AICAS 2022

Robotic Computing on FPGAs: Current Progress, Research Challenges, and Opportunities

Zishen Wan[1], Ashwin Lele[1], Bo Yu[2], Shaoshan Liu[2], Yu Wang[3], Vijay Janapa Reddi[4], Cong (Callie) Hao[1], Arijit Raychowdhury[1]

[1] *Georgia Institute of Technology, GA, USA*
[2] *PerceptIn, CA, USA*
[3] *Tsinghua University, Beijing, China*
[4] *Harvard University, MA, USA*

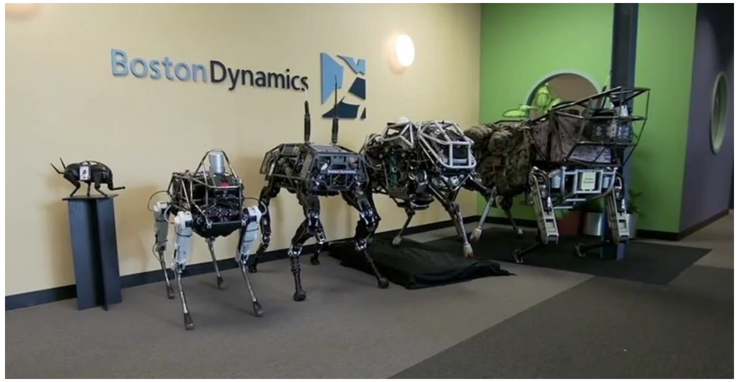
2022 IEEE International Conference on Artificial Intelligence Circuits and Systems
Virtual & Hybrid Conference

Motivation: Autonomous Systems

Drones

Self-Driving Cars

Robots



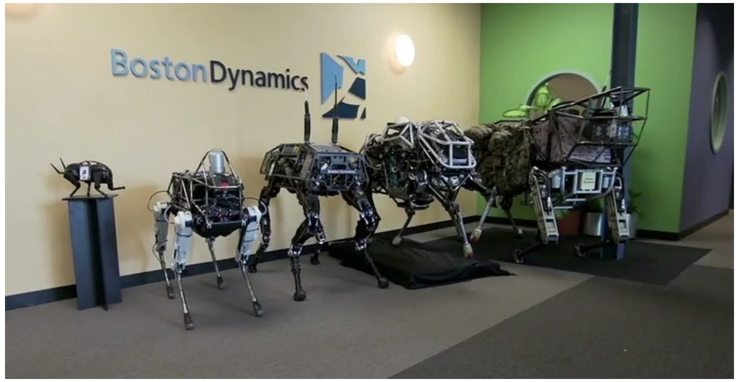
 [Source: V. Sze]

Motivation: Autonomous Systems

Drones

Self-Driving Cars

Robots

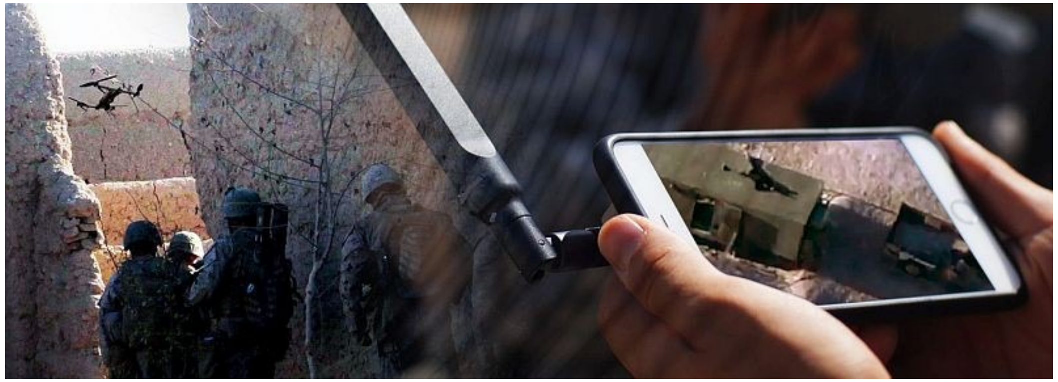


Applications

Search & Rescue

Package Delivery

Surveillance



[Source: V. Sze]

Outline

- Robotic Computing Systems
- Current Progress and Design Techniques
- Research Challenges and Future Directions

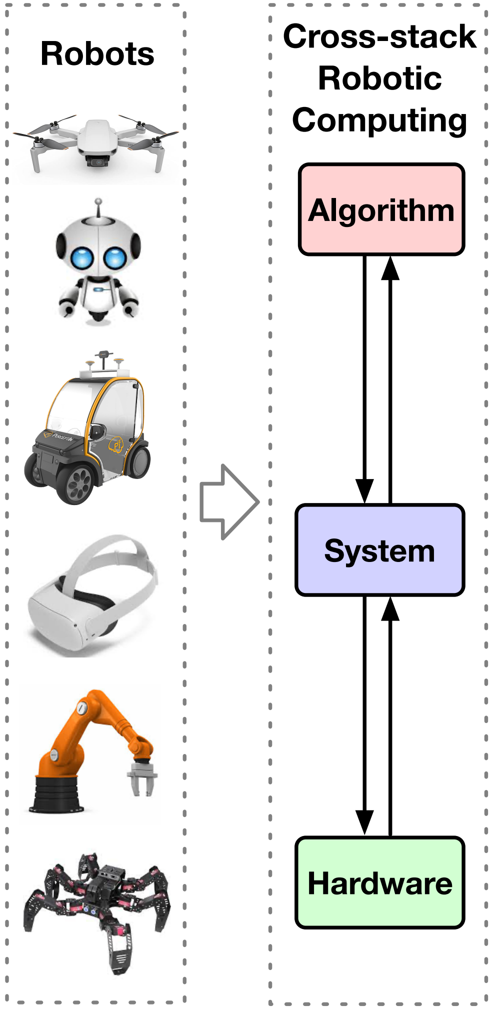
Outline

- Robotic Computing Systems
- Current Progress and Design Techniques
- Research Challenges and Future Directions

