Microsoft Research

Jason Oberg, Ken Eguro, Ray Bittner, and Alessandro Forin

RANDOM DECISION TREE BODY PART RECOGNITION USING FPGAS

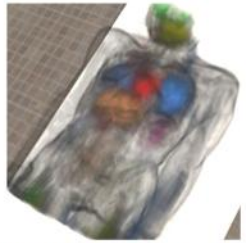
Randomized Decision Trees (RDT)

- Vision
  - Keypoint recognition
  - Object segmentation
  - **Human pose estimation**
  - Organ detection
- Speech recognition
- Web search
- Data mining

[Lepetit *et al.*, 06]
keypoint recognition

[Shotton *et al.*, 08]
object segmentation

[Rogez *et al.*, 08]
pose estimation



[Criminisi *et al.*, 09]
organ detection

Kinect Object Recognition

Platform	*FPS*
Atom 230 Stress	< 2
Cortex-A15 Stress	3

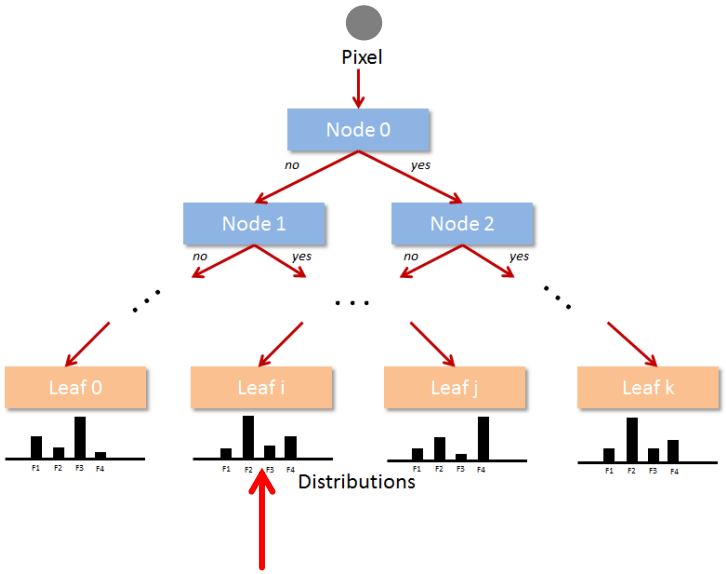
- Computationally intensive
  - All processing done on Xbox or PC
  - Requires "real" processor or GPU for 30 FPS
- What about mobile/embedded applications?
  - Atom and ARMs can't keep up

FPGAs for Kinect Vision

- Direct HW implementation needed
  - Power
  - Performance
  - Cost
- FPGAs are a good platform
  - Flexibility is important
  - New vision algorithms
  - Full system integration

How Kinect Uses RDTs

- Depth pixel enters at root
- Evaluate function at node
  - Look at another pixel
  - Subtract, compare with threshold
  - Go to left or right child
- Leaves contain classification probabilities
- Repeat for all pixels, all trees



90% chance this pixel is part of left hand, 10% chance it is part of torso

Memory Access is Important!

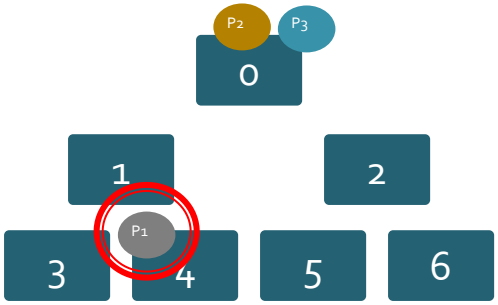
- Each node decision requires data-dependent access to database
  - Tree size is 25 MB, must use external DDR
- Processing is highly parallel, but computation/communication very small
- Speed of external memory access is bottleneck
  - What can we do to minimize & optimize DDR access?

