Harpo

RCC for Multi-scale Cellular Image Processing

Reid Porter, Al Conti, Jan Frigo, Maya Gokhale, Neal Harvey, Garrett Kenyon

1

Introduction to Cellular Algorithms

Lattice Gas Automata

2

The Harpo Framework

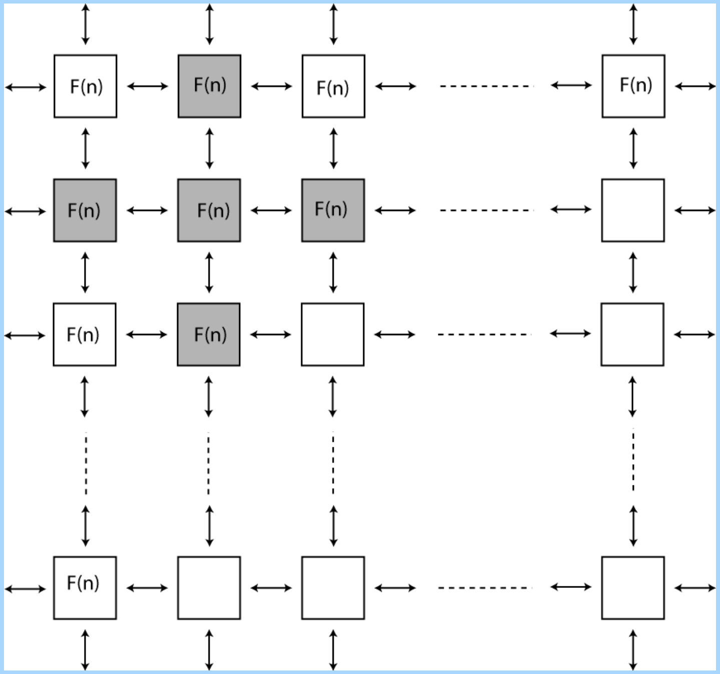
Overview, examples and performance assessment.

3

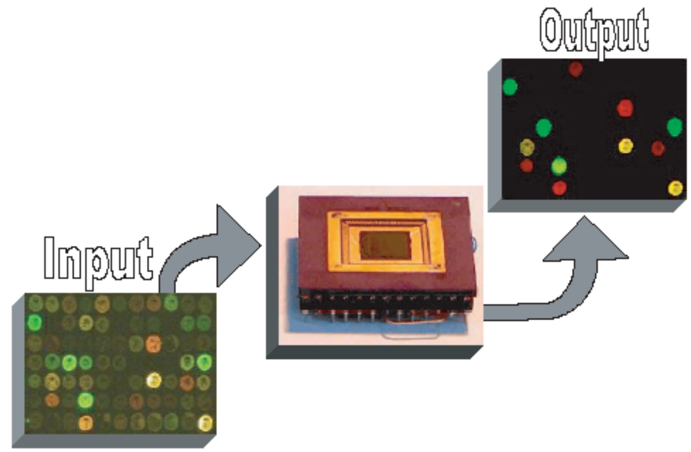
Example Applications

Plumes, people and movies..

Introduction to Cellular Algorithms



Simple, parallel and local.



Massively parallel input and output

Examples

- Cellular Automata, von Neumann, John, late 1940s.
- Cellular Neural \ Nonlinear Networks, L. Chua, L. Yang, IEEE Transactions on Circuits and Systems, 35 (10), 1988.
- Cellular image and video processing
  The Neocognitron and convolution neural networks.

-State variable update based on state variables within a local neighborhood: $\boldsymbol{x}_{t+1} = f\left(\boldsymbol{x}_t\right)$

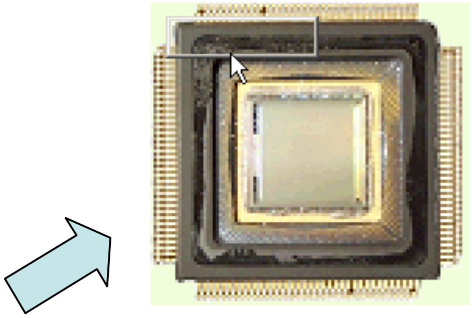
Cellular Automata: $f\left(\boldsymbol{x}\right) = \left(x^L \text{and}\, x^R\right) \text{or} \left(x^T \text{and}\, x^B\right)$

Cellular Nonlinear Networks: $f\left(\boldsymbol{x}\right) = \mathrm{T}\left(\sum_{i \in \boldsymbol{x}} w^i x^i\right)$

- Synchronous / Asynchronous, Discrete / Continuous, Homogenous / Inhomogeneous variants.

- Simple, parallel and local means many different technologies have been targeted.
- Reconfigurable Computers can exploit **fine grain parallelism** and are **commercial-off-the-shelf**.

20 Years of Cellular Algorithm Hardware



1985 Custom parallel processor architecture for Cellular Automata

The Connection Machine, W.D. Hillis, MIT Press, Cambridge, Mass, 1985.

1995 FPGA implementation of Cellular Automata

The Cellular Processing Machine CEPRA-8L, Rolf Hoffmann, K. Volhmann, Sobolewski, Mathematical Research, 81:179-188, 1994.

1996 ASIC implementation of Cellular Nonlinear Networks

CNN universal chip in CMOS technology., Espejo, S; Carmona, R; DominguezCastro, R; RodriguezVazquez, A, International Journal of Circuit Theory and Applications; Jan.-Feb. 1996; vol.24, no.1, p.93-109

2000 FPGA implementation of Cellular Nonlinear Networks

Low-cost, high-performance CNN simulator implemented in FPGA *Cellular Neural Networks and Their Applications, Perko, M. ; Fajfar, I. ; Tuma, T. ; Puhan, J. Proceedings of the 6th IEEE International Workshop on* Cellular Neural Networks; May, 2000.

2003 FPGA implementation of Multi-layered Cellular Networks

Optimizing Digital Hardware Perceptrons for Multi-Spectral Image Classification, Reid Porter et. al., J. Math. Imaging and Vision 19, 133-150, 2003.

Configurable multilayer CNN-UM emulator on FPGA, Z. Nagy, P. Szolgay, IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications; June 2003; vol.50, no.6, p.774-8.

2004 FPGA implementation of Cellular Automata continues…

Implementing cellular automata in FPGA logic, Halbach, M.; Hoffmann, R., 18th International Parallel and Distributed Processing Symposium, 26-30 April 2004, Santa Fe, NM, USA; p.258

2004 ASIC implementation of Cellular Nonlinear Networks continues…

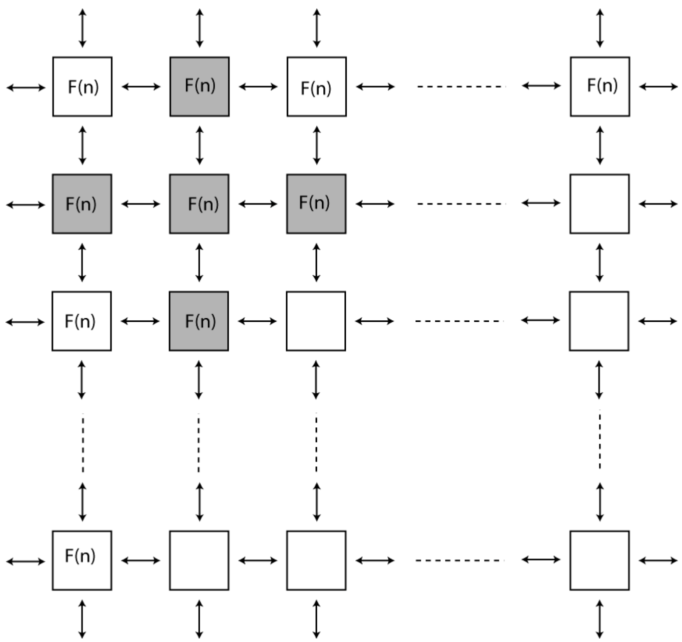
ACE16k: The third generation of mixed-signal SIMD-CNN ACE chips toward VSoCs, Rodriguez-Vazquez, A et. al., IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications; May 2004; vol.51, no.5, p.851-63

2005 FPGA implementation for flexible topology, multi-scale cellular algorithms

A Reconfigurable Computing Framework for Optimal Multi-scale Cellular Image Processing, submitted, 2005.