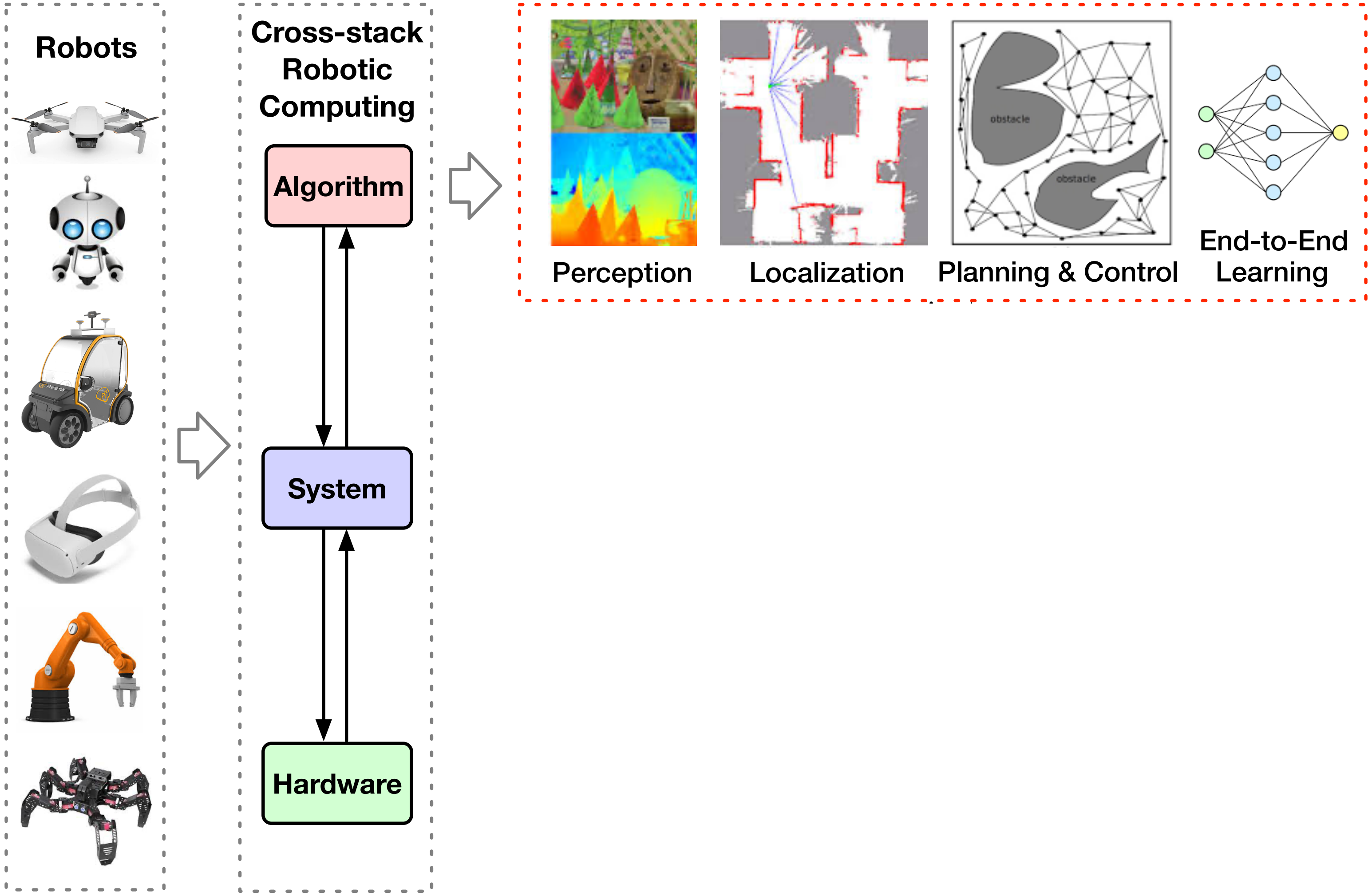
Cross-Layer Robotic Computing System

Robots

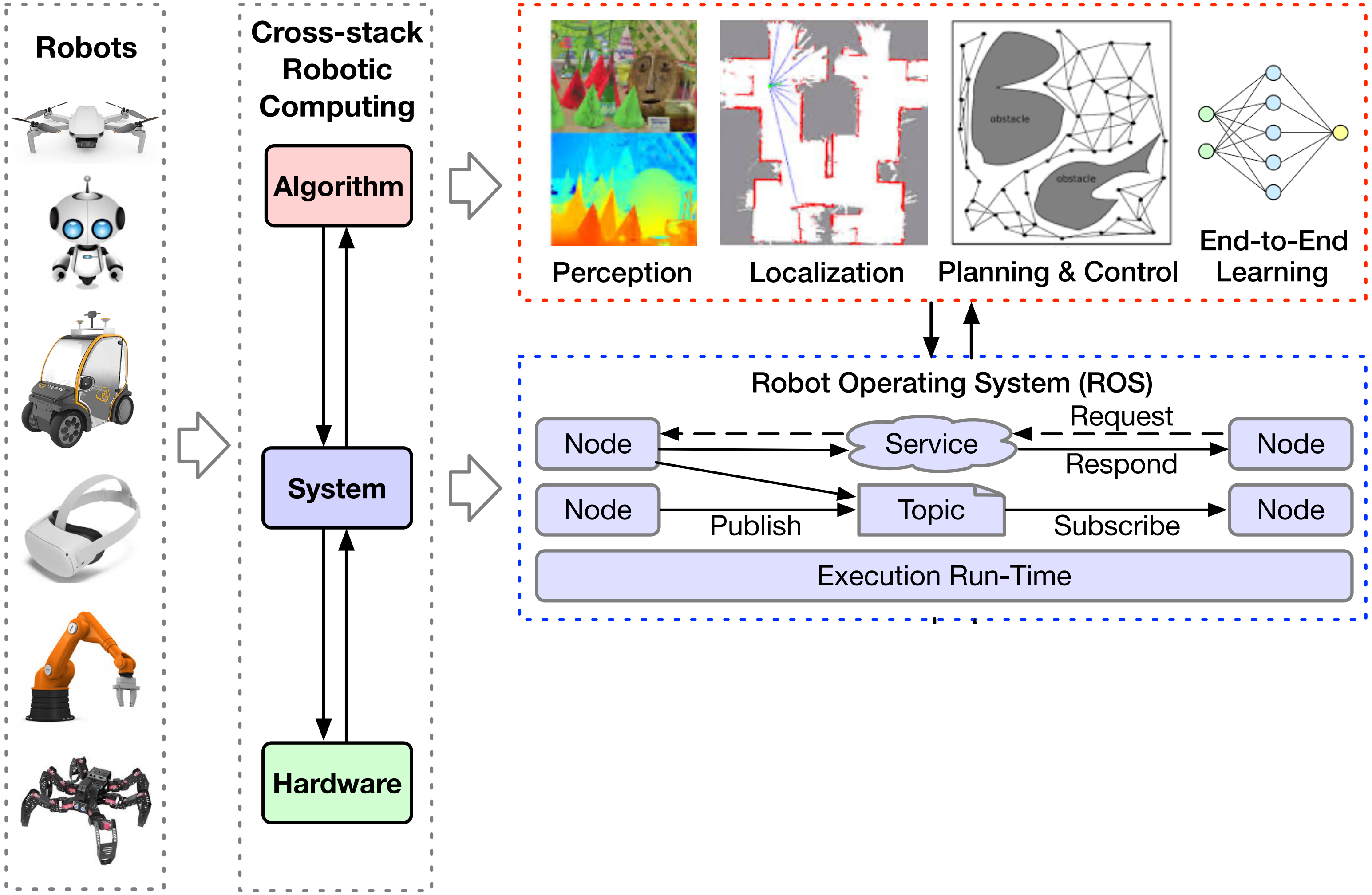
Cross-Layer Robotic Computing System



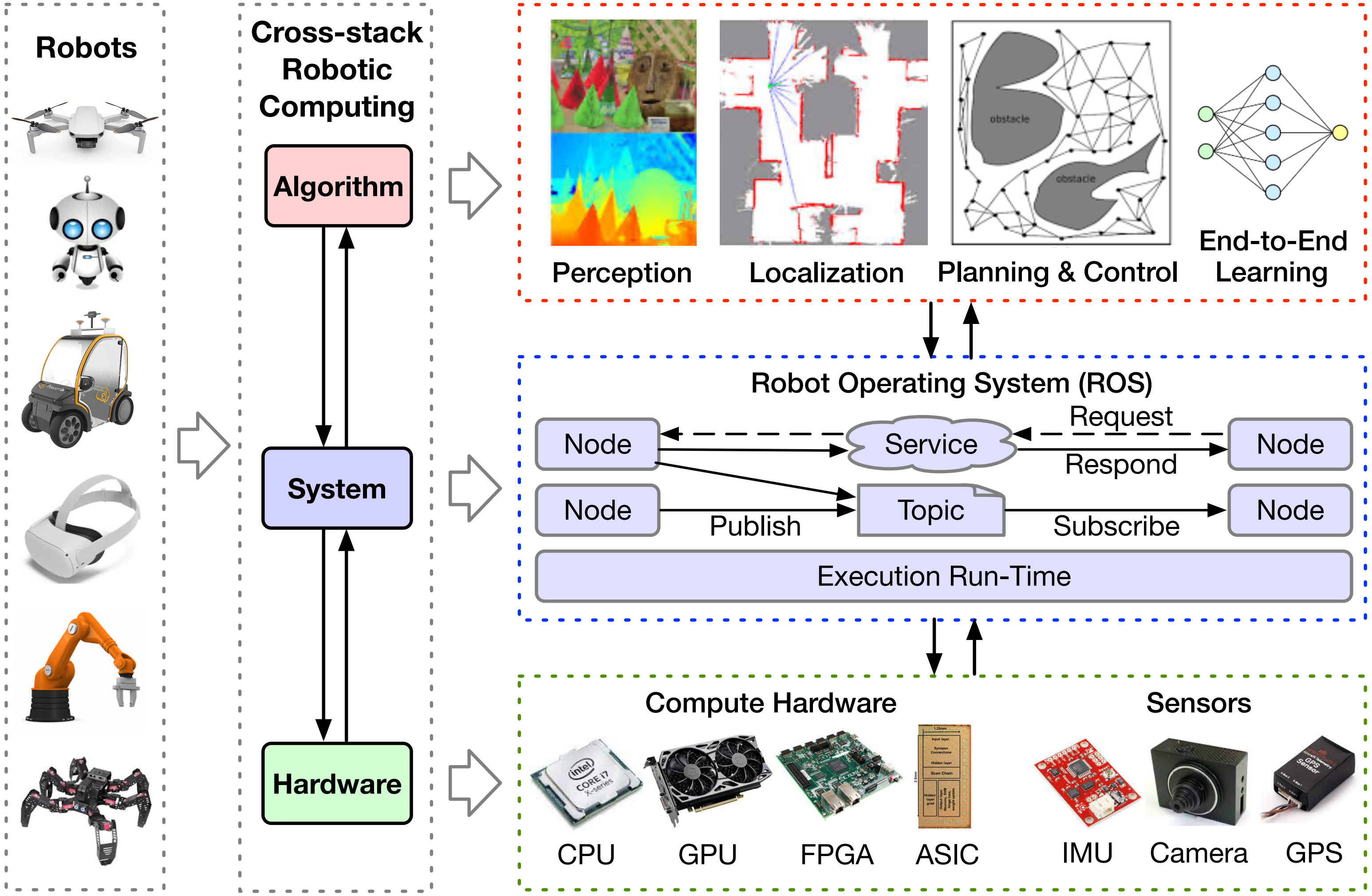
Cross-Layer Robotic Computing System



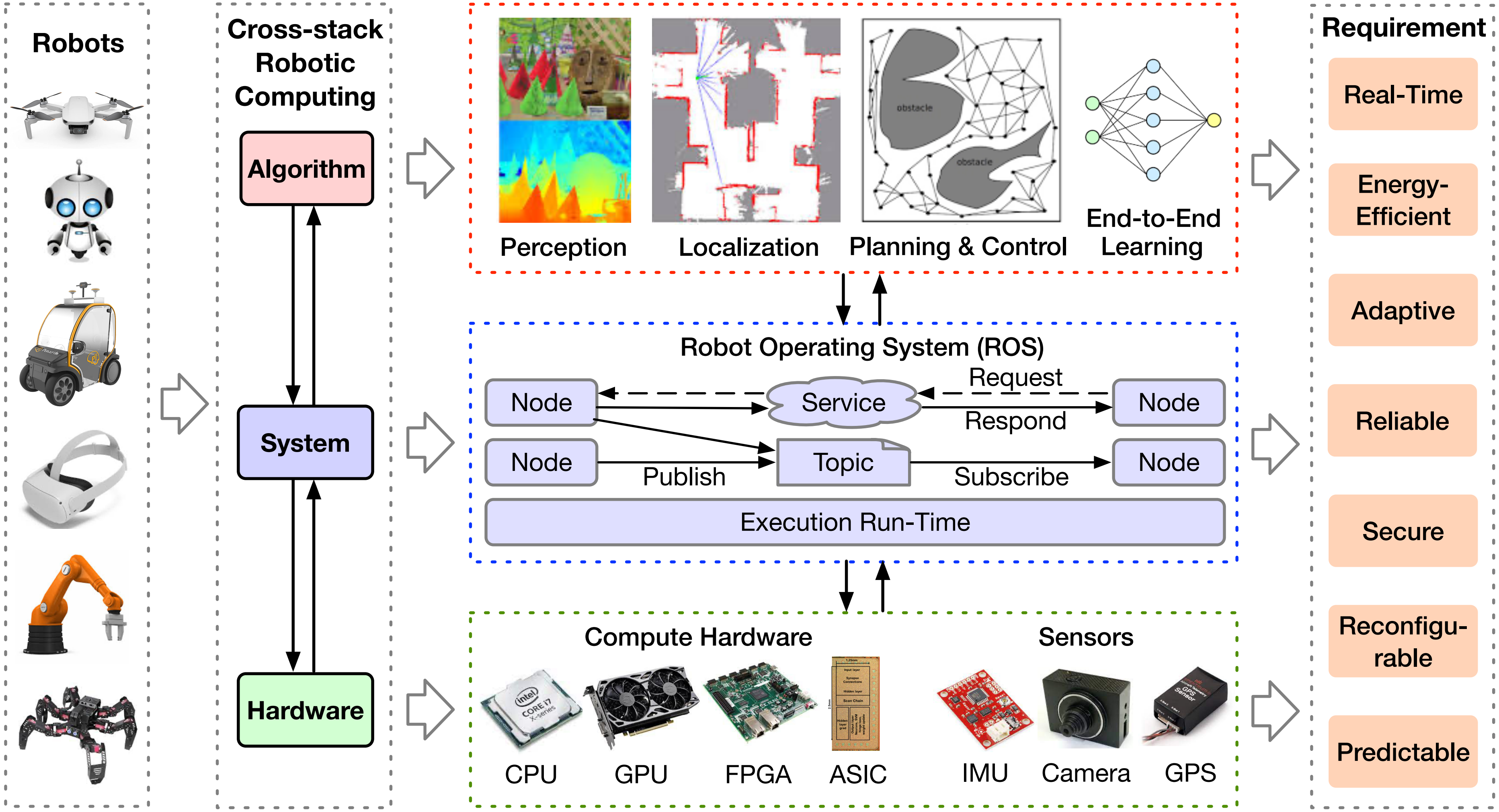
Cross-Layer Robotic Computing System



Cross-Layer Robotic Computing System

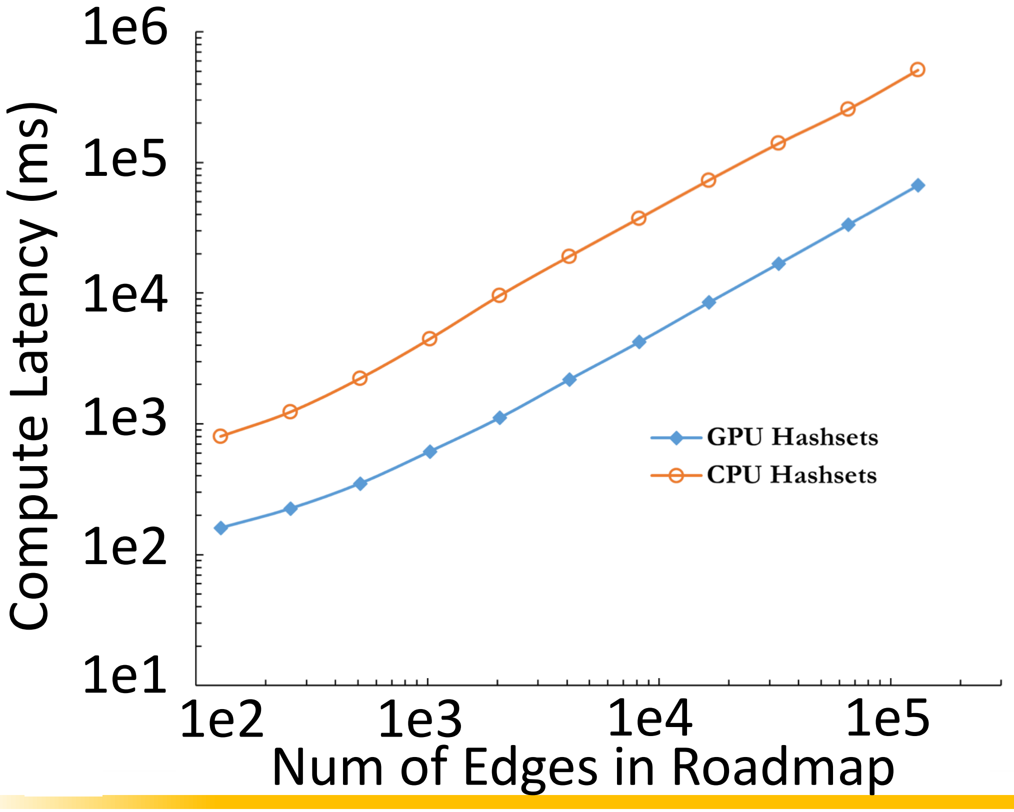


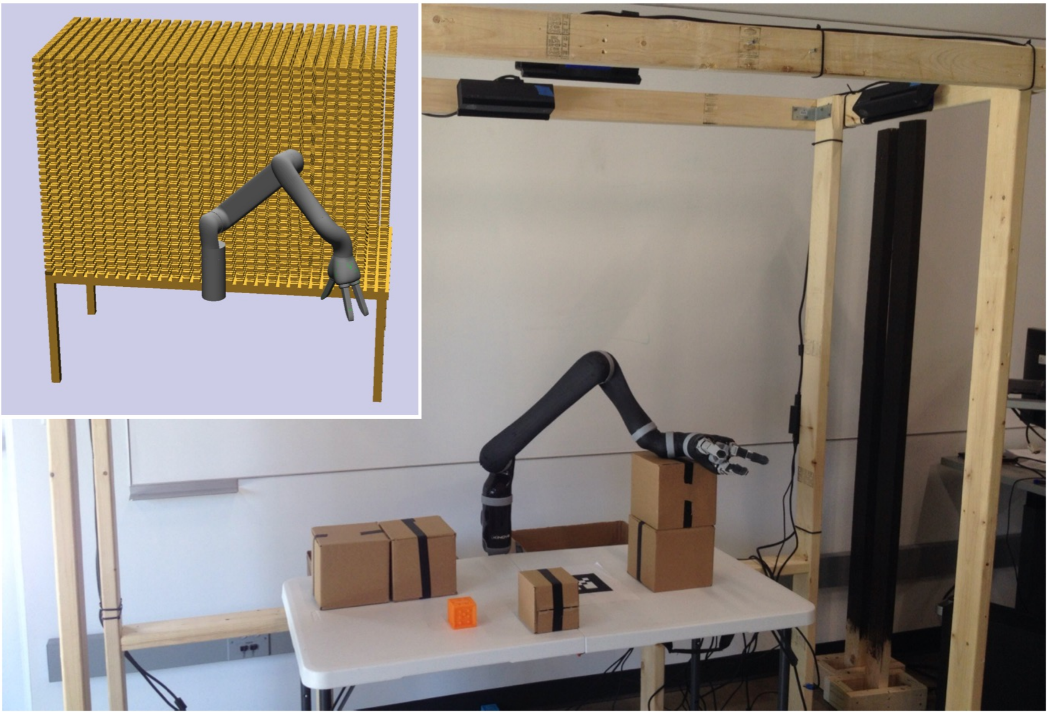
Cross-Layer Robotic Computing System



Robotic Computing Need Hardware Acceleration

- Take motion planning as an example: collision detection for each connecting path can be very expensive…!