Observations

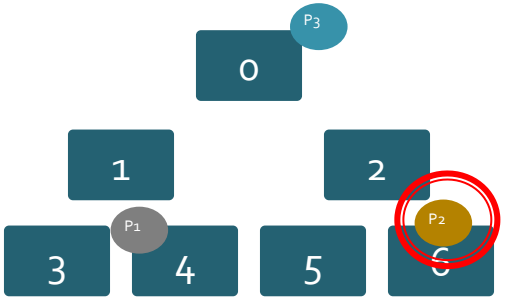
- DDR is organized into "pages"
  - Back-to-back accesses to same page → 2 cycles
  - Back-to-back accesses to different pages → 10 cycles
- All pixel processing is fully parallel
  - We can change the processing order without affecting the results

Depth-First Tree Traversal

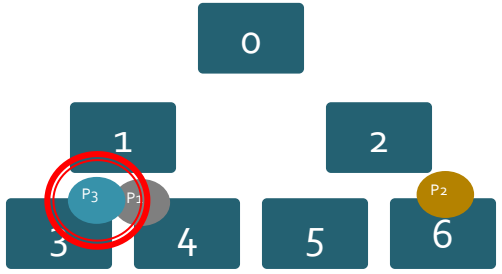
- Process each pixel from top to bottom of tree
  - Repeat for every pixel and tree
  - We are guaranteed to cross DDR pages!



Process Pixel 1



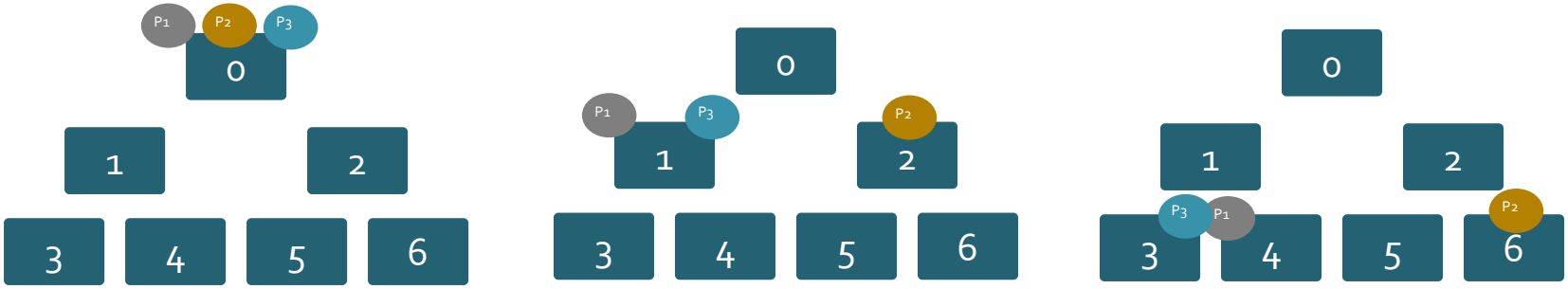
Process Pixel 2



Process Pixel 3

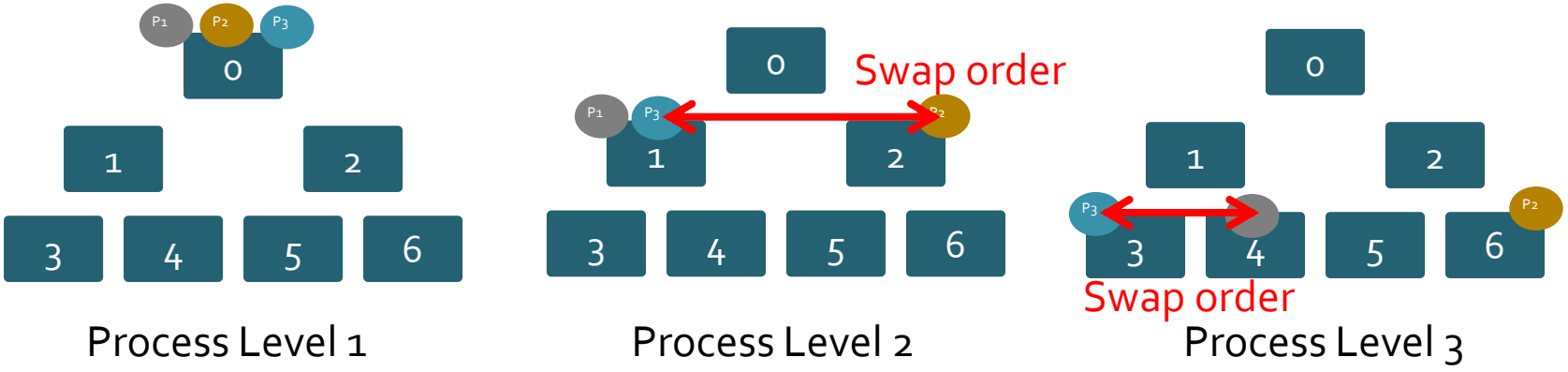
Breadth-First Tree Traversal

- Batch-process pixels through tree level before continuing to the next
  - Tends to reduce repeated node accesses
  - Tends to increase same-page accesses



