Navion: An Energy-Efficient Visual-Inertial Odometry Accelerator for Micro Robotics and Beyond

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

Massachusetts Institute of Technology

http://navion.mit.edu/
Contributors

Amr Suleiman
PhD Graduate in EECS

Zhengdong Zhang
PhD Candidate in EECS

Luca Carlone
Professor in AeroAstro

Sertac Karaman
Professor in AeroAstro

Vivienne Sze
Professor in EECS
Motivation: Autonomous Navigation

Self Driving Cars

UAVs: Unmanned Aerial Vehicles

Robots

[images] Electrek, Amazon, Knightscope, Boston Dynamics
Motivation: Miniaturized Robots

Very small form factor

Micro  Nano  Pico?
Motivation: Miniaturized Robots

Very small form factor

Micro

Nano

Pico?

Applications

Search & Rescue

Surveillance

How Does Autonomous Navigation Work?

Perception

Motion Planning

Control
How Does Autonomous Navigation Work?

Perception

Perception is the computation bottleneck

Motion Planning

Where to Go?

Control

[Kanellakis et al., JIRS 2017]
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

For example:

Insect-scale UAV (100mg)

<table>
<thead>
<tr>
<th>Lifting</th>
<th>Sensing</th>
<th>CPU, GPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>100 mW</td>
<td>100 mW</td>
<td>10 - 100 W</td>
</tr>
</tbody>
</table>

[R.J. Wood et al., IJRR 2012]
Navion: Energy-Efficient Visual-Inertial Odometry

- Energy-efficient & real-time localization and mapping
- Real-time performance of 20 fps at 2mW
- Peak performance of up to 171 fps at 24 mW
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

Visual-Inertial Odometry (VIO)

Localization

Mapping

IMU
Inertial Measurement Unit
Localization and Mapping Using VIO

Image sequence

IMU
Inertial Measurement Unit

Subset of SLAM algorithm
(Simultaneous Localization And Mapping)

Localization

Visual-Inertial Odometry (VIO)

Mapping
VIO: Frontend

Camera

Vision Frontend (VFE)

Process mono/stereo Images

Stereo Images
VIO: Frontend

Process mono/stereo Images
- Detect & track features ($L_i$)
**VIO: Frontend**

Process mono/stereo Images
- Detect & track features ($L_i$)
- Generate *Feature Tracks* $\rightarrow$ (keyframe IDs & feature coordinates)
VIO: Frontend

Camera

Vision Frontend (VFE)

Feature Tracks

IMU

Frontend (IFE)

Gyro. & Acc. Measurements

Stereo Images

KF

KF: Keyframe
Symposia on VLSI Technology and Circuits

VIO: Frontend

Camera

Vision Frontend (VFE)

Feature Tracks

Estimated States

IMU Frontend (IFE)

IMU

KF_1

KF_2

KF_3

KF_4

Stereo Images

KF: Keyframe

IMU_{12}: \{\Delta R_{12}, \Delta T_{12}\}

IMU_{23}: \{\Delta R_{23}, \Delta T_{23}\}

Preintegration

Preintegration

… Gyro. & Acc. Measurements
VIO: Frontend

Camera

Vision Frontend (VFE)

Feature Tracks

Estimated States

IMU Frontend (IFE)

IMU

IMU Measurements

Gyro. & Acc.

Preintegration

Preintegration

KF₁
KF₂
KF₃
KF₄

IMU₁₂: \{ΔR₁₂, ΔT₁₂\}
IMU₂₃: \{ΔR₂₃, ΔT₂₃\}

State:
Pose (Rotation R)
Location (Translation T)

Gyro. & Acc. Measurements
Update states ($x_i$) to minimize inconsistencies between measurements across time.
**VIO: Backend**

Non-linear least squares factor graph optimization

\[
\min_x \sum_{(i,j) \in E} \| r_{\text{IMU}}(x, \Delta \tilde{R}_{ij}, \Delta \tilde{p}_{ij}, \Delta \tilde{v}_{ij}) \|^2 + \sum_{k \in L} \sum_{i \in J_k} \| r_{\text{CAM}}(x, l_k, u_{ik}, u_{ik}) \|^2 + \| r_{\text{PRIOR}}(x) \|^2
\]

- IMU Factors
- Vision Factors
- Other Factors

**Factor Graph**

- Feature Tracks
- Estimated States

**Backend (BE)**

- Camera
- Vision Frontend (VFE)
- IMU Frontend (IFE)

- IMU

**4000+ factors**

Horizon at time \( t_k \):

