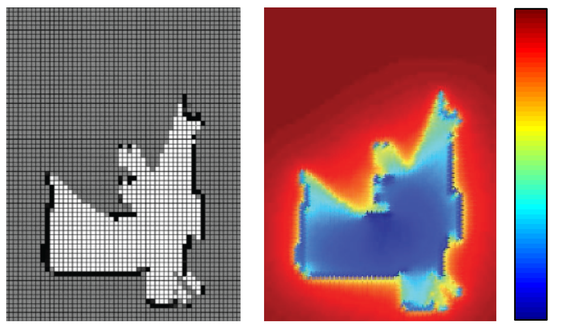
High-throughput Computation of Shannon Mutual Information on Chip

Peter Zhi Xuan Li*, Zhengdong Zhang*, Sertac Karaman, Vivienne Sze
Massachusetts Institute of Technology

1. Introduction & Summary of Contribution

Robotic Exploration

Where should the robot move next to learn the most about its environment?



Occupancy Map | Mutual Information (MI)

$$H(M|Z) = H(M) - I(M;Z)$$

Theoretically Proven Approach

Move to the location that **maximizes the mutual information** between prospective range measurements and the map.

Challenge

Computing the mutual information $I(M;Z)$ is **slow** (high time complexity) and thus becomes the **bottleneck** of autonomous exploration system.

Our Contribution

Multicore mutual information (MI) hardware accelerator consisting of:

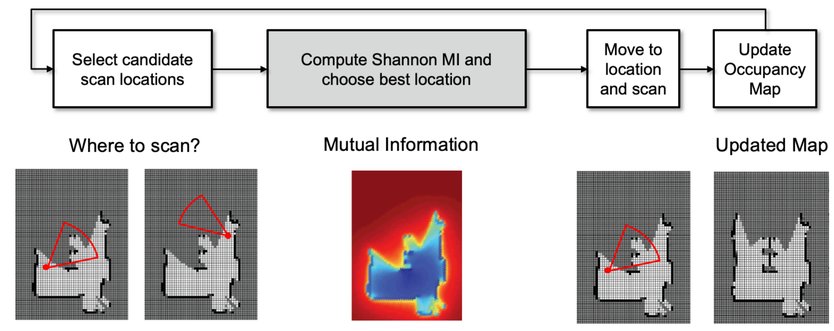
1) Specialized memory architecture
2) Fast & fair arbiter
3) 16 high-performance MI cores

System Performance

- Computes the mutual information for an **entire map** of 20m x 20m at 0.1m resolution **in under a second while consuming under 2W on an FPGA**

2. Autonomous Exploration Pipeline & MI Formulation

Typical Autonomous Exploration Pipeline



Computing Shannon MI at **more locations** allows for more optimal selection of the next scan location for exploration

Approximate Fast Shannon Mutual Information (FSMI)

[Zhang, et al., ICRA 2019]

$$I(M;Z) = \sum_{j=1}^{n} \sum_{k=j-\Delta}^{j+\Delta} P(e_j) C_k G_{|k-j|}$$

where

$$P(e_j) = \prod_{i<j} \bar{o}_i o_j$$

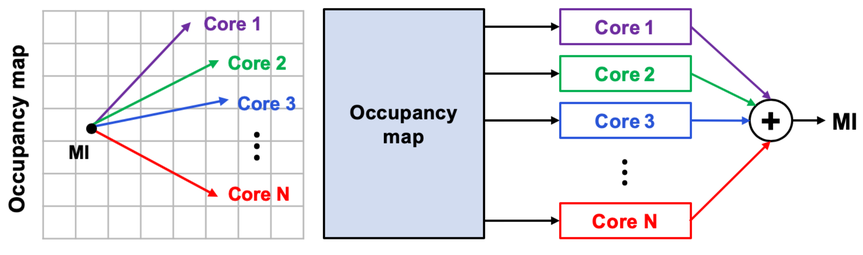
$$r_i = o_i/(1-o_i) = o_i/\bar{o}_i$$

$$C_k = \sum_{i<k} f(\delta_{emp}, r_i) + f(\delta_{occ}, r_k)$$

3. Hardware Architecture

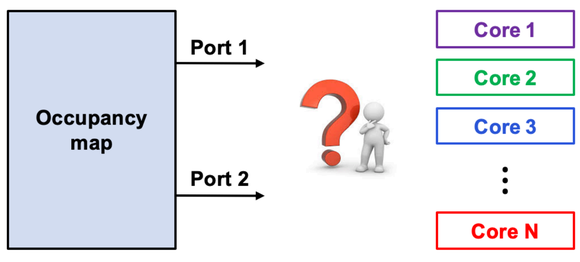
Challenge: Memory Bandwidth (Not Compute) Limits Performance

a) Algorithm is **highly parallelizable** across MI computation for every sensor beam