Sorted Breadth-First Tree Traversal

- Process all pixels through tree level before continuing, but sort pixels by child node
  - Guarantees any node is only requested once
  - Guarantees any page is only "opened" once



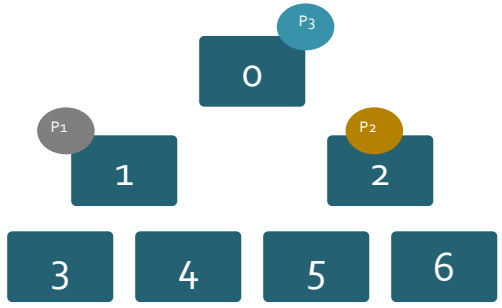
Effect of Ordering/Sorting

- Test image 160x120 pixels
  - Three 20-level trees
  - 1.1M tree node accesses

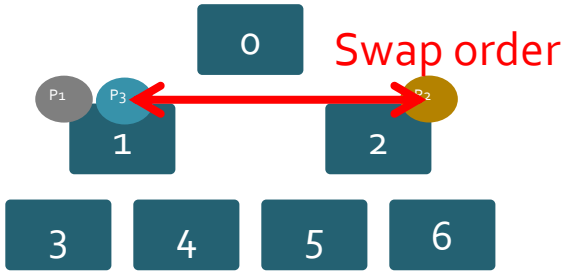
Processing Algorithm	*Cached Hits*	*Page Hits*	*Page Misses*	*Norm. BW*	% Max BW @ 30FPS
Depth First	124,314	394,086	633,600	1.000	1.070
Breadth First	892,202	173,111	86,687	0.170	0.180
Sorted Breadth First	**1,119,005**	**30,714**	**2,281**	**0.012**	**0.013**

Cost of Ordering/Sorting

- Depth-first
  - No temporary storage
- Breadth-first
  - Requires pointers
- Sorted breadth-first
  - Requires pointers
  - Requires computation for sorting



