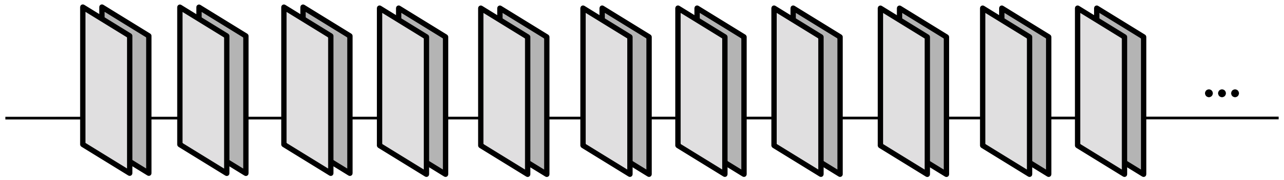
(Simultaneous Localization And Mapping)

VIO: Frontend

Camera 

↓

Vision Frontend (VFE)



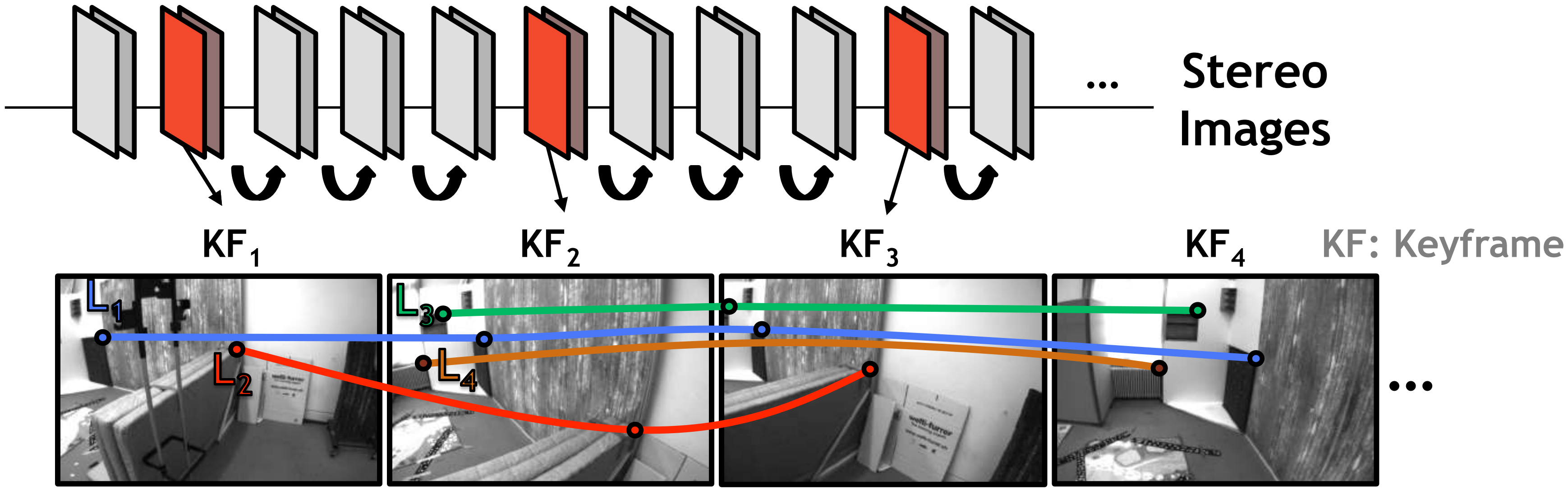
Stereo Images

Process mono/stereo Images

VIO: Frontend

Camera

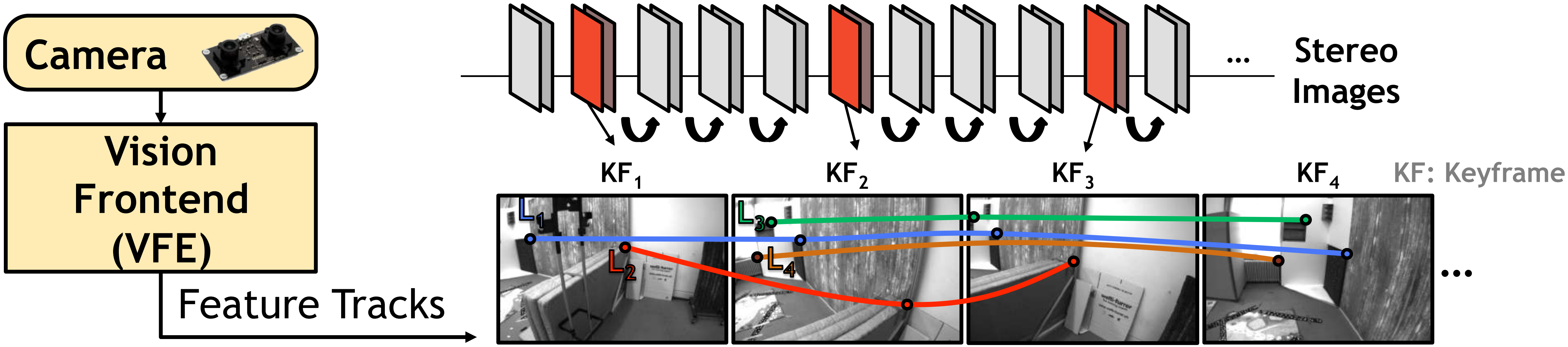
Vision Frontend (VFE)



<u>Process mono/stereo Images</u>

- Detect & track features (L_i)

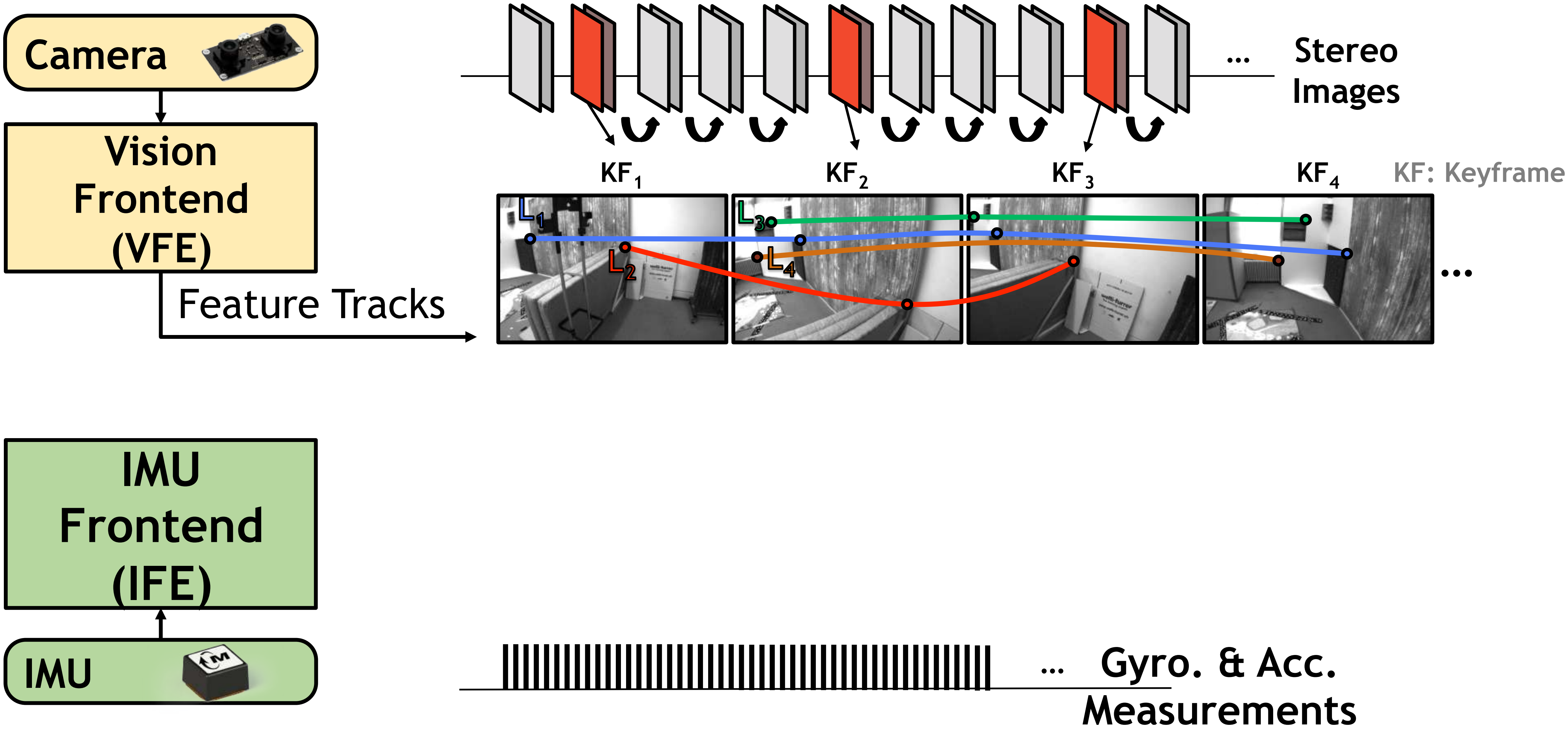
VIO: Frontend



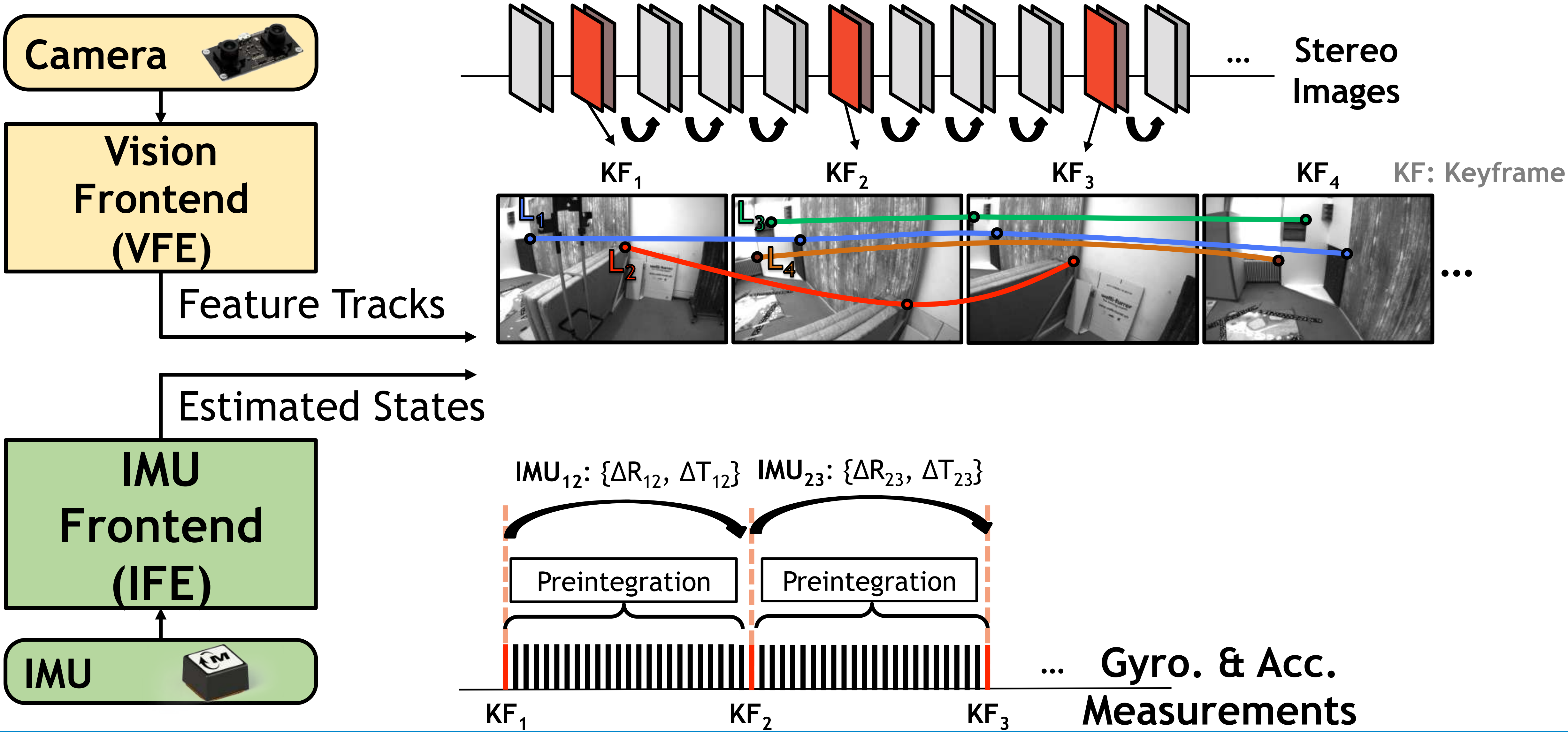
<u>Process mono/stereo Images</u>

- Detect & track features (L_i)
- Generate ***Feature Tracks*** -> (keyframe IDs & feature coordinates)

