Efficient Computing for Autonomy and Navigation

Vivienne Sze (@eems_mit)
Massachusetts Institute of Technology

In collaboration with Luca Carlone, Yu-Hsin Chen, Joel Emer, Keshav Gupta, Sertac Karaman, Tushar Krishna, Theia Henderson, Peter Li, Yi-Lun Liao, Fangchang Ma, James Noraky, Soumya Sudhakar, Amr Suleiman, Diana Wofk, Tien-Ju Yang, Zhengdong Zhang

Slides available at http://sze.mit.edu/slides
Low-Energy Autonomy and Navigation (LEAN) Group

A broad range of next-generation applications will be enabled by low-energy, miniature mobile robotics including insect-size flapping wing robots that can help with search and rescue, chip-size satellites that can explore nearby stars, and blimps that can stay in the air for years to provide communication services in remote locations. While the low-energy, miniature actuation, and sensing systems have already been developed in many of these cases, the processors currently used to run the algorithms for autonomous navigation are still energy-hungry. Our research addresses this challenge as well as brings together the robotics and hardware design communities.

We enable efficient computing on various key modules of other autonomous navigation systems including perception, localization, exploration and planning. We also consider the overall system by considering the energy cost of computing in conjunction with actuation and sensing.

Motion Planning

Many motion planning and control algorithms aim to design trajectories and controllers that minimize actuation energy. However, in low-energy robotics, computing such trajectories and controls themselves may consume a large amount of energy. We develop algorithms that optimize this trade-off.

Mutual Information for Exploration

Computing mutual information between the map and future measurements is critical to efficient exploration. Unfortunately, mutual information computation is computationally very challenging. We develop new algorithms and hardware for efficient computation of mutual information, and demonstrate real-time computation for the whole map in a reasonably-sized map.

Depth Sensing and Perception

Depth sensing is a critical function for robotic tasks such as localization, mapping and obstacle detection. State-of-the-art single-view depth estimation algorithms are based on fairly complex deep neural networks that are too slow for real-time inference on an embedded platform, for instance, mounted on a micro aerial vehicle. We address the problem of fast depth estimation on embedded systems.

Localization and Mapping

Autonomous navigation of miniatureized robots (e.g., nano/pico aerial vehicles) is currently a grand challenge for robotics research, due to the need for processing a large amount of sensor data (e.g., camera frames) with limited on-board computational resources. We focus on the design of a visual-inertial odometry (VIO) system in which the robot estimates its ego-motion (and a landmark-based map) from on-board camera and IMU data.

Group Website: [http://lean.mit.edu](http://lean.mit.edu)
Computing Challenge for Self-Driving Cars

Cameras and radar generate ~6 gigabytes of data every 30 seconds.

Self-driving car prototypes use approximately 2,500 Watts of computing power.

Generates wasted heat and some prototypes need water-cooling!
Robots Consuming < 1 Watt for Actuation

- Mini Autonomous Blimp (2017)
  - SOURCE: GEORGIA TECH
- 500 mW
  Seaglider (2003)
  - SOURCE: KONGSBERG
- 132 mW
  Chipsat (2016)
  - SOURCE: CORNELL
- 50 mW
  Robofly (2020)
  - SOURCE: UWASH.
- 31 mW
  Robobee (2019)
  - SOURCE: HARVARD
- 13.5 mW
  Robotic Water Strider (2015)
  - SOURCE: SEOUL NAT’L UNIVERSITY

Low Energy Robotics
- Miniature aerial vehicles
- Lighter than air vehicles
- Micro unmanned gliders
- Miniature satellites
Existing Processors Consume Too Much Power

< 1 Watt

> 10 Watts
Transistors Are Not Getting More Efficient

Slowdown of Moore’s Law and Dennard Scaling

General purpose microprocessors are not getting faster or more efficient

Need specialized hardware for significant improvements in speed and energy efficiency

Redesign computer from the ground up!
Efficient Computing with Cross-Layer Design

**Architectures**

*DCNN Accelerator*

- **Filter**
- **Input Image**
- **Output Image**
- **Buffer SRAM**

**14x12 PE Array**

- **Filt**
- **Img**
- **Psum**
- **Comp**
- **ReLU**

**Link Clock: Core Clock**

**Off-Chip DRAM**

- **64 bits**

**Systems**

*Object Categories / Positions*

1. at (x,y)
2. at (x,y)
3. at (x,y)

