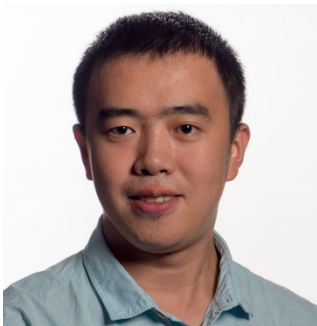# Visual-Inertial Odometry on Chip: An Algorithm-and-Hardware Co-design Approach

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

Massachusetts Institute of Technology

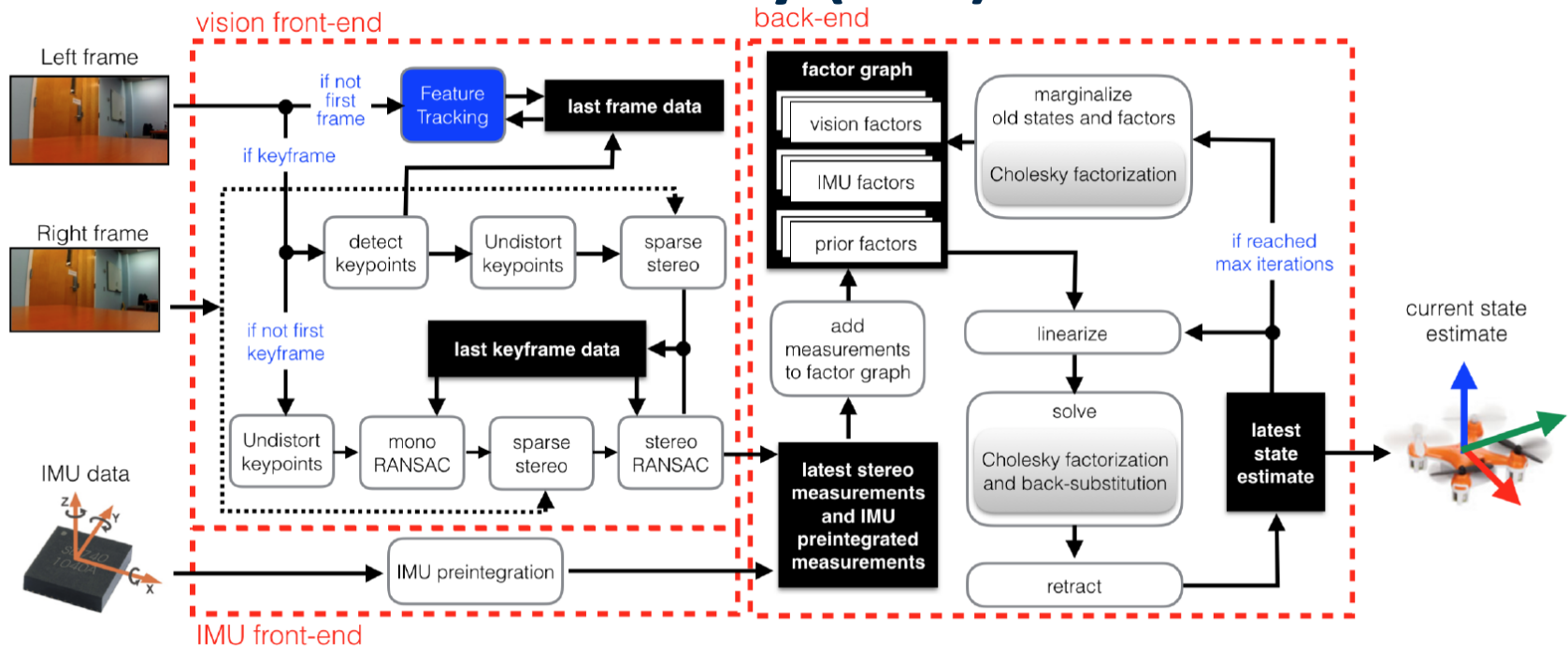navion.mit.edu

Nano Unmanned Aerial Vehicles (UAVs)