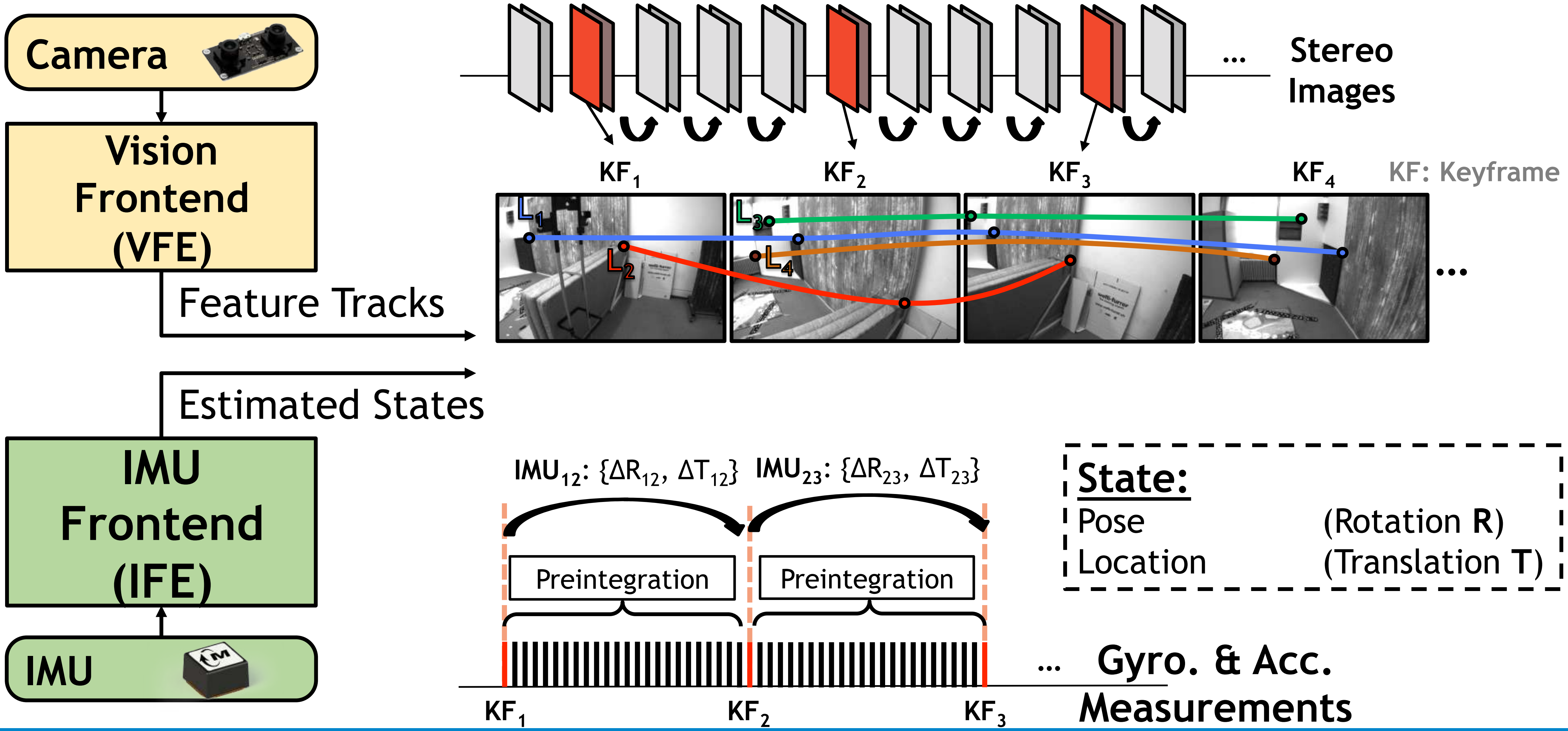
VIO: Frontend



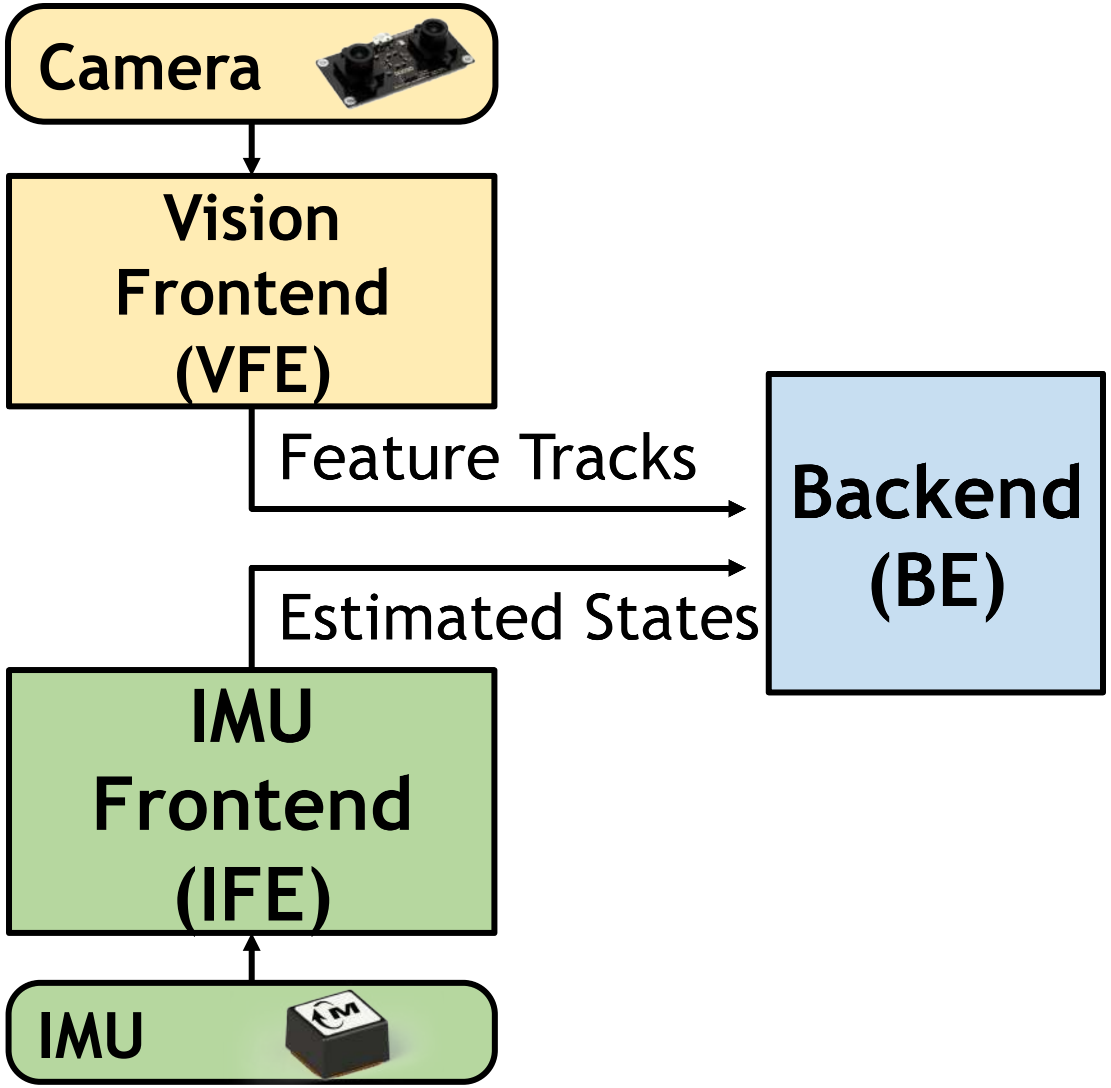
VIO: Frontend



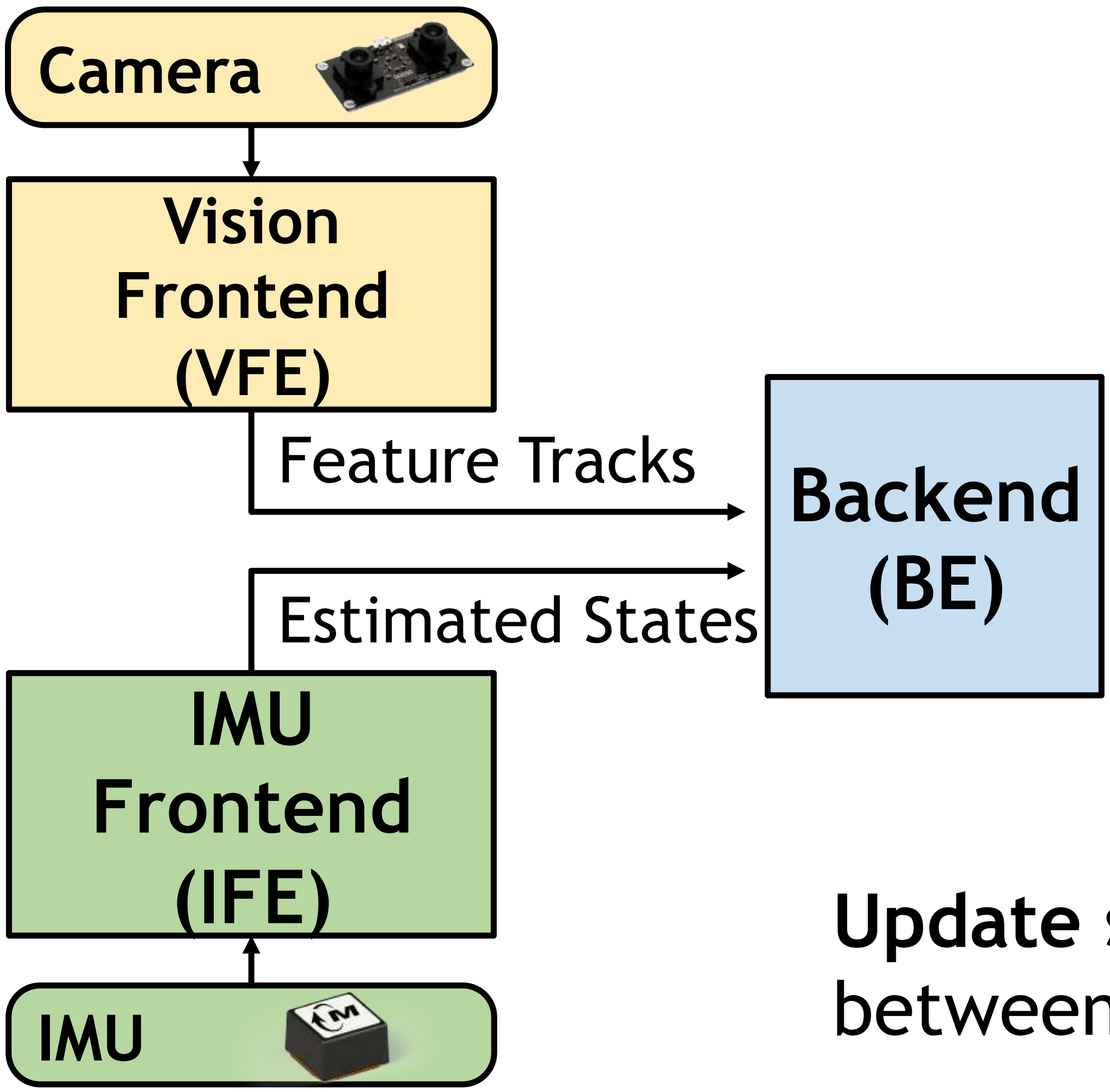
VIO: Frontend

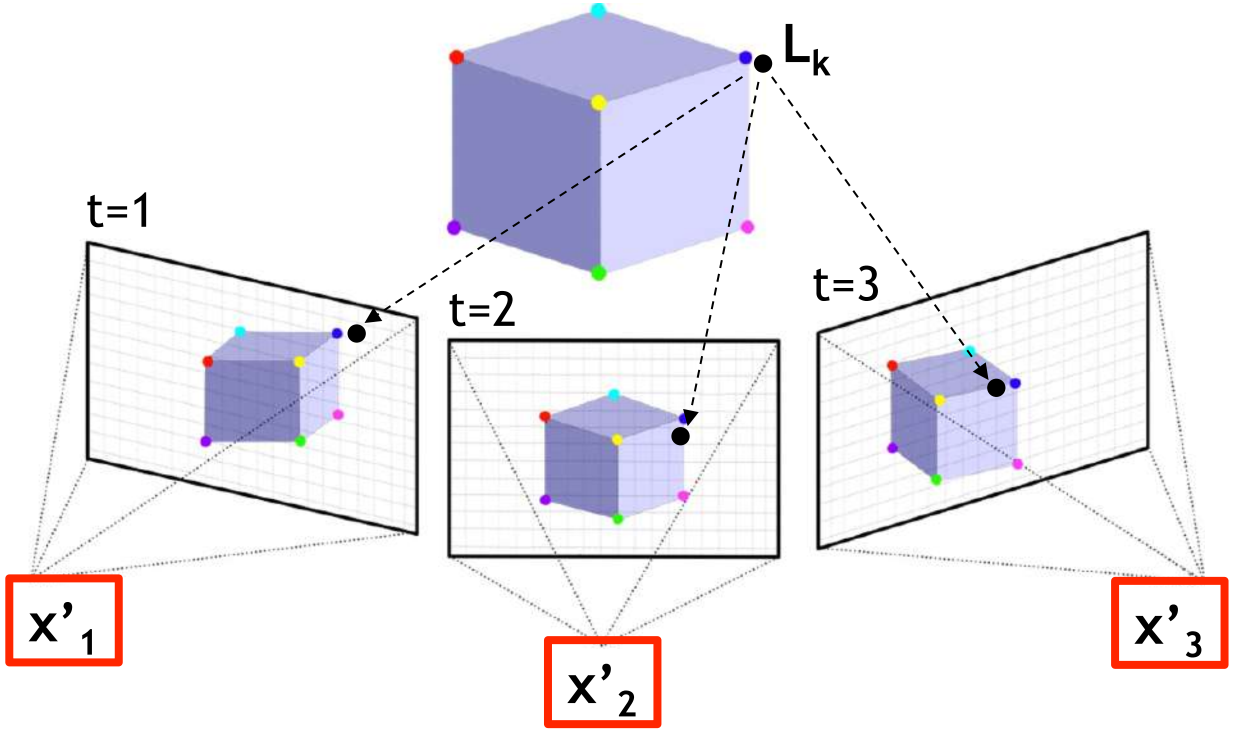


VIO: Backend


Camera
Vision Frontend (VFE)
Feature Tracks
Backend (BE)
Estimated States
IMU Frontend (IFE)
IMU