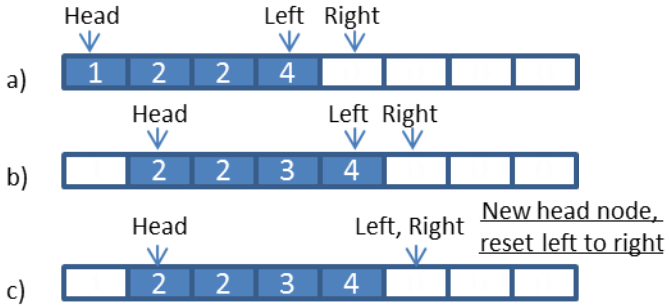
Breadth-first processing

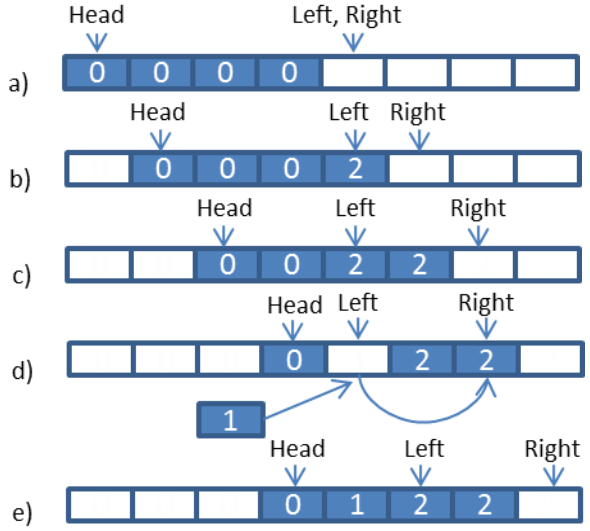


Sorted breadth-first process level 2

Efficiently Sorting Pixels in HW

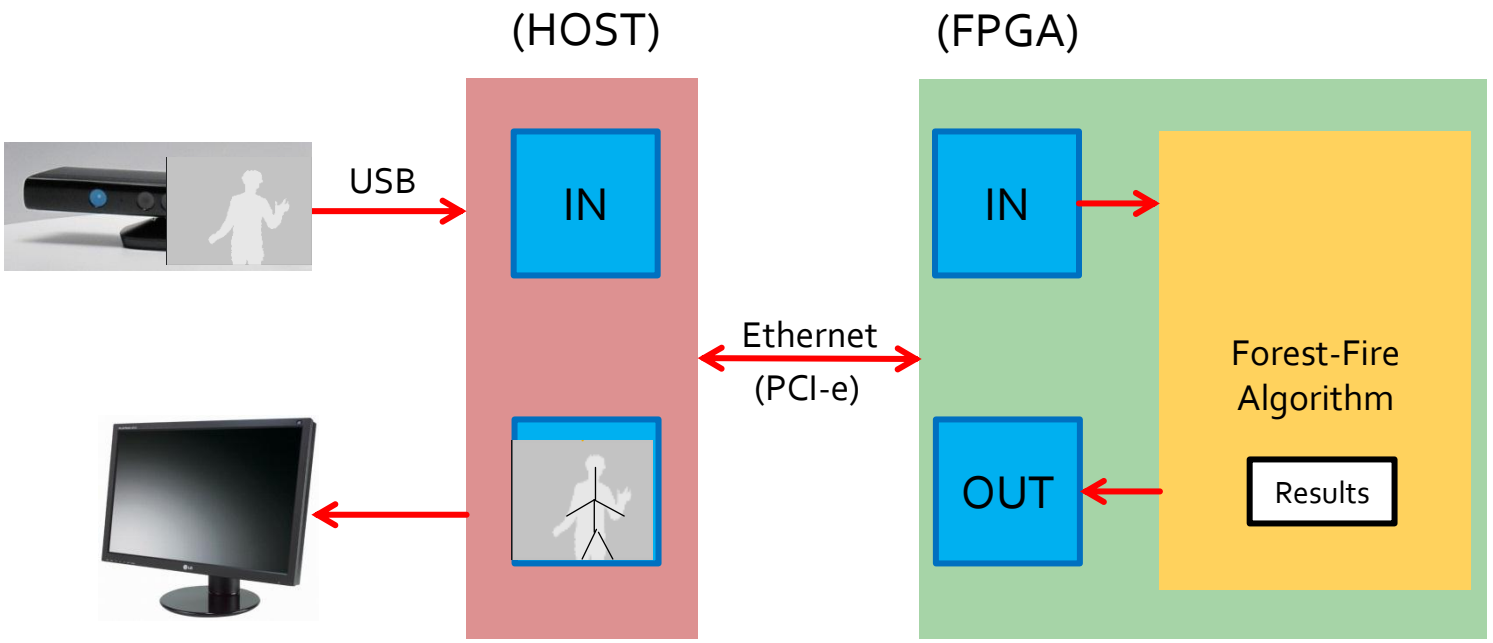
- Continuously sorted FIFO
  - Sort on insertion
  - Right children are pushed to 'Right'
  - Left children are inserted at 'Left'
    - Contents re-written to 'Right'
  - 'Left' reset to 'Right' once new # is popped by 'Head'





High-level Overview of Full System

1. Host PC captures **depth** frame from Kinect via USB
2. Host PC sends frame to FPGA via Ethernet
3. FPGA computes classification, sends back results
4. Host post-processes and displays



Resource Utilization

- Xilinx V6 LX240T
- Updated results since publication

	LUTs	*FF*	*BRAM*
Full System	**13,284 (8.8%)**	**16,144 (5.4%)**	**35 (8.4%)**
Forest Fire Core	*6,425 (4.3%)*	*7,552 (2.5%)*	*9 (2.1%)*
DDR3 Controller	*5,058 (3.4%)*	*7,496 (2.5%)*	*0 (0%)*
Eth → PC Interface	*1798 (1.2%)*	*1,092 (0.4%)*	*26 (6.3%)*

Performance

- Extra time can be used for building more of the application in hardware
  - any pre/post-processing done in software
  - higher-resolution camera

Algorithm	*Avg. Cycles @ 75Mhz*	*FPS*
BF Avg.	414,557 (1.0)	181
BFS Avg.	**349,492 (1.18x faster)**	**214**
BF Stress	1,714,525 (1.0)	44
BFS Stress	**1,349,706 (1.27x faster)**	**56**

Conclusions

- Random decision tree processing is used in many different applications
- Different platforms have different capabilities → different algorithmic considerations
- Low computation per communication doesn't mean it's bad for FPGAs & FPGA acceleration