Consumer Electronics

Search and Rescue
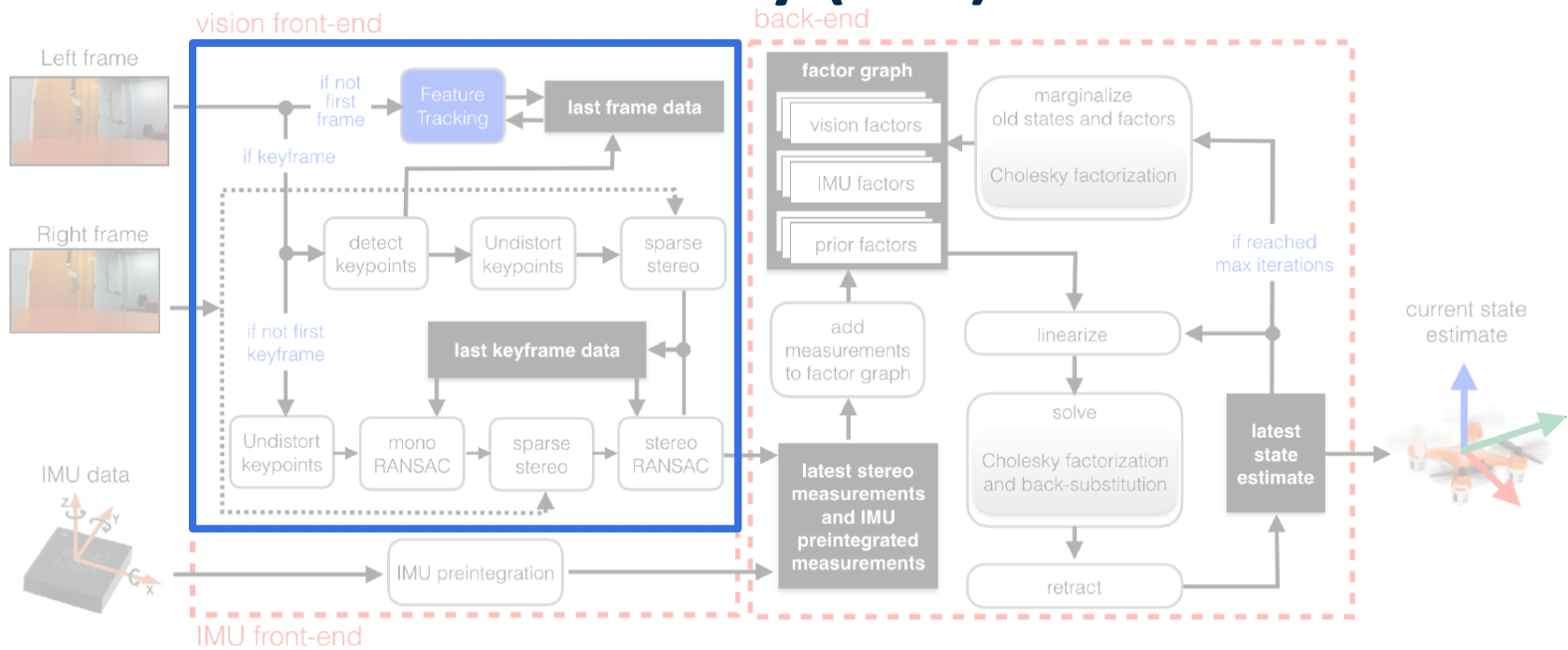
Fully-autonomous navigation without a map is essential

Visual Inertial Odometry (VIO)

Key component of autonomous navigation without a map

Visual Inertial Odometry (VIO)
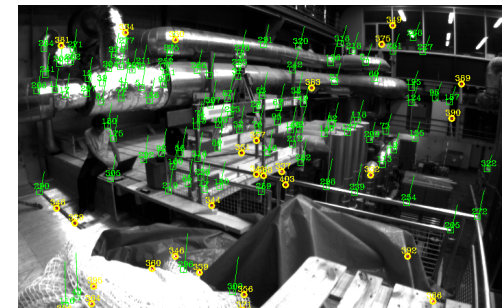motion estimation from camera and inertial sensor