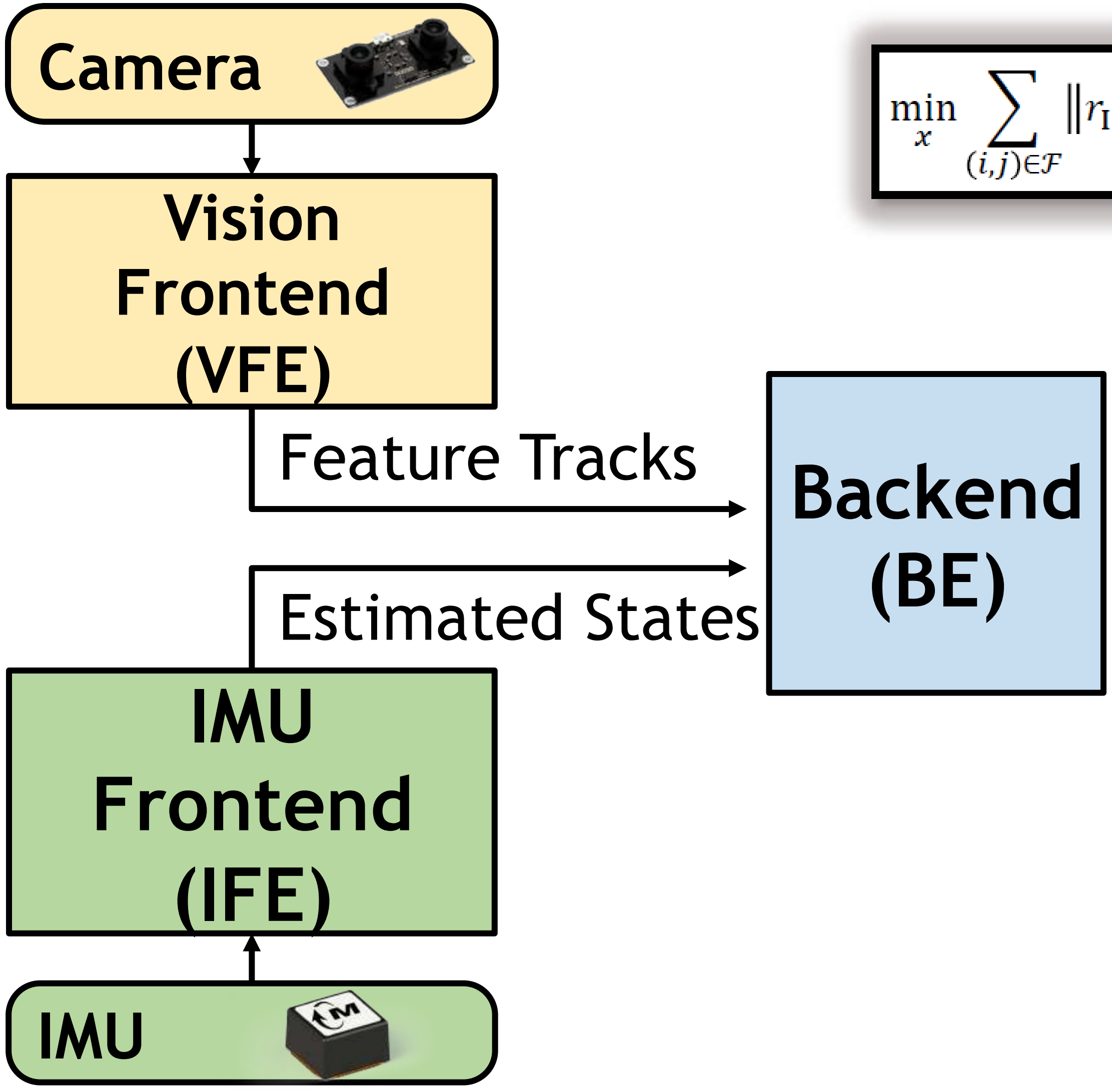
VIO: Backend





Update states (x_i) to minimize inconsistencies between measurements across time

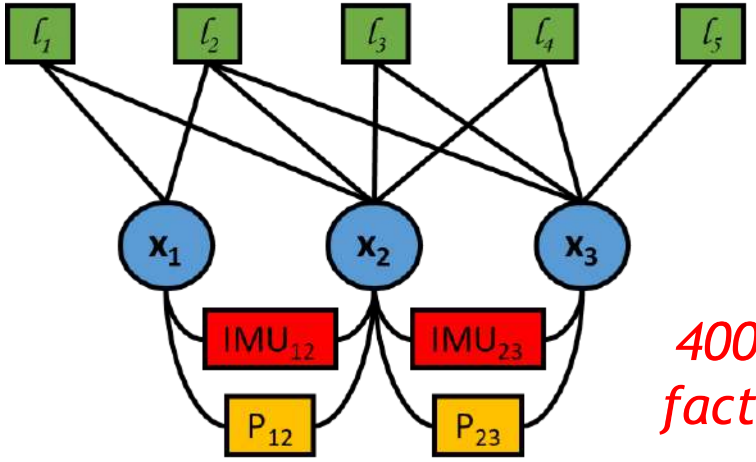
VIO: Backend



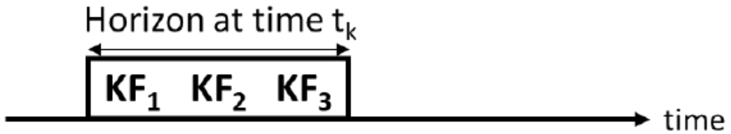
$$\min_{x} \sum_{(i,j)\in\mathcal{F}} \left\|r_{\mathrm{IMU}}\left(x,\Delta\tilde{R}_{ij},\Delta\tilde{p}_{ij},\Delta\tilde{v}_{ij}\right)\right\|^2 + \sum_{k\in\mathcal{L}}\sum_{i\in\mathcal{F}_k}\left\|r_{\mathrm{CAM}}\left(x,l_k,u^l_{ik},u^r_{ik}\right)\right\|^2 + \left\|r_{\mathrm{PRIOR}}(x)\right\|^2$$

IMU Factors *Vision Factors* *Other Factors*

Factor Graph



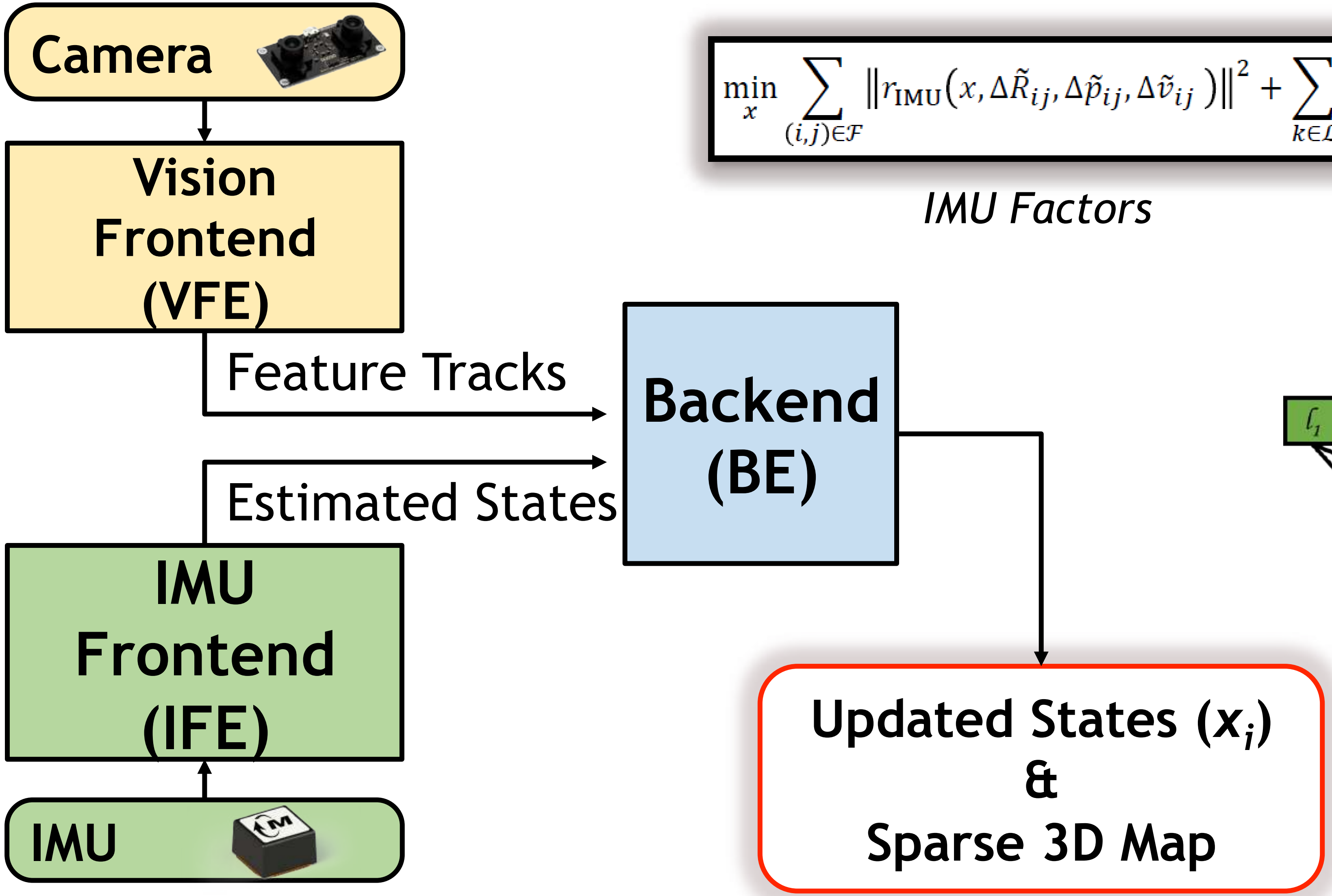
4000+ factors



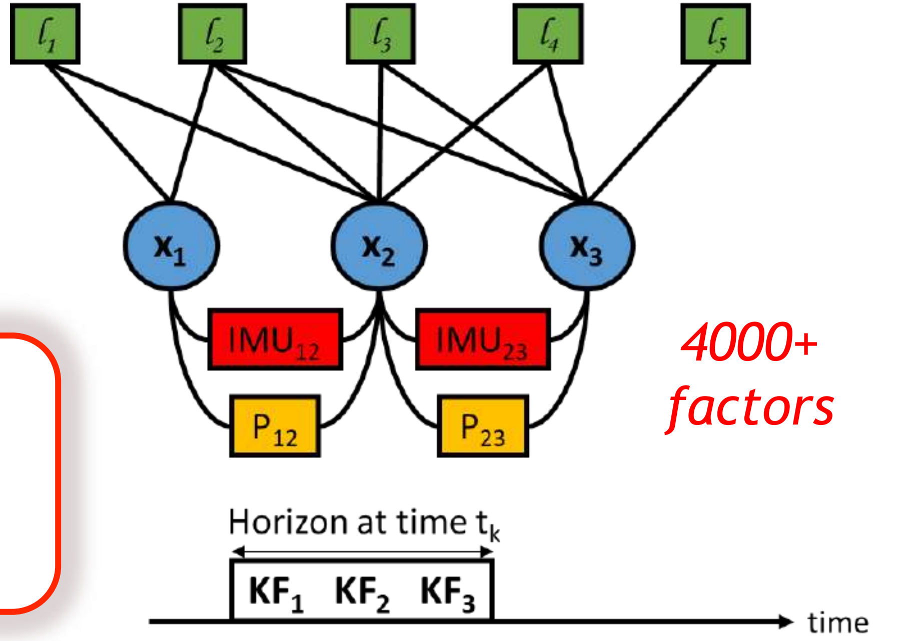
VIO: Backend

$$\min_{x} \sum_{(i,j)\in\mathcal{F}} \left\|r_{\mathrm{IMU}}\left(x, \Delta\tilde{R}_{ij}, \Delta\tilde{p}_{ij}, \Delta\tilde{v}_{ij}\right)\right\|^2 + \sum_{k\in\mathcal{L}} \sum_{i\in\mathcal{F}_k} \left\|r_{\mathrm{CAM}}\left(x, l_k, u_{ik}^{l}, u_{ik}^{r}\right)\right\|^2 + \left\|r_{\mathrm{PRIOR}}(x)\right\|^2$$

IMU Factors *Vision Factors* *Other Factors*



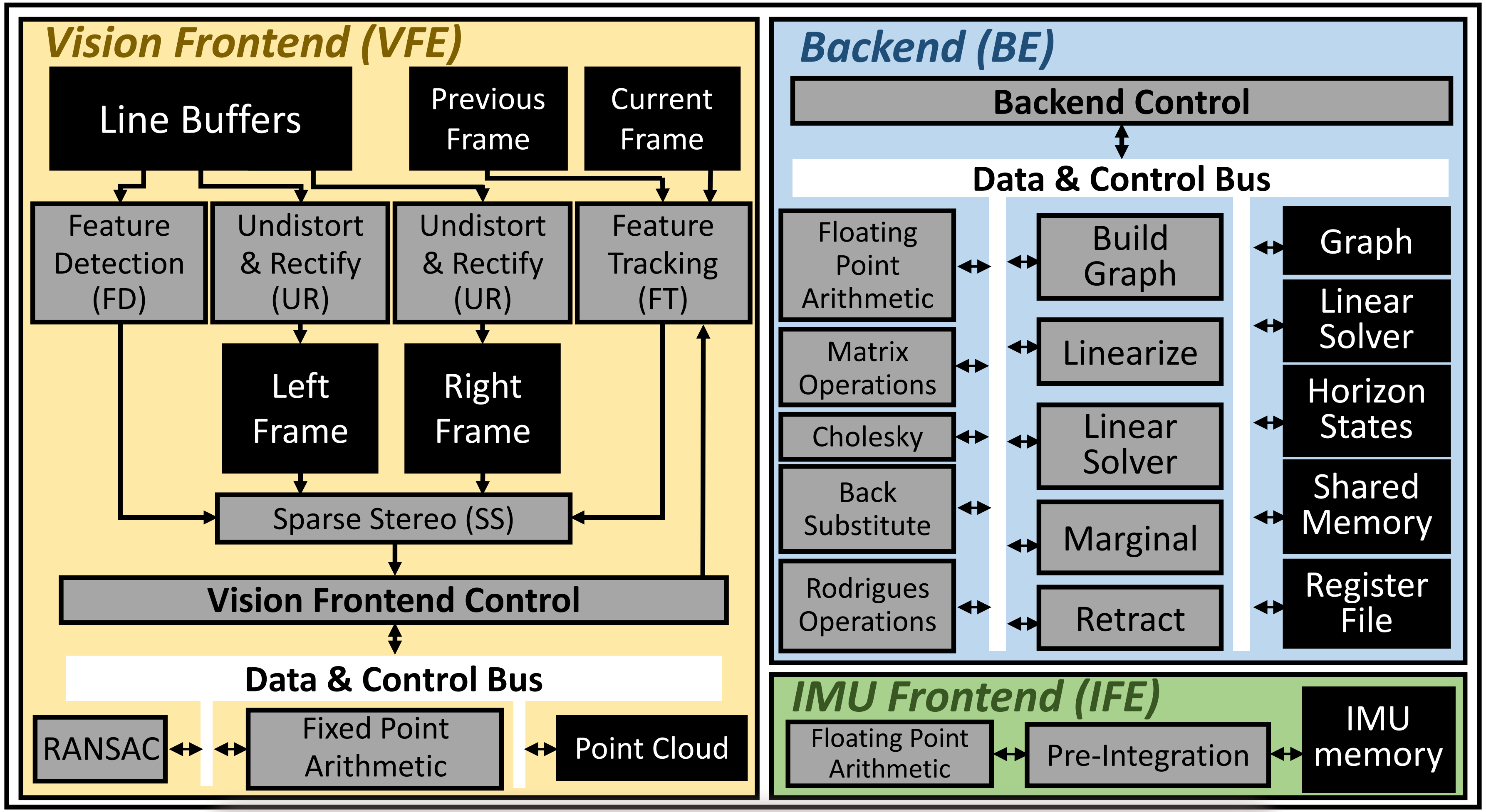
Factor Graph



Outline

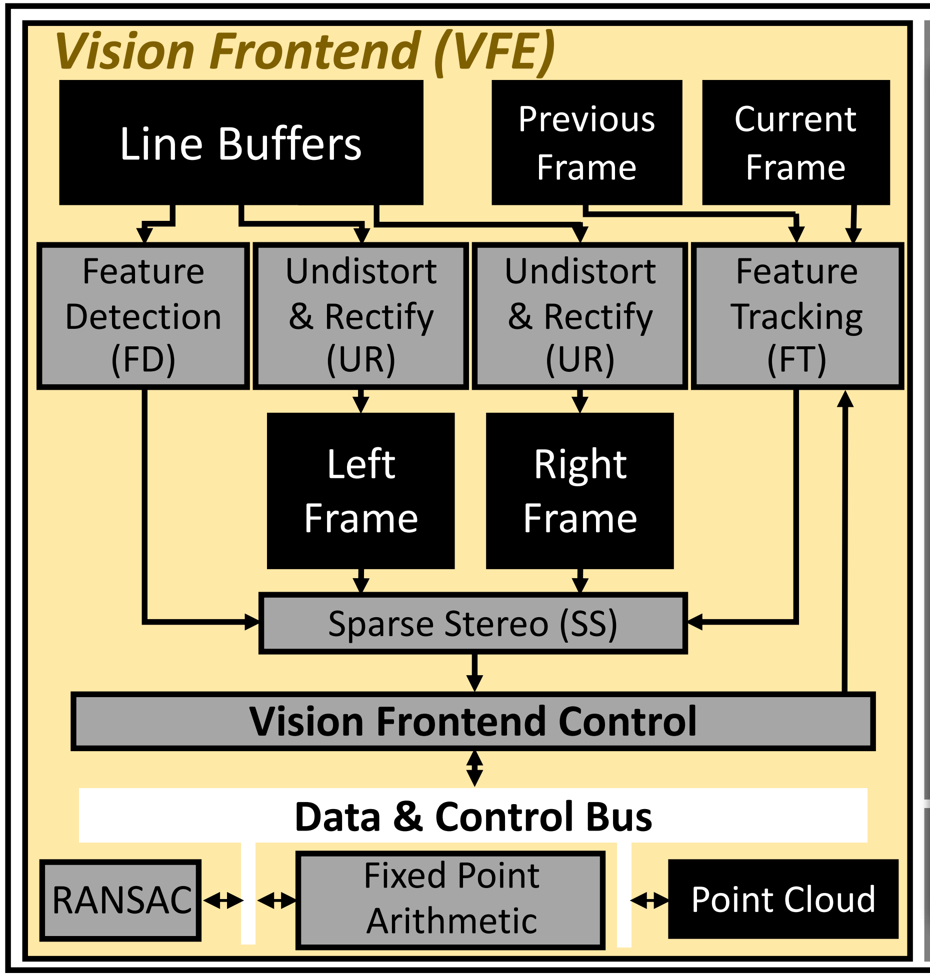
- Localization & Mapping: Visual-Inertial Odometry (VIO)
- Chip Architecture
- Main Contributions
- Chip Specifications and Comparisons
- Summary

