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

Amr Suleiman, Zhengdong Zhang, Luca Carlone, Sertac Karaman, Vivienne Sze
Massachusetts Institute of Technology, MA, USA

Abstract

This paper presents Navion, an energy-efficient accelerator for visual-inertial odometry (VIO) that enables autonomous navigation of miniaturized robots (e.g., nano drones), and virtual/augmented reality on portable devices. The chip uses inertial measurements and mono/stereo images to estimate the drone's trajectory and a 3D map of the environment. This estimate is obtained by running a state-of-the-art algorithm based on non-linear factor graph optimization, which requires large irregularly structured memories and heterogeneous computation flow. To reduce the energy consumption and footprint, the entire VIO system is fully integrated on chip to eliminate costly off-chip processing and storage. This work uses compression and exploits both structured and unstructured sparsity to reduce on-chip memory size by 4.1x. Parallelism is used under tight area constraints to increase throughput by 43%. The chip is fabricated in 65nm CMOS, and can process 752x480 stereo images at up to 171 fps and inertial measurements at up to 52 kHz, while consuming an average of 24mW. The chip is configurable to maximize accuracy, throughput and energy-efficiency across different environments. To the best of our knowledge, this is the first fully integrated VIO system in an ASIC.

Keywords: VIO, localization, mapping, nano drones, navigation.

Introduction

Simultaneous Localization and Mapping (SLAM) algorithms estimate the trajectory of a sensing device (e.g., robot, drone, self-driving car) while reconstructing a map of the environment [1]. The trajectory is the collection of the drone's state $\boldsymbol{x}$ (e.g., position $\boldsymbol{P}$, orientation $\boldsymbol{R}$) over time. These states are estimated based on measurements of the environment.

VIO is a special instance of SLAM where visual (camera) and inertial measurement unit (IMU) data are used for estimation [2] (Fig. 1). The frontend extracts measurements from the sensor inputs at the sensor rate (up to 171 fps for stereo and up to 52 kHz for IMU) and the backend updates the states at the keyframe (KF) rate of up to 90 fps. The vision frontend (VFE) tracks several landmarks (L_i) based on their 2D projections in the images (i.e., features f_i), and generates 3D feature tracks. Meanwhile, the IMU frontend (IFE) preintegrates the IMU's gyroscope and accelerometer measurements between KFs to give an estimate of the drone's state $\boldsymbol{x}$ [3]. The backend (BE) performs a factor graph optimization to fuse IMU and camera data by solving a high-dimensional non-linear least squares problem, whose solution describes the drone's trajectory and a sparse 3D landmarks map. The optimization is performed on measurements within a time window, referred to as the horizon.

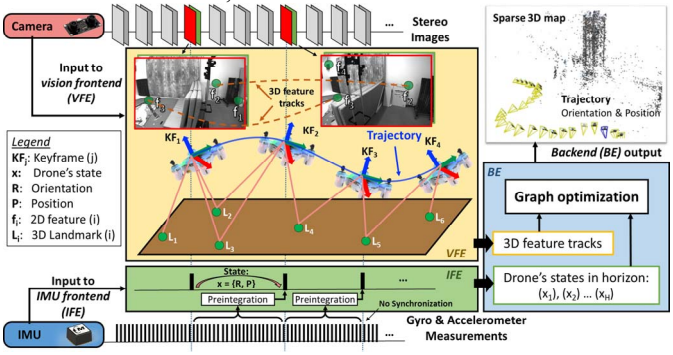


Fig. 1 Visual-Inertial Odometry algorithm

Prior ASICs implemented only partial navigation systems [4, 5]. For example, [4] presents a very accurate stereo engine, which is only a part of the whole VIO pipeline. In addition, [5] uses large off-chip resources for storage and to perform all image processing. This paper presents a fully integrated VIO implementation. Several optimizations are carried out to minimize the chip power and footprint for energy-constrained applications, while maintaining accuracy. Furthermore, configurable parameters (e.g., number of features, horizon length, etc.) enable efficient tradeoff between accuracy, throughput and power across different environments. The chip can also be configured to support different cameras and IMU sensors.