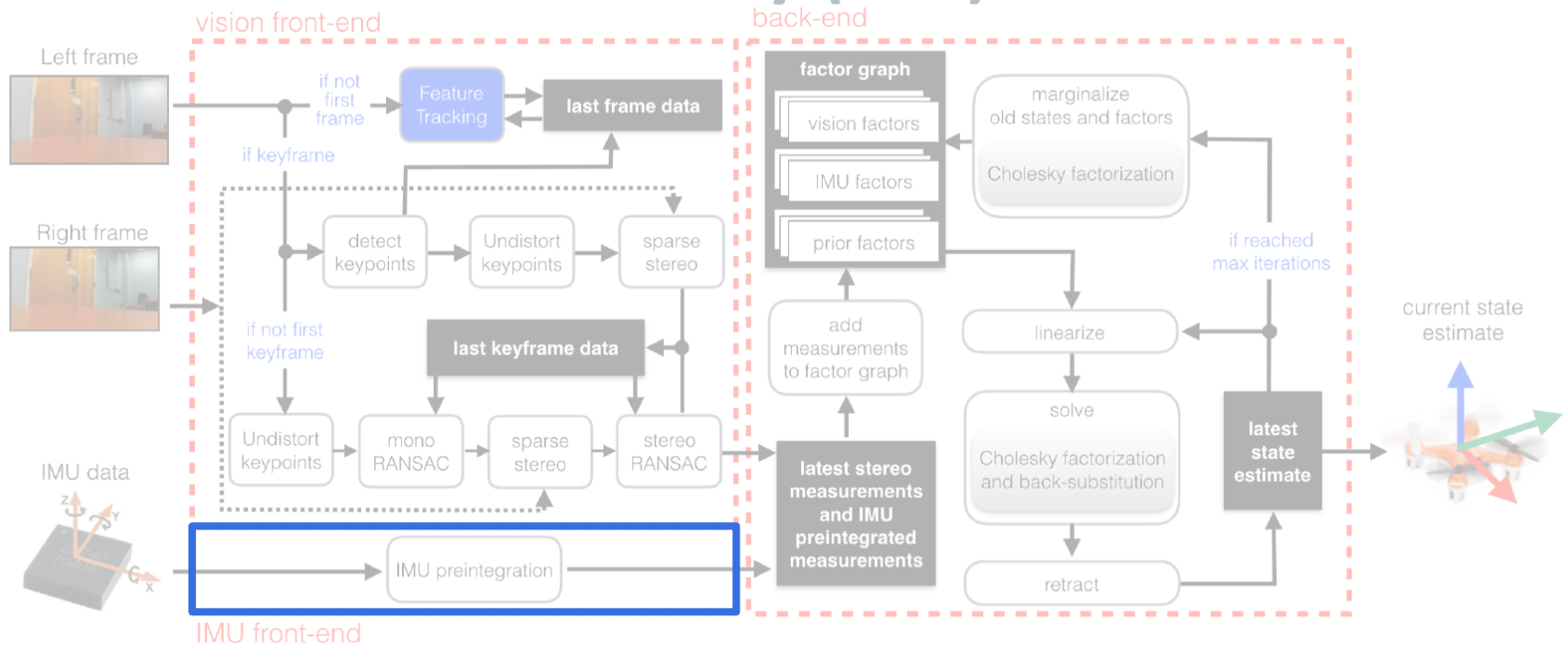# Visual Inertial Odometry (VIO)

Vision Frontend

Process Stereo Frame

Robust Tracking

Visual Inertial Odometry (VIO)