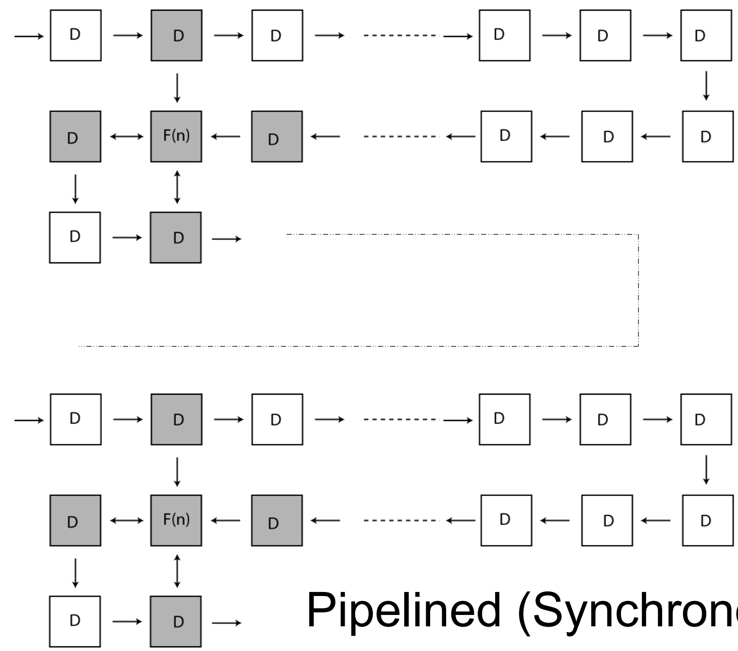
Synchronous, Discrete Cellular Algorithms in Space and Time

O(N) external bandwidth



N cells

O(1) external bandwidth

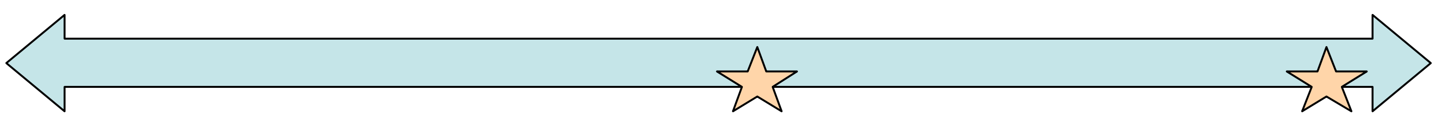


Pipelined (Synchronous)

Overhead: Edge data and latency

Data Parallel

Instruction Parallel



Cellular Automata

Harpo

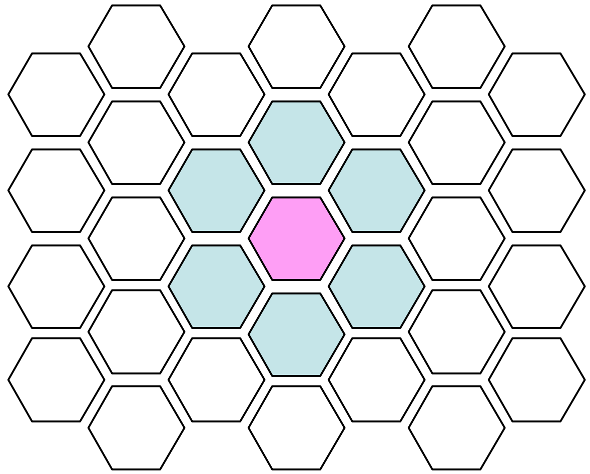
Computer vision

Lattice Gas Cellular Automata

Shown in 1980s to be mathematically equivalent to Navier-Stokes simulation.

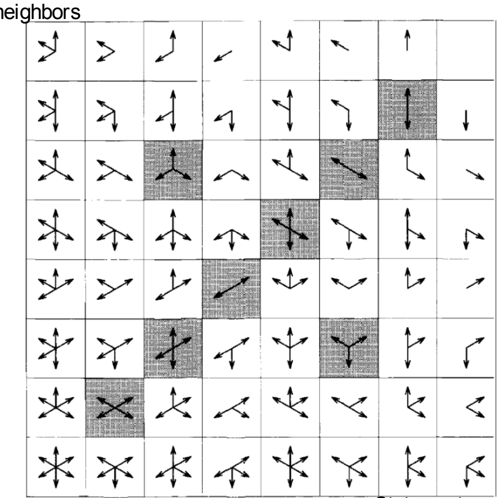
Advantages:

- Efficient and scalable.
- Complex geometry easily modeled.
- *Examples:* flow through porous media, automobile aerodynamics.



Hexagonal Lattice Network

State vector for 3 neighbors

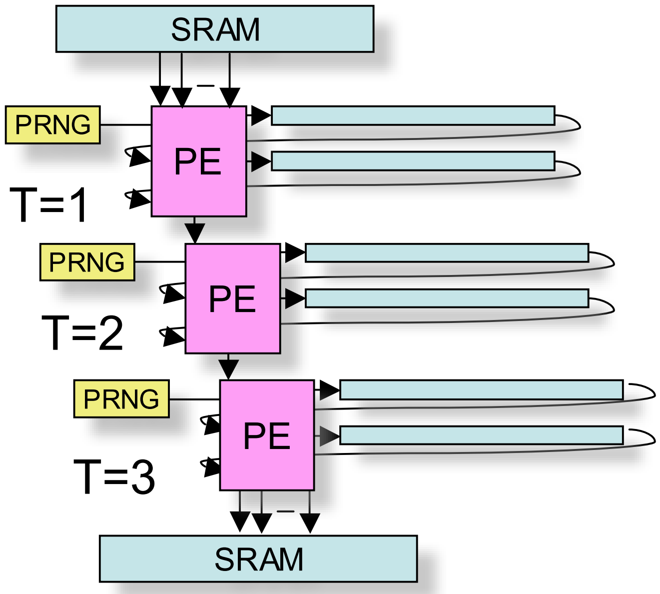


Update Matrix

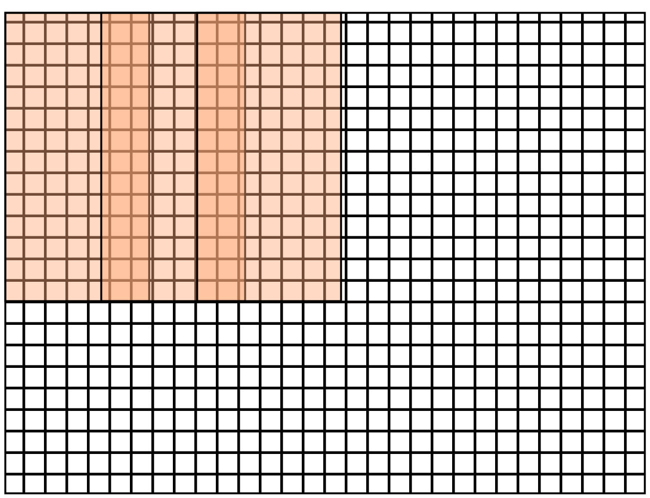
State vector for 3 neighbors

Snap-shot of lattice gas evolution

Cellular Automata Implementation



Pipelined CA


Tiling the large scale arrays

Shift registers on-chip
512 Deep * 14208 Wide

Shift registers off-chip
2M Deep * 216 Wide

Pipelined (222):
2048 * 27268 cells
169 cells / clock cycle (76% efficiency)

Pipelined * Parallel (55 * 16 = 880):
8192 * 6912 cells
840 cells / clock cycle (95% efficiency)
At 100MHz this means 84Gcells/sec

With communication (required for arbitrary sized arrays) we estimate 10Gcells/sec

