Resource Efficient Navigation Using Bitstream Computing

Unary Computing Workshop ’19

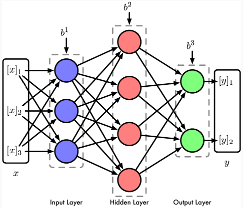
Kyle Daruwalla and Mikko Lipasti

22 June 2019

University of Wisconsin - Madison, Dept. of Elec. and Comp. Eng.

Motivation

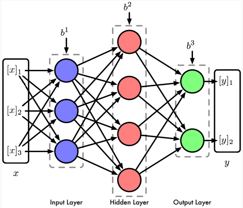
Autonomous navigation of unknown environments is a challenging computational problem



Motivation

Autonomous navigation of unknown environments is a challenging computational problem

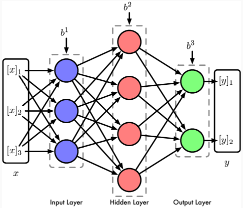
Task that the brain is uniquely efficient at solving



Motivation

Autonomous navigation of unknown environments is a challenging computational problem

Task that the brain is uniquely efficient at solving



Can we leverage the efficiency of the brain with current CV/ML applications?

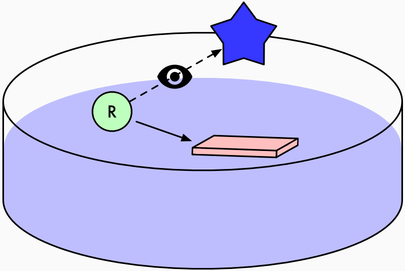
Presentation Summary

In this talk we will:

1. Point out the ineffectiveness of alternative methods
2. Frame the problem of navigation using computer vision
3. Identify bottlenecks that make FP/FXP implementations power-hungry
4. Apply bitstream computing to make implementations feasible
5. Discuss simulation and synthesized hardware results

Setup and Background

Problem Setup



Robot must navigate an unknown environment via visual cues (Morris Water Maze[1])

[1] Morris et al. 1982.

Reinforcement Learning

Divide environment in states (e.g. grid)

Divide environment in states (e.g. grid)

Define set of possible actions in each state

Assign value to each action

Reinforcement Learning

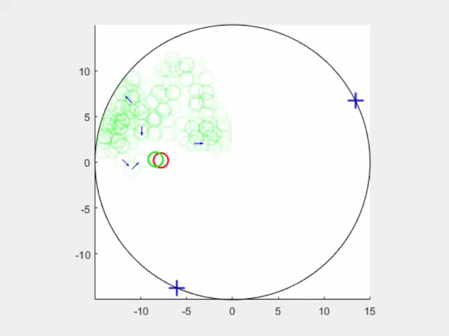
Divide environment in states (e.g. grid)

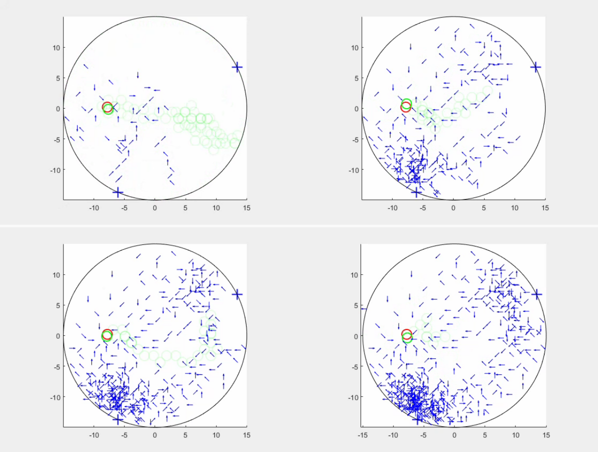
Define set of possible actions in each state

Assign value to each action

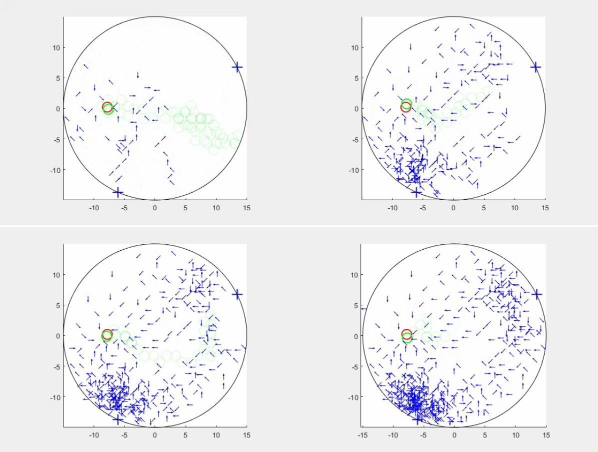
Iteratively explore the space and update values

Actions with high value are the best actions to take




10
5
0
-5
-10
-15
-10
-5
0
5
10
15

Reinforcement Learning



Warning!