Navion Chip Architecture



No off-chip storage or processing

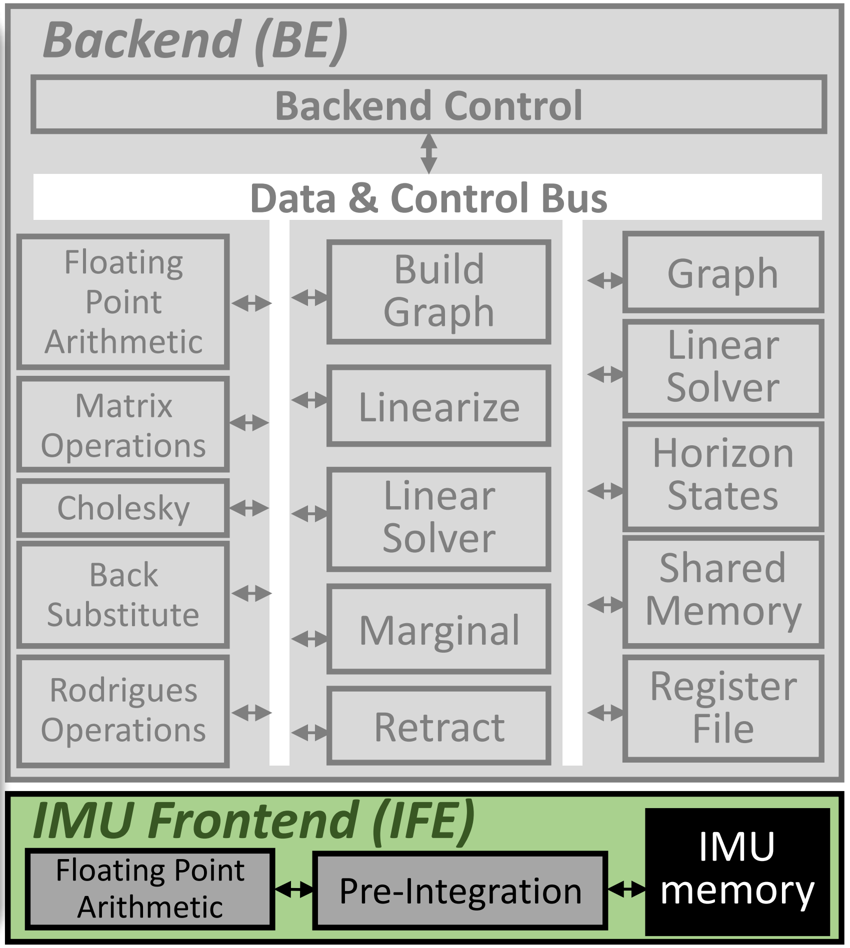
VFE: All Image Processing



- Fixed point arithmetic
- Parallel/pipelined image processing
- Mono & stereo cameras
- Runs at the sensor rate (<u>up to 171 fps</u>)
- **<u>Outputs at keyframe rate:</u>** Feature tracks

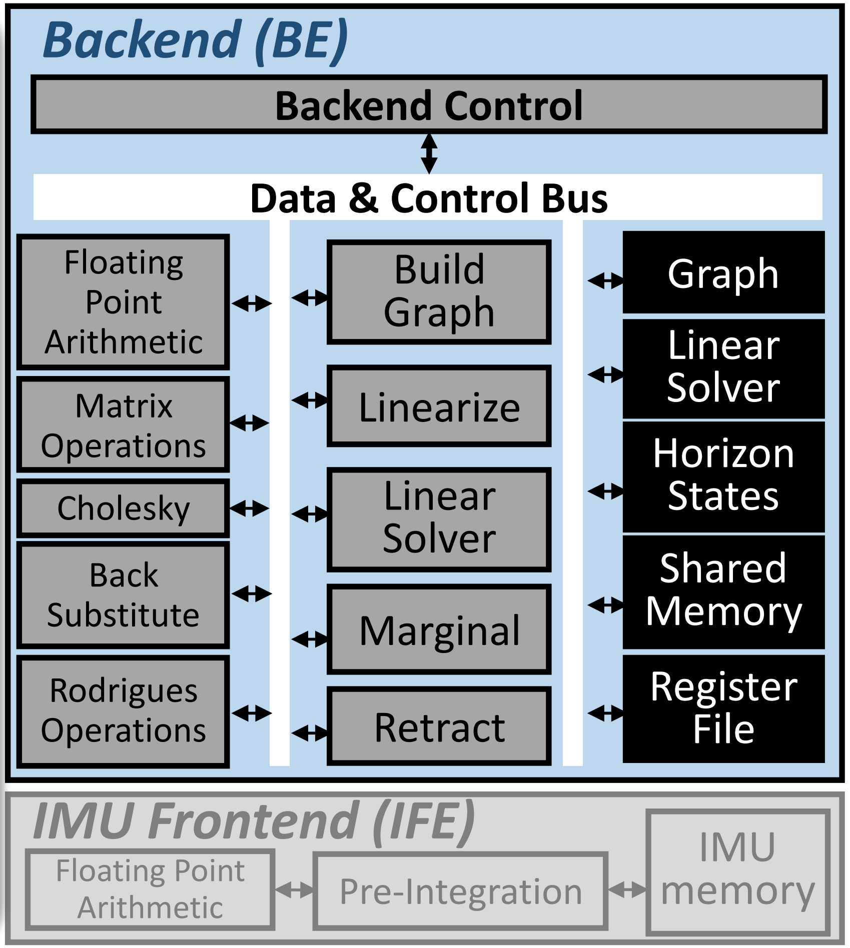
IFE: IMU Preintegration

- Double precision arithmetic
- Low cost: 2.4% area & 1.2% power
- Runs at the sensor rate (up to 52 kHz)
- **Outputs at keyframe rate:** Estimated state



BE: Fusing Sensors Data

- Double precision arithmetic
- Complex Finite State Machine (FSM)
- Runs at the keyframe rate (up to 90 fps)
- **Outputs at keyframe rate:** Updated state & 3D map



VIO Full Integration Challenges

- Vision Frontend (VFE)
  - Heterogeneous computation modules
    - Feature detection
    - Feature tracking
    - Stereo matching
    - Outliers rejection using RANSAC
    - …

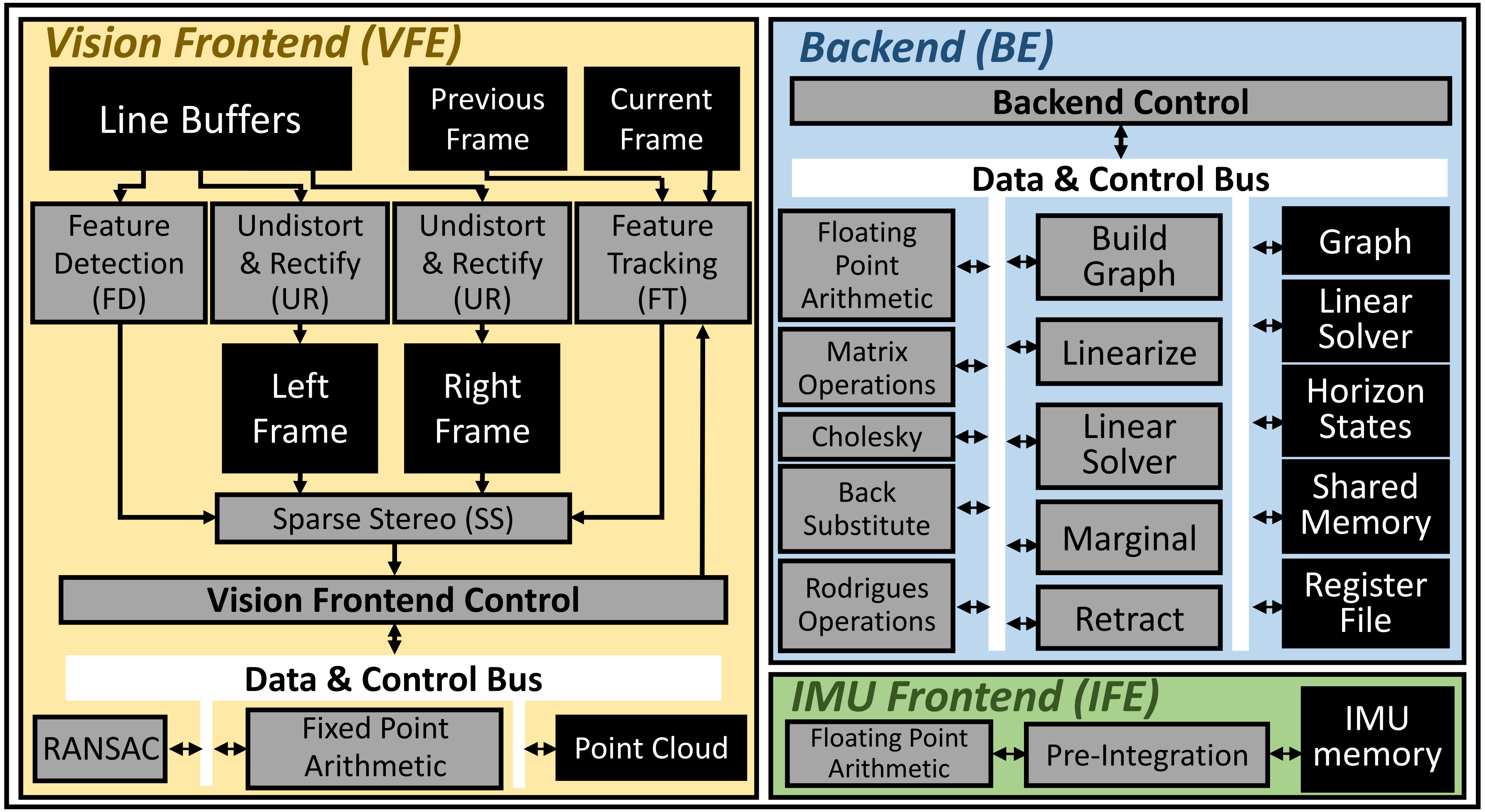
VIO Full Integration Challenges

- Vision Frontend (VFE)
  - Heterogeneous computation modules
    - Feature detection
    - Feature tracking
    - Stereo matching
    - Outliers rejection using RANSAC
    - …
- Backend (BE)
  - High dimensional and complex data structures
    - Large optimization problem (more than 4000 factors)
    - Dynamically changing factor graph
    - High computation precision (64-bit floating point)

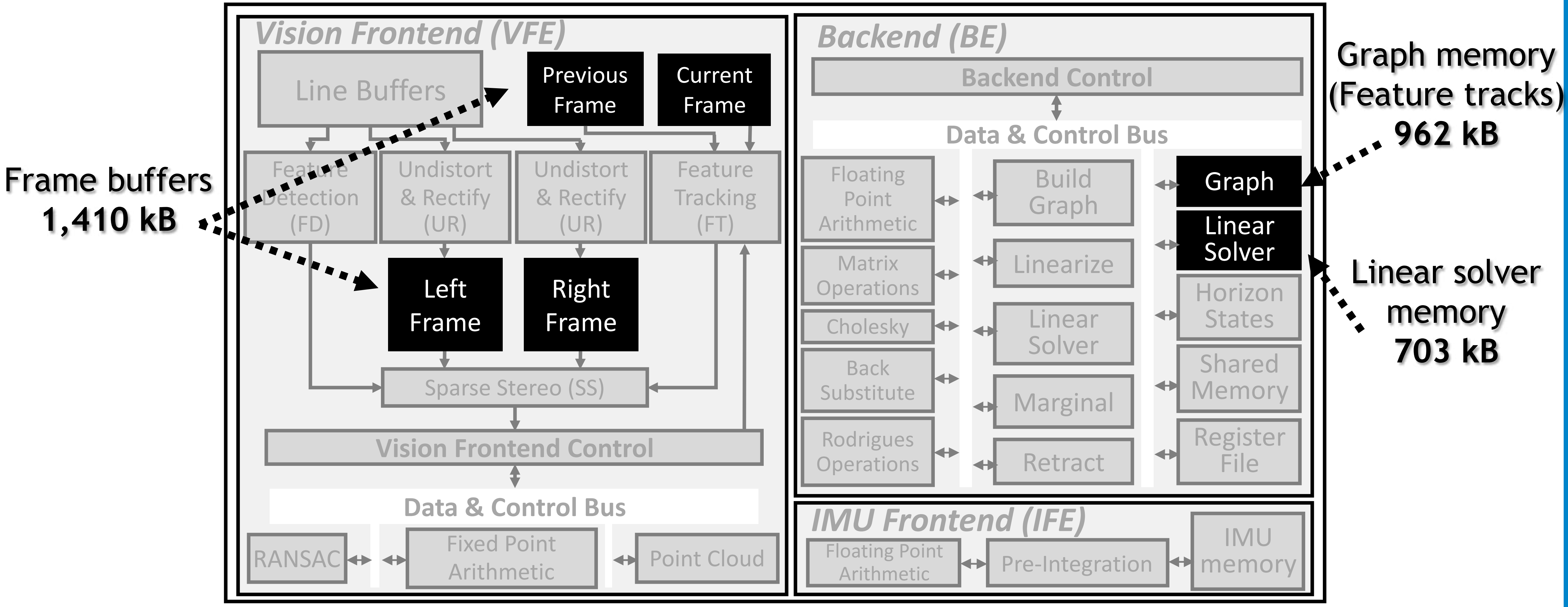
Outline

- Localization & Mapping: Visual-Inertial Odometry (VIO)
- Chip Architecture
- Main Contributions
- Chip Specifications and Comparisons
- Summary