Xeon 3.19GHz : 12 – *58* MCells / second
833 – 172 x speed-up

2 The HARPO Framework

1 **Introduction to Cellular Algorithms**

Lattice Gas Automata

2 **The Harpo Framework**

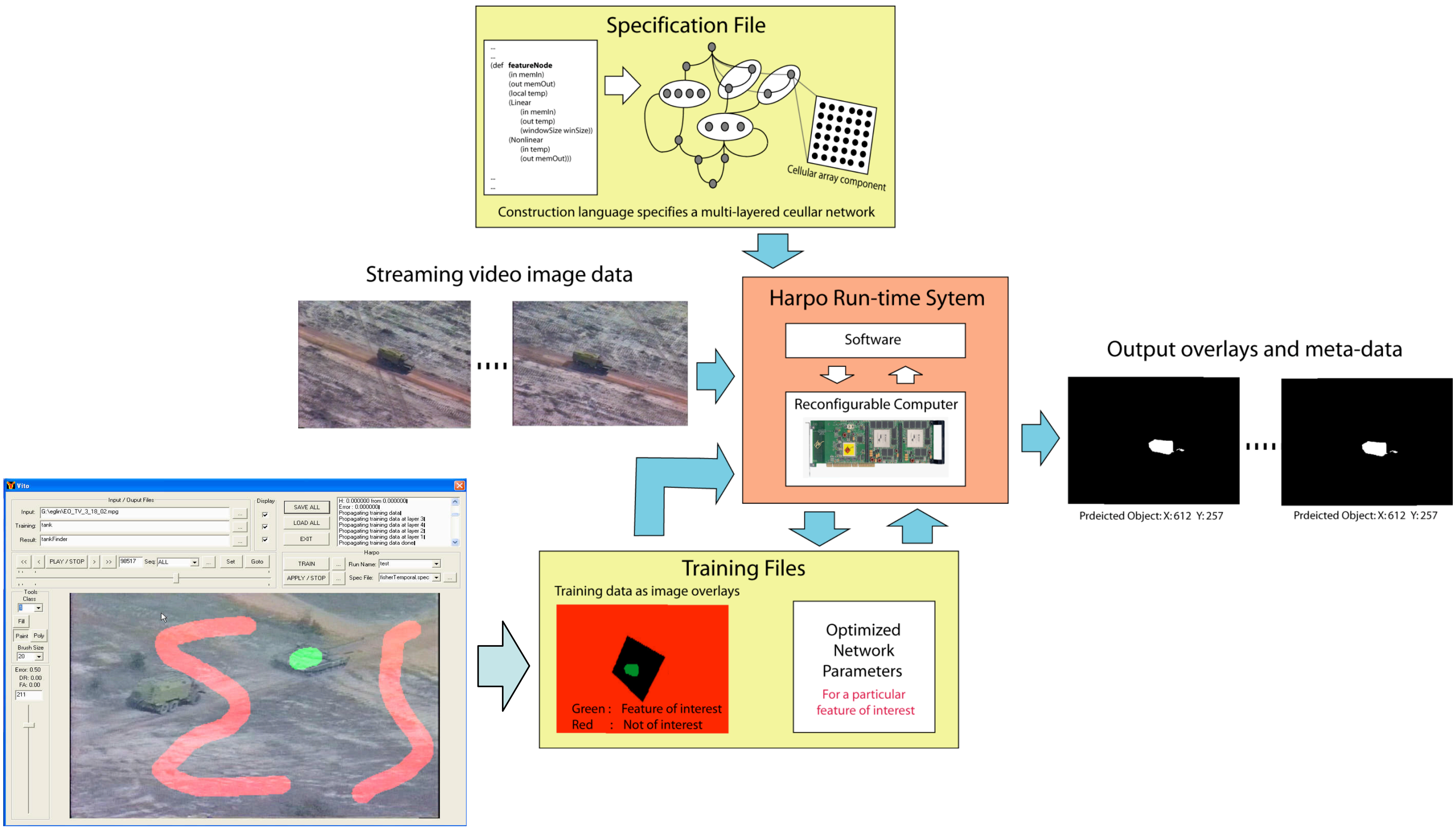
Overview, examples and performance assessment.

3 **Example Applications**

Plumes, people and movies..

Harpo System Overview

Flexibility: We can execute multiple cellular arrays at multiple scales in parallel.

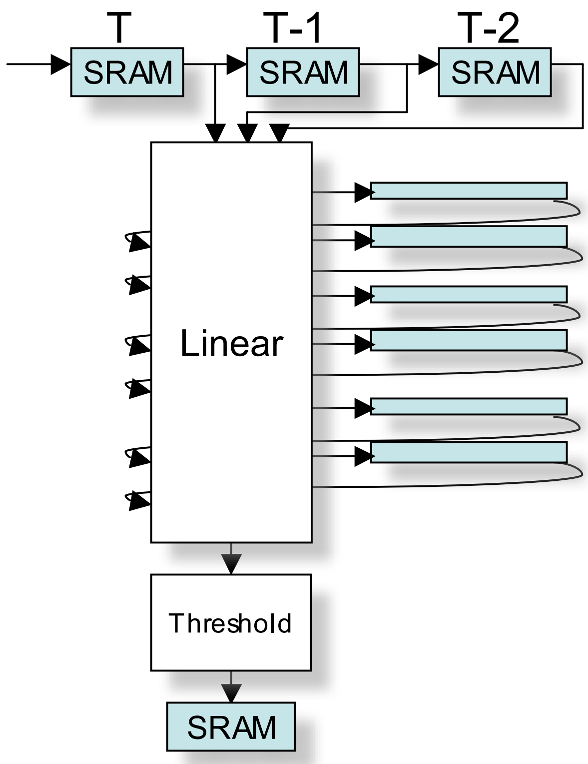


Programmability: Cellular algorithm design based on machine learning.

Flexibility: Cellular Algorithm Specification

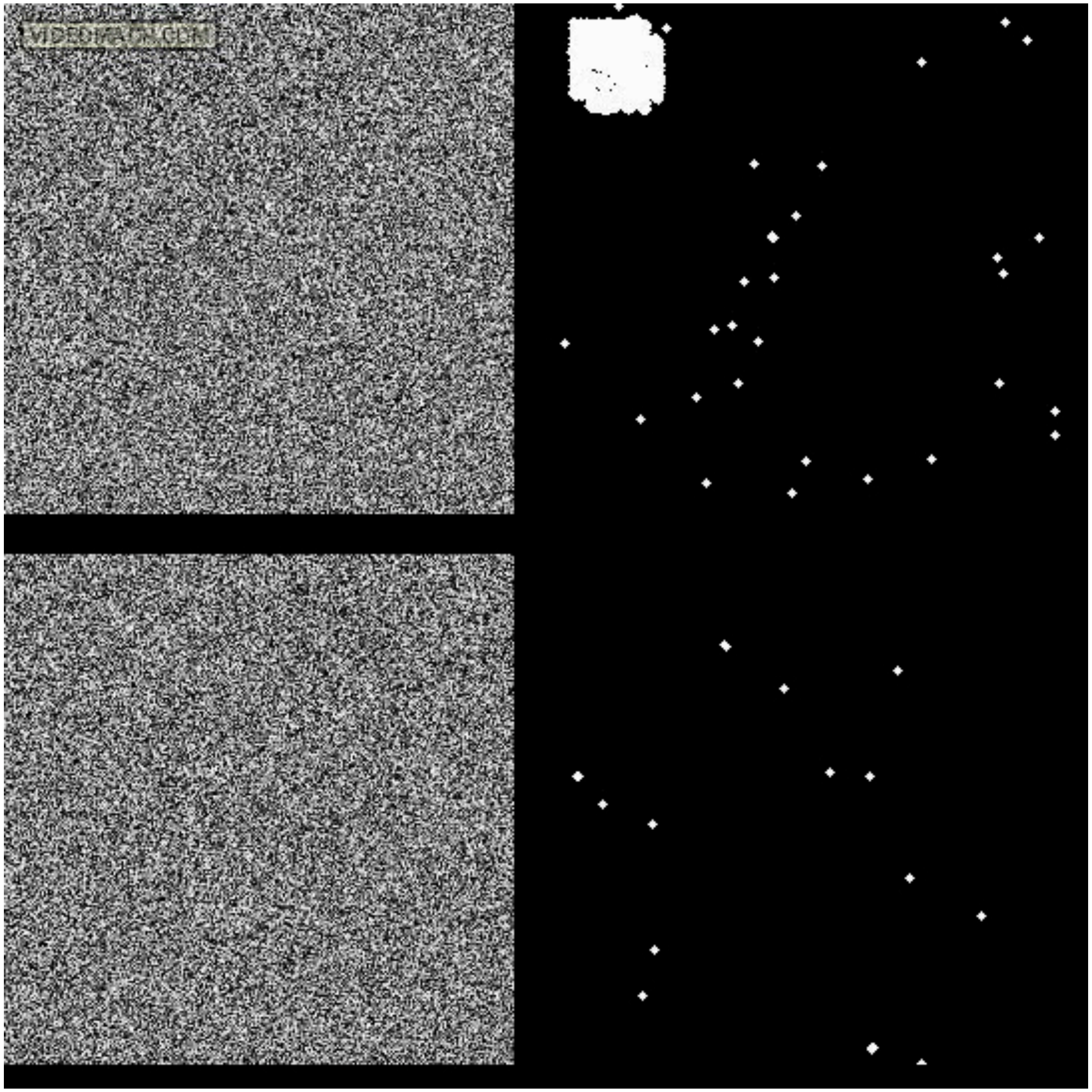
Scale language example

```
(def delay-Line (in memIn) (out memOut)
 (param taps)
 (set (index memOut 1) memIn)
 (replicate (i 2 taps)
      (Buffer
           (in (index memOut i-1)
           (out (index memOut i))))

(local memIn)
(local memOut)
(local lineOut)
(local (temp 1 3))

(delay-Line (in memIn) (out temp) (taps 3))
( Linear ( in temp ) ( out lineOut ) ( windowSize 5 ))
( Threshold ( in linOut ) ( out memOut ) ( initType abs))
```