**Circuits**

*On-Chip Buffer*

*Spatial PE Array*
Energy Dominated by Data Movement

<table>
<thead>
<tr>
<th>Operation</th>
<th>Energy (pJ)</th>
</tr>
</thead>
<tbody>
<tr>
<td>8b Add</td>
<td>0.03</td>
</tr>
<tr>
<td>16b Add</td>
<td>0.05</td>
</tr>
<tr>
<td>32b Add</td>
<td>0.1</td>
</tr>
<tr>
<td>16b FP Add</td>
<td>0.4</td>
</tr>
<tr>
<td>32b FP Add</td>
<td>0.9</td>
</tr>
<tr>
<td>8b Multiply</td>
<td>0.2</td>
</tr>
<tr>
<td>32b Multiply</td>
<td>3.1</td>
</tr>
<tr>
<td>16b FP Multiply</td>
<td>1.1</td>
</tr>
<tr>
<td>32b FP Multiply</td>
<td>3.7</td>
</tr>
<tr>
<td>32b SRAM Read (8KB)</td>
<td>5</td>
</tr>
<tr>
<td>32b DRAM Read</td>
<td>640</td>
</tr>
</tbody>
</table>

Relative Energy Cost

Memory access is **orders of magnitude** higher energy than compute

[Horowitz, ISSCC 2014]
Autonomous Navigation Uses a Lot of Data

Semantic Understanding

• High frame rate
• Large resolutions
• Data expansion

Geometric Understanding

• Growing map size

2 million pixels

10x-100x more pixels

[Pire, RAS 2017]
Visual-Inertial Localization

Determines location/orientation of robot from images and IMU (also used by headset in Augmented Reality and Virtual Reality)

*Subset of SLAM algorithm (Simultaneous Localization And Mapping)
Localization at Under 25 mW

*First chip* that performs **complete** Visual-Inertial Odometry

**Front-End for camera**
*(Feature detection, tracking, and outlier elimination)*

**Front-End for IMU**
*(pre-integration of accelerometer and gyroscope data)*

**Back-End Optimization of Pose Graph**

Consumes **684×** and **1582×** less energy than mobile and desktop CPUs, respectively

[Zhang, RSS 2017], [Suleiman, VLSI-C 2018]

Joint work with Sertac Karaman
Key Methods to Reduce Data Size

**Navion:** Fully integrated system – no off-chip processing or storage

http://navion.mit.edu

Use *compression* and *exploit sparsity* to reduce memory down to 854KB

[Vivienne Sze](http://sze.mit.edu) @eems_mit

[Suleiman, VLSI-C 2018] Best Student Paper Award
Understanding the Environment

Depth Estimation

State-of-the-art approaches use Deep Neural Networks, which require up to several hundred millions of operations and weights to compute! >100x more complex than video compression

Semantic Segmentation
Deep Neural Networks (DNNs) have become a cornerstone of AI

- Computer Vision
- Speech Recognition
- Game Play
- Medical

Deep Neural Networks (DNNs) have become a cornerstone of AI.
Properties We Can Leverage

• Operations exhibit **high parallelism**
  → **high throughput** possible

• Memory Access is the Bottleneck

![Diagram showing memory read, MAC, and memory write]

**Worst Case:** all memory R/W are **DRAM** accesses

• Example: AlexNet has **724M** MACs
  → **2896M** DRAM accesses required
Properties We Can Leverage

• Operations exhibit **high parallelism**
  → **high throughput** possible

• **Input data reuse** opportunities (up to 500x)

**Convolutional Reuse**
(Activations, Weights)
CONV layers only
(sliding window)

**Fmap Reuse**
(Activations)
CONV and FC layers

**Filter Reuse**
(Weights)
CONV and FC layers
(batch size > 1)
Exploit Data Reuse at Low-Cost Memories

Normalized Energy Cost:
- 1× (Reference)
- 2×
- 6×
- 200×

* measured from a commercial 65nm process

Specialized hardware with small (< 1kB) low cost memory near compute

Farther and larger memories consume more power
Deep Neural Networks at Under 0.3W