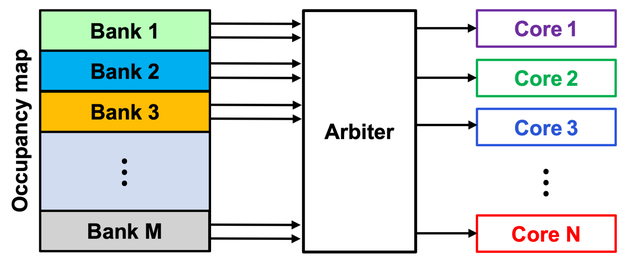
Ideally, each core computes MI for each beam using **its own memory read port** to independently access the map

b) Standard, low power SRAM has only **two ports!**



Data delivery, specifically **memory bandwidth** (not compute), limits the system throughput

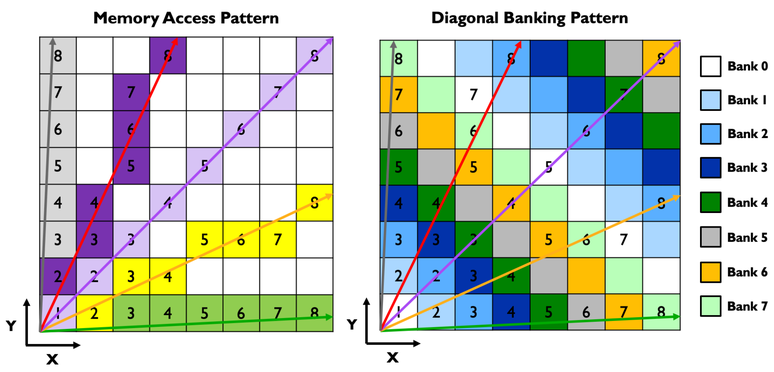
c) **Partition** and store the occupancy map into several memory banks to **increase memory bandwidth**



Specialized map partitioning pattern & arbiter are required to optimize data delivery to the cores

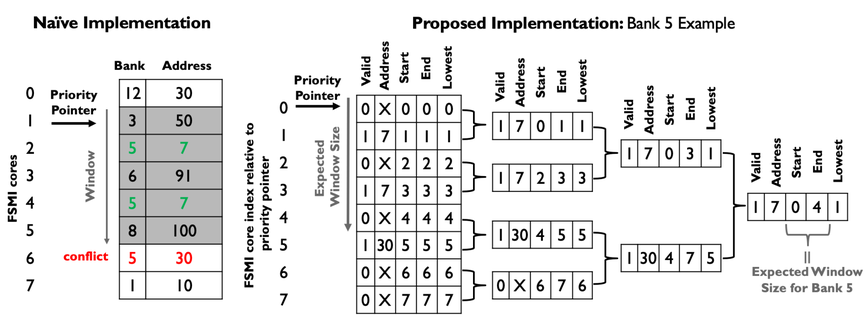
Proposed Solution & Detailed Implementation

1. **Diagonal partitioning** of the occupancy map **minimizes** memory read conflicts among cores



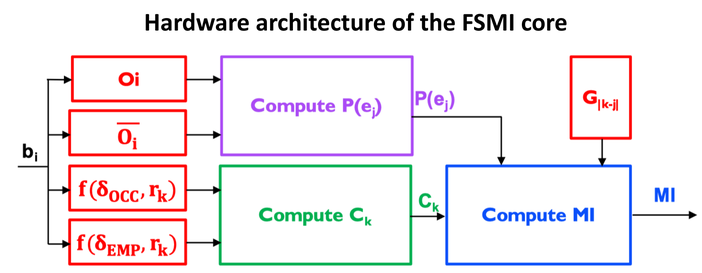
- All cores access the **same column or row** of the map every cycle (left Figure)
- Diagonal banking pattern **minimizes read conflicts** by making cores access different banks every cycle (right Figure)