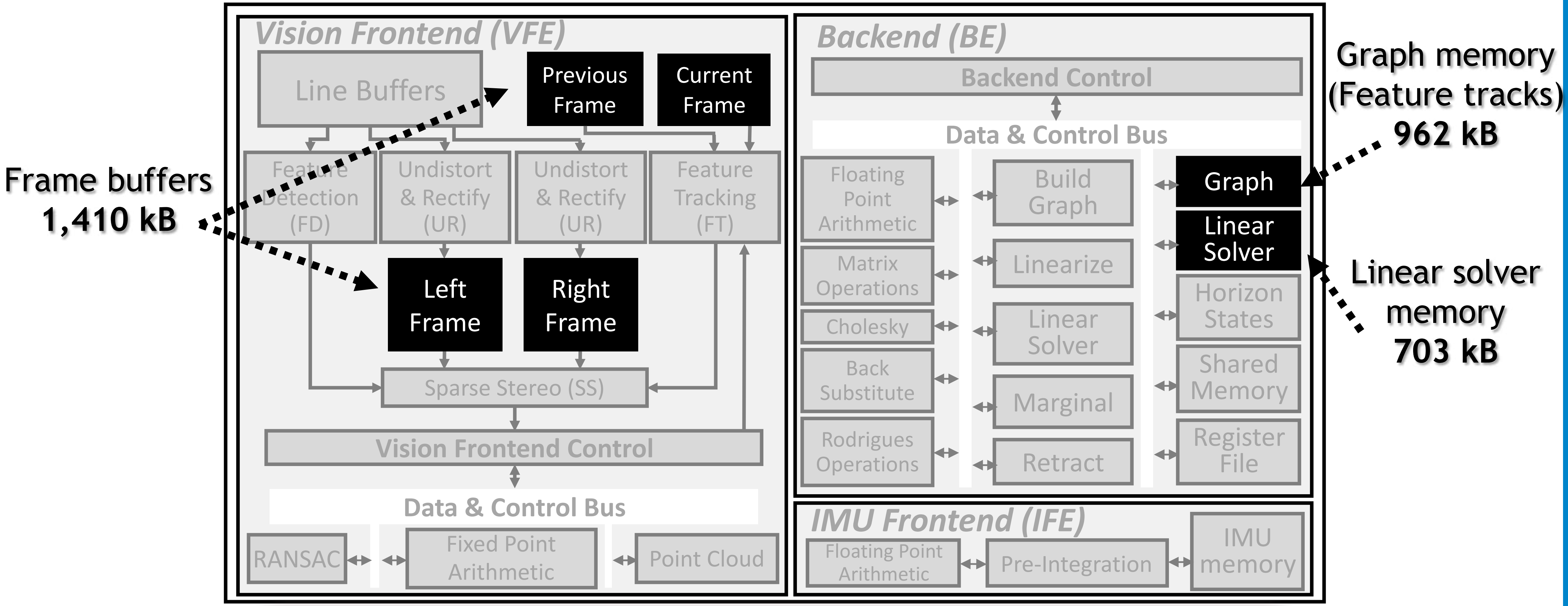
Enabling Full Integration



Enabling Full Integration



Enabling Full Integration



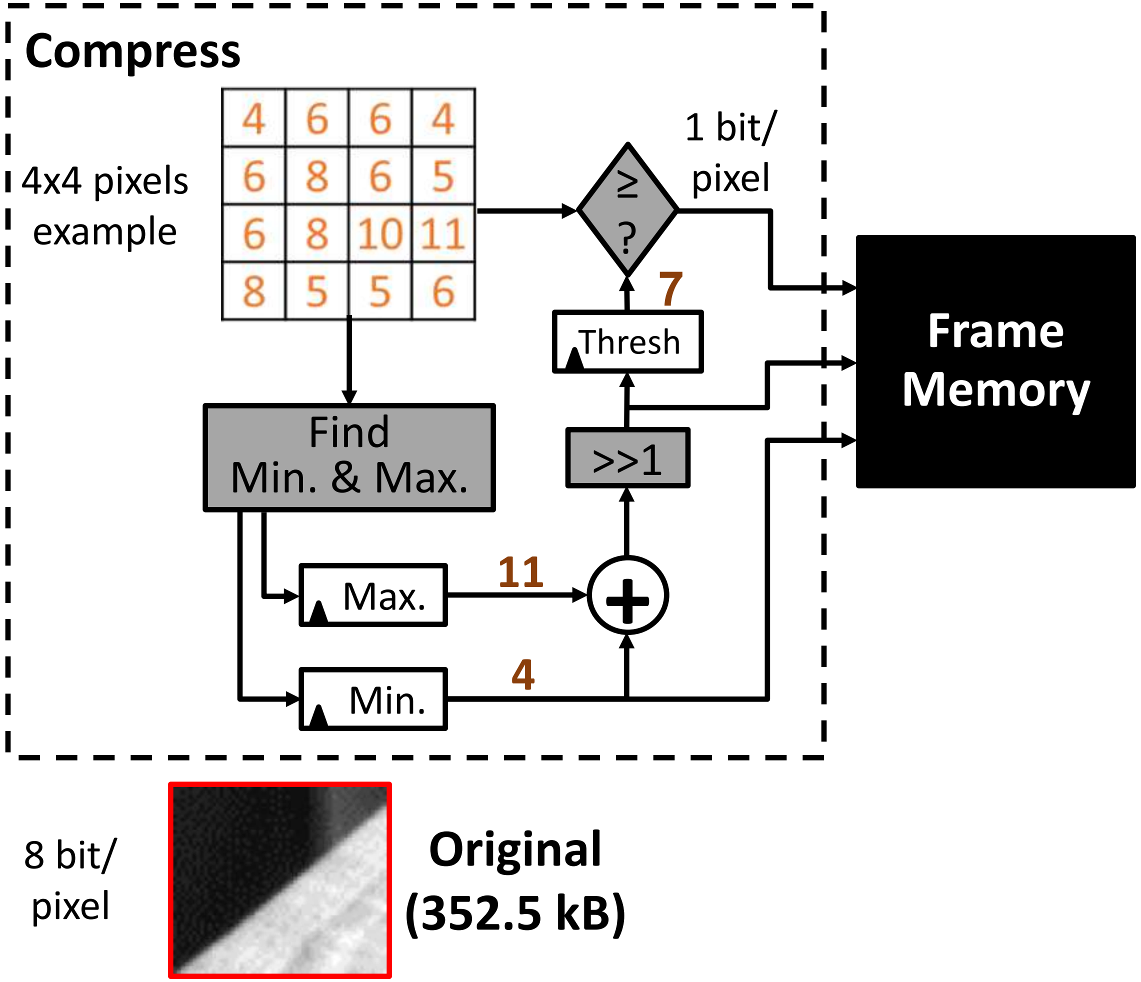
Use compression and exploit sparsity

Method 1

Data Compression

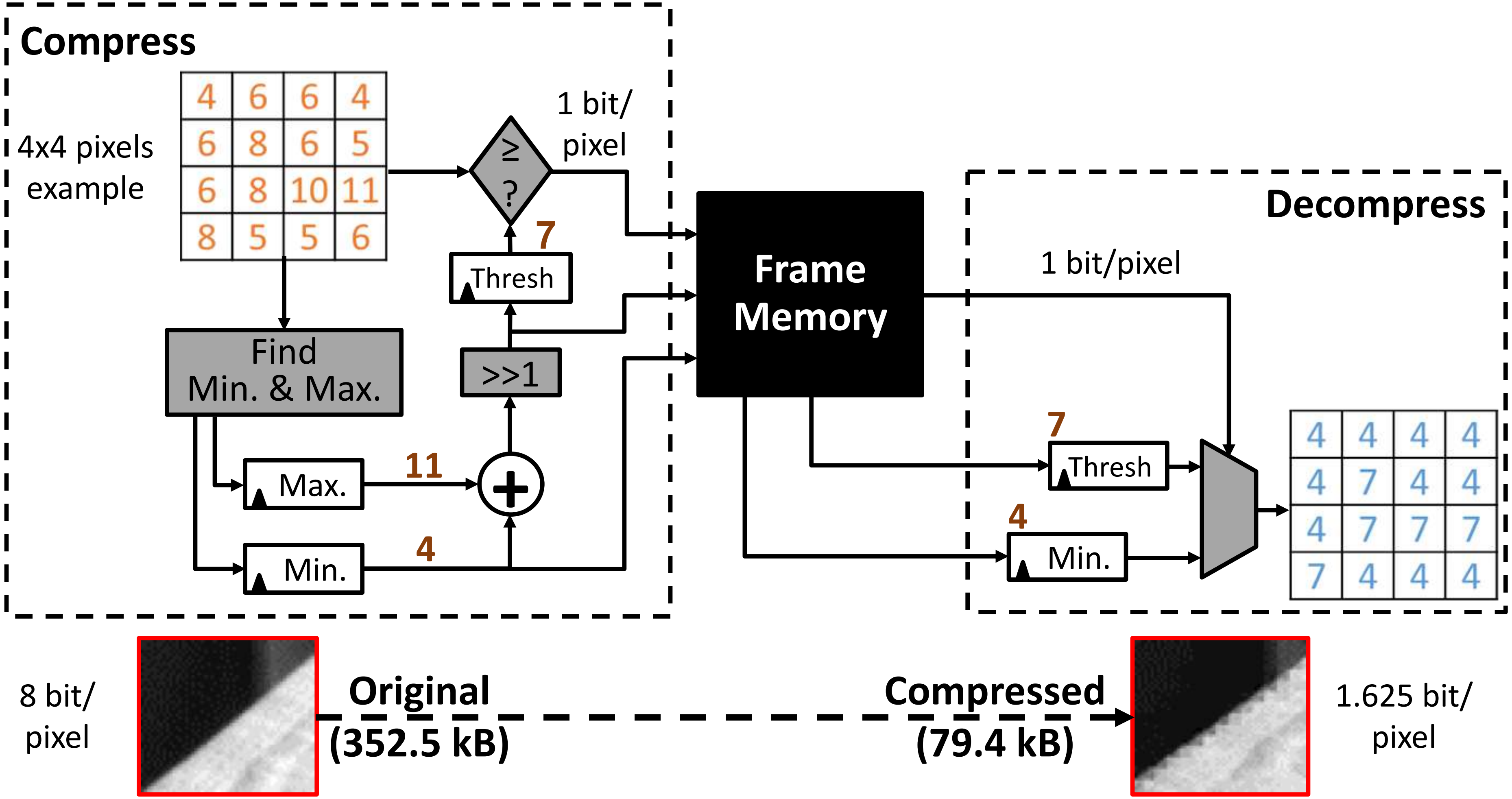
Frame Buffer: Image Compression

- Block-wise Lossy Image Compression



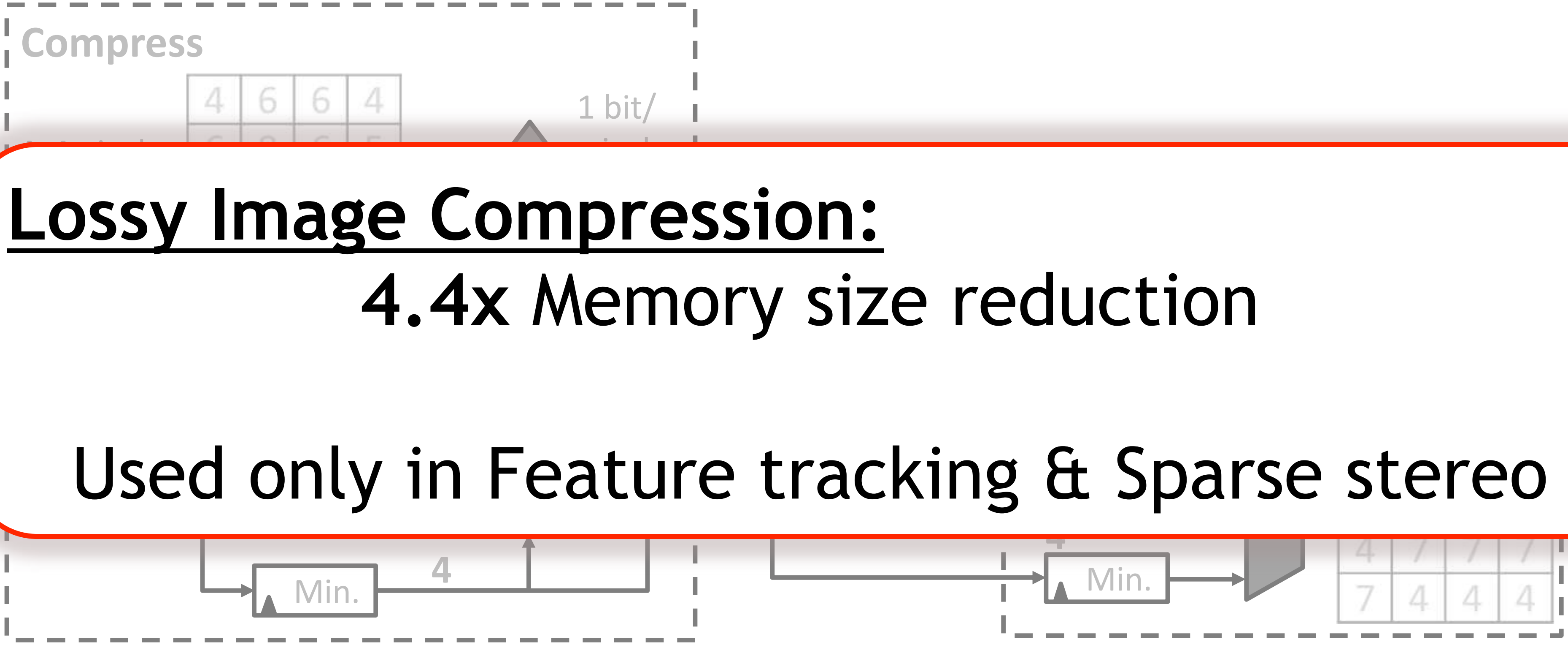
Frame Buffer: Image Compression

- Block-wise Lossy Image Compression



Frame Buffer: Image Compression

- Block-wise Lossy Image Compression

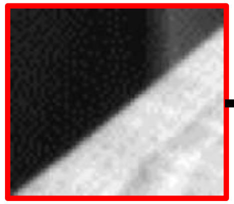


Lossy Image Compression:

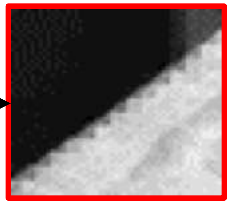
4.4x Memory size reduction

Used only in Feature tracking & Sparse stereo

8 bit/ pixel



Original (352.5 kB) → **Compressed (79.4 kB)**



1.625 bit/ pixel

Method 2

Exploit Sparsity

(Structured & Unstructured)