IMU Frontend
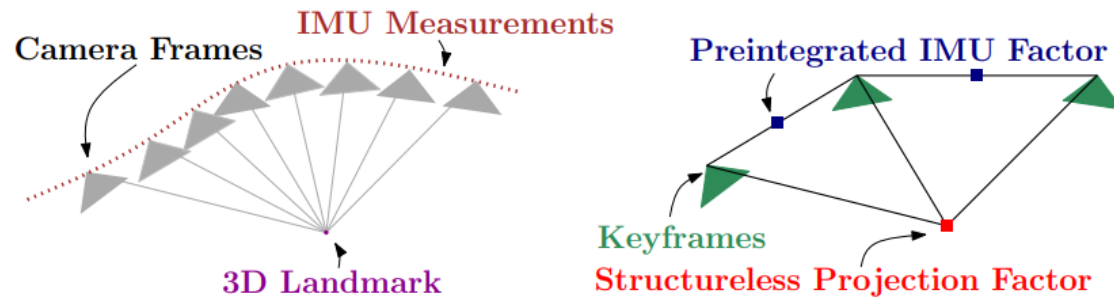
IMU Preintegration by Forster, et, al.

[1] Forster, et, al. IMU Preintegration on Manifold for Efficient Visual-Inertial Maximum-a-Posteriori Estimation, RSS 2015