Slow to learn and converge!

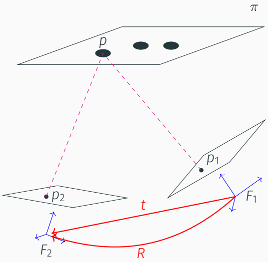
Reinforcement Learning

RL does not leverage the structure of the space.

Can we use CV to more efficiently navigate?

Homography Setup

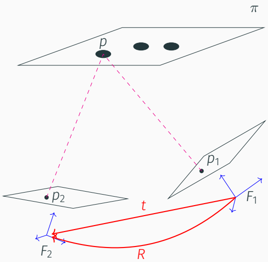
Use perspective maps of the same feature point at the current and target locations



Find relationship between p_1 and p_2 to determine rotation (R) and translation (t)

Homography Setup

Use perspective maps of the same feature point at the current and target locations



$$p_2 \sim Hp_1$$

$$p_1 = \begin{bmatrix} x & y & 1 \end{bmatrix}^\top \quad p_2 = \begin{bmatrix} u & v & 1 \end{bmatrix}^\top$$

Find relationship between p_1 and p_2 to determine rotation (R) and translation (t)

Homography Summary

Process of finding relationship between pairs of feature points:

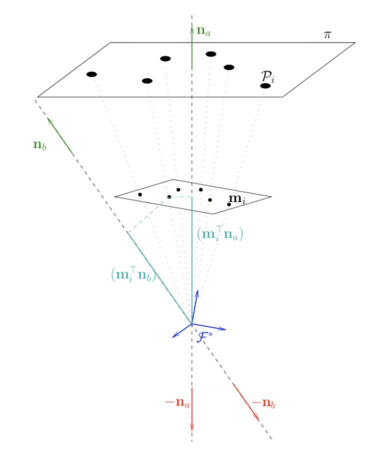
1. Homography estimation (finding H):
   requires singular value decomposition of 8×9 matrix[2] [3]
2. Homography decomposition ($H \Rightarrow R + nt^\top$):
   requires singular value decomposition of 3×3 matrix[4]

---

[2] Dubrofsky 2009.
[3] Hartley 1997.
[4] Malis and Vargas 2007.

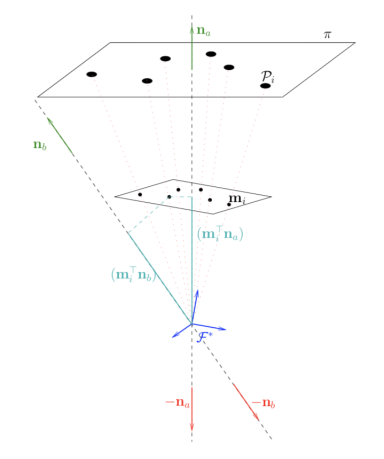
Caveats to Homography Technique

- In general, there are eight solutions



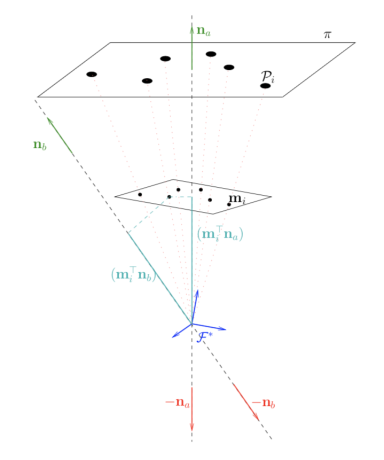
Caveats to Homography Technique

- In general, there are eight solutions
- Zhang SVD-based decomposition naturally elimates four solutions (by assuming that the camera cannot cross through the reference plane)



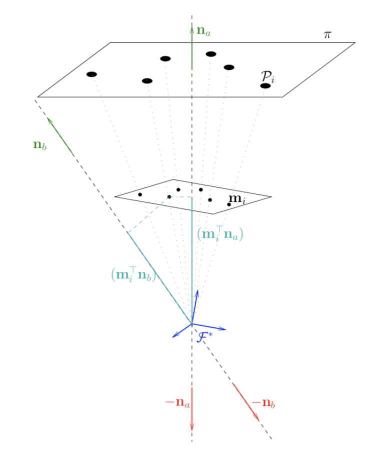
Caveats to Homography Technique

- In general, there are eight solutions
- Zhang SVD-based decomposition naturally elimates four solutions (by assuming that the camera cannot cross through the reference plane)
- Need to apply reference point visibility to eliminate two more solutions



Caveats to Homography Technique

- In general, there are eight solutions
- Zhang SVD-based decomposition naturally elimates four solutions (by assuming that the camera cannot cross through the reference plane)
- Need to apply reference point visibility to eliminate two more solutions
- Need to use multiple decompositions to select final solution