Sequence used in training

Output from system



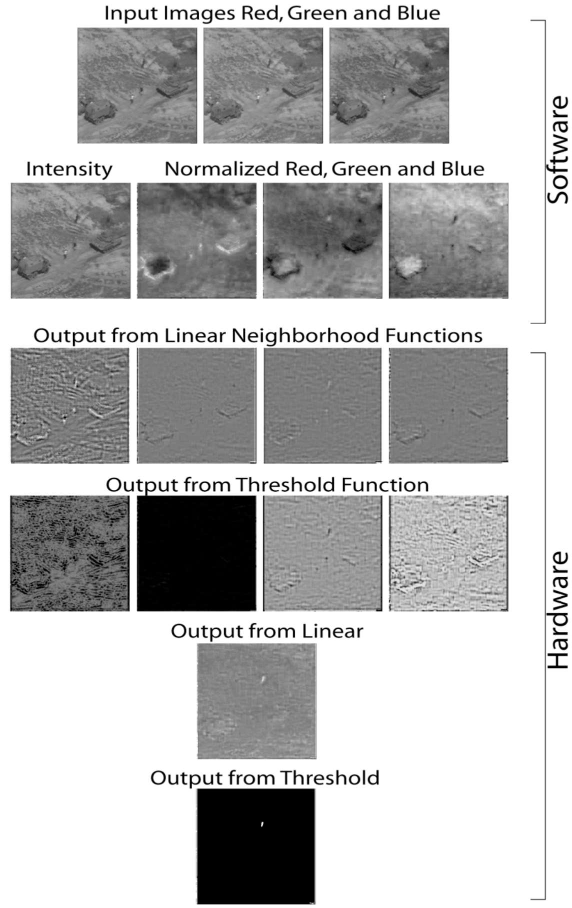
Test Sequence

Output from system

Hardware / Software Co-design

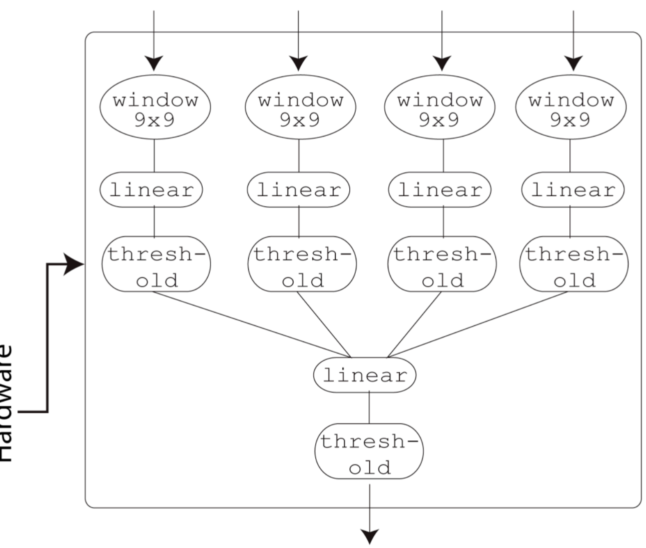
```
(def feature-Node  (in memIn) (out memOut)
    (local temp)
    (Linear
        (in memIn)
        (out temp)
        (paramType    gabor)
        (initType random))
    (Threshold
        (in temp)
        (out memOut)
        (initType random)))

(def multi-Layer-Net (in memIn) (out memOut)
    (param numFeatures)
    (local (temp 1 numFeatures))
    (replicate (i 1 numFeatures)
        (feature-Node
            (in memIn)
            (out (index temp i))
            (windowSize 9)))
    (feature-Node
        (in temp)
        (out memOut)
        (trainFlag true)))

(def pre-Process (in memIn) (out memOut)
    (Combine (in memIn)
             (out (index memOut 1))
             (funcType squared))
    (Normalize (in memIn)
               (out (index memOut 2))
               (funcType squared)))
;;;;;;;;;;;;;;;;;;;;;
;; Top level
;;;;;;;;;;;;;;;;;;;;;
(local memIn)
(local temp)
(local memOut)
(pre-Process
    (in memIn)
    (out temp))
(Evolve
    (Hardware
        ( multi-Layer-Net
            (in temp)
            (out memOut)
            (numFeatures 4))))
```

Scale language example



(a)

Output image from each layer
704 pixels by 480 pixels



(b)

Hardware Sub-Network

Corresponding Network

Resource Usage and Speed-up

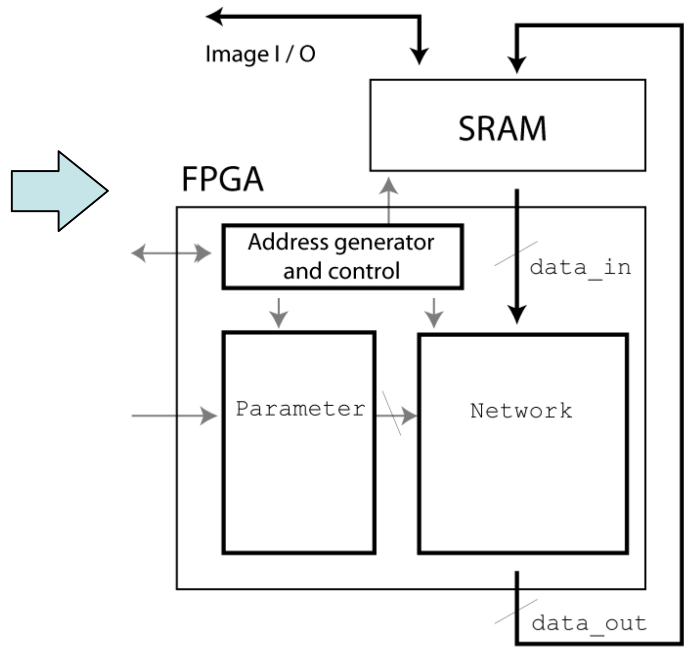
Hardware Resource Usage

Hardware API

	Resources	% Usage
On-chip RAM	4 / 444 Blocks	1
Logic	3187 / 44096 Slices	7

Application 1

	Number of Resources	% Utilization
Observed		
MULT18x18s	328	73
RAMB16s	12	3
Slices	12988	30



Performance Assessment

	P4 3.18GHz (s)	RCC (s) 100MHz	Speed-up (x)
Hardware Pipeline	2.969	0.004	742
Communication	-	0.010	
Data Preparation	-	0.015	
Sub-Total	2.969	0.029	102
Software layers	0.313	0.313	
Complete Application	3.282	0.342	9.6

Multi-scale Cellular Image Processing

- For many problems local information is not enough (e.g. object recognition).