\( \text{KF}_1 \quad \text{KF}_2 \quad \text{KF}_3 \)
VIO: Backend

Non-linear least squares factor graph optimization

\[
\min_x \sum_{(i,j) \in F} \left\| r_{\text{IMU}}(x, \Delta \tilde{r}_{ij}, \Delta \tilde{p}_{ij}, \Delta \tilde{v}_{ij}) \right\|^2 + \sum_{k \in L} \sum_{i \in F_k} \left\| r_{\text{CAM}}(x, t_k, u^i_{lk}, u^i_{Ik}) \right\|^2 + \left\| r_{\text{PRIOR}}(x) \right\|^2
\]

IMU Factors

Vision Factors

Other Factors

Updated States (\(x_i\)) & Sparse 3D Map

Factor Graph

4000+ factors

Horizon at time \(t_k\)

KF1, KF2, KF3

Camera

Vision Frontend (VFE)

Feature Tracks

Estimated States

IMU Frontend (IFE)

IMU

Backend (BE)

Updated States (\(x_i\)) & Sparse 3D Map
Outline

• Localization & Mapping: Visual-Inertial Odometry (VIO)
• Chip Architecture
• Main Contributions
• Chip Specifications and Comparisons
• Summary
Navion is a **fully integrated** system: No off-chip storage or processing
VFE: All Image Processing

- Fixed point arithmetic
- Parallel/pipelined image processing
- Mono & stereo cameras
- Operates at the sensor rate (up to 171 fps)
- Outputs at keyframe rate: Feature tracks
IFE: IMU Preintegration

- Double precision arithmetic
- Low cost: 2.4% area & 1.2% power
- Operates 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)
- Operates 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

**Vision Frontend (VFE)**
- Line Buffers
- Previous Frame
- Current Frame
- Feature Detection (FD)
- Undistort & Rectify (UR)
- Undistort & Rectify (UR)
- Feature Tracking (FT)
- Left Frame
- Right Frame
- Sparse Stereo (SS)

**Backend (BE)**
- Backend Control
- Data & Control Bus
- Build Graph
- Linearize
- Linear Solver
- Marginal
- Retract
- Graph
- Linear Solver
- Horizon States
- Shared Memory
- Register File

**IMU Frontend (IFE)**
- Floating Point Arithmetic
- Pre-Integration
- IMU memory
Enabling Full Integration

Vision Frontend (VFE)
- Line Buffers
- Feature Detection (FD)
- Undistort & Rectify (UR)
- Left Frame
- Right Frame
- Sparse Stereo (SS)

Vision Frontend Control

Data & Control Bus

Backend (BE)
- Backend Control
- Data & Control Bus
- Build Graph
- Linearize
- Linear Solver
- Horizon States
- Shared Memory
- Register File
- Graph
- Linear Solver
- Marginal
- Retract
- Rodigues Operations
- Back Substitute
- Cholesky
- Matrix Operations
- Floating Point Arithmetic

IMU Frontend (IFE)
- RANSAC
- Fixed Point Arithmetic
- Point Cloud
- IMU memory

Graph memory (Feature tracks) 962 kB
Linear solver memory 703 kB
Frame buffers 1,410 kB

Use compression and exploit sparsity
Method 1
Data Compression
Frame Buffer: Image Compression

- Lossy Block-wise Image Compression

Compress:

<table>
<thead>
<tr>
<th>4</th>
<th>6</th>
<th>6</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>8</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>8</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>8</td>
<td>5</td>
<td>5</td>
<td>6</td>
</tr>
</tbody>
</table>

4x4 pixels example

Find Min. & Max.

Min. 4

Max. 11

\[ \text{Threshold} \]

\[ \geq 7 \]

\[ \gg 1 \]

1 bit/pixel

Frame Memory

Original (352.5 kB)
Frame Buffer: Image Compression

- Lossy Block-wise Image Compression

Compress

4x4 pixels example

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>6</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>8</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>8</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>8</td>
<td>5</td>
<td>5</td>
<td>6</td>
</tr>
</tbody>
</table>

Find Min. & Max.

Min.

Max.

\[\geq\]?

\[\gg 1\]

1 bit/pixel

Decompress

Frame Memory

Thresh

7

5

4

1 bit/pixel

Original (352.5 kB)

Compressed (79.4 kB)

1.625 bit/pixel

8 bit/pixel

4 4 4 4

4 7 4 4

4 7 7 7

7 4 4 4

Compress

Thresh

4

7

Min.