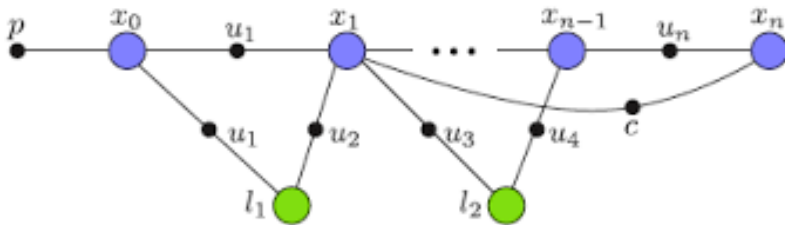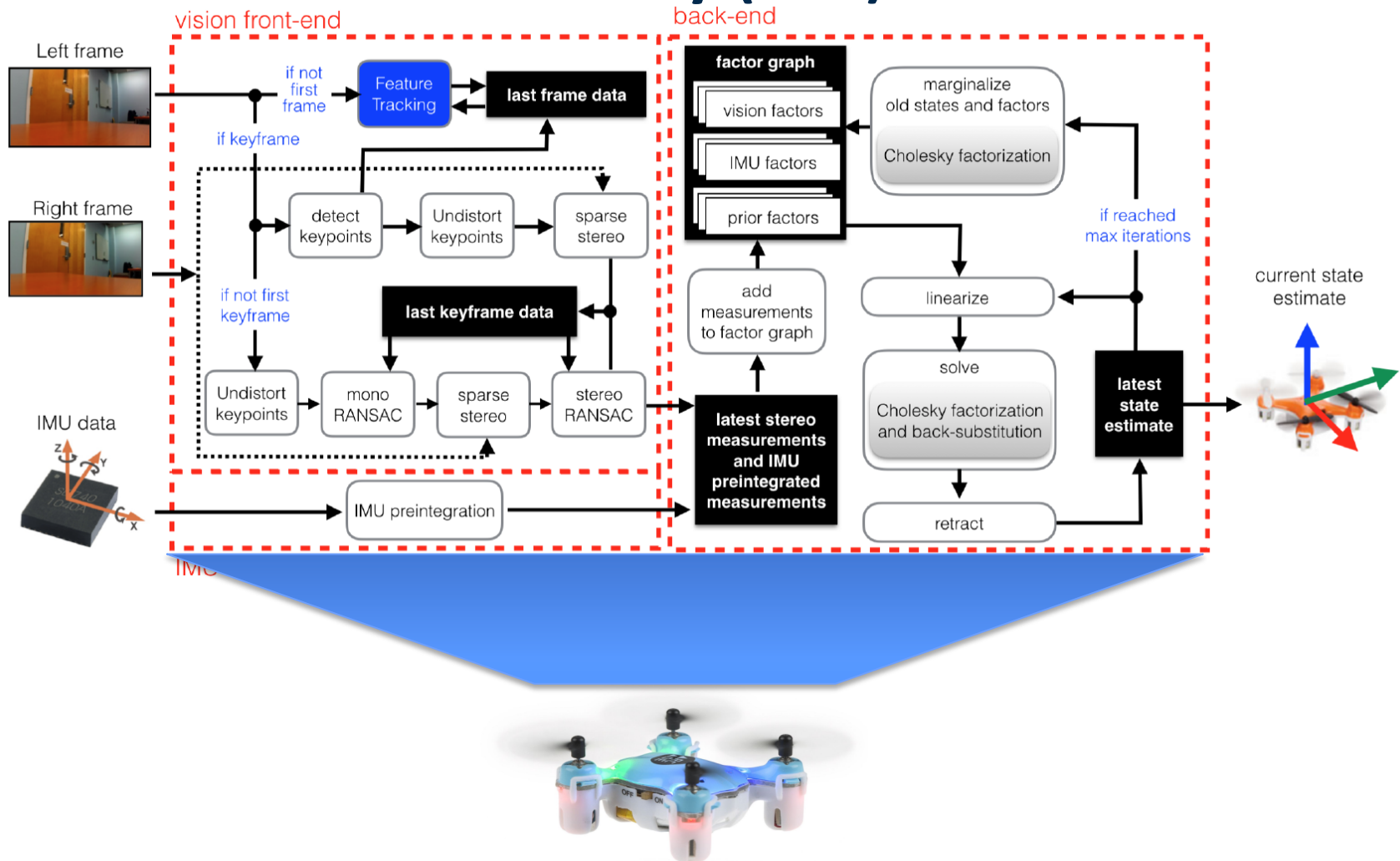
Visual Inertial Odometry (VIO)
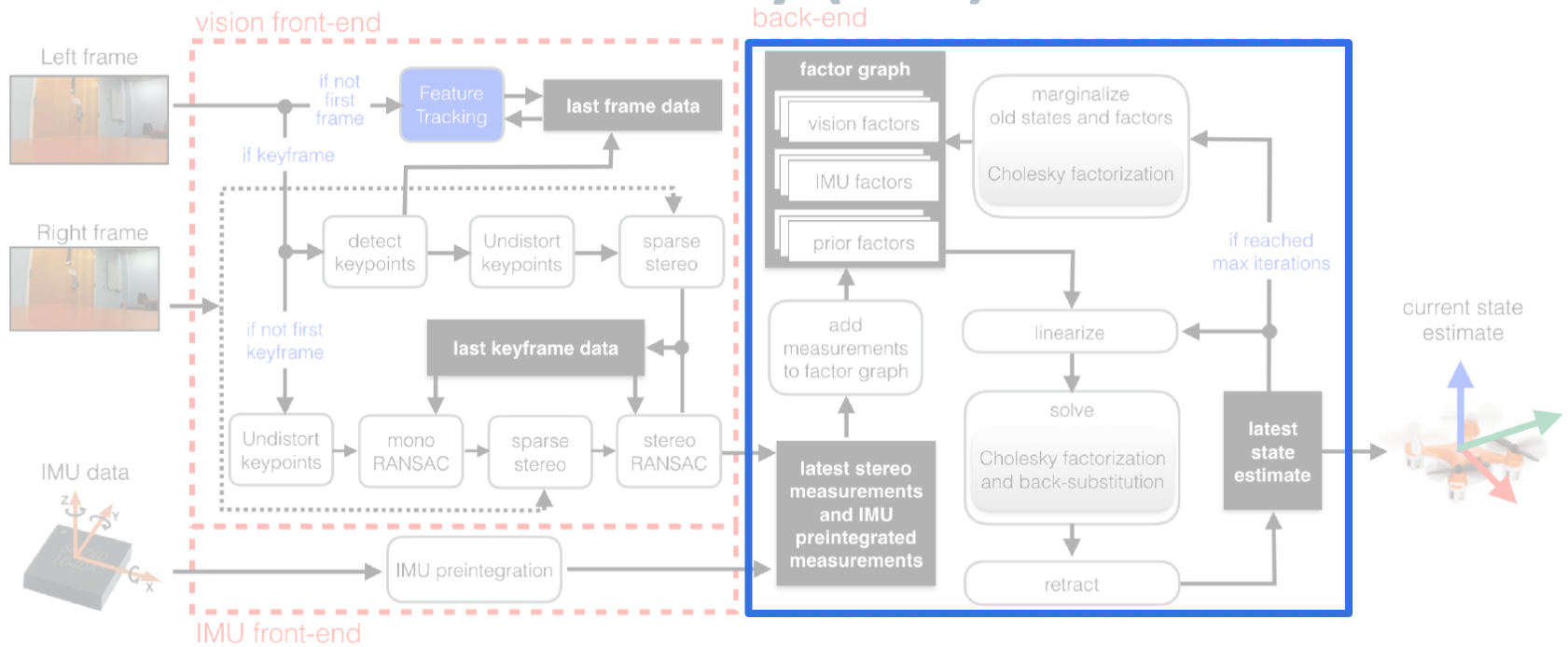
Backend