- Hierarchy is used to incrementally introduce more global information.

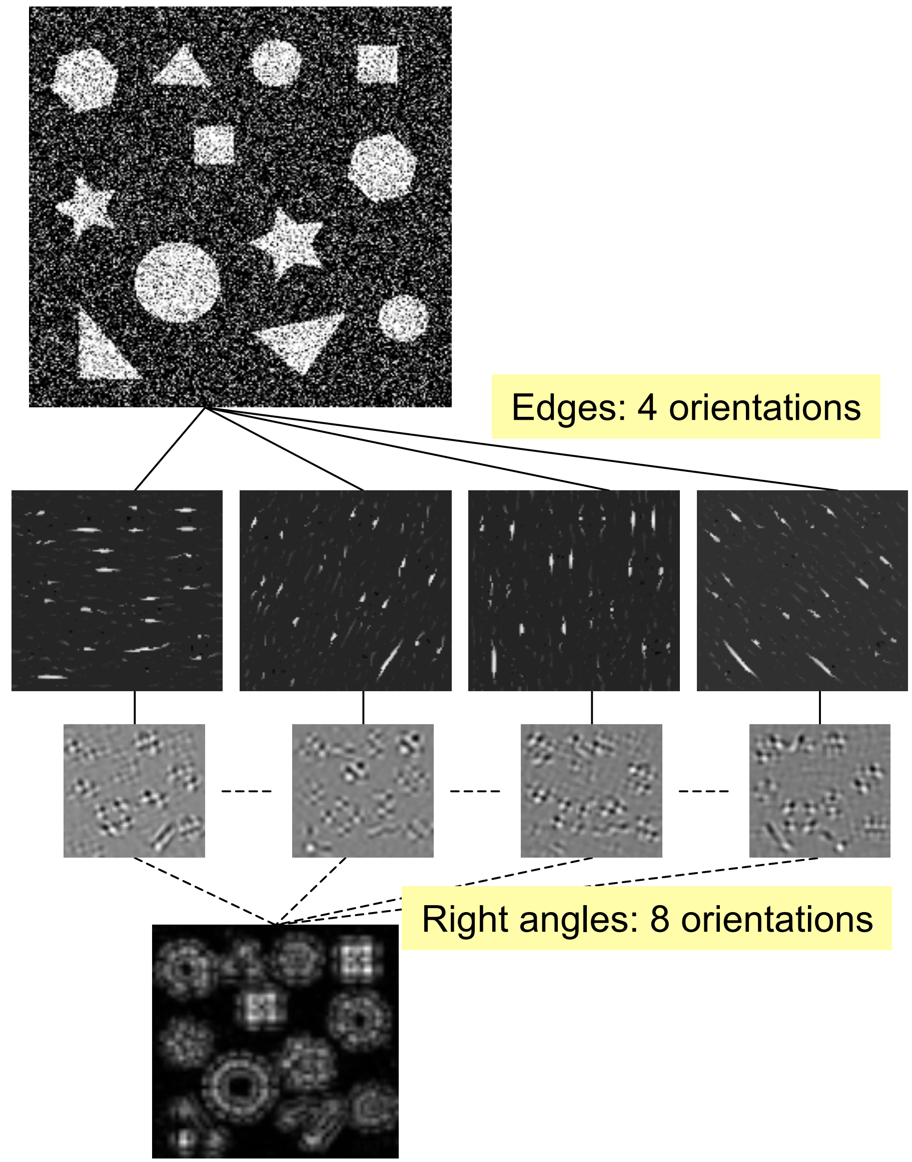
- Multi-scale processing in parallel is challenging since the data volume changes.

- Most other architectures process each scale in a different pass.

- Device capacity is reaching the point where the entire multi-scale algorithms can be implemented in parallel.

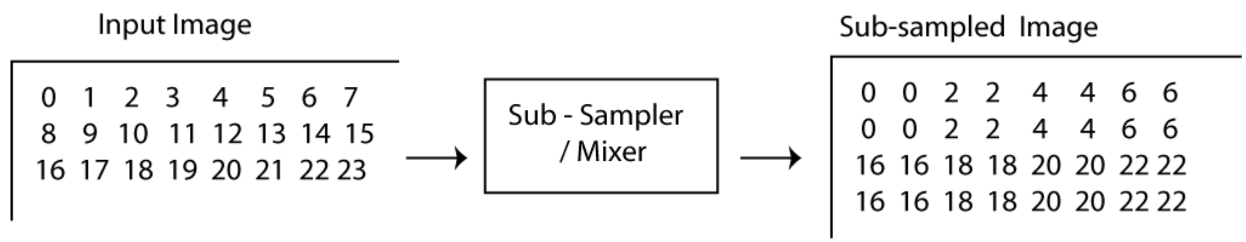
- Many multi-scale solutions of interest have strong local dependencies between layers at different scales. Therefore can be difficult to exploit the parallelism within a single scale.

- Cellular image processing is very regular.

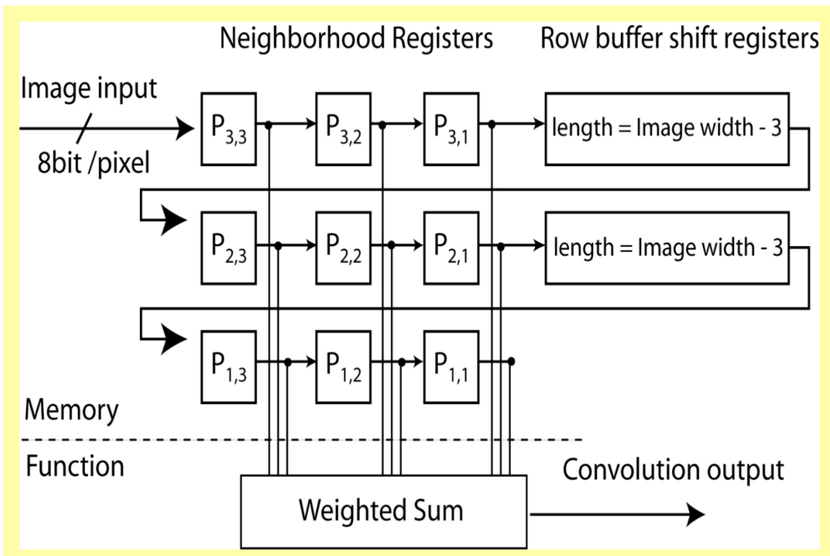


A multi-scale application: finding corners.