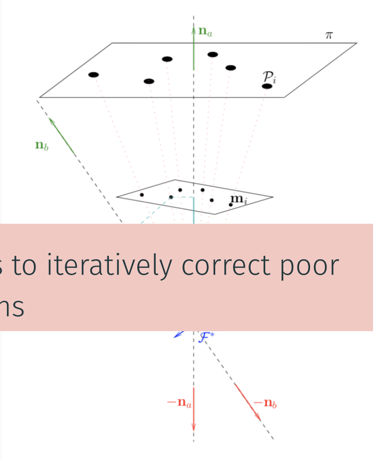


Caveats to Homography Technique

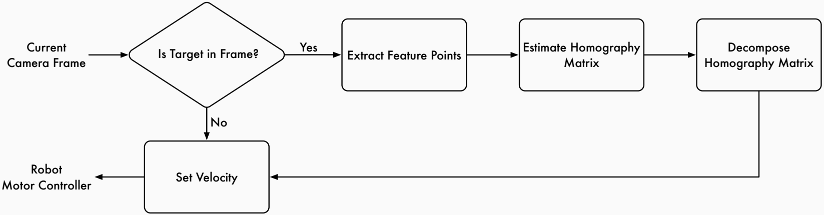
- In general, there are eight solutions
- Zhang SVD-based decomposition naturally elimates four solutions (by assuming that the camera cannot cross through the

Must repeatedly do this process to iteratively correct poor solutions

point visibility to eliminate two more solutions
- Need to use multiple decompositions to select final solution



Overall Data Flow Graph



Stochastic Computing

Major Bottlenecks

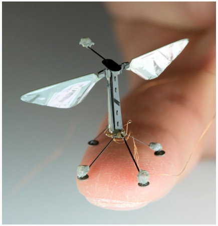
Would like to make algorithm feasible for PAVs ($< 35\,\mathrm{mW}$)

Most operations are matrix multiplication

- Implemented by prior work [5]

Need to take SVD of a 8×9 and 3×3 matrix

- Major bottleneck
- How can we do this stochastically?



[5] Shukla, Jorgensen, and Lipasti 2017

The SVD of a matrix $A \in \mathbb{R}^{m \times n}$ is

$$A = U\Sigma V^\top$$

$$U = \begin{bmatrix} \vdots & \vdots & & \vdots \\ u_1 & u_2 & \dots & u_r \\ \vdots & \vdots & & \vdots \end{bmatrix} \quad \Sigma = \begin{bmatrix} \sigma_1 & 0 & \dots & 0 \\ 0 & \sigma_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sigma_r \end{bmatrix} \quad V = \begin{bmatrix} \vdots & \vdots & & \vdots \\ v_1 & v_2 & \dots & v_r \\ \vdots & \vdots & & \vdots \end{bmatrix}$$

Algorithm 1 Iterative SVD[6]

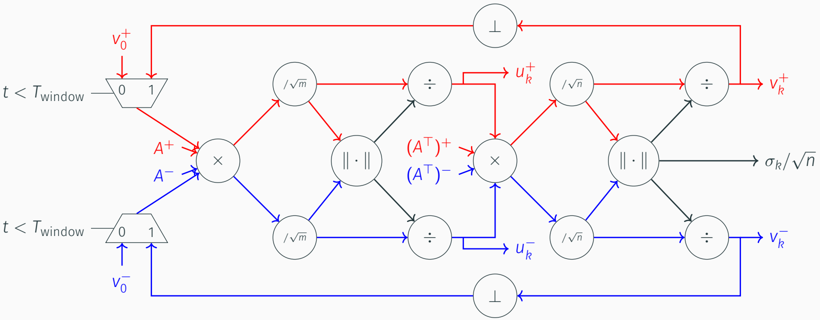
Require: Input matrix $A \in \mathbb{R}^{m \times n}$ and initial guess $v_0 \in \mathbb{R}^n$

1: **for** $k = 1, 2, \ldots$ (until convergence) **do**
2: $\quad w_k = Av_{k-1}$
3: $\quad \alpha_k = \|w_k\|_2 = \sqrt{w_k^\top w_k}$
4: $\quad u_k = w_k / \alpha_k$
5: $\quad z_k = A^\top u_k$
6: $\quad \sigma_k = \|z_k\|_2 = \sqrt{z_k^\top z_k}$
7: $\quad v_k = z_k / \sigma_k$
8: **end for**
9: **return** First left/right singular vectors, u_k & v_k, and first singular value, σ_k

Similar to prior work on pseudoinverse[7] and eigenvalue decomposition[8] using stochastic computing

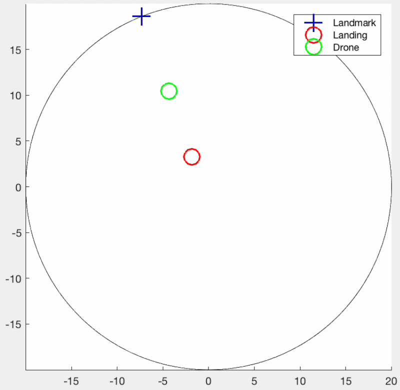
[6] Bentbib and Kanber 2015.
[7] Shukla, Jorgensen, and Lipasti 2017.
[8] Ting and Hayes 2014.