2. Round-robin arbiter **fairly and quickly resolves** memory read conflicts among cores



- **Naïve implementation** (left Figure) executes in **linear time O(T)**, T = # of cores
- **Proposed implementation** (right Figure) executes in logarithmic time **O(log(T)+log(B)),** T = # of cores, B = # of banks. (T=16, B=16 in our design)

3. High-performance mutual information cores using **parallelized and pipelined** internal operations

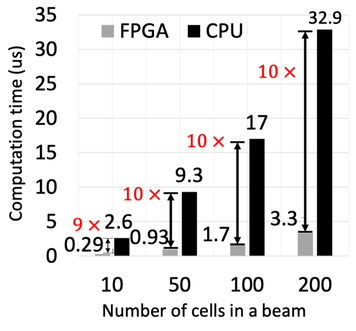


Note: Parameters (highlighted in red) needed for FSMI computation are generated by **Look Up Tables** using quantized occupancy value b_i from the map

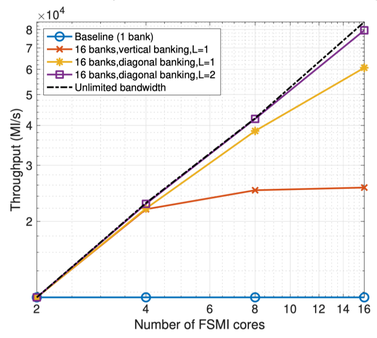
- **$P(e_j)$, C_k and I(M;Z)** are executed in **parallel** while their individual operations are **pipelined internally**
- MI for each beam is computed in **n+15 cycles**, where n is the number of cells in a sensor beam
- Final hardware system contains **16 cores**

4. Results

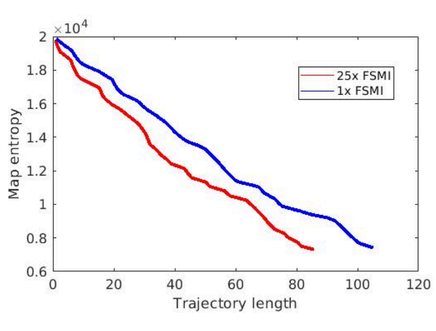
Single core performance:
10x compared with a single core (thread) Intel Xeon CPU



Multicore performance:
throughput within **94%** of theoretical limit (unlimited bandwidth)



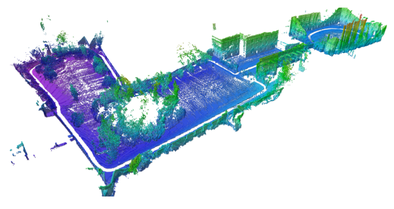
Exploration path:
20% shorter trajectory



System throughput: entire MI map with 200x200 cells at **2Hz**,
>100x faster than a single core (thread) Intel Xeon CPU
Power: **2W** on Xilinx XC7Z045 FPGA, **10x less** than Intel Xeon CPU

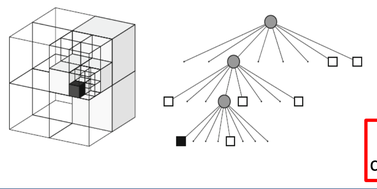
5. Extending MI Computation to 3D Environments

Computing MI on a **3D map** requires a significant amount of storage and computation

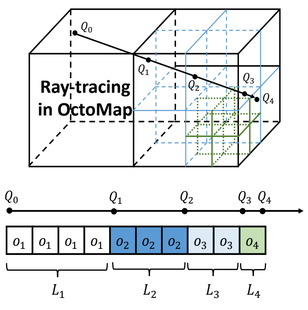


Compress the 3D map with OctoMap

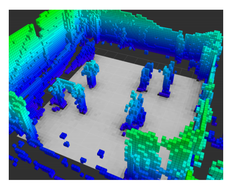
[Hornung, et al., Autonomous Robots, 2013]



Compute FSMI on the Compressed 3D Map



A 3D environment featuring an arch, a giant cat, a box and a tree



3D OctoMap where color indicates the height of a voxel

Computing FSMI directly on a 3D OctoMap achieves an **acceleration ratio of 8×**

$n_r \ll n$, significant reduction if the constants are comparable $O(n) \rightarrow O(n_r)$

For more information, please see:

Z. Zhang et al., FSMI: Fast computation of Shannon Mutual Information for information-theoretic mapping, *arXiv 2019*
http://arxiv.org/abs/1905.02238