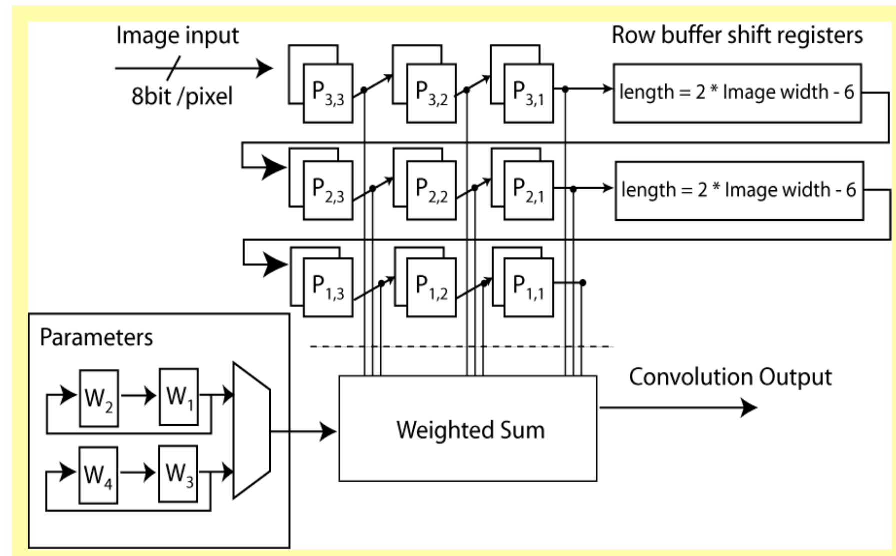
Pipelined Multi-scale Processing

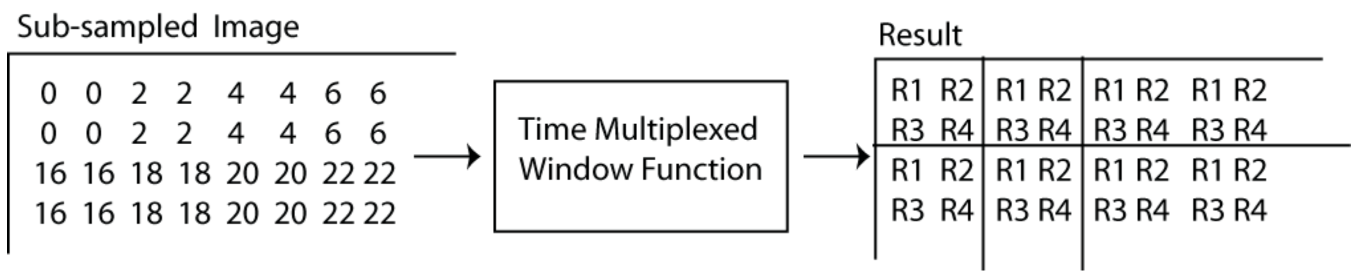


Typical neighborhood function

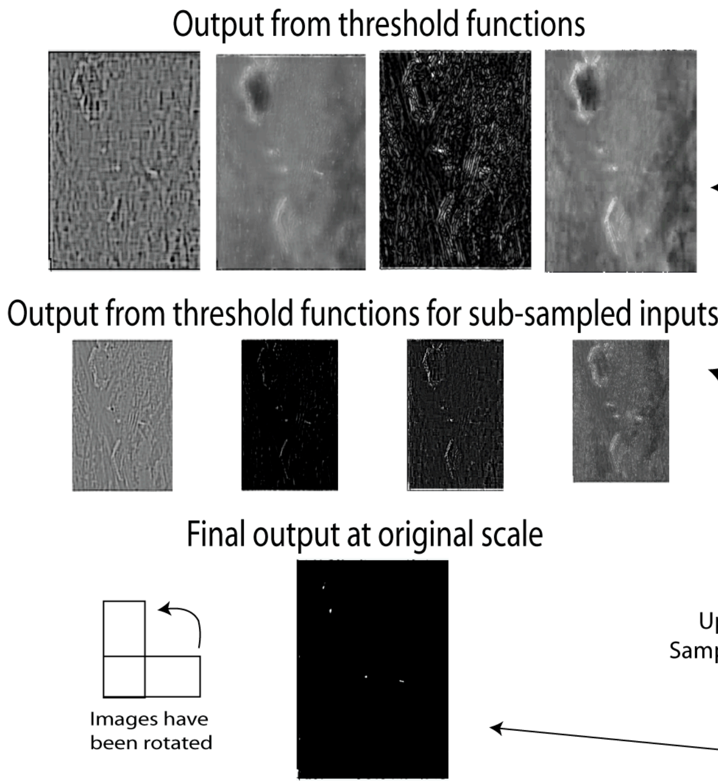


Generalized neighborhood function



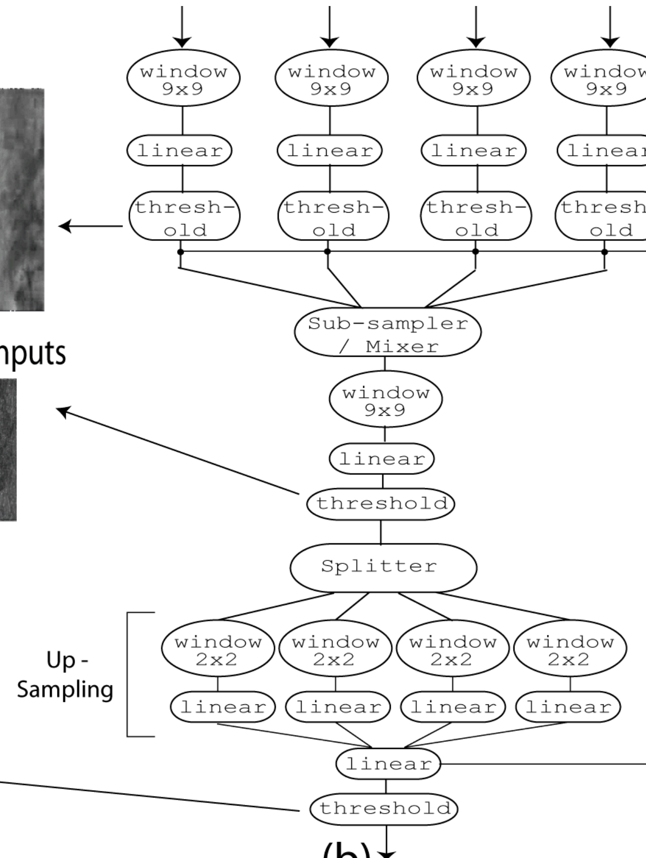


Multi-scale Example



(a)

Output images from hardware sub-network



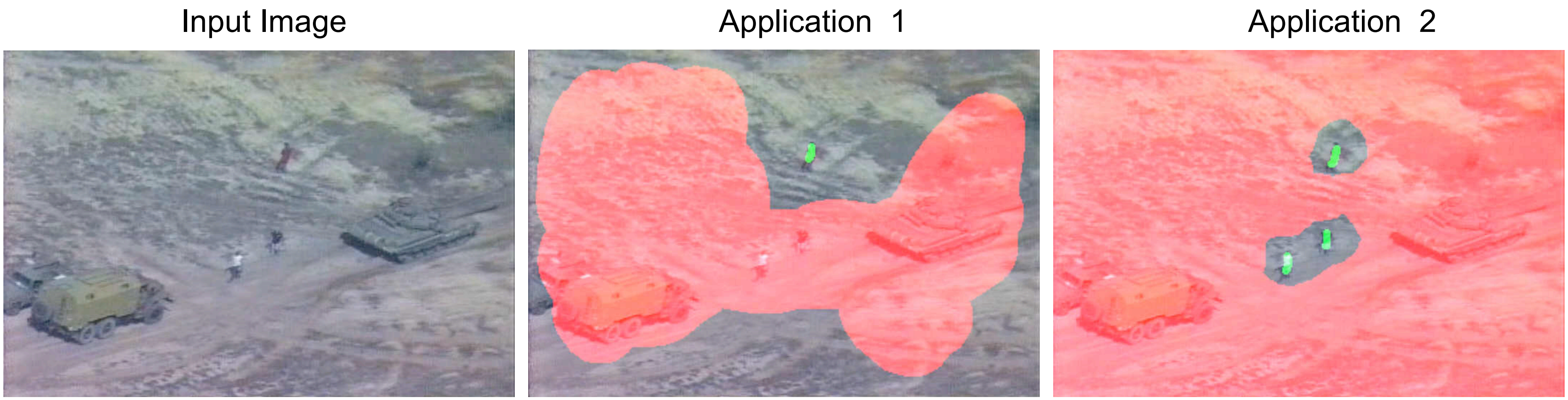
(b)

Hardware Sub-Network

Resource	% Util
Mults	93
Ram	4
Slices	65

	P4 3.18GHz (s)	RCC (s) 66MHz	Speed-up (x)
Hardware Pipeline	3.860	0.006	643
Communication	-	0.010	
Data Preparation	-	0.015	
Sub-Total	3.860	0.031	124
Software layers	0.313	0.313	
Complete Application	4.173	0.344	12.2

Programmability: Co-design and Learning Performance



Hardware and software can be used in a number of different ways during learning:

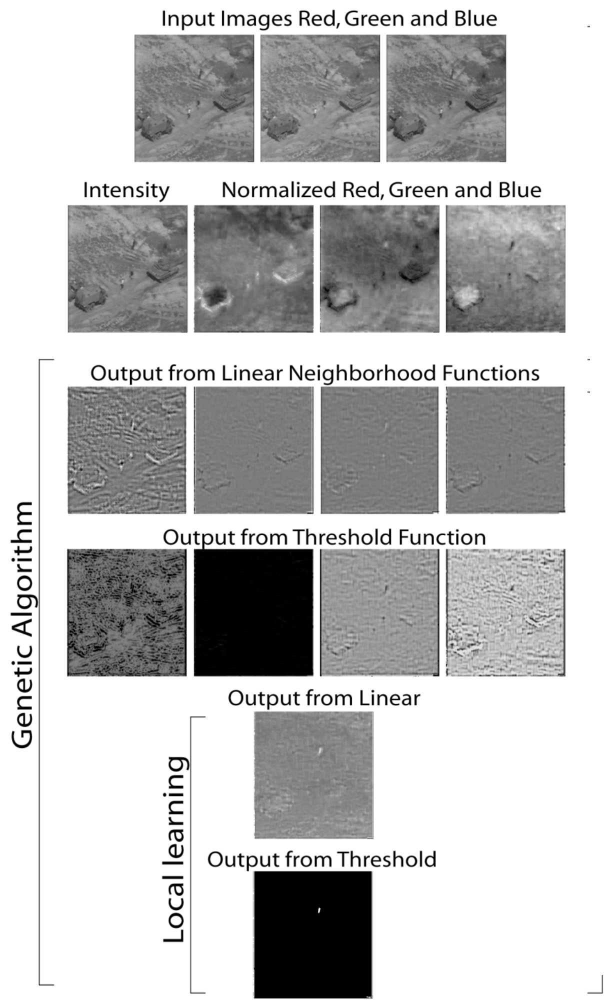
- Genetic algorithms (GA) are a global optimization technique. The *sample and test* approach is ideal for hardware/software since we only read back error for each test.

- A linear discriminant can be found with an efficient local optimization method but data must get back to the micro-processor to invert a matrix.

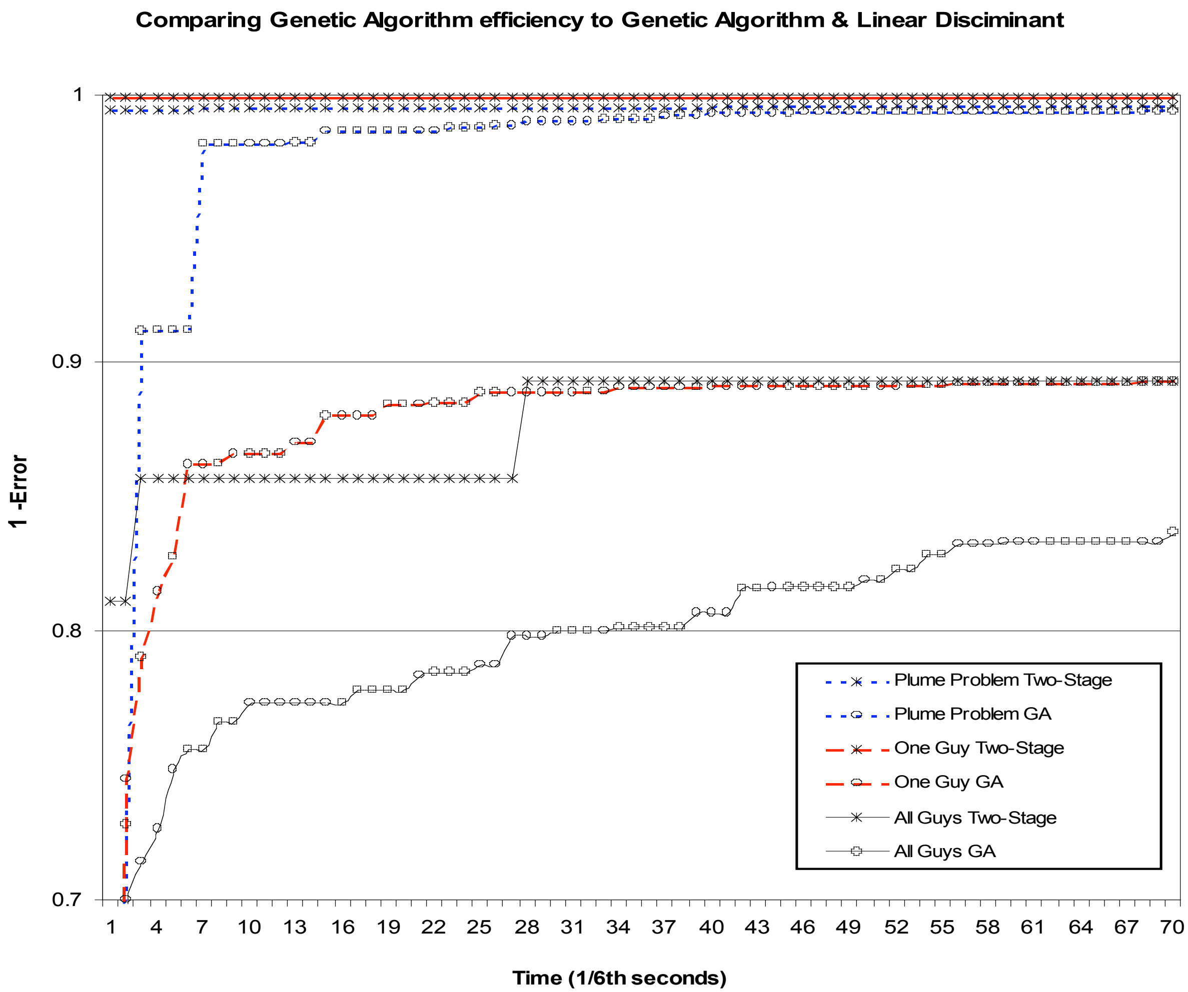
Pipeline Evaluation Time:	0.0026 seconds
Communication Time:	0.004 seconds
Discriminant in software:	0.15 seconds
Optimal threshold in software:	0.05 seconds

- With the GA we can execute **79x** more evaluations than with the 2-stage approach.

Hardware / software Co-design and Learning Performance



(a)



(b)

3 Example Applications

1 **Introduction to Cellular Algorithms**

Lattice Gas Automata

2 **The Harpo Framework**

Overview, examples and performance assessment.

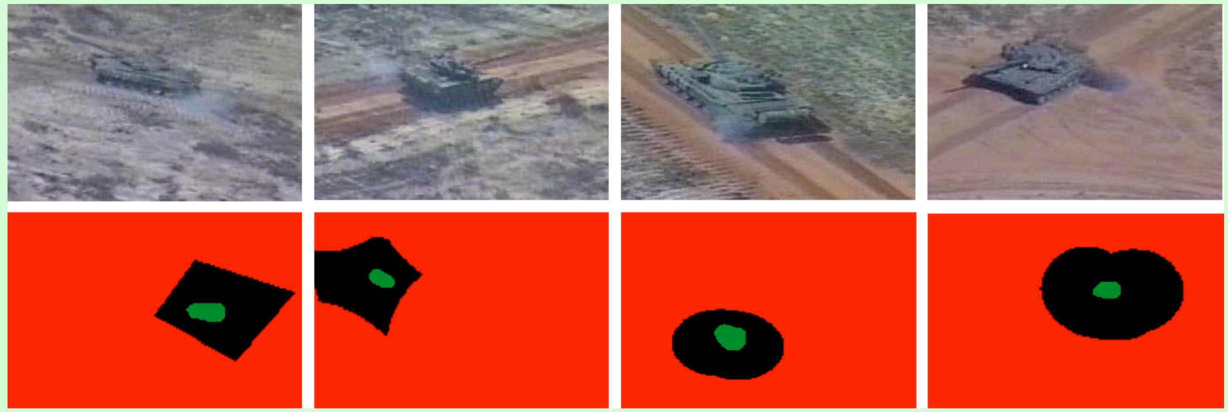
3 **Example Applications**

Plumes, people and movies..