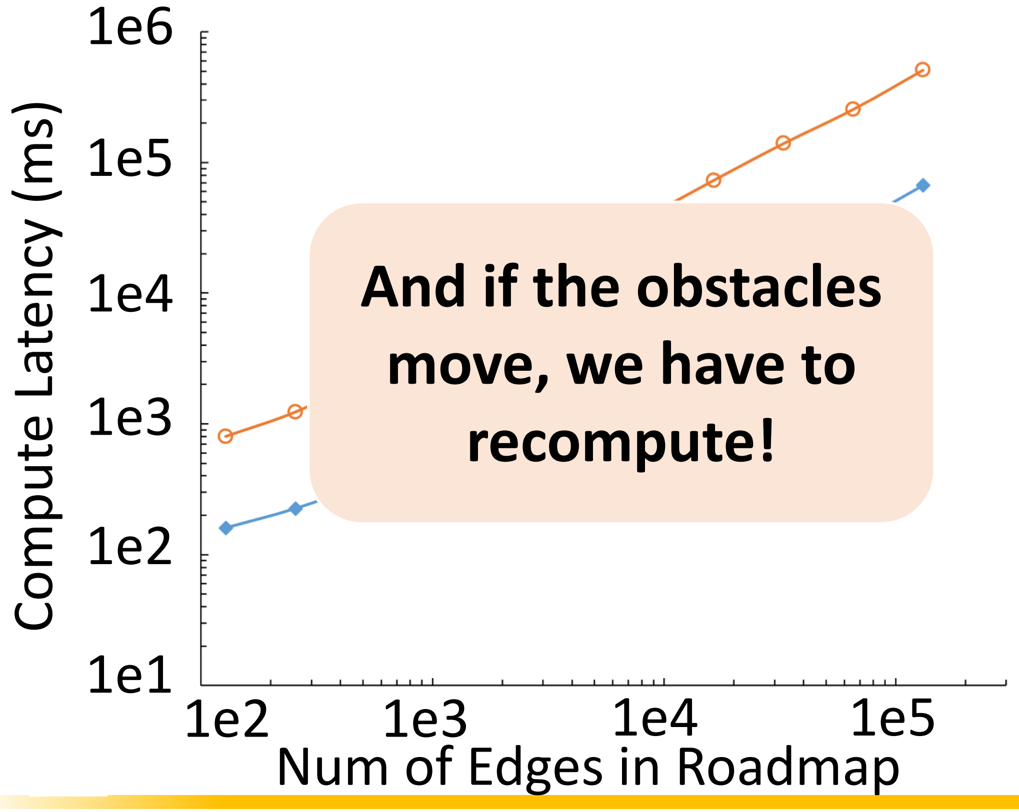


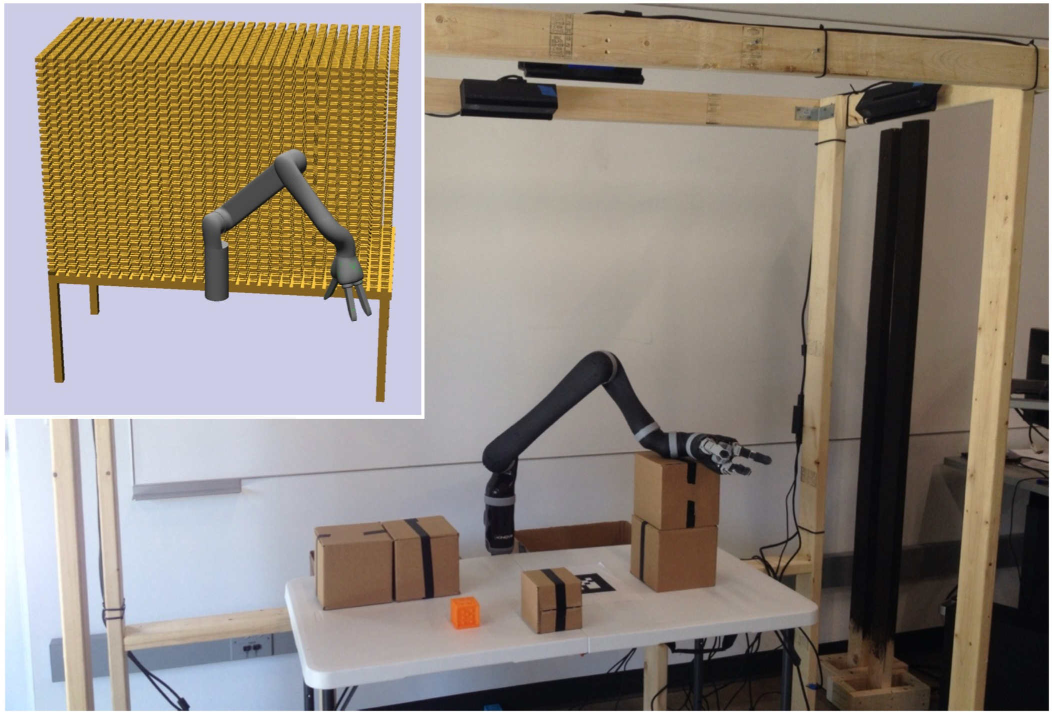


[Murray, MICRO'16]

Robotic Computing Need Hardware Acceleration

- Take motion planning as an example: collision detection for each connecting path can be very expensive…!

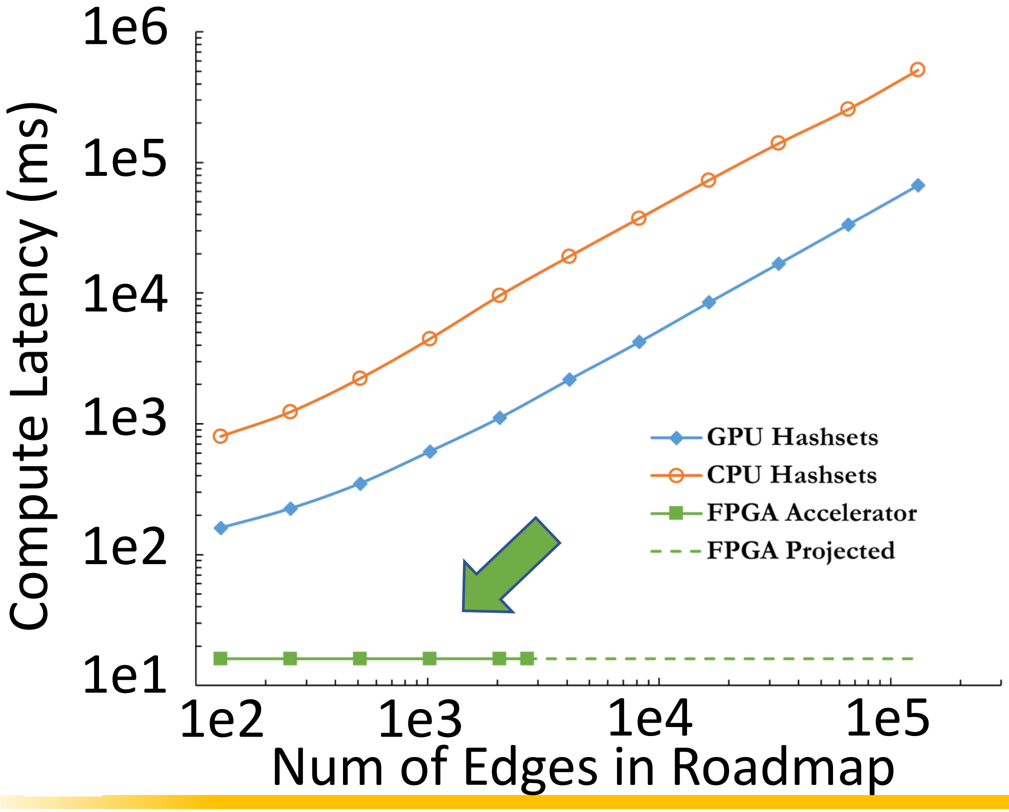




[Murray, MICRO'16]

Robotic Computing Need Hardware Acceleration

- Take motion planning as an example: collision detection for each connecting path can be very expensive…!

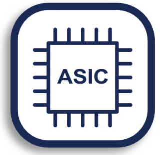


[Murray, MICRO'16]

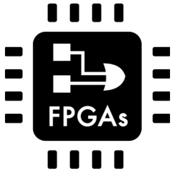
Robotic Computing Need Hardware Acceleration

- Which Hardware Platform for Robotic Computing Acceleration?

  - GPUs/CPUs' **power consumption** is orders of magnitude higher than requirements of resource-constrained scenarios.
  - GPUs/CPUs' general-purpose nature leads to **time inefficiencies** (real-time requirement) and more **vulnerable to cybersecurity threats** (safety requirement)



  - ASICs typically have the highest energy-efficiency, but their **limited configurability** has difficulty adapting to new robotic scenarios, as the robotic computing **algorithms are still evolving very fast**.

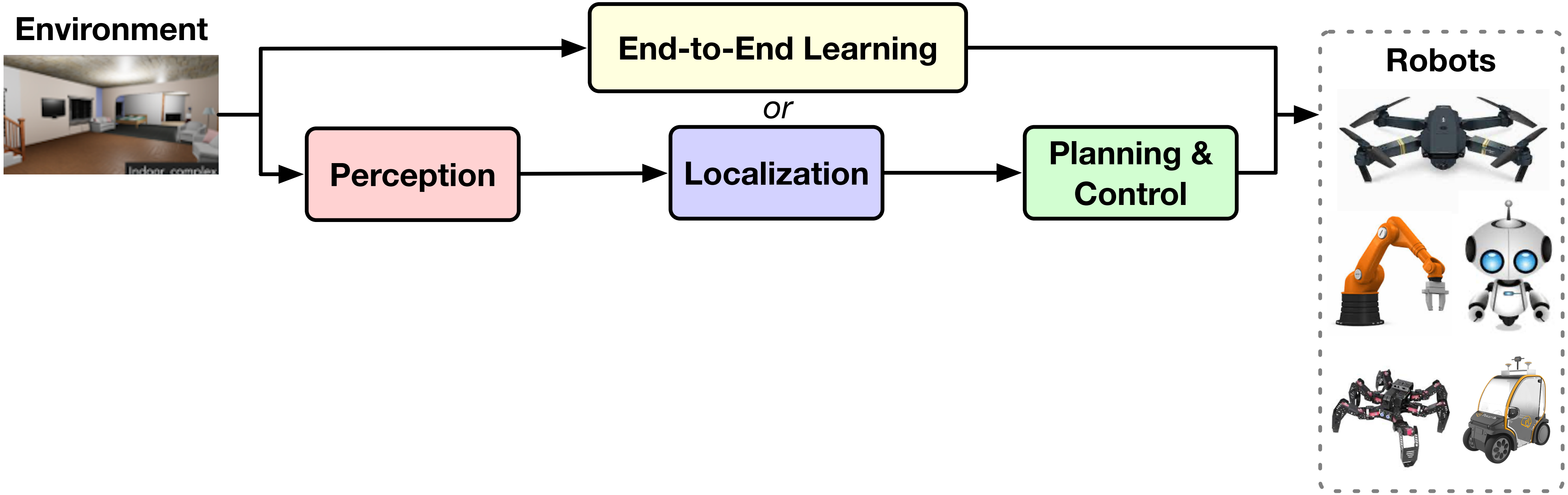


  - **FPGAs have some unique advantages –**
    Compared to GPUs/CPUs: higher energy-efficiency, low power, higher performance
    Compared to ASICs: higher reconfigurability, adaptivity, faster time-to-market, and longer useful life time.