Max.
Frame Buffer: Image Compression

- Lossy Block-wise Image Compression

**Lossy Image Compression:**

4.4x Memory size reduction

Used only in Feature tracking & Sparse stereo

8 bit/pixel

Original (352.5 kB) 1.625 bit/pixel

Compressed (79.4 kB)
Method 2

Exploit Sparsity

(Structured & Unstructured)
Linear Solver Memory: Structured Sparsity

\[ \min_x \sum_{(i,j) \in F} \| r_{\text{imu}}(x, \Delta \tilde{R}_{ij}, \Delta \tilde{p}_{ij}, \Delta \tilde{v}_{ij}) \|^2 + \sum_{k \in \mathcal{L}} \sum_{i \in \mathcal{F}_k} \| r_{\text{cam}}(x, l_k, u_{ik}^l, u_{ik}^r) \|^2 + \| r_{\text{prior}}(x) \|^2 \]

Linearize

\[ H\delta = \hat{\epsilon} \]

Solve a large linear system for \( \delta \)
Linear Solver Memory: Structured Sparsity

\[
\min_x \sum_{(i,j) \in F} \| r_{IMU}(x, \Delta \tilde{R}_{ij}, \Delta \tilde{p}_{ij}, \Delta \tilde{v}_{ij}) \|^2 + \sum_{k \in E} \sum_{i \in F_k} \| r_{CAM}(x, l_k, u^l_{ik}, u^r_{ik}) \|^2 + \| r_{PRIOR}(x) \|^2
\]

\( H \delta = \hat{\epsilon} \)  Solve a large linear system for \( \delta \)

\( H \) is:
- Symmetric

<table>
<thead>
<tr>
<th>Memory size (kB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full</td>
</tr>
<tr>
<td>703</td>
</tr>
</tbody>
</table>

\( 2x \)
Linear Solver Memory: Structured Sparsity

\[
\min_x \sum_{(i,j) \in F} \| r_{IMU}(x, \Delta \hat{R}_{ij}, \Delta \hat{p}_{ij}, \Delta \hat{v}_{ij}) \|^2 + \sum_{k \in L} \sum_{i \in F_k} \| r_{CAM}(x, l_k, u_{ik}^l, u_{ik}^r) \|^2 + \| r_{PRIOR}(x) \|^2
\]

\[ H\delta = \epsilon \]

Solve a large linear system for \( \delta \)

**H** is:
- Symmetric
- Sparse
  (Black: non zero)

<table>
<thead>
<tr>
<th>Memory size (kB)</th>
<th>Full</th>
<th>Sym</th>
<th>Sym + Sparse</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>703</td>
<td>353</td>
<td>134</td>
</tr>
<tr>
<td>Size factor</td>
<td>2x</td>
<td></td>
<td>5.2x</td>
</tr>
</tbody>
</table>
Linear Solver Memory: Structured Sparsity

\[
\min_x \sum_{(i,j) \in F} \| r_{IMU}(x, \Delta \hat{R}_{ij}, \Delta \hat{p}_{ij}, \Delta \hat{v}_{ij}) \|^2 + \sum_{k \in \mathcal{L}} \sum_{i \in F_k} \| r_{CAM}(x, l_k, u^l_{ik}, u^r_{ik}) \|^2 + \| r_{PRIOR}(x) \|^2
\]

\[
H \delta = \hat{\varepsilon}
\]

Solve a large linear system for \( \delta \)

\( H \) is:
- Symmetric
- Sparse
  (Black: non zero)

Memory size (kB):
- Full: 703
- Sym: 353
- Sym + Sparse: 134

Processing time (ms):
- Full: 48.2
- Sparse: 6.7

Back-substitution: 2x
Cholesky Factorization: 5.2x

7.2x improvement for Sparse over Full
Linear Solver Memory: Structured Sparsity

\[
\min_x \sum_{(i,j) \in \mathcal{F}} \|r_{IMU}(x, \Delta \hat{R}_{ij}, \Delta \hat{p}_{ij}, \Delta \hat{v}_{ij})\|^2 + \sum_{k \in \mathcal{L}} \sum_{i \in \mathcal{F}_k} \|r_{CAM}(x, l_k, u_{ik}, u'_{ik})\|^2 + \|r_{PRIOR}(x)\|^2
\]

\[H\delta = \varepsilon\]

Storing symmetric non-zero values:
5.2x Memory size reduction

Skip processing zeros:
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
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