Architecture Overview

Fig. 2 shows the chip architecture and a timing diagram of the processing modules. VFE processes images with fixed-point arithmetic, supporting both mono and stereo cameras. All image processing is done on-chip, including Harris corner feature detection (FD), Lucas-Kanade feature tracking (FT), and block matching sparse stereo (SS). Undistortion & rectification (UR) is also included to eliminate any off-chip processing. RANSAC is used to remove outliers from the tracked features. These modules operate in parallel to increase throughput by 43% at the cost of 77kB additional memory. Most of the VFE, except FT, is active only at KFs; clock gating reduces the power consumption by over 10%. The IFE computes the preintegration of the gyroscope and accelerometer measurements. It accounts for only 2.4% area and 1.2% power of the chip.

The BE fuses VFE and IFE data by solving a non-linear factor graph optimization at the KF rate using Gauss-Newton method in double precision arithmetic. The main modules in BE operate serially as shown in the timing diagram in Fig. 2. The factor graph is built in the *Graph* memory. It defines a non-linear objective function of the drone's state using the summarized measurements from VFE and IFE. The problem is then linearized and solved in-place in the *Linear Solver* memory to determine how the state should be updated based on the measurements in the current horizon. These updates are applied to the states in the *Horizon States* memory. Marginalization is used to remove old measurements from the optimization [3]. The complex finite state machine (FSM) controlling the BE is divided into smaller FSMs to enable sharing for reduced area (e.g., matrix operations, Cholesky (CH), Back-substitution (BS), etc.). The *Shared* memory and register file serve as a memory hierarchy to reduce data movement energy of the intermediate data.

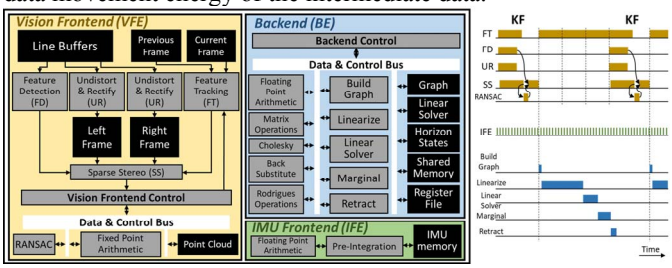


Fig. 2 Chip architecture and timing diagram

Image Compression

VFE has 4 full frame memories (Fig. 2) to support FT and SS and to avoid re-reading pixels from off-chip, requiring a total of 1410kB. Lossy image compression is used (Fig. 3), with a block-wise quantization, such that each 4x4 block of pixels uses

only 26 bits to store its dynamic range (1b/pixel), threshold (5b) and minimum value (5b). With a small overhead and a 1.4kB line buffer, compression reduces the frame memory size by 4.4x and power by 4.9x. Compressed frames are not used in FD due to blocking and quantization noise as shown in Fig. 3. The image compression increases the average error by only 0.06%.