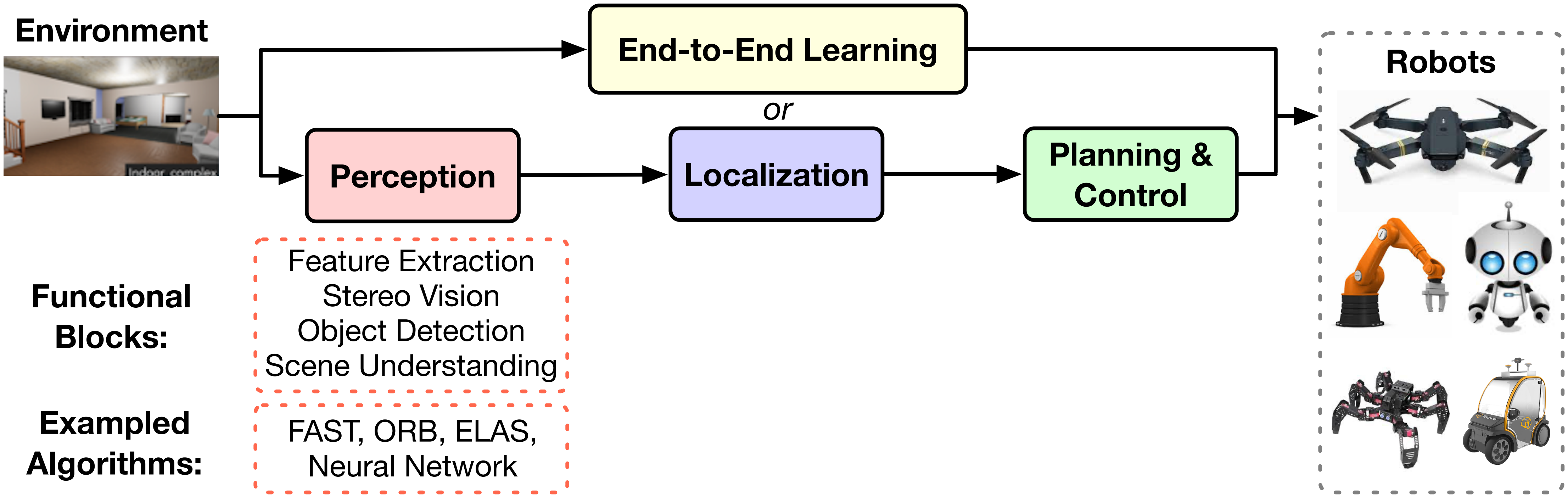
Outline

- Robotic Computing Systems
- Current Progress and Design Techniques
- Research Challenges and Future Directions

FPGA-Based Robotic Computing



FPGA-Based Robotic Computing



[HPCA’21]
[AICAS’21]
[FPL’20]

FPGA-Based Robotic Computing

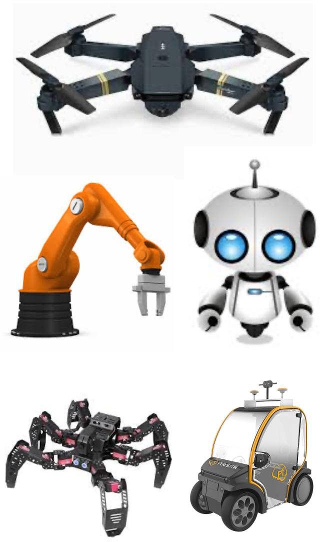
Environment

End-to-End Learning

or

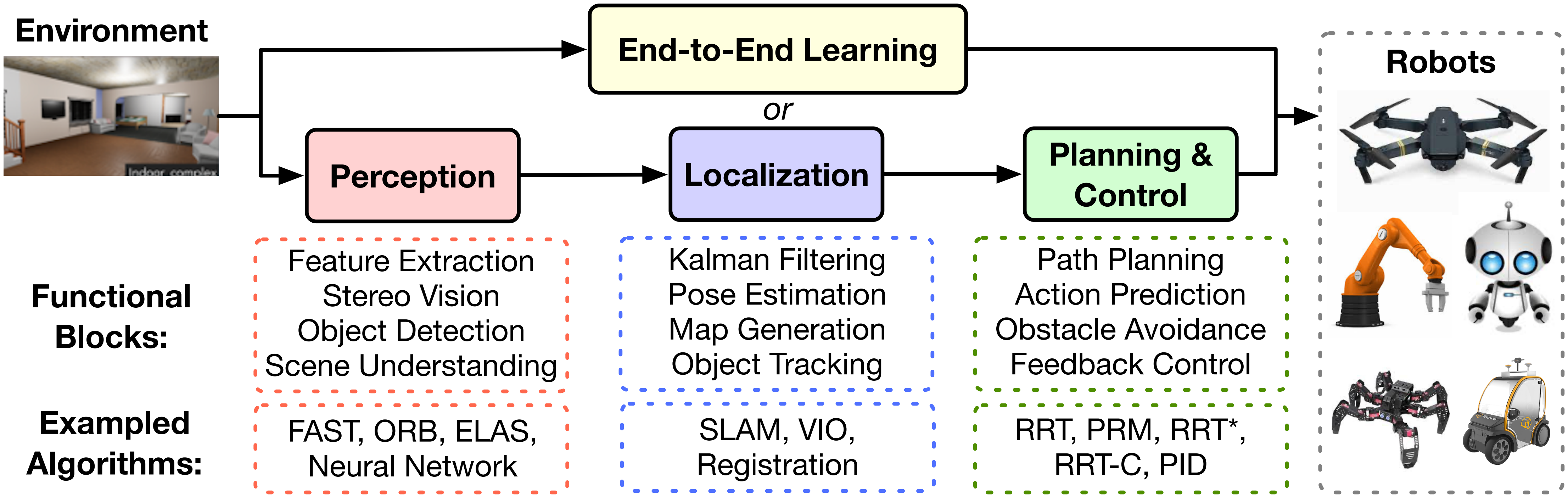
Perception → **Localization** → **Planning & Control**

Robots



	Perception	Localization
Functional Blocks:	Feature Extraction Stereo Vision Object Detection Scene Understanding	Kalman Filtering Pose Estimation Map Generation Object Tracking
Exampled Algorithms:	FAST, ORB, ELAS, Neural Network	SLAM, VIO, Registration
	[HPCA’21] [AICAS’21] [FPL’20]	[CICC’22] [MICRO’21] [TC’20] [DAC’20]

FPGA-Based Robotic Computing

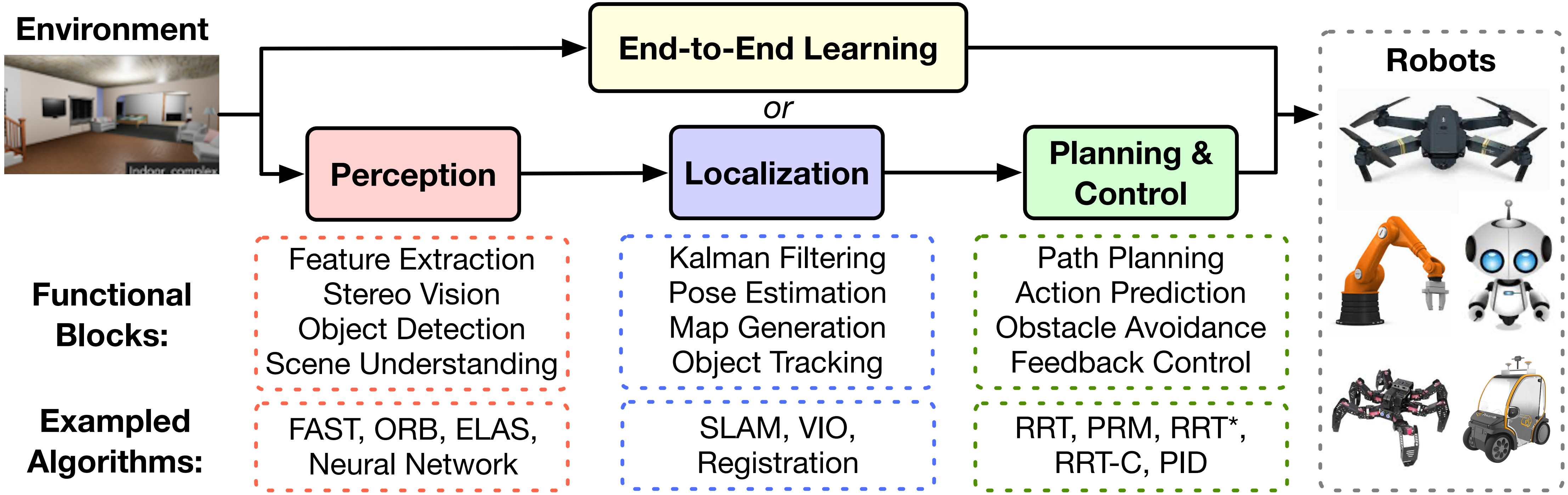


[HPCA'21]
[AICAS'21]
[FPL'20]

[CICC'22]
[MICRO'21]
[TC'20]
[DAC'20]

[MICRO'16]
[ASAP'19]

FPGA-Based Robotic Computing



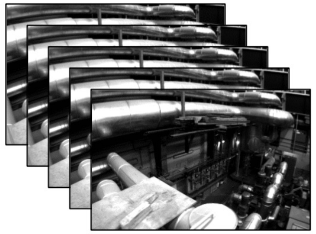
Exampled design techniques:

SW: Robotic-specific hardware-friendly algorithms and data structure, dynamic scheduling, ROS support

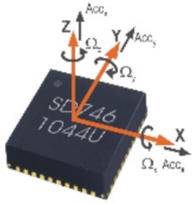
HW: Robotic-specific architecture, sparsity, locality, pipeline and reduced data movement

Exampled Design (SLAM)

Camera



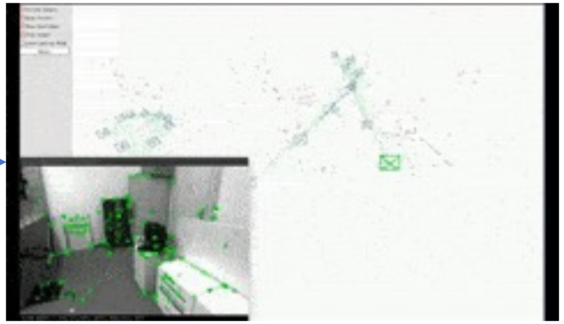
IMU
(Inertial Measurement Unit)



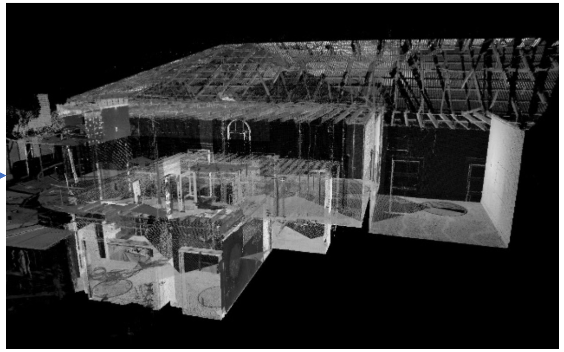
Feature Tracks

Estimated States

Simultaneous Localization and Mapping (SLAM)



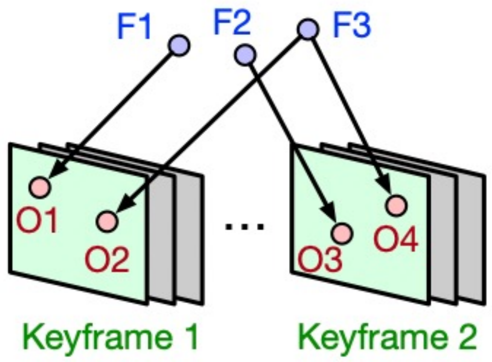
Localization



Mapping

[Source: V. Sze]

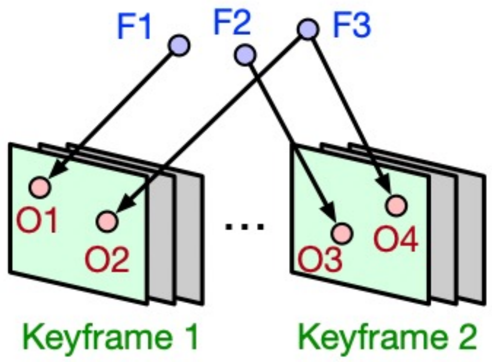
Exampled Design (SLAM): Data Reuse

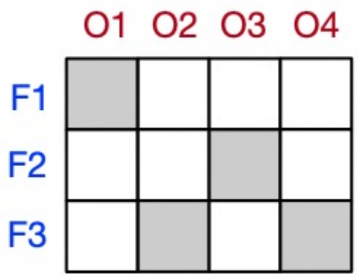


2 **K**eyframes
3 **F**eature Points (F1~F3)
4 **O**bservations (O1~O4)

[Wan, CICC'22]

Exampled Design (SLAM): Data Reuse





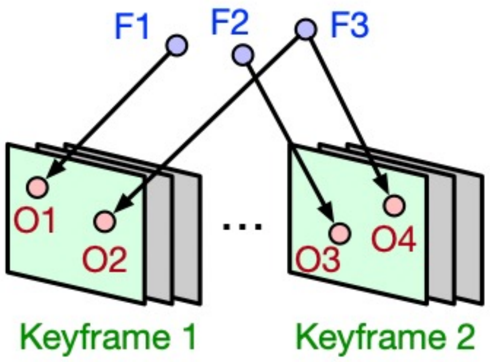
Jacobian Matrix

2 **K**eyframes
3 **F**eature Points (F1~F3)
4 **O**bservations (O1~O4)

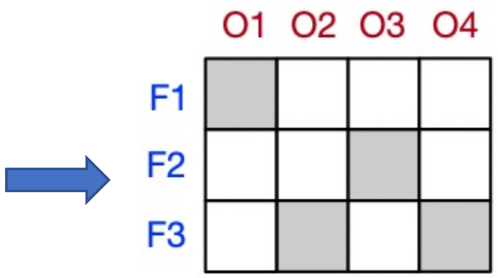
<feature point, observation>
pairs have non-zero values

[Wan, CICC’22]