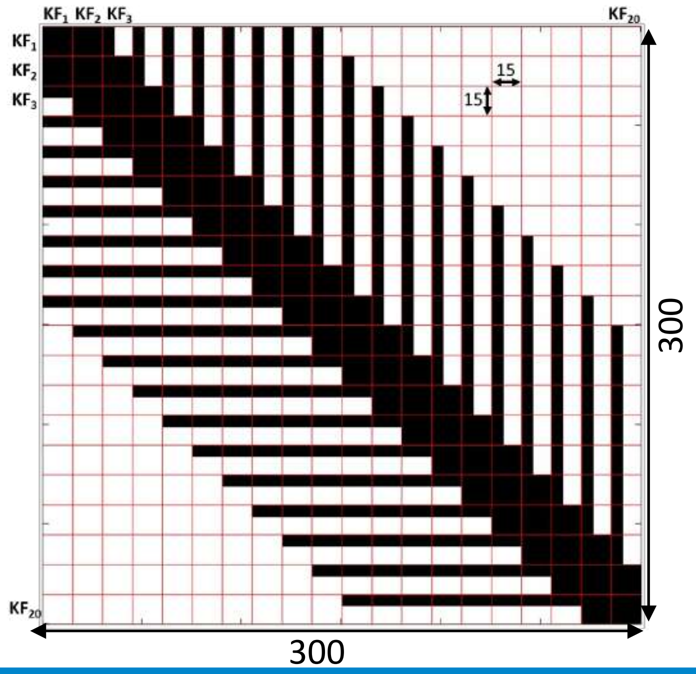
Linear Solver memory: Structured Sparsity

$$\min_x \sum_{(i,j)\in\mathcal{F}} \left\| r_{\mathrm{IMU}}\left(x, \Delta\tilde{R}_{ij}, \Delta\tilde{p}_{ij}, \Delta\tilde{v}_{ij}\right)\right\|^2 + \sum_{k\in\mathcal{L}} \sum_{i\in\mathcal{F}_k} \left\| r_{\mathrm{CAM}}\left(x, l_k, u_{ik}^l, u_{ik}^r\right)\right\|^2 + \left\| r_{\mathrm{PRIOR}}(x)\right\|^2$$

Linearize

$$\boldsymbol{H}\delta = \acute{\varepsilon}$$

Solve a large linear system for δ



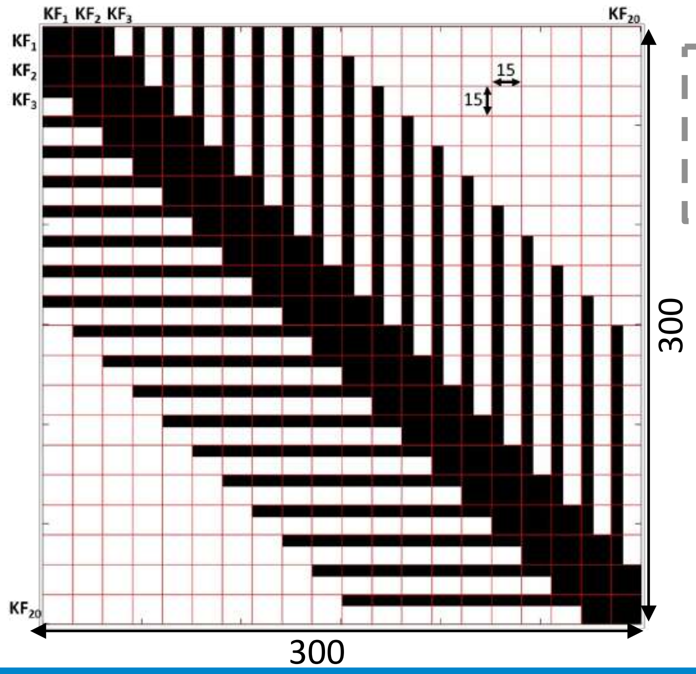
Linear Solver memory: Structured Sparsity

$$\min_x \sum_{(i,j)\in\mathcal{F}} \left\| r_{\mathrm{IMU}}(x, \Delta\tilde{R}_{ij}, \Delta\tilde{p}_{ij}, \Delta\tilde{v}_{ij}) \right\|^2 + \sum_{k\in\mathcal{L}} \sum_{i\in\mathcal{F}_k} \left\| r_{\mathrm{CAM}}(x, l_k, u^l_{ik}, u^r_{ik}) \right\|^2 + \left\| r_{\mathrm{PRIOR}}(x) \right\|^2$$

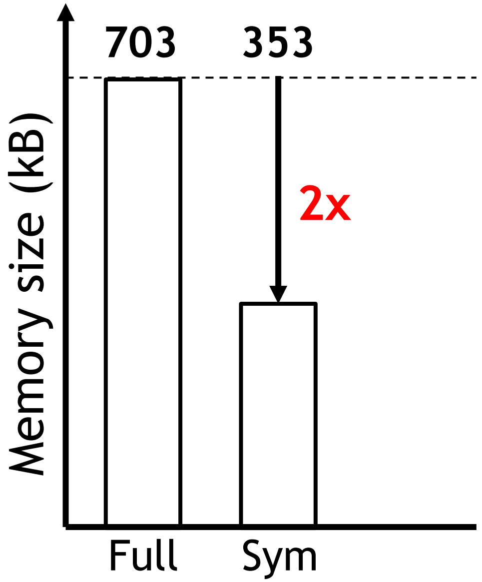
Linearize

$$\boxed{\boldsymbol{H}\delta = \acute{\varepsilon}}$$

Solve a large linear system for δ



H is:
- Symmetric



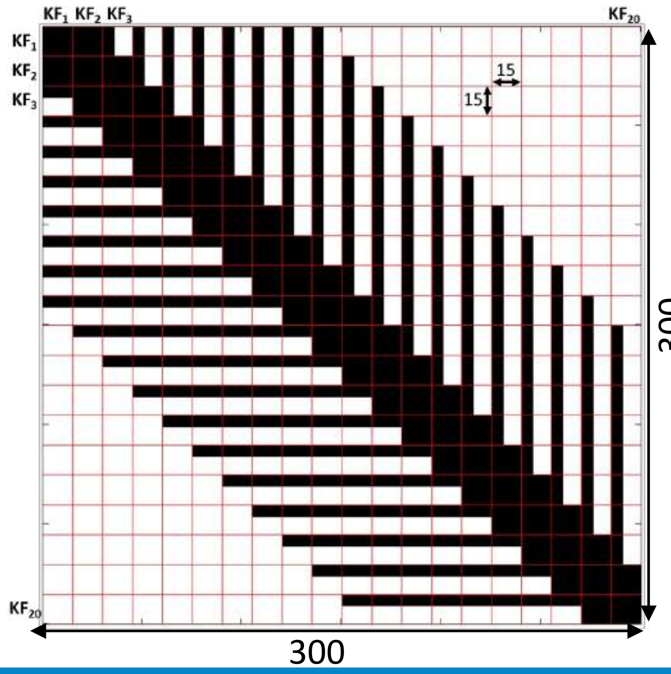
Linear Solver memory: Structured Sparsity

$$\min_{x} \sum_{(i,j)\in\mathcal{F}} \left\|r_{\mathrm{IMU}}\left(x,\Delta\tilde{R}_{ij},\Delta\tilde{p}_{ij},\Delta\tilde{v}_{ij}\right)\right\|^2 + \sum_{k\in\mathcal{L}}\sum_{i\in\mathcal{F}_k}\left\|r_{\mathrm{CAM}}\left(x,l_k,u^l_{ik},u^r_{ik}\right)\right\|^2 + \left\|r_{\mathrm{PRIOR}}(x)\right\|^2$$

Linearize

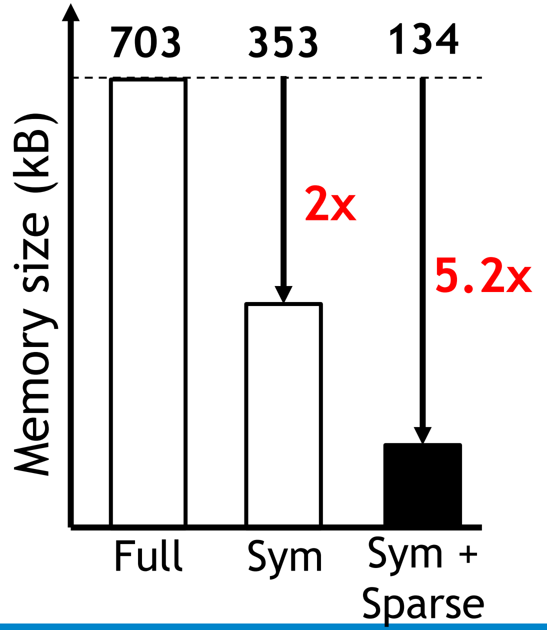
$$\boxed{H\delta = \acute{\varepsilon}}$$

Solve a large linear system for δ



H is:
- Symmetric
- Sparse

(Black: non zero)



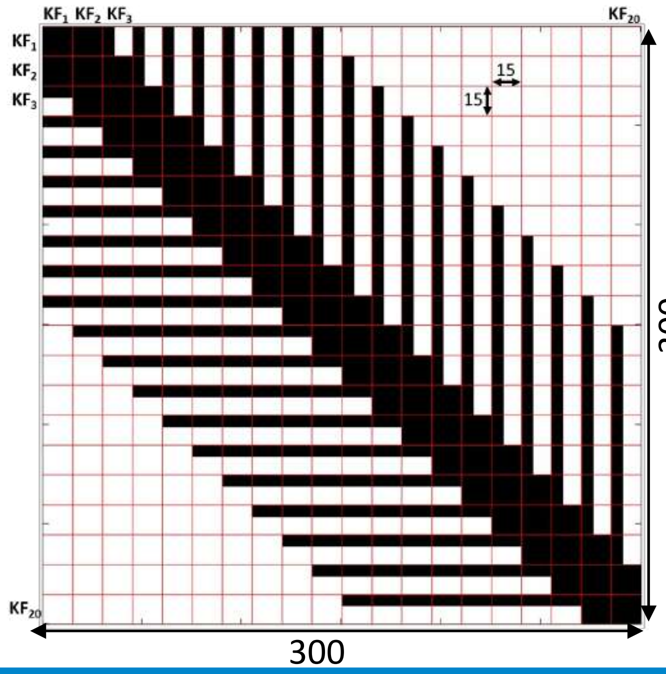
Linear Solver memory: Structured Sparsity

$$\min_{x} \sum_{(i,j)\in\mathcal{F}} \left\| r_{\mathrm{IMU}}\left(x, \Delta\tilde{R}_{ij}, \Delta\tilde{p}_{ij}, \Delta\tilde{v}_{ij}\right)\right\|^2 + \sum_{k\in\mathcal{L}} \sum_{i\in\mathcal{F}_k} \left\| r_{\mathrm{CAM}}\left(x, l_k, u_{ik}^{l}, u_{ik}^{r}\right)\right\|^2 + \left\| r_{\mathrm{PRIOR}}(x)\right\|^2$$

Linearize

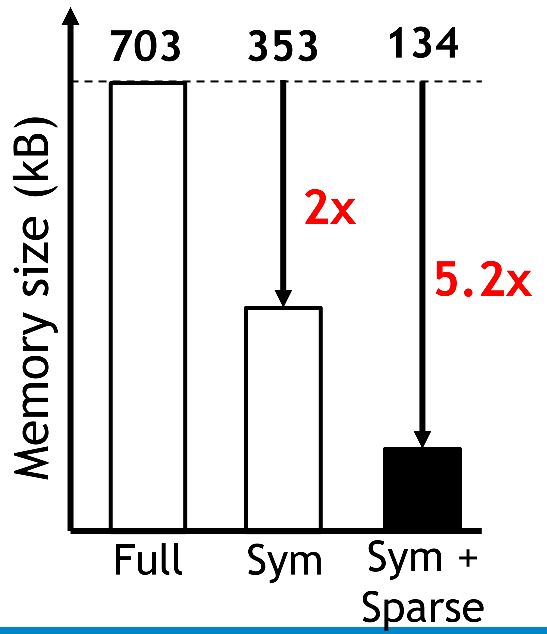
$$\boldsymbol{H}\delta = \acute{\varepsilon}$$

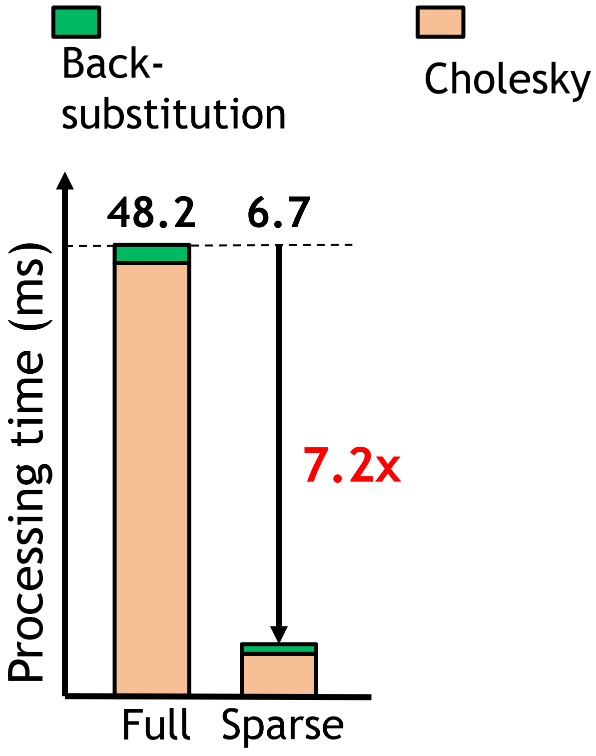
Solve a large linear system for δ



H is:
- Symmetric
- Sparse

(Black: non zero)





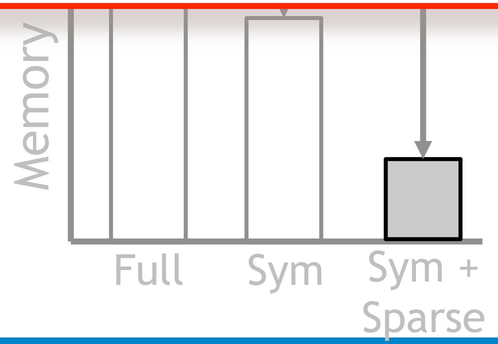
Linear Solver memory: Structured Sparsity

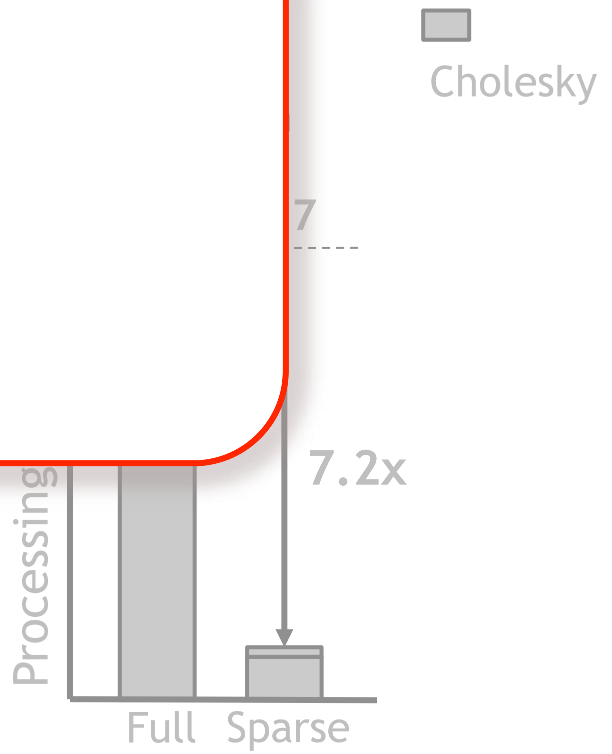
$$\min_{x} \sum_{(i,j)\in\mathcal{F}} \left\|r_{\mathrm{IMU}}(x, \Delta\tilde{R}_{ij}, \Delta\tilde{p}_{ij}, \Delta\tilde{v}_{ij})\right\|^2 + \sum_{k\in\mathcal{L}} \sum_{i\in\mathcal{F}_k} \left\|r_{\mathrm{CAM}}(x, l_k, u^l_{ik}, u^r_{ik})\right\|^2 + \left\|r_{\mathrm{PRIOR}}(x)\right\|^2$$