Factor graph based optimization

Output trajectory and 3D point cloud

Visual Inertial Odometry (VIO)

Goal: Run VIO locally on the nano/pico UAVs

Challenge: Power and Speed
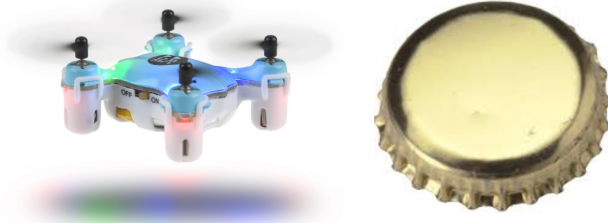
Bottle-cap-sized **nano** UAV

Goal
- **Power**: < 2 W
- **Keyframe rate**: > 5 fps

Challenge: Power and Speed

Bottle-cap-sized **nano** UAV

Goal
- **Power**: < 2 W
- **Keyframe rate**: > 5 fps

	Goal	Desktop CPU	Embedded CPU
Keyframe rate	> 5 fps	**8.4 fps**	**2 fps**
Power	< 2 W	**28.2 W**	**2.5 W**
		Too high power	**Too slow**

General Purpose Computing not good enough!

Our Choice:
Low-Power Specialized Hardware

FPGA
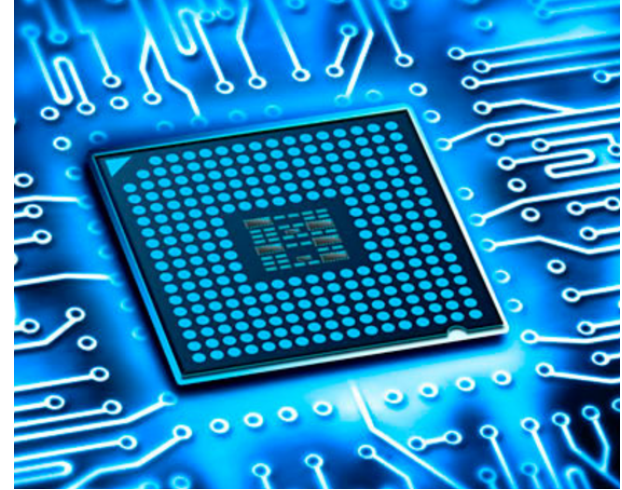
ASIC

Low power if only use on-chip memory (e.g., 3MB on FPGA)

Standard VIO algorithms do not fit,
we need an **algorithm-and-hardware co-design** approach