Eyeriss: Energy-Efficient Dataflow
http://eyeriss.mit.edu

Exploits data reuse for 100x reduction in memory accesses from global buffer and 1400x reduction in memory accesses from off-chip DRAM

Overall >10x energy reduction compared to a mobile GPU (Nvidia TK1)

Results for AlexNet

Joint work with Joel Emer
Features: Energy vs. Accuracy

Accuracy (Average Precision)

Energy/ Pixel (nJ)

- Exponential
- Linear

Measured in 65nm*

- VGG16²
- AlexNet²
- HOG¹

Exponential

Video Compression

Accuracy measured in on VOC 2007 Dataset

1. DPM v5 [Girshick, 2012]

* Only feature extraction. Does not include data, classification energy, augmentation and ensemble, etc.

[Suleiman, VLSI 2016]

[Suleiman, ISSCC 2016]

[Suleiman, ISCAS 2017]
Energy-Efficient Processing of DNNs

A significant amount of algorithm and hardware research on energy-efficient processing of DNNs

We identified various limitations to existing approaches

Hardware Architectures for Deep Neural Networks

ISCA Tutorial
June 24, 2017
Website: http://eyeriss.mit.edu/tutorial.html

http://eyeriss.mit.edu/tutorial.html

V. Sze, Y.-H. Chen, T-J. Yang, J. Emer,
“Efficient Processing of Deep Neural Networks: A Tutorial and Survey,”
Proceedings of the IEEE, Dec. 2017
Design of Efficient DNN Algorithms

Popular efficient DNN algorithm approaches

Network Pruning

Efficient Network Architectures

Examples: SqueezeNet, MobileNet

... also reduced precision

• Focus on reducing number of MACs and weights
• Does it translate to energy savings and reduced latency?

[Chen*, Yang*, SysML 2018]
Number of MACs and Weights are Not Good Proxies

# of operations (MACs) does not approximate latency well

Source: Google

# of weights alone is not a good metric for energy
(All data types should be considered)

Energy breakdown of GoogLeNet

[https://energyestimation.mit.edu/]
[Yang, CVPR 2017]
Energy-Aware Pruning

Directly target energy and incorporate it into the optimization of DNNs to provide greater energy savings

• Sort layers based on energy and prune layers that consume the most energy first

• **Energy-aware pruning** reduces AlexNet energy by 3.7x w/ similar accuracy

• Outperforms magnitude-based pruning by 1.7x

[Yang, CVPR 2017]

Pruned models available at http://eyeriss.mit.edu/energy.html
NetAdapt: Platform-Aware DNN Adaptation

- **Automatically adapt DNN** to a mobile platform to reach a target latency or energy budget
- Use **empirical measurements** to guide optimization (avoid modeling of tool chain or platform architecture)
- **Few hyperparameters** to reduce tuning effort
- **>1.7x speed up** on MobileNet w/ similar accuracy

[Yang, ECCV 2018]

Code available at [http://netadapt.mit.edu](http://netadapt.mit.edu)

Joint work with Google’s Mobile Vision Team
FastDepth: Fast Monocular Depth Estimation

Depth estimation from a single RGB image desirable, due to the relatively low cost and size of monocular cameras.

Auto Encoder DNN Architecture (Dense Output)
FastDepth: Fast Monocular Depth Estimation

Apply *NetAdapt*, compact network design, and depth wise decomposition to decoder layer to enable depth estimation at **high frame rates on an embedded platform** while still maintaining accuracy.

Models available at [http://fastdepth.mit.edu](http://fastdepth.mit.edu)

Joint work with Sertac Karaman

[Wofk*, Ma*, ICRA 2019]
NetAdapt v2: Reduce Adaption Time

Reduce time to find efficient DNN that adapts to hardware by up to 5.8x

Typical Steps in Neural Architecture Search (NAS):
1) Train super-network (search space of DNNs)
2) Sample and evaluate different DNNs
3) Fine tune the final DNN

Contributions
• **Ordered dropout**: train multiple DNNs in single forward pass (reduce step 1)
• **Channel-level bypass**: merge layer depth and channel width into a single search dimension (reduce step 2)
• **Multi-layer coordinate descent optimizer**: consider joint effect of multiple layers (reduce step 2 & support non-differentiable metrics, e.g., latency)

More info at [http://netadapt.mit.edu](http://netadapt.mit.edu)

[Vivienne Sze](http://sze.mit.edu)  @eems_mit  [Yang, CVPR 2021]
Measuring Uncertainty in DNN Monocular Depth Estimation

Need to estimate uncertainty (sensor noise model) for robot decision making.