Linearize

$$H\delta = \acute{\varepsilon}$$

(Black: non zero)





<u>Storing symmetric non-zero values:</u>

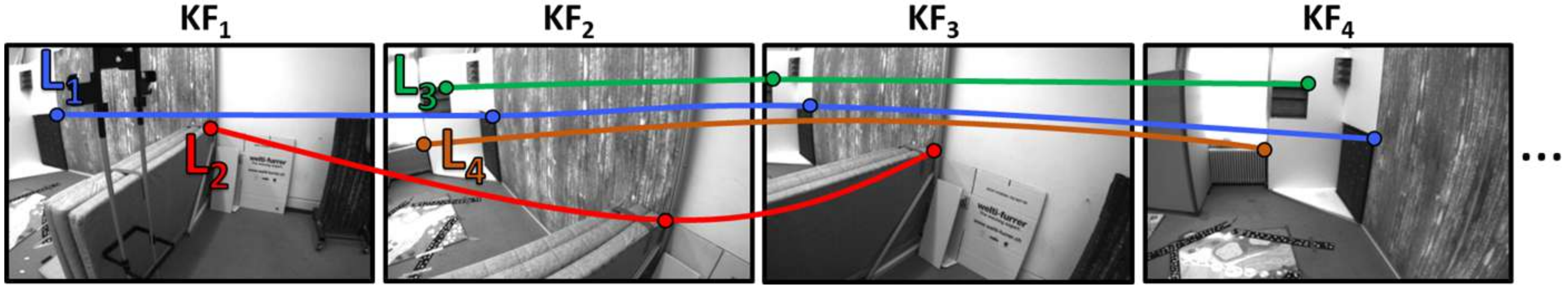
5.2x Memory size reduction

<u>Skip processing zeros:</u>

7.2x Speed up

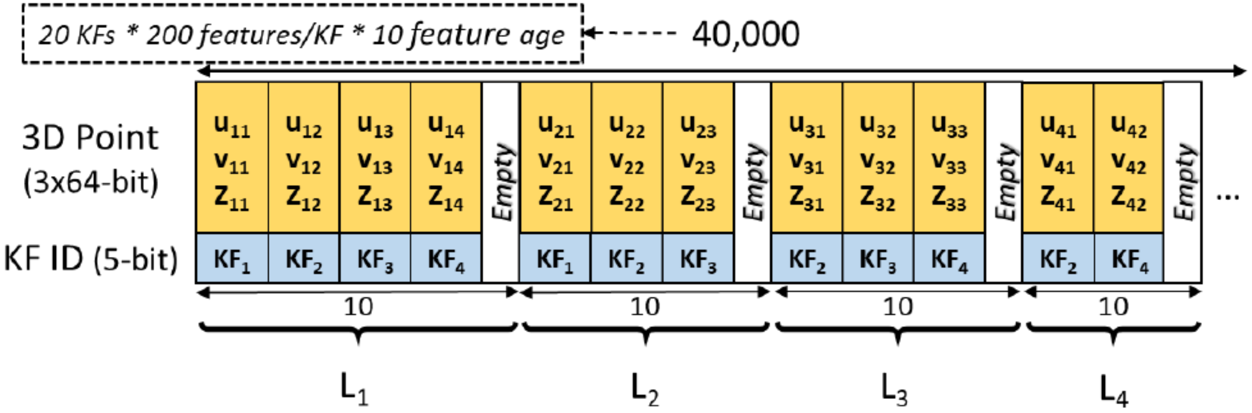
Feature Tracks: Unstructured Sparsity

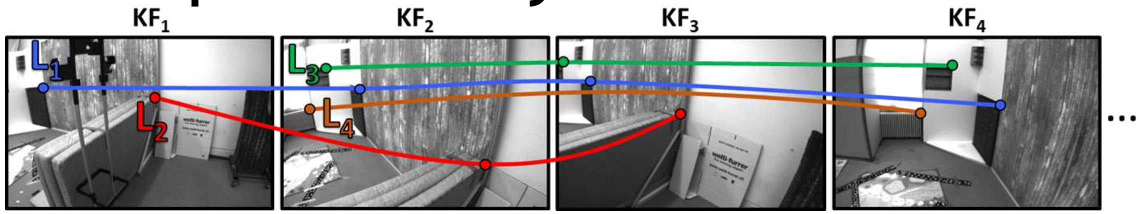
- Feature Tracks accounts for 88% of the Graph memory



Feature Tracks: Unstructured Sparsity

- Feature Tracks accounts for 88% of the Graph memory

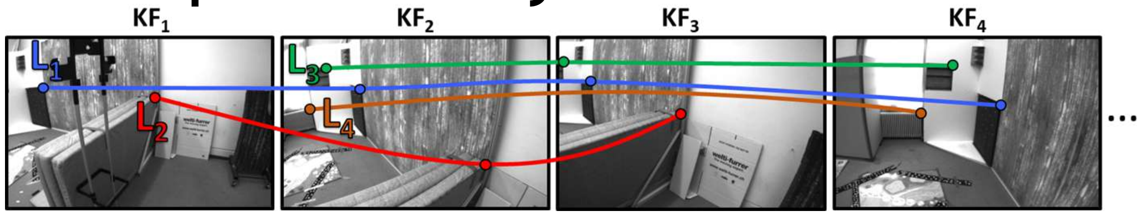


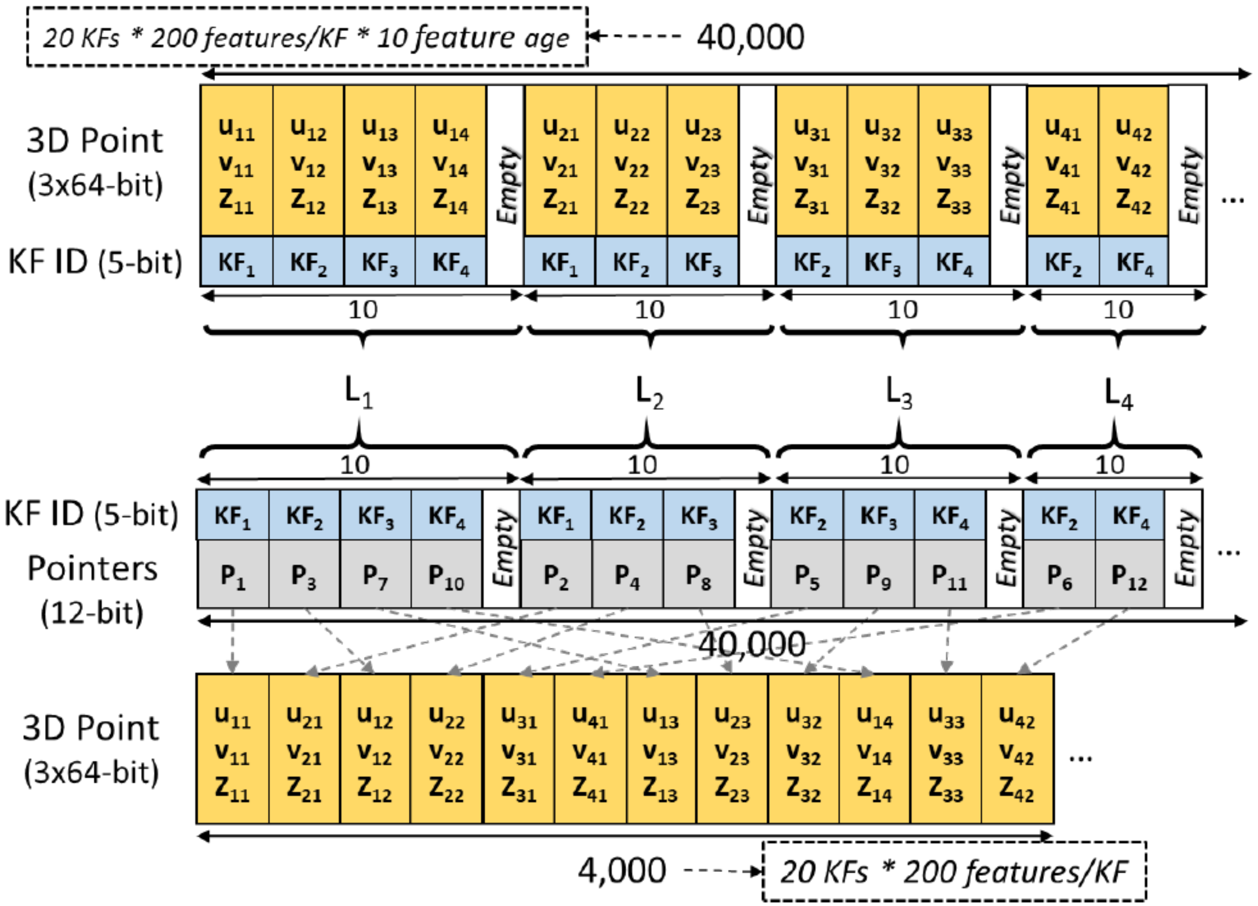


One Memory (962 kB)

Feature Tracks: Unstructured Sparsity

- Feature Tracks accounts for 88% of the Graph memory





One Memory (962 kB)

Two-stage Memory (177 kB)

Feature Tracks: Unstructured Sparsity

- Feature Tracks accounts for 88% of the Graph memory

<u>Feature tracks two-stage storage:</u>

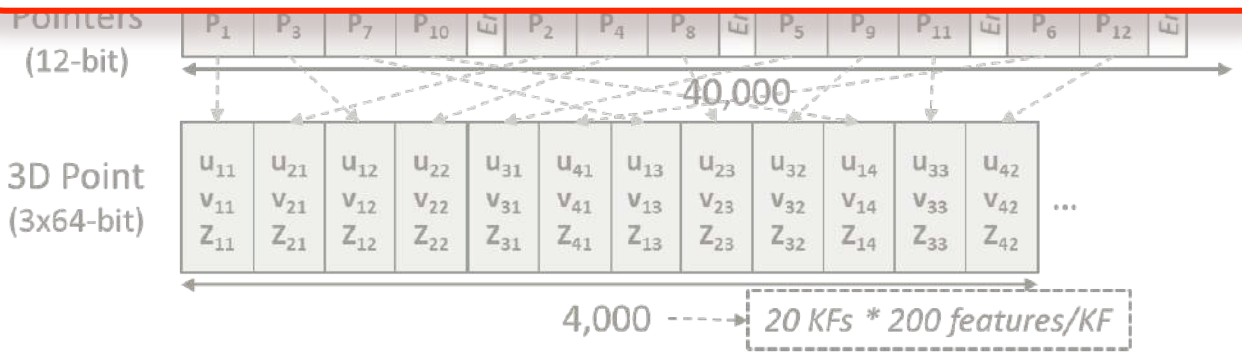
5.4x Memory size reduction

<u>Overhead:</u>

1 extra cycle access latency

One Memory (962 kB)

Two-stage Memory (177 kB)



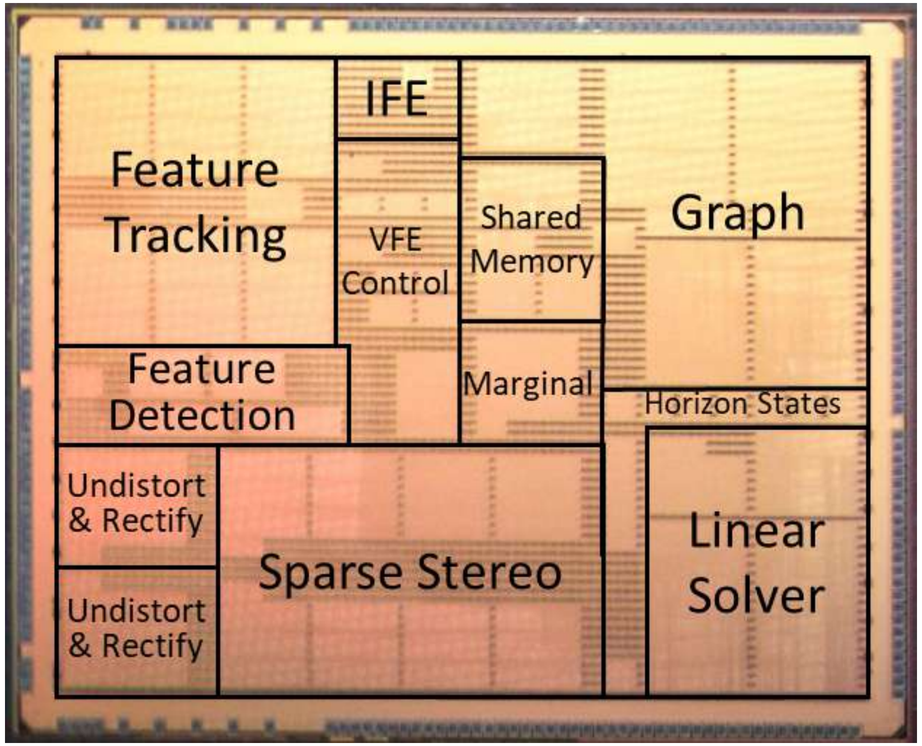
Outline

- Localization & Mapping: Visual-Inertial Odometry (VIO)
- Chip Architecture
- Main Contributions
- Chip Specifications and Comparisons
- Summary