**Feature tracks two-stage storage:**

5.4x Memory size reduction

**Overhead:**

1 extra cycle access latency
Outline

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

Silicon Specifications

<table>
<thead>
<tr>
<th>Specification</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technology</td>
<td>65nm CMOS</td>
</tr>
<tr>
<td>Chip area (mm²)</td>
<td>4.0 x 5.0</td>
</tr>
<tr>
<td>Logic gates</td>
<td>2,043 kgates</td>
</tr>
<tr>
<td>Resolution</td>
<td>752 x 480</td>
</tr>
<tr>
<td>SRAM</td>
<td>854 kB</td>
</tr>
</tbody>
</table>

Over 250 configurable parameters to adapt to different sensors and environments
Memory Optimization

![Memory Optimization Diagram](image)

- Feature Tracking
- Feature Detection
- Sparse Stereo
- Linear Solver
- IFE
- Shared Memory
- Marginal States

Diagram showing memory size reductions:
- Baseline: 3.5 MB
- Image Compression: 1.5x to 2.4 MB
- Two-stage Storage: 2.6x to 1.6 MB
- Sparse LS Memory: 4.1x to 854 kB
Navion Evaluation

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

Examples of Easy Sequences

Examples of Difficult Sequences

*Dark scenes*  *Motion blur*
Navion Evaluation

• Peak Performance @ Max Configuration
  – VFE: 28 - 171 fps (71 fps average)
  – BE: 16 - 90 fps (19 fps average)
• Average Power Consumption: 24mW
• Trajectory Error: 0.28%

• Real-Time Performance @ Optimized Configuration
  – VF: 20 fps
  – BE: 5 fps
• Average Power Consumption: 2mW
• Trajectory Error: 0.27%
Navion Evaluation

- Average numbers over the 11 EuRoC dataset sequences

<table>
<thead>
<tr>
<th>Platform</th>
<th>Xeon (E5-2667)</th>
<th>ARM (Cortex A15)</th>
<th>Navion (Peak w/ Max Config)</th>
<th>Navion (Real-time w/ Optimized Config)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trajectory Error (%)</td>
<td>0.22%</td>
<td></td>
<td>0.28%</td>
<td>0.27%</td>
</tr>
<tr>
<td>Camera rate (fps)</td>
<td>63</td>
<td>19</td>
<td>71</td>
<td>20</td>
</tr>
<tr>
<td>Keyframe rate (fps)</td>
<td>12</td>
<td>2</td>
<td>19</td>
<td>5</td>
</tr>
<tr>
<td>Average Power (W)</td>
<td>27.9</td>
<td>2.4</td>
<td>0.024</td>
<td>0.002</td>
</tr>
<tr>
<td>Energy (mJ/KF)</td>
<td>3,638</td>
<td>1,573</td>
<td>2.3</td>
<td>0.7</td>
</tr>
</tbody>
</table>
**Navion Evaluation**

- Average numbers over the 11 EuRoC dataset sequences

<table>
<thead>
<tr>
<th>Platform</th>
<th>Xeon (E5-2667)</th>
<th>ARM (Cortex A15)</th>
<th>Navion (Peak w/ Max Config)</th>
<th>Navion (Real-time w/ Optimized Config)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trajectory Error (%)</td>
<td>0.22%</td>
<td>0.28%</td>
<td>0.27%</td>
<td></td>
</tr>
<tr>
<td>Camera rate (fps)</td>
<td>63</td>
<td>19</td>
<td>71</td>
<td>20</td>
</tr>
<tr>
<td>Keyframe rate (fps)</td>
<td>12</td>
<td>2</td>
<td>19</td>
<td>5</td>
</tr>
<tr>
<td>Average Power (W)</td>
<td>27.9</td>
<td>2.4</td>
<td>0.024</td>
<td>0.002</td>
</tr>
<tr>
<td>Energy (mJ/KF)</td>
<td>3,638</td>
<td>1,573</td>
<td>2.3</td>
<td>0.7</td>
</tr>
</tbody>
</table>

**Navion Energy:**

- 684x or 2,247x less than embedded ARM CPU
- 1,582x or 5,197x less than server Xeon CPU
Navion System Demo

https://youtu.be/X5VZkPo_704
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
Navion

http://navion.mit.edu


Other Related Works

Object Detection w/ Deformable Parts Models
[Suleiman et al., VLSI 2016]

Eyeriss: Deep Neural Networks
[ISSCC 2016, ISCA 2016]
Eyeriss v2
https://arxiv.org/abs/1807.07928

Semantic Understanding

Website: http://www.rle.mit.edu/eems/
Questions

http://navion.mit.edu/