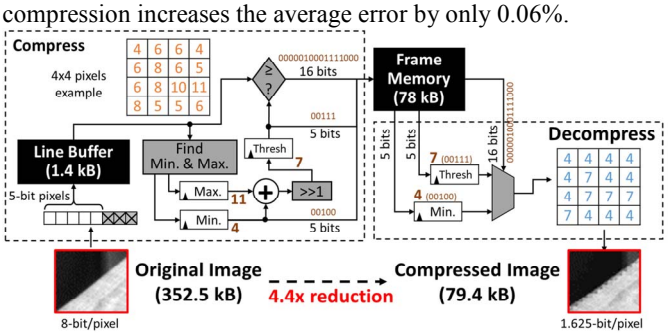


Fig. 3 Lossy image compression on Frame Memories

Feature Tracks Data Structure

Fig. 4 shows the data structure used to store the 3D feature tracks, which contains each feature's 3D position and its corresponding KF ID. It accounts for 88% of the *Graph* memory. Each feature has an *age* indicating how many KFs it appears in before it is discarded. The chip supports a configurable feature age with a maximum of 10 KFs. Storing tracks in one memory requires 962kB as shown in Fig. 4. However, this 40,000-entry memory is sparsely populated with a maximum of 4,000 measurements depending on the image sequence. A two-stage memory architecture is used instead: the first *sparse* memory stores the KFs ID with pointers to the second *dense* memory, which stores the 3D positions. This reduces the graph memory size by 5.4x, and increases access latency by only one cycle.

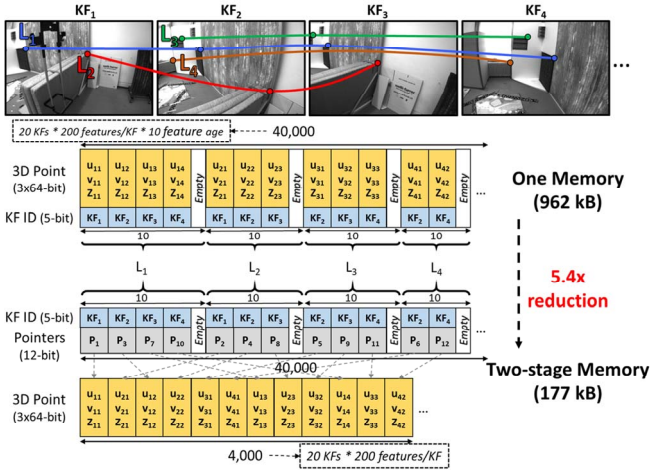


Fig. 4 Two-stage storage of feature tracks in Graph Memory

Linear Solver Sparsity

After linearizing the optimization problem, a linear solver (LS) is used to solve for state update $\boldsymbol{\Delta x}$ given $\boldsymbol{H\Delta x=\varepsilon}$ using CH and BS as shown in Fig. 5. The Hessian matrix $\boldsymbol{H}$ and $\boldsymbol{\varepsilon}$ describe how VFE and IFE data affect each of the 20 KFs in the horizon, where each KF has 15 variables, hence the matrix size of 300x300. Matrices $\boldsymbol{H}$, and its Cholesky decomposition $\boldsymbol{L}$, reuse the same memory for in-place computation. The memory size can be reduced by 2x by exploiting the symmetry in $\boldsymbol{H}$ and $\boldsymbol{L}$, and storing only the matrix lower triangle. Fig. 5 shows the fixed sparsity of $\boldsymbol{H}$ and $\boldsymbol{L}$. A memory wrapper masks the read/write operations in the zero locations and generates memory addresses based on matrix coordinates. Sparse storage reduces the linear solver memory size by 5.2x, with an area overhead less than 1%. The LS also exploits sparsity for a 7.2x speed up by skipping the zero locations when traversing the matrix.