Algorithm-and-Hardware Co-design

Step 1: **Specify Performance and Resource Goals**

| Power | Form factors | Accuracy | Speed |

Mini-ITX

Standard-ATX

Step 2: **Define Design Space**, *D*

$$D = H \times A \times I \times P$$

Hardware choices   Algorithm choices   Implementation choices   Parameter choices

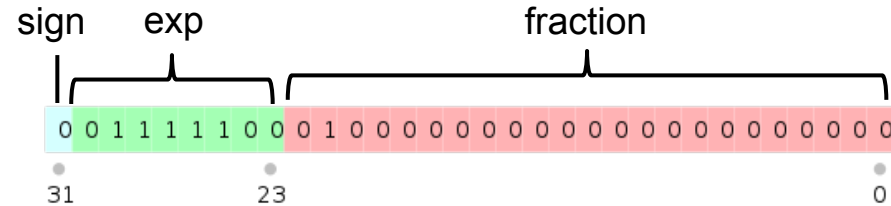---

Step 3: Explore Design Space via **Iterative Split Co-Design**

Example 1
Reduced Precision of Data Representation

Cost

≪

Fixed point

sign   exp   fraction

0 0 1 1 1 1 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

31   23   0

Floating point

estimation error

Resources

0.4   1.2

0.3   1

0.2   0.8

0.1   0.6

                                       0.4
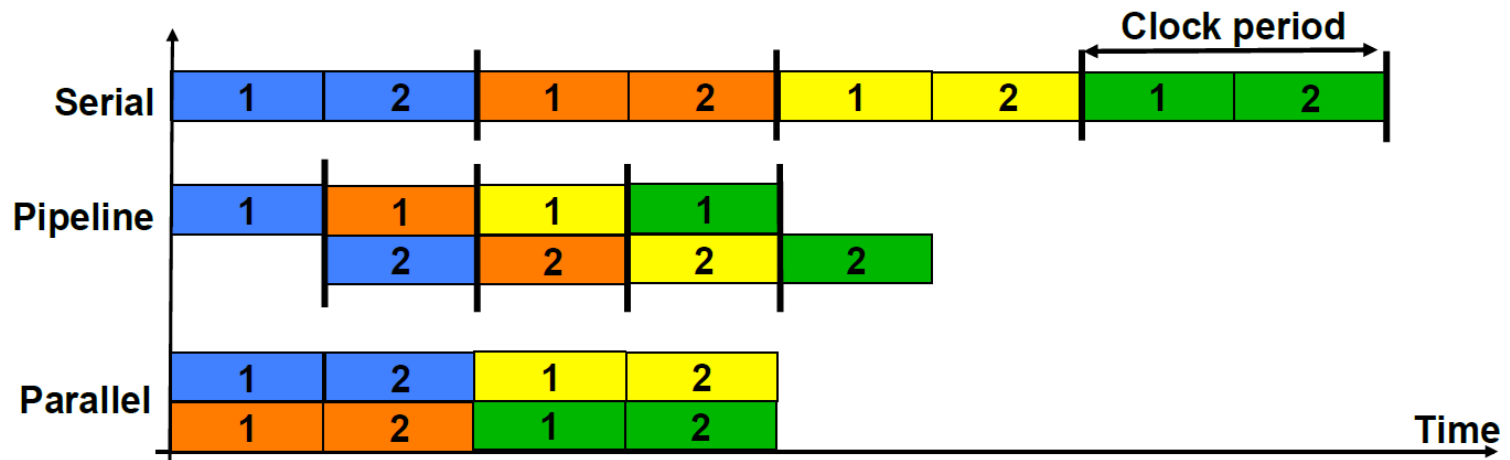
32   21   16   12

Number of bits
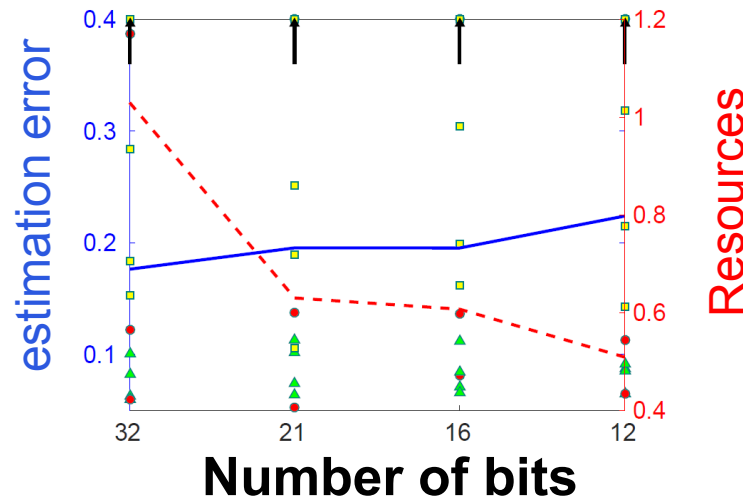
Reduce vision front-end to 16 bits fixed-point
for efficient **accuracy vs. memory trade-off**

IIliī

Example 2
Hardware Design Choices

$+, \times \qquad \div, \sqrt{}$

Avoid division and sqrt as much as possible

Parallelism and pipelining increase speed, but also increase power/resources. **Use carefully!**

Many Other Design Choices!!