Popular approaches involve running multiple DNNs on the same input.

[Sudhakar, ICRA 2022] Joint work with Sertac Karaman
Uncertainty from Motion (UfM)

It exploits **temporal redundancy** in video inputs by **merging outputs** that belong to the same point in 3D space across multiple views to estimate uncertainty.

UfM needs to run only **one** DNN per input.

seen for first time
seen for k’th time
not seen

RGB input
Depth pred.
Uncertainty pred.

Frame 1
Frame 2
Frame 3

[Joint work with Sertac Karaman]

[Vivienne Sze](http://sze.mit.edu/)  
[@eems_mit](http://twitter.com/eems_mit)

[Sudhakar, ICRA 2022]
Mapping with Gaussian Mixture Models

Convert depth images to Gaussian Mixture Models (GMMs) to construct a compact 3D map of an environment.

While existing approaches focus on reducing map size, they do not account for the memory cost during the conversion process.

- 2D Depth Image
- Gaussian Mixture Models (blue)

307,200 pixels (3.5MB)

Around 1000 parameters (12-18 kB)

[Li, ICRA 2022] Joint work with Sertac Karaman
Single Pass Gaussian Fitting (SPGF)

### SPGF Approach: Scanline Segmentation + Segment Fusion

- **Single pass** reduces storage of inputs and temporary variables
- **Row-by-row based approach** allows for accurate and efficient inference of surface geometries in a single pass
SPGF Results on TUM RGB-D Room

Comparison of SPGF with other approaches at similar accuracy and compactness

Hierarchical EM (H-EM): [Eckart, CVPR 2016], Normal Distance Transform (NDT): [Saarinen, IJRR 2013], Region Growing (RG): [Dhawale, RSS 2020]

SPGF only uses KBs of memory overhead and achieves real-time on a low-power ARM Cortex-57 CPU

Note: All algorithms were similarly optimized in C++
Where to Go Next: Planning and Mapping

Robot Exploration
**Mutual-Information-Based Exploration**

**Robot Exploration:** Decide where to go by computing Shannon Mutual Information

1. Select candidate scan locations
2. Compute Shannon MI and choose best location
3. Move to location and scan
4. Update Occupancy Map

Where to scan?  
Mutual Information  
Updated Map

Joint work with Sertac Karaman
Information Theoretic Mapping

Occupancy grid map, $M$

Mutual information map, $I(M; Z)$

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

Perspective updated map entropy

Current map entropy

Mutual information
FSMI: Fast Shannon Mutual Information

Shannon Mutual Information
(between ray $Z$ and map $M$)
[Julian, *IJRR* 2014]

$$I(M; Z) = \sum_{i=1}^{n} \int_{z \geq 0} P(z) f(\delta_i(z), r_i) \, dz$$

No closed form solution. Requires expensive numerical integration at resolution $\lambda_z$. $O(n^2 \lambda_z)$

**FSMI: Fast Shannon Mutual Information**

Evaluate MI for all cells in entire ray altogether removes numerical integration. $O(n^2)$

**Approximate FSMI**

Approximate noise model of depth sensor with truncated Gaussian*. $O(n)$

*Charrow et al., ICRA 2015

---

Vivienne Sze [http://sze.mit.edu/](http://sze.mit.edu/) @eems_mit

[Zhang, ICRA 2019]

Joint work with Sertac Karaman
FSMI: Fast Shannon Mutual Information

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$O(n^2 \lambda_z)$</td>
<td>$O(n^2)$</td>
<td>$O(n)$</td>
<td>$O(n)$</td>
</tr>
</tbody>
</table>

Measured run time per ray (µsec) on an Intel Xeon core (desktop):