Exampled Design (SLAM): Data Reuse

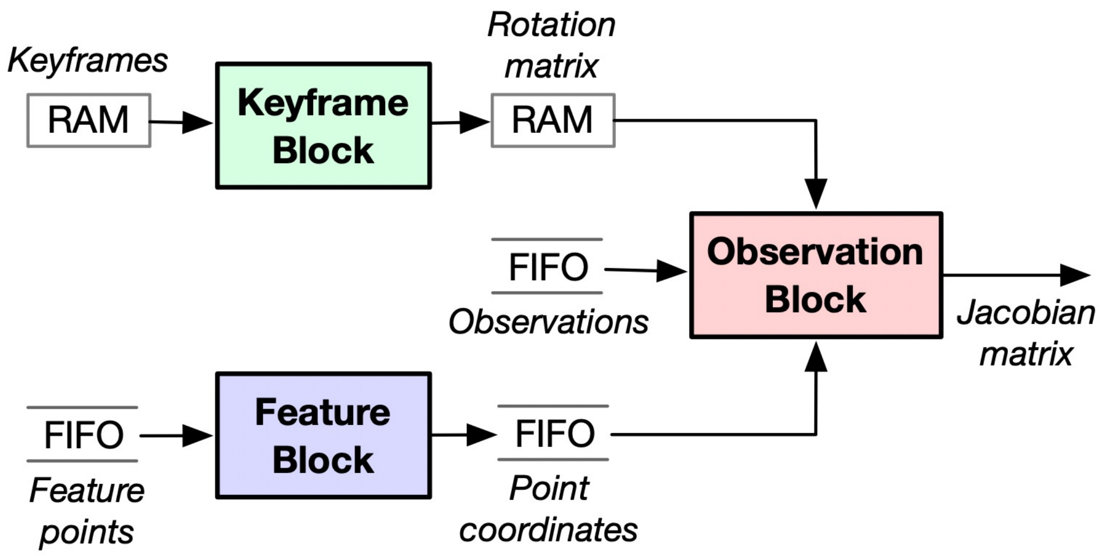


2 **K**eyframes
3 **F**eature Points (F1~F3)
4 **O**bservations (O1~O4)



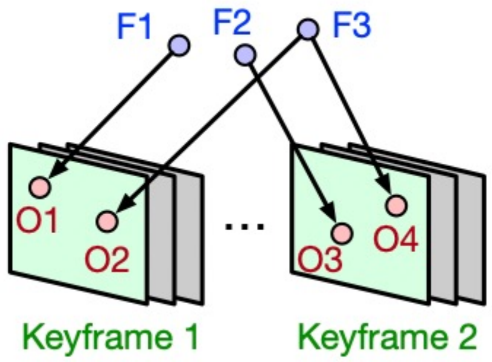
Jacobian Matrix

<feature point, observation> pairs have non-zero values

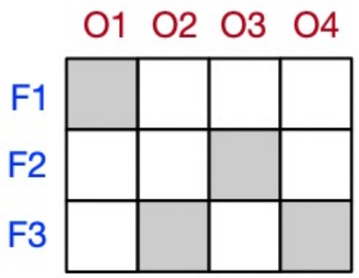


[Wan, CICC'22]

Exampled Design (SLAM): Data Reuse

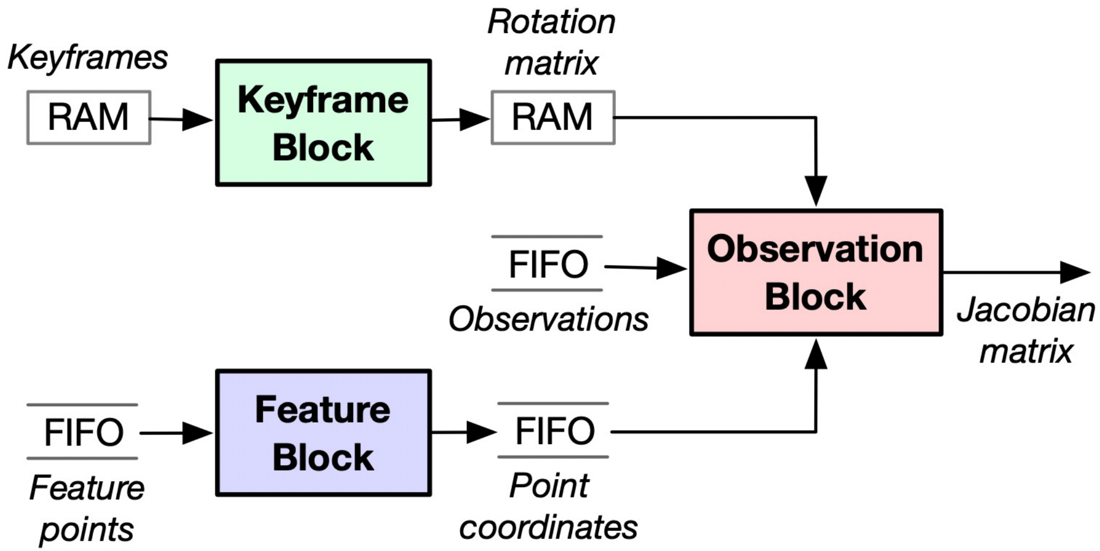


2 **K**eyframes
3 **F**eature Points (F1~F3)
4 **O**bservations (O1~O4)



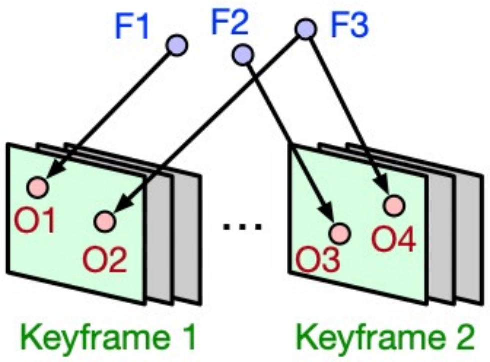
Jacobian Matrix

<feature point, observation> pairs have non-zero values

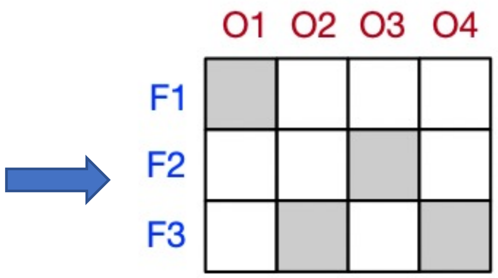


Three-Level Block Designs:
- Keyframe-level: Rotation matrix of keyframes
- Feature-level: 3D coordinates
- Observation-level: Jacobian matrix

Exampled Design (SLAM): Data Reuse

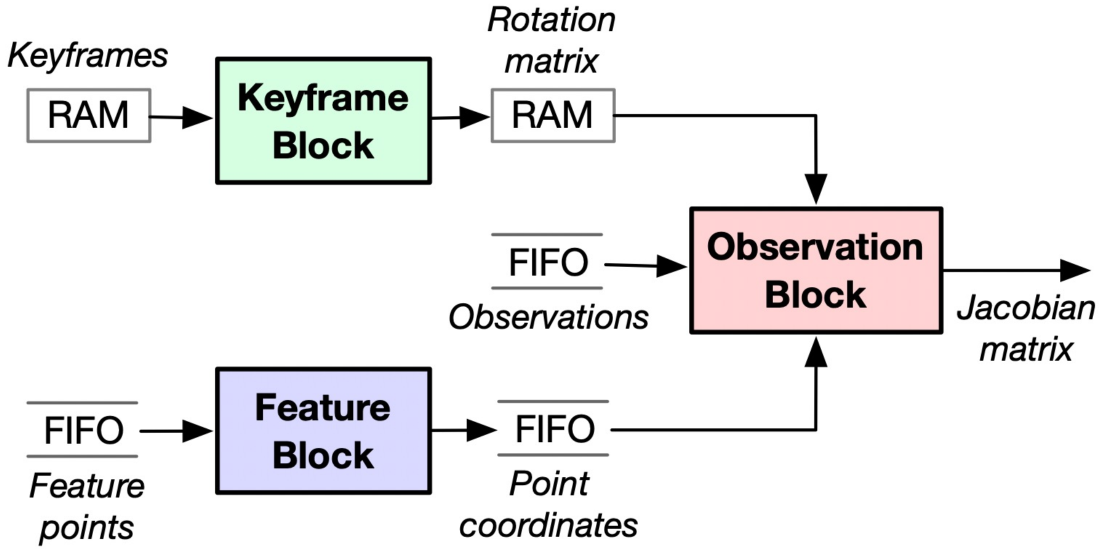


2 **K**eyframes
3 **F**eature Points (F1~F3)
4 **O**bservations (O1~O4)



Jacobian Matrix

<feature point, observation> pairs have non-zero values



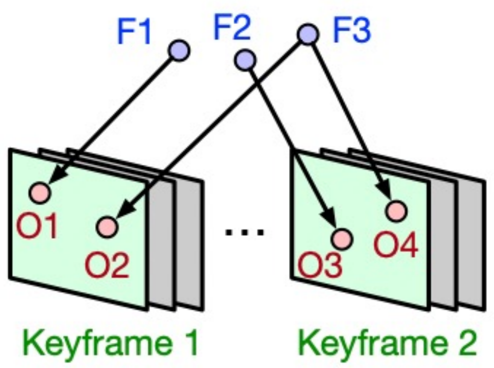
Two-Level Data Reuses:

- Feature-reuse: across associated observations
- Keyframe-reuse: over all obsn. within keyframe

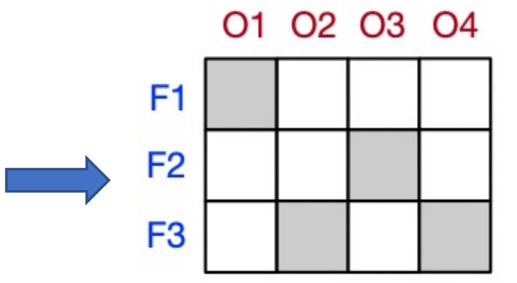
Three-Level Block Designs:

- Keyframe-level: Rotation matrix of keyframes
- Feature-level: 3D coordinates
- Observation-level: Jacobian matrix

Exampled Design (SLAM): Data Reuse

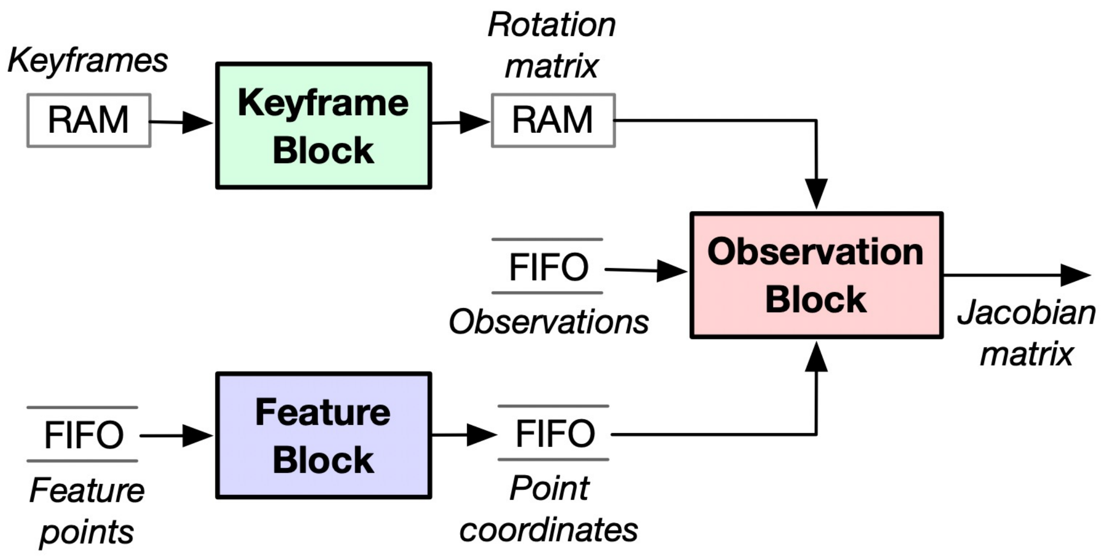


2 **K**eyframes
3 **F**eature Points (F1~F3)
4 **O**bservations (O1~O4)



Jacobian Matrix

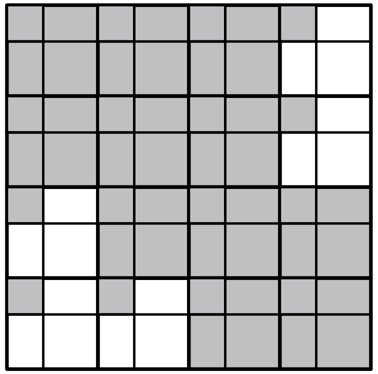
<feature point, observation> pairs have non-zero values



Two-Level Data Reuses:
- Feature-reuse: across associated observations → feature (row)-stationary
- Keyframe-reuse: over all obsn. within keyframe

Three-Level Block Designs:
- Keyframe-level: Rotation matrix of keyframes
- Feature-level: 3D coordinates
- Observation-level: Jacobian matrix