Segmenting Exhaust Plumes

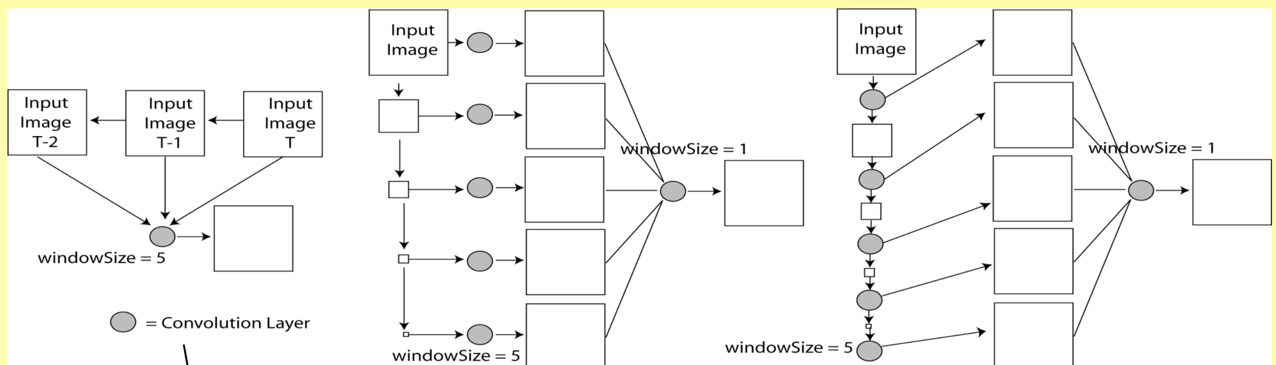
Example training frames

- 8 sequences marked
- From period of about 15min
- 4 frames used for training
- 4 frames used for testing



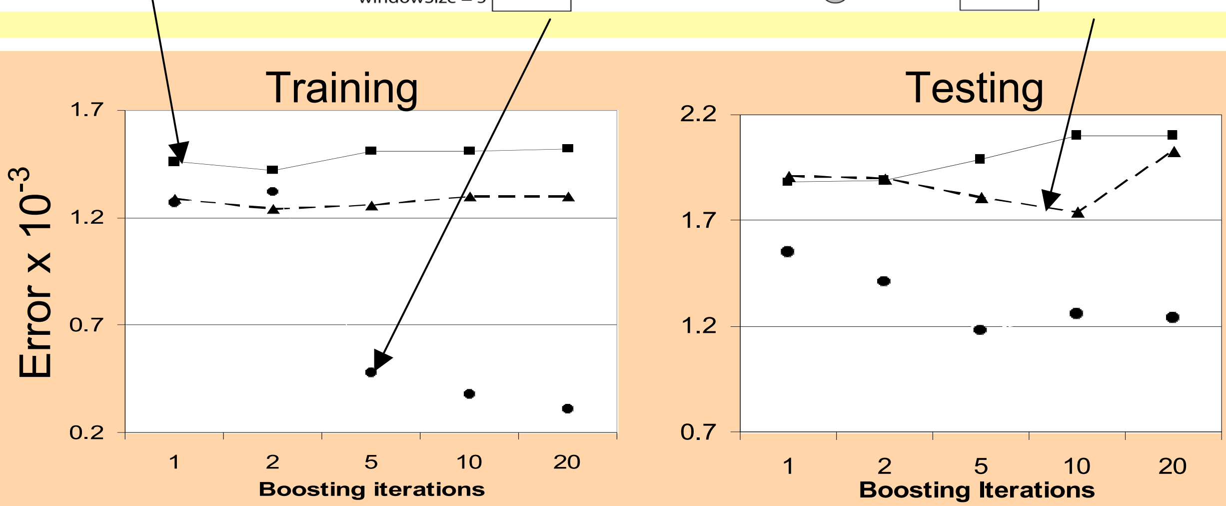
Architectures considered

- Temporal
- Multi-Scale
- Multi-Stage



Estimated performance

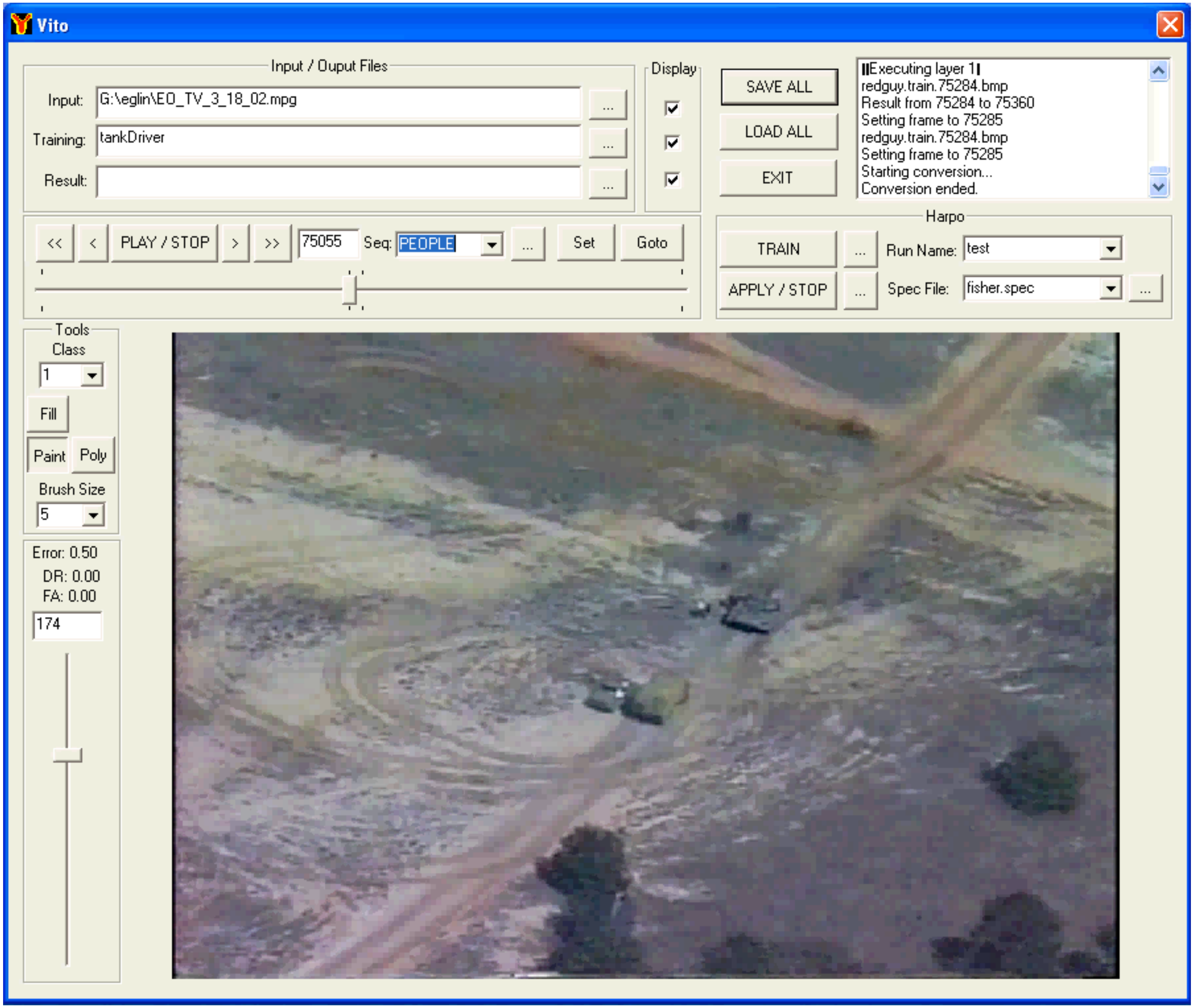
- Network optimized via greedy learning
- *Boosting* used to build ensembles of varying complexity.



Segmenting Exhaust Plumes

- Eglin Skymaster UAV test platform
- 3-color S-VHS (30 frames/sec)

Point-and-click feature extraction



Point-and-click feature extraction

A framework to meet increasingly complex recognition problems.


Future Work: Report key video sequences where the object of interest interacts with more general object classes and / or behaves in particular ways e.g.

1. Extract a family of objects and features in parallel.
2. More complex algorithms to detect interesting behavior and events.

Summary

- Cellular Algorithms map extremely well to RCC

- Some applications are naturally cellular, but research is often required.

- Harpo abstracts memory management and provides a path to automatic generation of extremely efficient hardware.

- Pipelined multi-scale processing is an excellent match to RCC since datapaths can be customized for each application to optimize resource usage.

- Harpo currently targets a single co-processor node.

- Future work will investigate cluster configurations.

Finding all the people