- Original MI: 188046
- FSMI: 132
- CSQMI: 29
- Approximate FSMI: 17

Measured run time per ray (µsec) on an ARM Cortex-A57 core (embedded):

- CSQMI: 422
- Approximate FSMI: 149

Approximate FSMI is over 1000x faster than original MI and 1.7 – 2.8x faster than CSQMI

Experimental Results (4x Real Time)

Exploration with a mini race car using motion capture for localization

Occupancy map with planned path using RRT* (compute MI on all possible paths)

MI surface

[Zhang, ICRA 2019] Joint work with Sertac Karaman
**Building Hardware to Compute FSMI**

**Motivation:** Compute MI faster for faster exploration!

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

Algorithm is *embarrassingly* parallel!
High throughput *should* be possible with multiple cores.

**Process beams in parallel with multiple cores**
Challenge is Data Delivery to All Cores

Power consumption of memory scales with number of ports.

**Low power SRAM limited to two-ports!**

Data delivery, specifically memory bandwidth, limits the throughput (not compute)
Specialized Memory Architecture

Break up map into separate memory banks and novel storage pattern to minimize read conflicts when processing different rays in parallel.
Experimental Results

Specialized banking, efficient memory arbiter and packing multiple values at each address results in throughput within 94% of theoretical limit (unlimited bandwidth).

Compute MI for an entire map of 20m x 20m at 0.1m resolution in under a second while consuming under 2W on a ZC706 FPGA (100x faster than CPU at 10x lower power).

[Li, RSS 2019]
FCMI: Fast Continuous Mutual Information

Reformulate with a *continuous* occupancy map framework and exploit recursive structure when computing MI across *entire* map

\[ \text{FSMI: } O(nLH^2) \rightarrow \text{FCMI: } O(LH^2) \]

*Two orders of magnitude speed up over FSMI!*

\( n = \text{cells per ray} \)
\( L = \text{number of rays} \)
\( H^2 = \text{size of map} \)
Several Orders of Magnitude Speed up Via Co-Design

For a 200x200 Map
(Note: Speed up increases for larger maps)

- **Optimize memory subsystem** (banking) for multi-beam parallel processing
- **Reformulate** using a continuous occupancy map framework and exploit recursive structure
- **Evaluate** MI for all cells in entire ray altogether, removes numerical integration
- **Optimize memory subsystem**, time-interleave cores and approximate computing

Compute mutual information for the **entire map** in real time for the first time!

[Shannon MI] [FSMI (CPU)] [FSMI (hardware)] [FCMI (CPU)] [FCMI (hardware)]

[Julian, IJRR 2014] [Zhang, ICRA 2019] [Li, RSS 2019] [Henderson, ICRA 2020] [Gupta, IROS 2021]

Joint work with Sertac Karaman
Balancing Actuation and Computing Energy

**Motion Planning**
Find a feasible (obstacle-free) path [typically optimize for shortest path]

**Low-Energy Robotics**
Actuation and computing energy are similar order of magnitude

---

Energy to move 1 more meter \( (P_a/v [\text{W/(m/s)}]) \)

Energy to compute 1 more second \( (P_c [\text{W}]) \)

---

Joint work with Sertac Karaman

[Sudhakar, ICRA 2020]
Balancing Actuation and Computing Energy

Baseline
(compute 20,000 samples)

Compute Energy Included Motion Planning (CEIMP)
A framework to balance the energy spent on computing a path and the energy spent on moving along that path (Don’t think too hard!)

Vivienne Sze http://sze.mit.edu/ @eems_mit

[Sudhakar, ICRA 2020] Joint work with Sertac Karaman
Low Power 3D Time of Flight Imaging

- Pulsed Time of Flight: Measure distance using round trip time of laser light for each image pixel
  - Illumination + Imager Power: 2.5 – 20 W for range from 1 - 8 m

- Use computer vision techniques and passive images to estimate changes in depth without turning on laser
  - CMOS Imaging Sensor Power: < 350 mW

Real-time Performance on Embedded Processor
VGA @ 30 fps on Cortex-A7 CPU (< 0.5W active power)

[Noraky, TCSVT 2019]

Estimated Depth Maps
Results of Low Power Depth ToF Imaging

Mean Relative Error: 0.7%
Duty Cycle (on-time of laser): 11%

Noraky, TCSVT 2019
Summary

- Efficient computing is critical for advancing the progress of autonomous robots, particularly at the smaller scales. → Critical step to making autonomy ubiquitous!