Exampled Design (SLAM): Symmetric & Sparsity

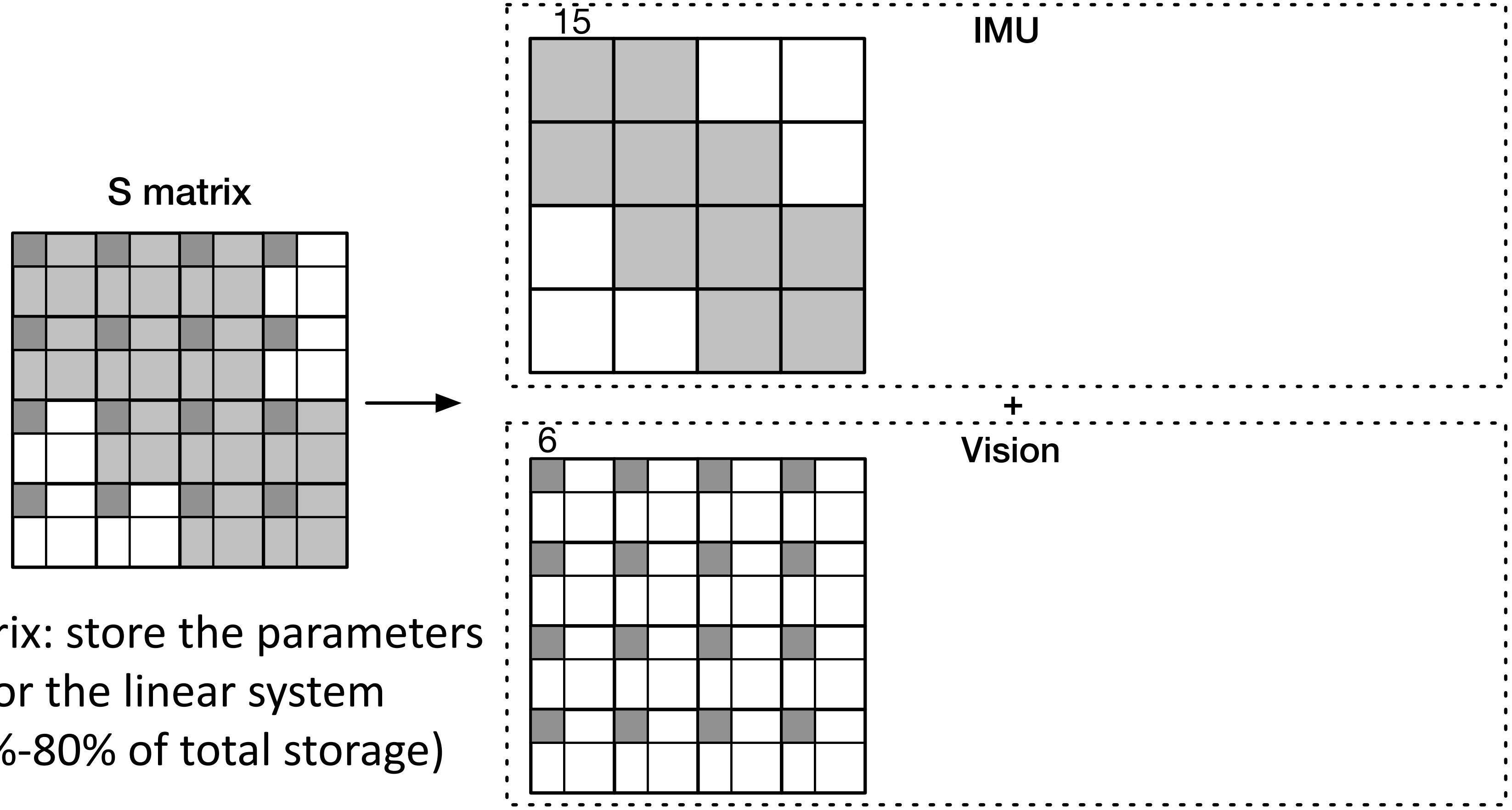


S matrix: store the parameters for the system (40%-80% of total storage)

720 kb

[Wan, CICC'22]

Exampled Design (SLAM): Symmetric & Sparsity

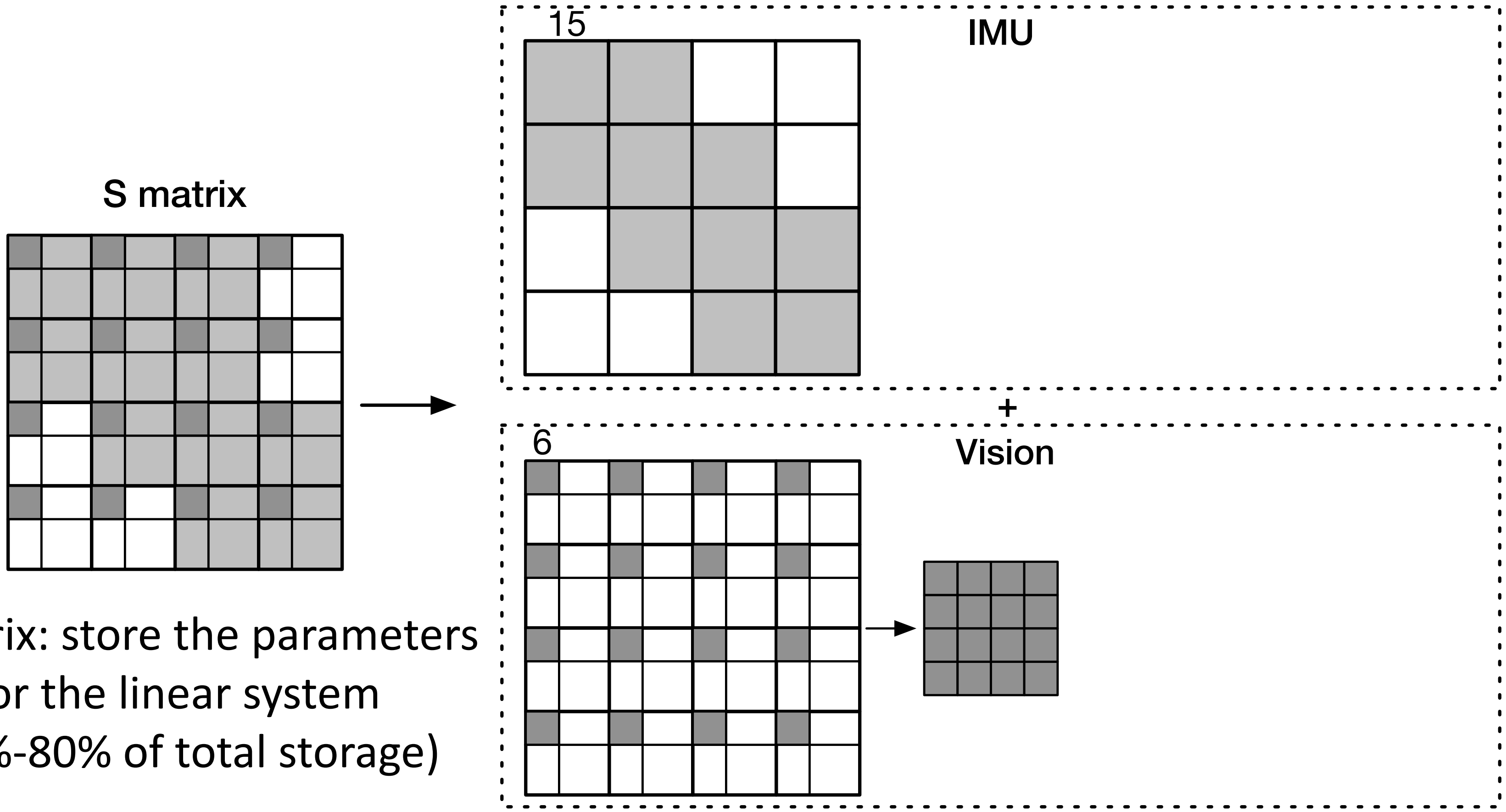


S matrix: store the parameters for the linear system (40%-80% of total storage)

720 kb

[Wan, CICC'22]

Exampled Design (SLAM): Symmetric & Sparsity



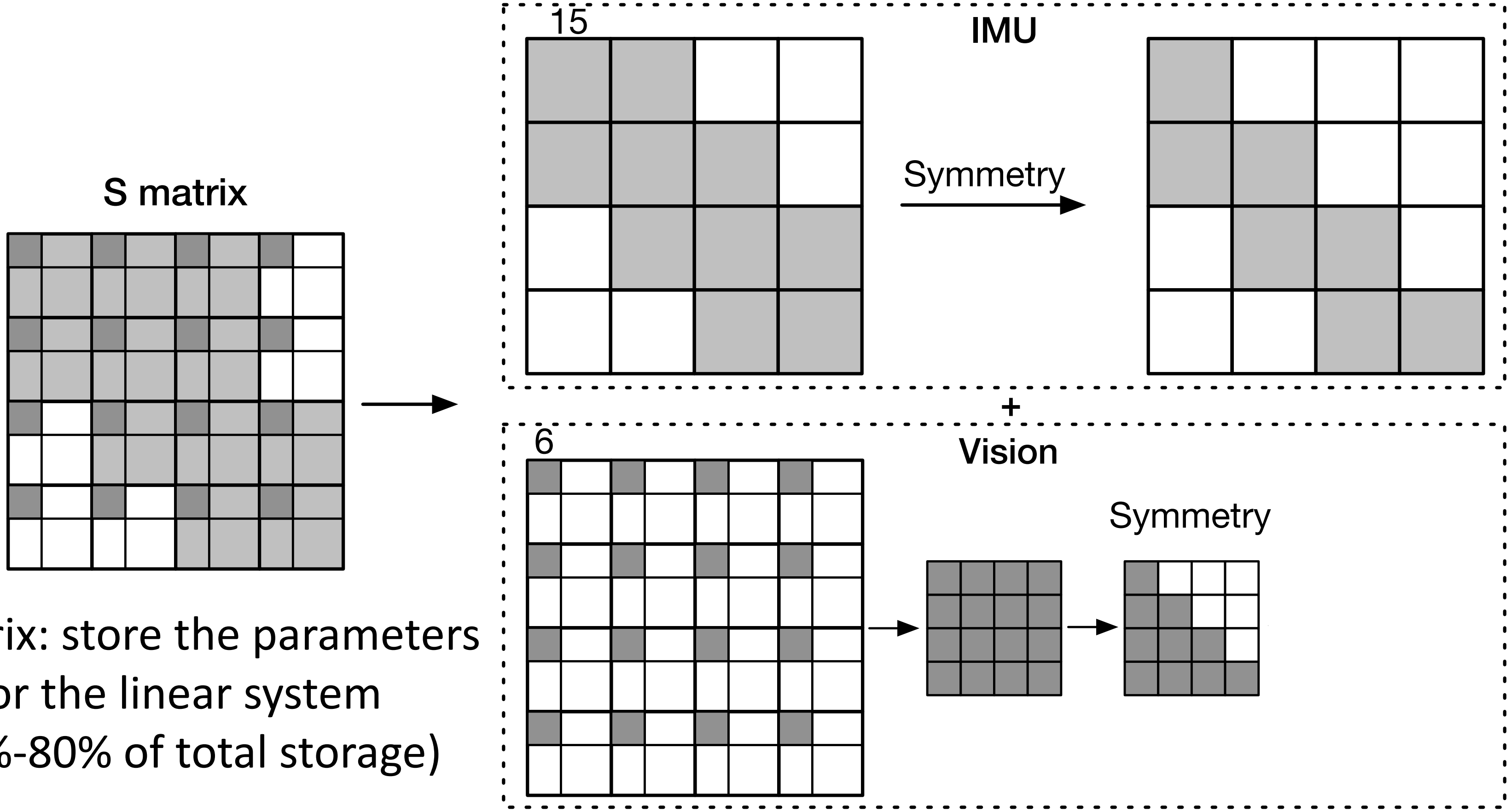
S matrix: store the parameters for the linear system (40%-80% of total storage)

720 kb

[Wan, CICC’22]

Exampled Design (SLAM): Symmetric & Sparsity

S matrix

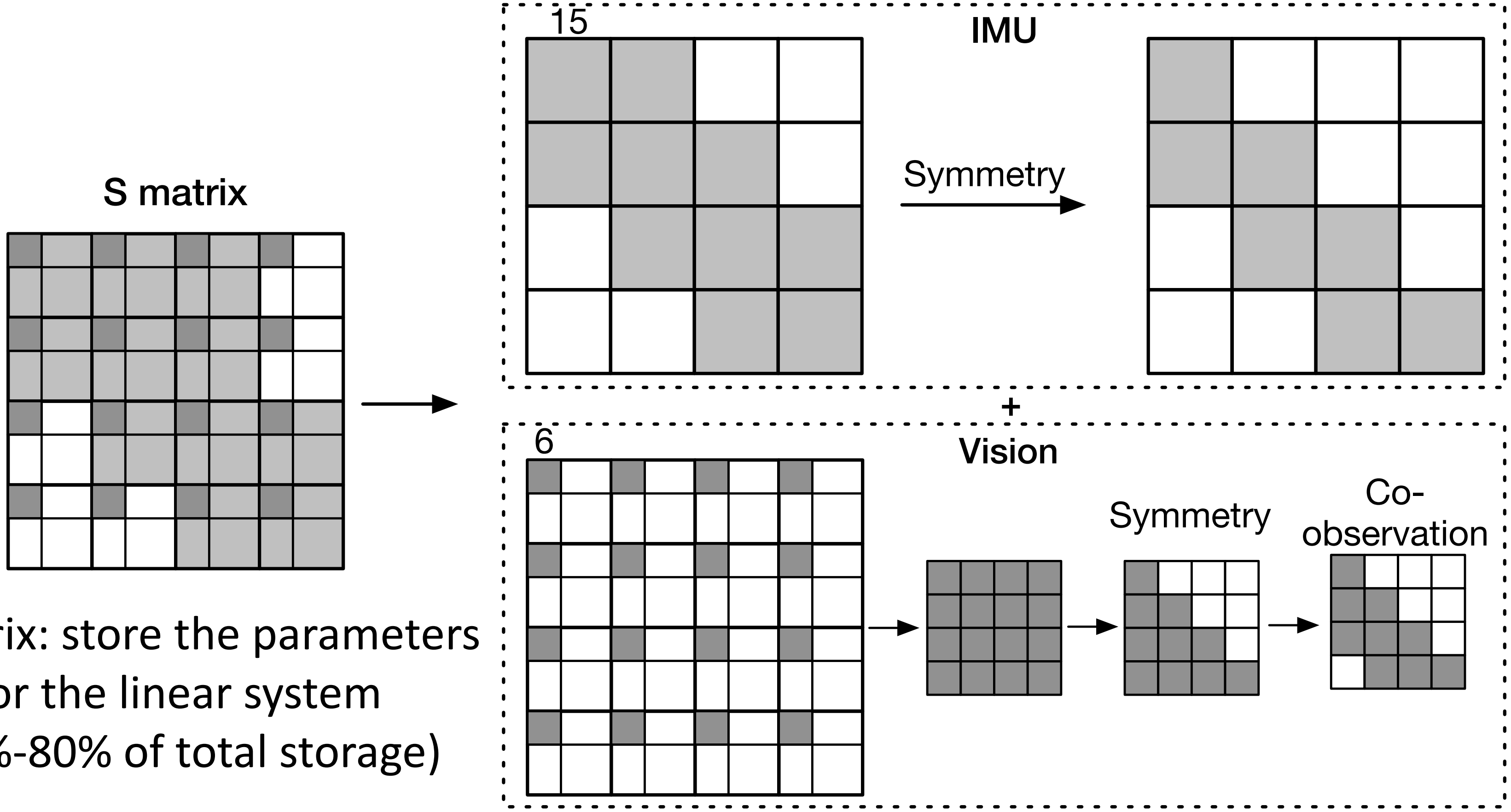


S matrix: store the parameters for the linear system (40%-80% of total storage)