$$D = H \times A \times I \times P$$

H **Hardware choices**	A **Algorithm choices**	I **Implementation choices**	P **Parameter choices**
desktop-CPU	Tracking?	On the fly computation	Max feature num
embedded-CPU	RANSAC?	Pipelining	Template size
embedded-GPU	Sparse vs dense solver?	Parallelism	Max tracking levels
FPGAs	SVD in triangulation?	Reduced precision	Intra-keyframe time
ASICs	GN vs LM?	Low cost arithmetic	Nr. GN iterations
	Relinearization for Marginalization?	...	...
	...		

IIiT

Result: Co-Designed VIO on FPGA
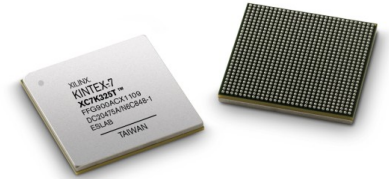
	Goal	d-CPU	e-CPU	**FPGA (ours)**
Error (m)	≤ 0.2	**0.15**	**0.15**	**0.19**
Keyframe rate (fps)	≥ 5	**8.4**	**2**	**5**
Power (W)	~2	**28.2**	**2.5**	**1.5**

Too high power   **Too slow**   *Best of both worlds!*

The co-designed FPGA implementation only requires **2.1 MB** memory!

Contributions

- **Systematically explore the co-design space of VIO** towards a design that meets the desired trade-off

- **A VIO implementation on FPGA** that has **20 fps tracking**, **5 fps keyframe** and only requires **2.1 MB memory** and **consumes 1.5 W**
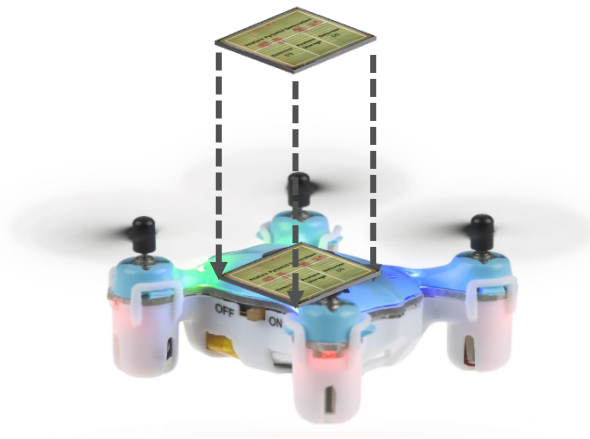
ASIC coming soon!

Stay tuned: navion.mit.edu