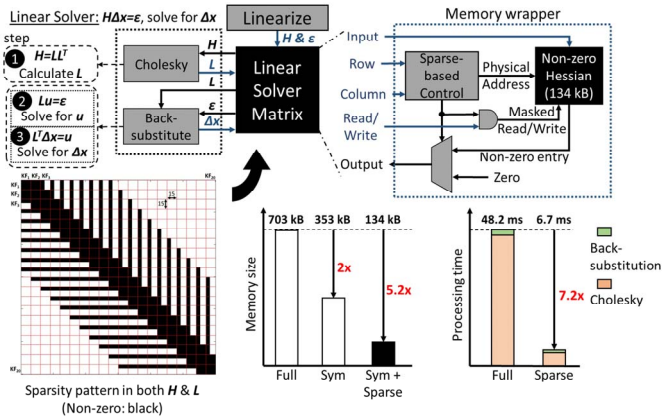


Fig. 5 Exploit sparsity to reduce Linear Solver Memory

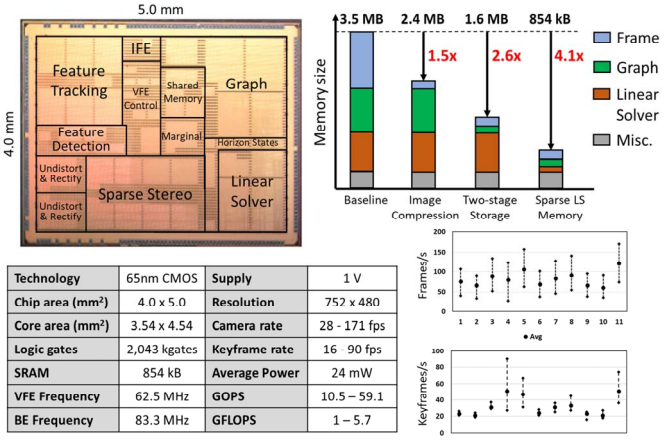
Implementation and Results

The chip is implemented in 65nm CMOS. It is tested using the EuRoC MAV dataset [6], which is a challenging drone dataset with different sequences having high-speed rotations, low lighting conditions and blurred images. The chip can process 752x480 stereo images at a rate of 28 - 171 fps in real-time, depending on the sequence, operating at 1V with a measured average power consumption of 24mW. It updates the states and the sparse 3D map at KF rate of 16 - 90 fps. Fig. 6 shows the die photo and the chip specifications, along with a plot of the throughput range of the 11 sequences in the dataset. Fig. 7 shows a trajectory output example compared to the ground truth with an average error of 0.28%. Finally, compared to previous work, this work uses 2.5x less memory than [2], and consumes less power than [5], which performs only BE, while this work performs full VIO processing.

Acknowledgement This work was funded in part by the AFOSR YIP FA9550-16-1-0228 and by the NSF CAREER 1350685.

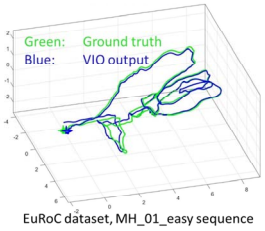
References

[1] C. Cadena et al., IEEE Trans. Robotics vol. 32 no. 6 pp. 1309-1332, 2016.
[2] Z. Zhang et al., Proceedings of Robotics: Science and Systems, 2017.
[3] C. Foster et al., IEEE Trans Robotics, vol. 33, no. 1, pp. 1-21, 2016.
[4] Z. Li et al., ISSCC, pp. 62-63, 2017.
[5] I. Hong et al., JSSC, vol. 50, no. 11, pp. 2513-2523, 2015.
[6] M. Burri et al., International Jr. Robotics, vol. 35, no. 10, 2016.



Technology	65nm CMOS	Supply	1 V
Chip area (mm²)	4.0 x 5.0	Resolution	752 x 480
Core area (mm²)	3.54 x 4.54	Camera rate	28 - 171 fps
Logic gates	2,043 kgates	Keyframe rate	16 - 90 fps
SRAM	854 kB	Average Power	24 mW
VFE Frequency	62.5 MHz	GOPS	10.5 – 59.1
BE Frequency	83.3 MHz	GFLOPS	1 – 5.7

Fig. 6 Die photo and summary of the chip specifications



	[2]			[5]	This work
Platform	Xeon	ARM	FPGA	ASIC (65nm)	ASIC (65nm)
VFE	✓	✓	✓	x	✓
BE	✓	✓	✓	✓	✓
Error rate	0.20%	0.20%	0.24%	NA	0.28%
VFE max fps	21	5	20	NA	171
BE max fps	13	3	5	92	90
Power (W)	26.10	2.33	1.46	0.027	0.024
Memory (MB)	16	2.5	2.1	0.09	0.83

Fig. 7 Comparison table and an example of the chip output