720 kb

[Wan, CICC'22]

Exampled Design (SLAM): Symmetric & Sparsity

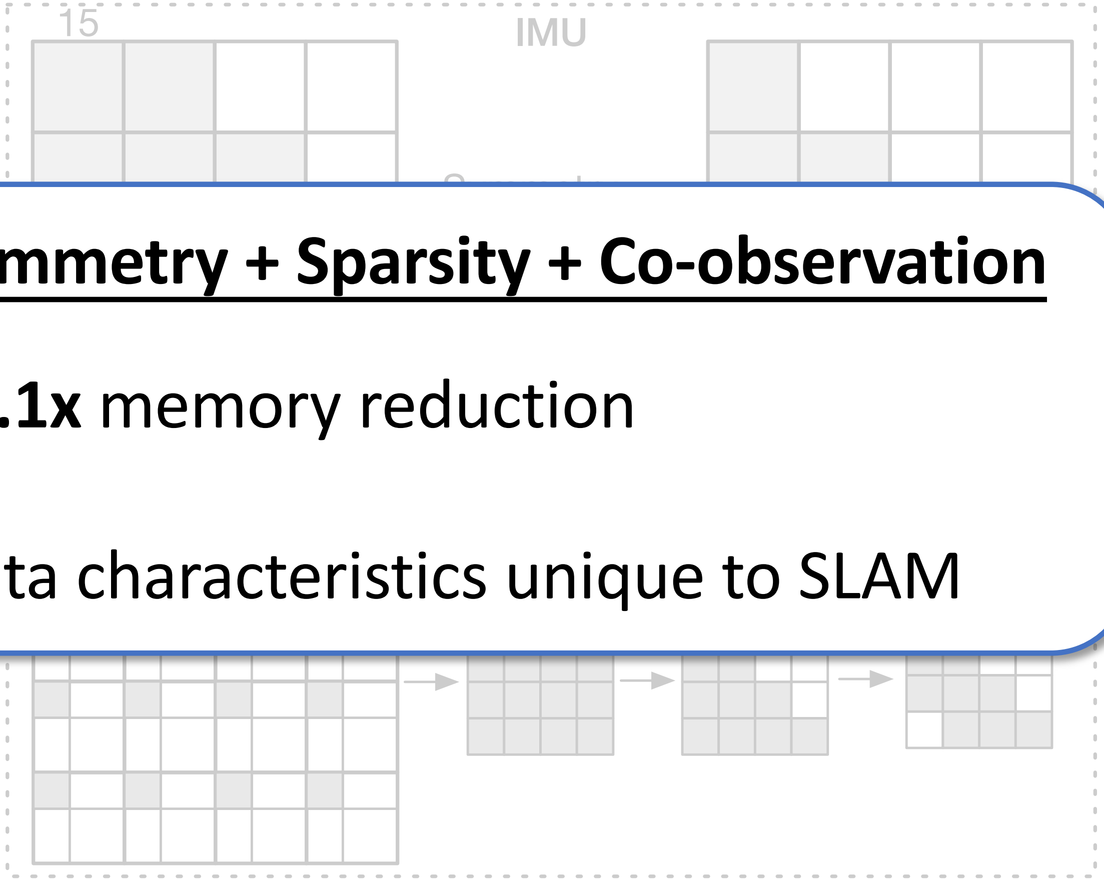


S matrix: store the parameters for the linear system (40%-80% of total storage)

720 kb → 175.97 kb
4.1x reduction

[Wan, CICC'22]

Exampled Design (SLAM): Symmetric & Sparsity



Data Layout + Symmetry + Sparsity + Co-observation

4.1x memory reduction

Exploiting data characteristics unique to SLAM

S matrix: store the parameters for the linear system (40%-80% of total storage)

720 kb → 175.97 kb (4.1x reduction)

[Wan, CICC'22]

Outline

- Robotic Computing Systems
- Current Progress and Design Techniques
- Research Challenges and Future Directions

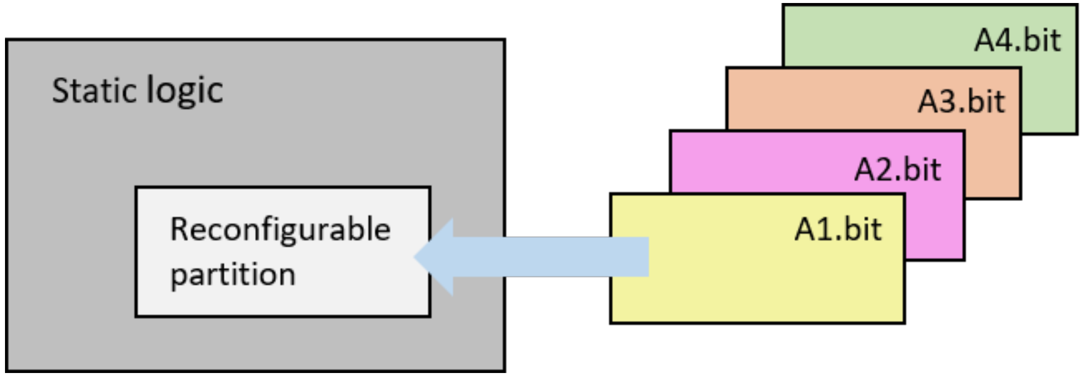
Challenges

- Dynamic changing workloads
- Unoptimized general solutions
- Diverse hardware components
- Inefficient ROS support
- Large #algorithms and #hardware
- Tedious development procedure
- Inaccurate performance evaluation

Challenges and Research Opportunities

- Dynamic changing workloads
- Unoptimized general solutions
- Diverse hardware components
- Inefficient ROS support
- Large #algorithms and #hardware
- Tedious development procedure
- Inaccurate performance evaluation

Reconfiguring robotic computing at runtime



Partial Reconfiguration of FPGA

Challenges and Research Opportunities

- Dynamic changing workloads
- Unoptimized general solutions
- Diverse hardware components
- Inefficient ROS support
- Large #algorithms and #hardware
- Tedious development procedure
- Inaccurate performance evaluation

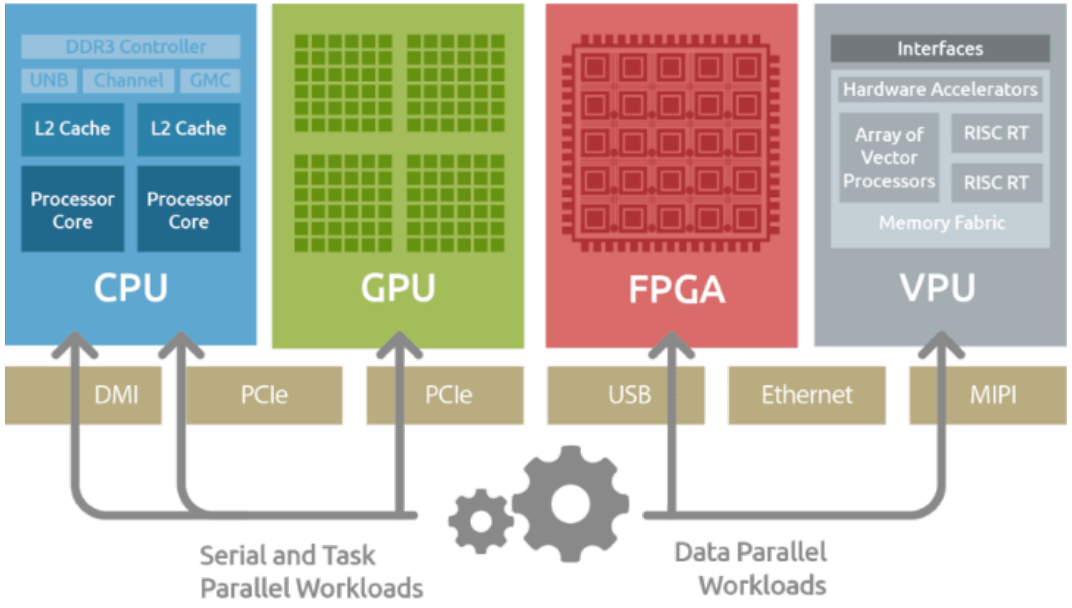
Modularizing robotic computing kernels design

- Build optimized building blocks for robotic kernels, as libraries or packages.
- During design phase, directly import these robotic-specific libraries and building blocks.

Challenges and Research Opportunities

- Dynamic changing workloads
- Unoptimized general solutions
- Diverse hardware components
- Inefficient ROS support
- Large #algorithms and #hardware
- Tedious development procedure
- Inaccurate performance evaluation

Mapping robotic computing on heterogeneous platforms



Challenges and Research Opportunities

- Dynamic changing workloads
- Unoptimized general solutions
- Diverse hardware components
- Inefficient ROS support
- Large #algorithms and #hardware
- Tedious development procedure
- Inaccurate performance evaluation

Connecting FPGA to ROS ecosystem

- Better interface with FPGA and ROS.
- Accelerate inter-process and intra-process between ROS nodes.
- Dynamically and efficiently mapping ROS to heterogeneous compute platforms.

(ROS: Robot Operating System)

Challenges and Research Opportunities

- Dynamic changing workloads
- Unoptimized general solutions
- Diverse hardware components
- Inefficient ROS support
- Large #algorithms and #hardware
- Tedious development procedure
- Inaccurate performance evaluation

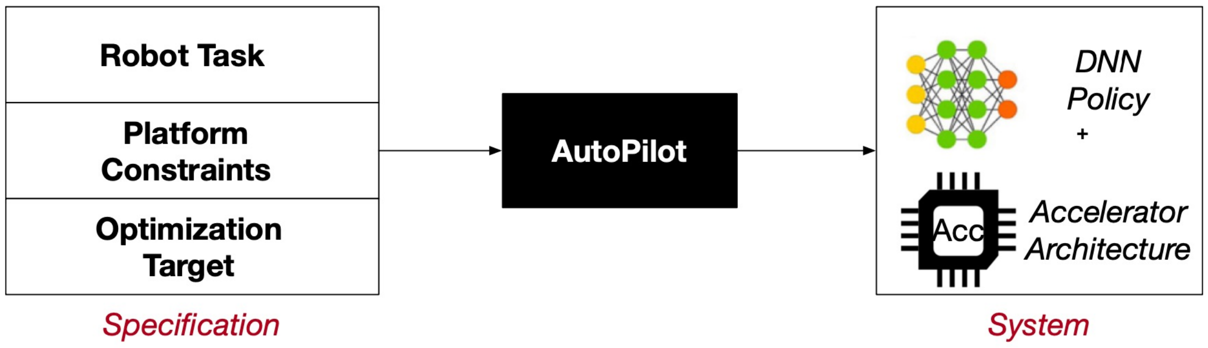
Benchmarking robotic computing kernels

- Benchmark a robotic algorithm across various hardware platforms.
- Benchmark various robotic algorithms within the same hardware.