Navion Chip

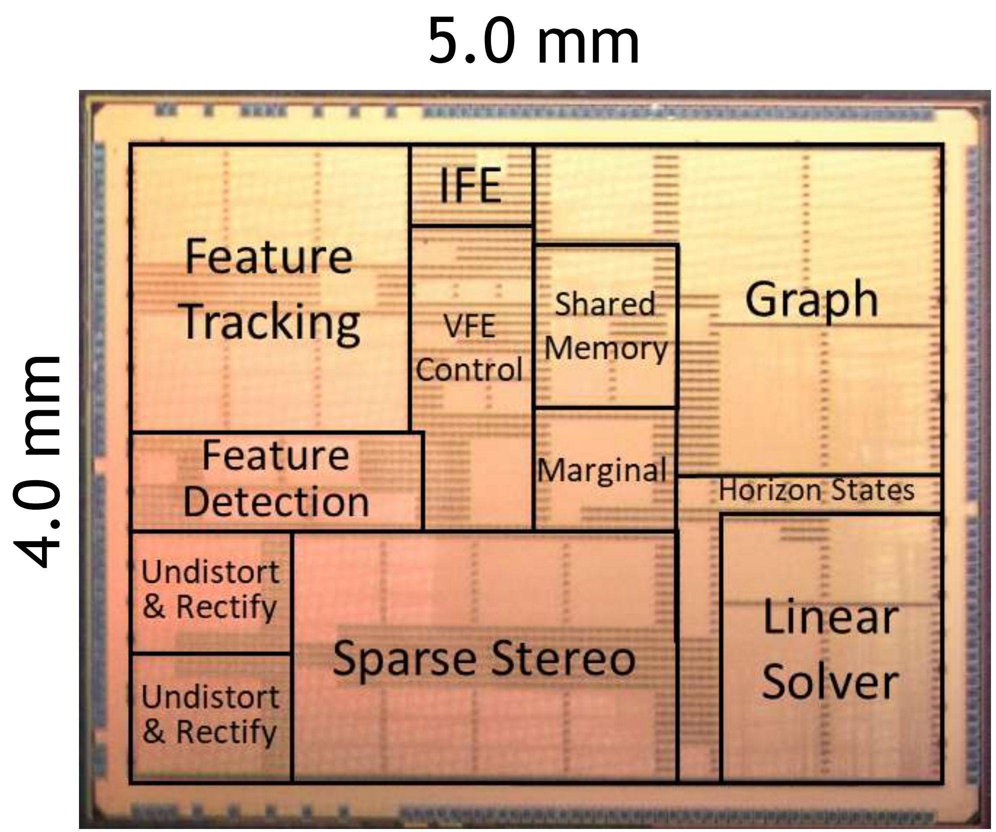
5.0 mm

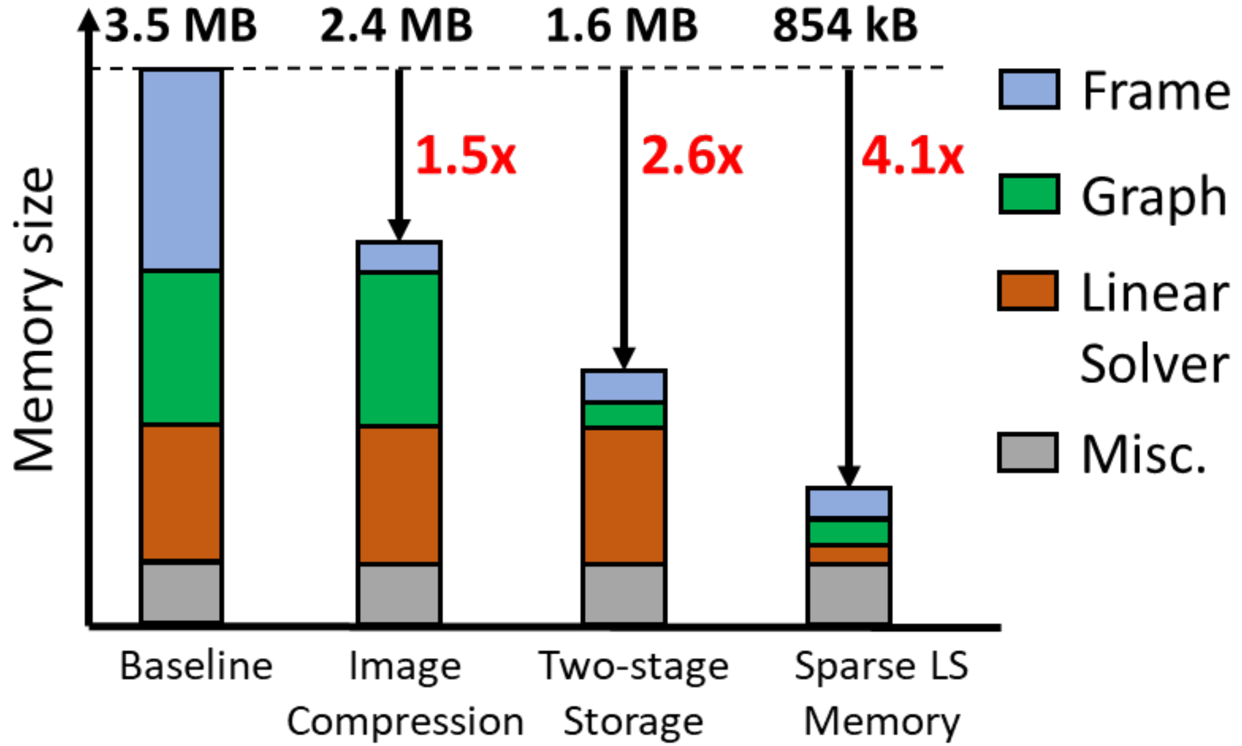
4.0 mm



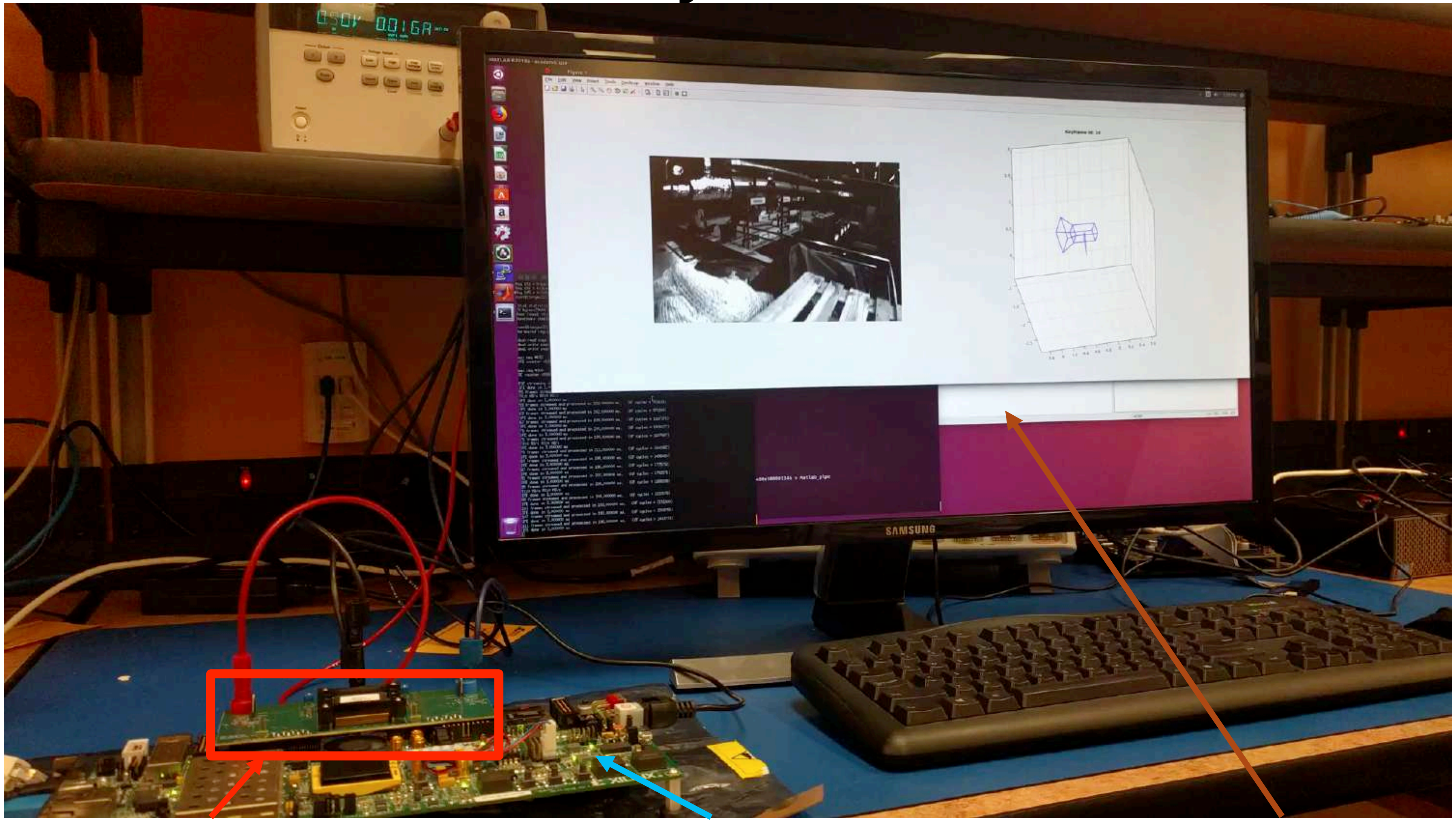
Technology	65nm CMOS
Chip area (mm^2)	4.0 x 5.0
Logic gates	2,043 kgates
Resolution	752 x 480
SRAM	854 kB
Camera rate	**28 - 171 fps**
Keyframe rate	**16 - 90 fps**
Average Power	**24 mW**
GOPS	10.5 – 59.1
GFLOPS	1 – 5.7

Memory Optimization





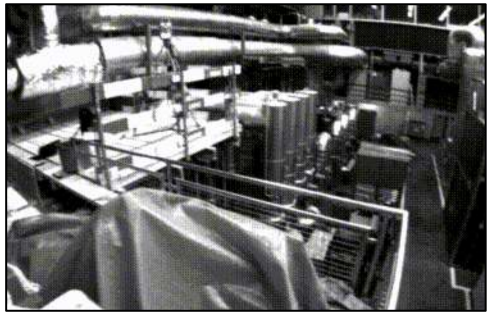
Navion System Demo



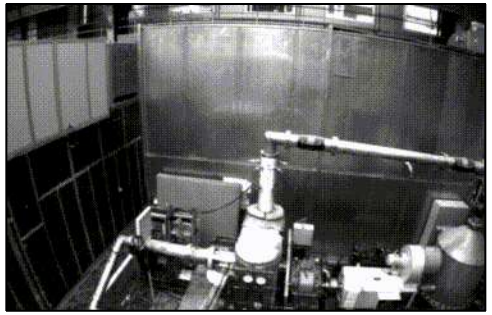
Navion chip PCB Xilinx Zynq FPGA Board Results

Navion Evaluation

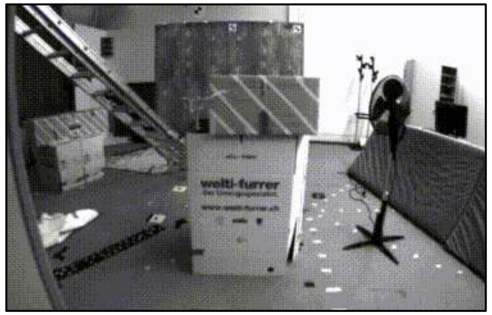
- EuRoC dataset
  - A very challenging, and widely used UAV dataset
  - 11 sequences with three categories: easy, medium & difficult




Examples of easy Sequences



Dark scenes



Motion blur

Examples of difficult Sequences

Navion Evaluation

- Average numbers over the 11 EuRoC dataset sequences

Platform	Xeon (E5-2667)	ARM (Cortex A15)	Navion
Trajectory Error (%)	0.22%		0.28%
Camera rate (fps)	63	19	**71**
Keyframe rate (fps)	12	2	**19**
Average Power (W)	27.9	2.4	**0.024**
Energy (nJ/pixel)	2,531	1,094	**1.6**

Navion Evaluation

- Average numbers over the 11 EuRoC dataset sequences

Platform	Xeon (E5-2667)	ARM (Cortex A15)	Navion
Trajectory Error (%)	0.22%		0.28%
Camera rate (fps)	63	19	**71**
Keyframe rate (fps)	12	2	**19**
Average Power (W)	27.9	2.4	**0.024**
Energy (nJ/pixel)	2,531	1,094	**1.6**

Navion Energy:

684x less than embedded ARM CPU

1,582x less than server Xeon CPU

Outline

- Localization & Mapping: Visual-Inertial Odometry (VIO)
- Chip Architecture
- Main Contributions
- Chip Specifications and Comparisons
- Summary

Summary

- First **full integration** of VIO pipeline on chip for robot perception

Summary

- First **full integration** of VIO pipeline on chip for robot perception
- Leverage compression and sparsity to reduce memory size
  - **4.4x** reduction with image compression
  - **5.2x** reduction with structured sparsity in linear solver
  - **5.4x** reduction with unstructured sparsity in feature tracks

Summary

- First **full integration** of VIO pipeline on chip for robot perception
- Leverage compression and sparsity to reduce memory size
  - **4.4x** reduction with image compression
  - **5.2x** reduction with structured sparsity in linear solver
  - **5.4x** reduction with unstructured sparsity in feature tracks
- Navion is **2 to 3 orders of magnitude** more energy efficient than CPU

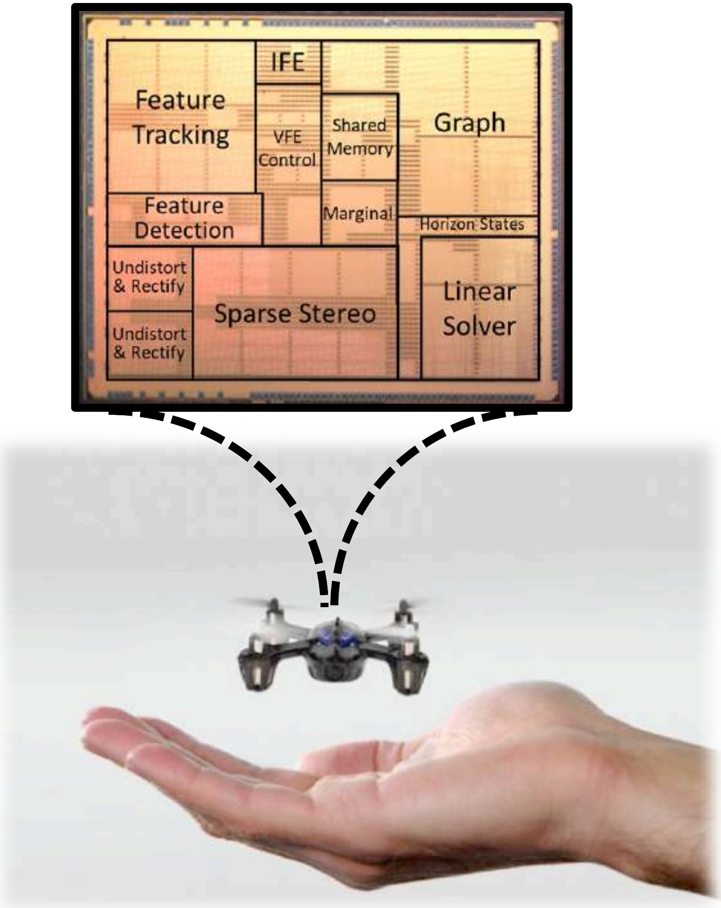
Summary

- First **full integration** of VIO pipeline on chip for robot perception
- Leverage compression and sparsity to reduce memory size
  - **4.4x** reduction with image compression
  - **5.2x** reduction with structured sparsity in linear solver
  - **5.4x** reduction with unstructured sparsity in feature tracks
- Navion is **2 to 3 orders of magnitude** more energy efficient than CPU

Acknowledgment

AFOSR YIP and NSF CAREER

Questions



http://navion.mit.edu/