Iterative SVD Block Diagram



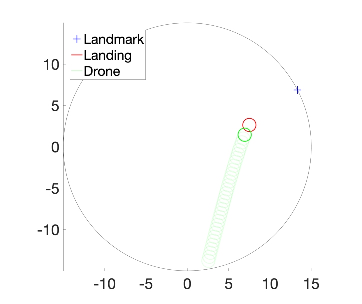
Results

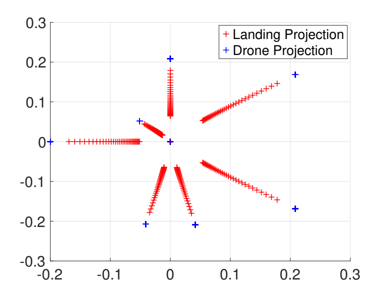
Simulated Results



Homography Navigation

Simulated Results





Hardware Implementation

Iterative SVD implemented in BitSAD

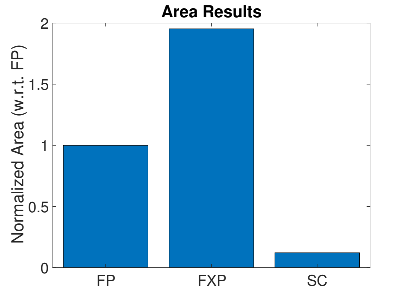
Mapped to ultra-low power Lattice LM4K FPGAs

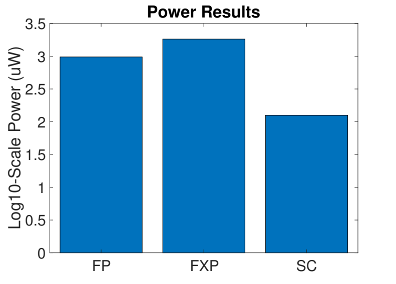
FP/FXP implementations done using Vivado HLS

FP/FXP cannot fit on Lattice FPGAs

- But we assume ideal partitioning
- FP requires 8 chips
- FXP requires 15 chips

Hardware Results





Conclusion

Concluding Remarks

We have demonstrated:

- An iterative stochastic computing algorithm for SVD
- Simulated navigation of an unknown environment using well-known computer vision techniques
- Stochastic computing implementations have much lower resource consumption

Remaining concerns:

- Extend this approach to other PAV applications
- Address latency issue for real-time deadlines
- Objects blocking field of view
    - Break main goal into series of navigation tasks?

Questions?

Bentbib, A. H. and A. Kanber (2015). "Block power method for SVD decomposition". In: *Analele Stiintifice ale Universitatii Ovidius Constanta, Seria Matematica* 23.2, pp. 45–58. ISSN: 18440835. DOI: 10.1515/auom-2015-0024.

Dubrofsky, Elan (2009). "Homography Estimation". PhD thesis. The University of British Columbia.

Hartley, Richard I. (1997). "In defense of the eight-point algorithm". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 19.6, pp. 580–593. ISSN: 01628828. DOI: 10.1109/34.601246. URL: http://ieeexplore.ieee.org/document/601246/.

Malis, Ezio and Manuel Vargas (2007). "Deeper understanding of the homography decomposition for vision-based control". In: *Sophia* 6303.6303, p. 90. ISSN: 0036-8075. DOI: 10.1126/science.318.5857.1691b. URL: http://hal.archives-ouvertes.fr/inria-00174036/.

Morris, R. G. M. et al. (1982). "Place navigation impaired in rats with hippocampal lesions". In: *Nature* 297.5868, pp. 681–683. ISSN: 0028-0836. DOI: 10.1038/297681a0. URL: http://www.nature.com/articles/297681a0.

References iii

Shukla, Rohit, Erik Jorgensen, and Mikko Lipasti (2017). "Evaluating hopfield-network-based linear solvers for hardware constrained neural substrates". In: *Proceedings of the International Joint Conference on Neural Networks* 2017-May, pp. 3938–3945. DOI: 10.1109/IJCNN.2017.7966352.

Ting, Pai Shun and John Patrick Hayes (2014). "Stochastic logic realization of matrix operations". In: *Proceedings - 2014 17th Euromicro Conference on Digital System Design, DSD 2014*, pp. 356–364. DOI: 10.1109/DSD.2014.75.