Challenges and Research Opportunities

- Dynamic changing workloads
- Unoptimized general solutions
- Diverse hardware components
- Inefficient ROS support
- Large #algorithms and #hardware
- Tedious development procedure
- Inaccurate performance evaluation

Automating robotic computing design flow



[Krishnan, arXiv'21]

- Push button framework
- Intelligently search huge design space to pick optimal hardware and algorithm

Challenges and Research Opportunities

- Dynamic changing workloads
- Unoptimized general solutions
- Diverse hardware components
- Inefficient ROS support
- Large #algorithms and #hardware
- Tedious development procedure
- Inaccurate performance evaluation

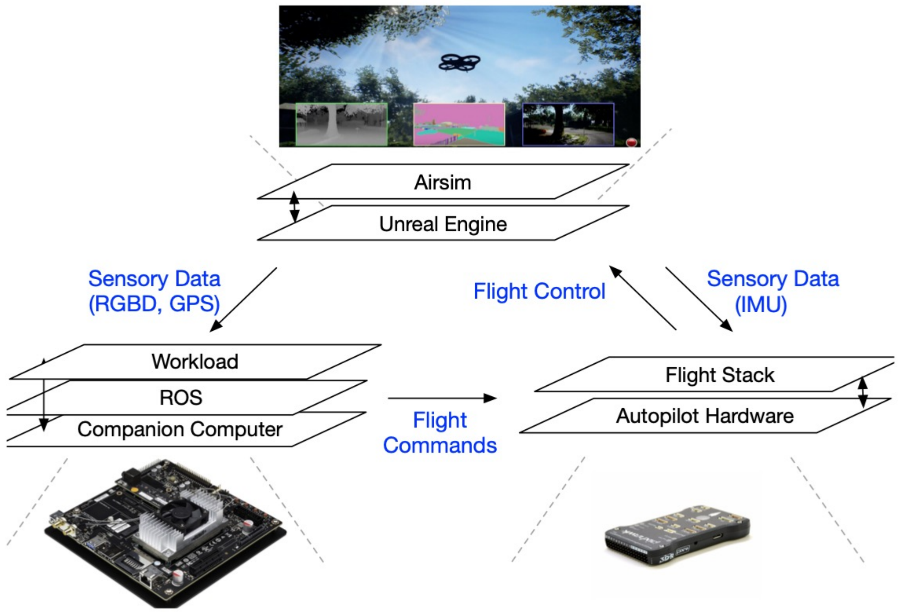
Building customized robotic computing with the open-source framework

- Defining and building an open-source FPGA-based RISC-V robotics-on-chip processor with open-source frameworks

Challenges and Research Opportunities

- Dynamic changing workloads
- Unoptimized general solutions
- Diverse hardware components
- Inefficient ROS support
- Large #algorithms and #hardware
- Tedious development procedure
- Inaccurate performance evaluation

Integrating robotic computing hardware in a simulation loop



[Boroujerdian, MICRO'18]

Reference

Robotic Computing on FPGAs: Current Progress, Research Challenges, and Opportunities

Zishen Wan[1], Ashwin Lele[1], Bo Yu[2], Shaoshan Liu[2], Yu Wang[3], Vijay Janapa Reddi[4], Cong Hao[1], and Arijit Raychowdhury[1]

[1] School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, USA
[2] PerceptIn, Fremont, CA, USA
[3] Department of Electronic Engineering, Tsinghua University, Beijing, China
[4] School of Engineering and Applied Sciences, Harvard University, Cambridge, MA, USA
{zishenwan, alele9, callie.hao}@gatech.edu, arijit.raychowdhury@ece.gatech.edu
{bo.yu, shaoshan.liu}@perceptin.io, yu-wang@tsinghua.edu.cn, vj@eecs.harvard.edu

***Abstract*—Robotic computing has reached a tipping point, with a myriad of robots (e.g., drones, self-driving cars, logistic robots) being widely applied in diverse scenarios. The continuous proliferation of robotics, however, critically depends on efficient computing substrates, driven by real-time requirements, robotic size-weight-and-power constraints, cybersecurity considerations, and dynamically changing scenarios. Within all platforms, FPGA is able to deliver both software and hardware solutions with low power, high performance, reconfigurability, reliability, and adaptivity characteristics, serving as the promising computing substrate for robotic applications. This paper highlights the current progress, design techniques, challenges, and open research challenges in the domain of robotic computing on FPGAs.**

I. Introduction

Robotic computing is on the rise. A myriad of robots such as drones, legged robots, and self-driving cars are on the verge of becoming an integral part of our life [1], [2]. Robotics is typically an art of system integration both in software and hardware (Fig. 1). The continuous proliferation of robots, however, face computing challenges, raised from the higher performance requirements, resource constraints, miniaturization of machine form factors, dynamic operating scenarios, and cybersecurity considerations. Therefore, it is essential to choose a proper computing substrate for robotic system that can meet real-time and power requirements and adapt to changing workloads.

CPUs and GPUs are two widely-used computing platforms, however, their performance and efficiency are still incompetent in real-time computation for complex robots. Take the motion planning task as an example, CPU typically takes a few seconds to find the collision-free trajectory [3], making it too slow for complex navigation tasks. GPUs can finish planning tasks in hundreds of milliseconds, still insufficient for many scenarios while at hundreds of watts cost [4]. ASICs are recently developed for specific robotic workloads with low power and high performance [5]–[7], but their fixed architecture has difficulty in adapting to rapid-evolving robotic algorithms and dynamic scenarios, and is vulnerable to cybersecurity threats.

As an alternative, we believe FPGA is the promising compute substrate for robotic applications. First, FPGA increases the performance with massive parallelism and deeply pipelined datapath, making it capable of meeting real-time requirements with high energy efficiency compared to CPUs and GPUs. Second, FPGA can adaptively generate custom architectures and update with the fast-evolving of robotic algorithms without going through re-fabrication as ASIC [8]. Third, FPGA is flexible in dealing with highly diverse robotic workloads, especially with partial reconfiguration allowing modification part of the operating board. Fourth, FPGA provides reliable design by leveraging reconfiguration to patch flows, compared to potential vulnerabilities detected in fixed architectures [9], which is especially essential in safety-critical scenarios [10]. Overall, FPGA has the potential to deliver high-performance, low-power, reconfigurable, adaptive, and secure features in robotic computing, and is booming in autonomous applications. However, several challenges, such as tedious development procedures, inefficient system support, and huge design space, remain in the FPGA-based robotic computing and impede the way ahead.

In this paper, we will discuss the current progress, challenges, and opportunities for FPGA-based robotic computing. Section II introduces the cross-layer stack of robotic system. Section III presents current FPGA accelerators and systems for robotic computing, with an emphasis on design techniques. Section IV discusses challenges and opportunities for FPGA-based robotic computing, and our view of the road ahead.

II. Cross-Layer Robotic Computing Systems

This section introduces the abstraction layers of the robotic computing stack. We traverse down Fig. 1 to explain robotic-specific algorithms and systems building blocks.

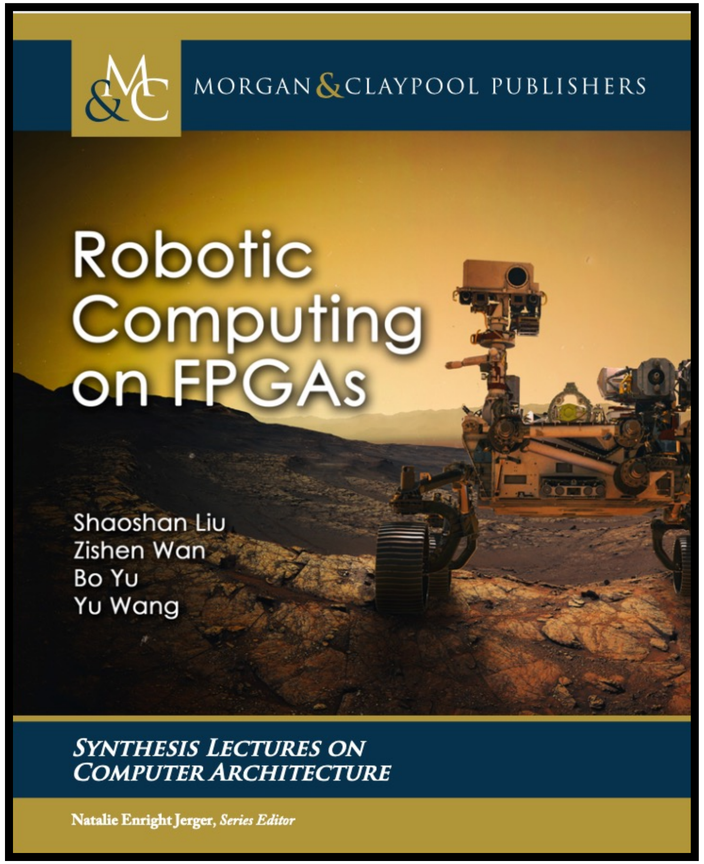
A. Robotic-Computing Algorithm Layer

Fig. 2 illustrates the representative algorithm building blocks in robotic computing, including sense-plan-act (perception, localization, planning, control) and end-to-end learning.

Perception. The goal of perception is to sense the dynamic surroundings and build a reliable and detailed representation based on sensory data (e.g., camera, IMU, GPS, LiDAR). Perception usually includes feature extraction, stereo vision, object detection, scene understanding, etc. In feature extraction, key points are usually detected using FAST feature and ORB

[Wan, AICAS 2022]

Reference



[Wan, Synthesis Lectures on Comp Arch 2021]

Robotic Computing on FPGAs: Current Progress, Research Challenges, and Opportunities

Zishen Wan[1], Ashwin Lele[1], Bo Yu[2], Shaoshan Liu[2], Yu Wang[3], Vijay Janapa Reddi[4], Cong Hao[1], and Arijit Raychowdhury[1]

[1] School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, USA
[2] PerceptIn, Fremont, CA, USA
[3] Department of Electronic Engineering, Tsinghua University, Beijing, China
[4] School of Engineering and Applied Sciences, Harvard University, Cambridge, MA, USA
{zishenwan, alele9, callie.hao}@gatech.edu, arijit.raychowdhury@ece.gatech.edu
{bo.yu, shaoshan.liu}@perceptin.io, yu-wang@tsinghua.edu.cn, vj@eecs.harvard.edu

***Abstract*—Robotic computing has reached a tipping point, with a myriad of robots (e.g., drones, self-driving cars, logistic robots) being widely applied in diverse scenarios. The continuous proliferation of robotics, however, critically depends on efficient computing substrates, driven by real-time requirements, robotic size-weight-and-power constraints, cybersecurity considerations, and dynamically changing scenarios. Within all platforms, FPGA is able to deliver both software and hardware solutions with low power, high performance, reconfigurability, reliability, and adaptivity characteristics, serving as the promising computing substrate for robotic applications. This paper highlights the current progress, design techniques, challenges, and open research challenges in the domain of robotic computing on FPGAs.**

I. Introduction

Robotic computing is on the rise. A myriad of robots such as drones, legged robots, and self-driving cars are on the verge of becoming an integral part of our life [1], [2]. Robotics is typically an art of system integration both in software and hardware (Fig. 1). The continuous proliferation of robots, however, face computing challenges, raised from the higher performance requirements, resource constraints, miniaturization of machine form factors, dynamic operating scenarios, and cybersecurity considerations. Therefore, it is essential to choose a proper computing substrate for robotic system that can meet real-time and power requirements and adapt to changing workloads.

CPUs and GPUs are two widely-used computing platforms, however, their performance and efficiency are still incompetent in real-time computation for complex robots. Take the motion planning task as an example, CPU typically takes a few seconds to find the collision-free trajectory [3], making it too slow for complex navigation tasks. GPUs can finish planning tasks in hundreds of milliseconds, still insufficient for many scenarios while at hundreds of watts cost [4]. ASICs are recently developed for specific robotic workloads with low power and high performance [5]–[7], but their fixed architecture has difficulty in adapting to rapid-evolving robotic algorithms and dynamic scenarios, and is vulnerable to cybersecurity threats.

As an alternative, we believe FPGA is the promising compute substrate for robotic applications. First, FPGA increases the performance with massive parallelism and deeply pipelined datapath, making it capable of meeting real-time requirements with high energy efficiency compared to CPUs and GPUs. Second, FPGA can adaptively generate custom architectures and update with the fast-evolving of robotic algorithms without going through re-fabrication as ASIC [8]. Third, FPGA is flexible in dealing with highly diverse robotic workloads, especially with partial reconfiguration allowing modification part of the operating board. Fourth, FPGA provides reliable design by leveraging reconfiguration to patch flows, compared to potential vulnerabilities detected in fixed architectures [9], which is especially essential in safety-critical scenarios [10]. Overall, FPGA has the potential to deliver high-performance, low-power, reconfigurable, adaptive, and secure features in robotic computing, and is booming in autonomous applications. However, several challenges, such as tedious development procedures, inefficient system support, and huge design space, remain in the FPGA-based robotic computing and impede the way ahead.

In this paper, we will discuss the current progress, challenges, and opportunities for FPGA-based robotic computing. Section II introduces the cross-layer stack of robotic system. Section III presents current FPGA accelerators and systems for robotic computing, with an emphasis on design techniques. Section IV discusses challenges and opportunities for FPGA-based robotic computing, and our view of the road ahead.

II. Cross-Layer Robotic Computing Systems

This section introduces the abstraction layers of the robotic computing stack. We traverse down Fig. 1 to explain robotic-specific algorithms and systems building blocks.

A. Robotic-Computing Algorithm Layer

Fig. 2 illustrates the representative algorithm building blocks in robotic computing, including sense-plan-act (perception, localization, planning, control) and end-to-end learning.

Perception. The goal of perception is to sense the dynamic surroundings and build a reliable and detailed representation based on sensory data (e.g., camera, IMU, GPS, LiDAR). Perception usually includes feature extraction, stereo vision, object detection, scene understanding, etc. In feature extraction, key points are usually detected using FAST feature and ORB

[Wan, AICAS 2022]

[Wan, Circuits and Systems Magazine 2021]

Thank You!

Contact: Zishen Wan
Email: zishenwan@gatech.edu
Webpage: https://zishenwan.github.io