- In order to meet computing demands in terms of power and speed, need to redesign computing hardware from the ground up → Focus on data movement!

- Specialized hardware creates new opportunities for the co-design of algorithms and hardware → Innovation opportunities for the future of robotics!
Acknowledgements

Research conducted in the MIT Energy-Efficient Multimedia Systems Group would not be possible without the support of the following organizations:

[Logos of various organizations]

Vivienne Sze [http://sze.mit.edu/] @eems_mit
Low-Energy Autonomy and Navigation (LEAN) Group

A broad range of next-generation applications will be enabled by low-energy, miniature mobile robotics including insect-size flapping wing robots that can help with search and rescue, chip-size satellites that can explore nebulae stars, and blimps that can stay in the air for years to provide communication services in remote locations. While low-energy, miniature actuators, and sensing systems have already been developed, in many of these cases, the processors currently used to run the algorithms for autonomous navigation are still energy-hungry. Our research addresses this challenge as well as brings together the robotics and hardware design communities.

We enable efficient computing on various key modules of other autonomous navigation systems including perception, localization, exploration and planning. We also consider the overall system by considering the energy cost of computing in conjunction with actuation and sensing.

Motion Planning

Many motion planning and control algorithms aim to design trajectories and controllers that minimize actuation energy. However, in low-energy robotics, computing such trajectories and controls themselves may consume a large amount of energy. We develop algorithms that optimize this trade-off.

Mutual Information for Exploration

Computing mutual information between the map and future measurements is critical to efficient exploration. Unfortunately, mutual information computation is computationally very challenging. We develop new algorithms and hardware for efficient computation of mutual information, and demonstrate real-time computation for the whole map in a reasonably-sized map.

Depth Sensing and Perception

Depth sensing is a critical function for robotic tasks such as localization, mapping and obstacle detection. State-of-the-art single-view depth estimation algorithms are based on fairly complex deep neural networks that are too slow for real-time inference on an embedded platform. For instance, mounted on a micro aerial vehicle, we address the problem of fast depth estimation on embedded systems.

Localization and Mapping

Autonomous navigation of miniature-sized robots (e.g., nano/pico aerial vehicles) is currently a grand challenge for robotics research, due to the need for processing a large amount of sensor data (e.g., camera frames) with limited on-board computational resources. We focus on the design of a visual-inertial odometry (VIO) system in which the robot estimates its ego-motion (and a landmark-based map) from on-board camera and IMU data.

Group Website: [http://lean.mit.edu](http://lean.mit.edu)

Vivienne Sze [http://sze.mit.edu](http://sze.mit.edu) / [eems_mit](https://twitter.com/eems_mit)
Resources on Efficient Processing of DNNs

http://eyeriss.mit.edu/tutorial.html

Joint work with Joel Emer

Vivienne Sze ☑️ http://sze.mit.edu/ ✏️@eems_mit
Additional Resources

Talks and Tutorial Available Online
http://sze.mit.edu/slides
References

• Energy-Efficient Visual Inertial Localization
  – Project website: http://navion.mit.edu

• Efficient Map Compression
  – Project website: https://lean.mit.edu/highlights/localization-mapping
**References**

- **Efficient Processing for Deep Neural Networks**
  - Project website: [http://eyeriss.mit.edu](http://eyeriss.mit.edu)
References

• Co-Design of Algorithms and Hardware for Deep Neural Networks

• Monocular Depth Estimation using Deep Neural Networks
  – Project website: https://lean.mit.edu/highlights/depth-sensing
• **Fast Shannon Mutual Information for Robot Exploration**
  
  ‒ **Project website:** [https://lean.mit.edu/highlights/mutual-information](https://lean.mit.edu/highlights/mutual-information)
  
  
  
  
  
References

• Balancing Actuation and Computation
  – Project website: https://lean.mit.edu/highlights/motion-planning

• Low Power Time of Flight Imaging