Multiprocessor SoCs for Video Applications

Wayne Wolf, Tiehan Lv,
Burak Ozer, Jason Fritts

Outline

- Problem and methodology.
- Ozer, Lv, Wolf: smart camera system.
- Fritts, Wolf: Programmable VSPs.
- Ozer, Lv, Wolf: Optimizing the smart camera software.
- Wolf, Lv, Ozer: Architectures for smart cameras.

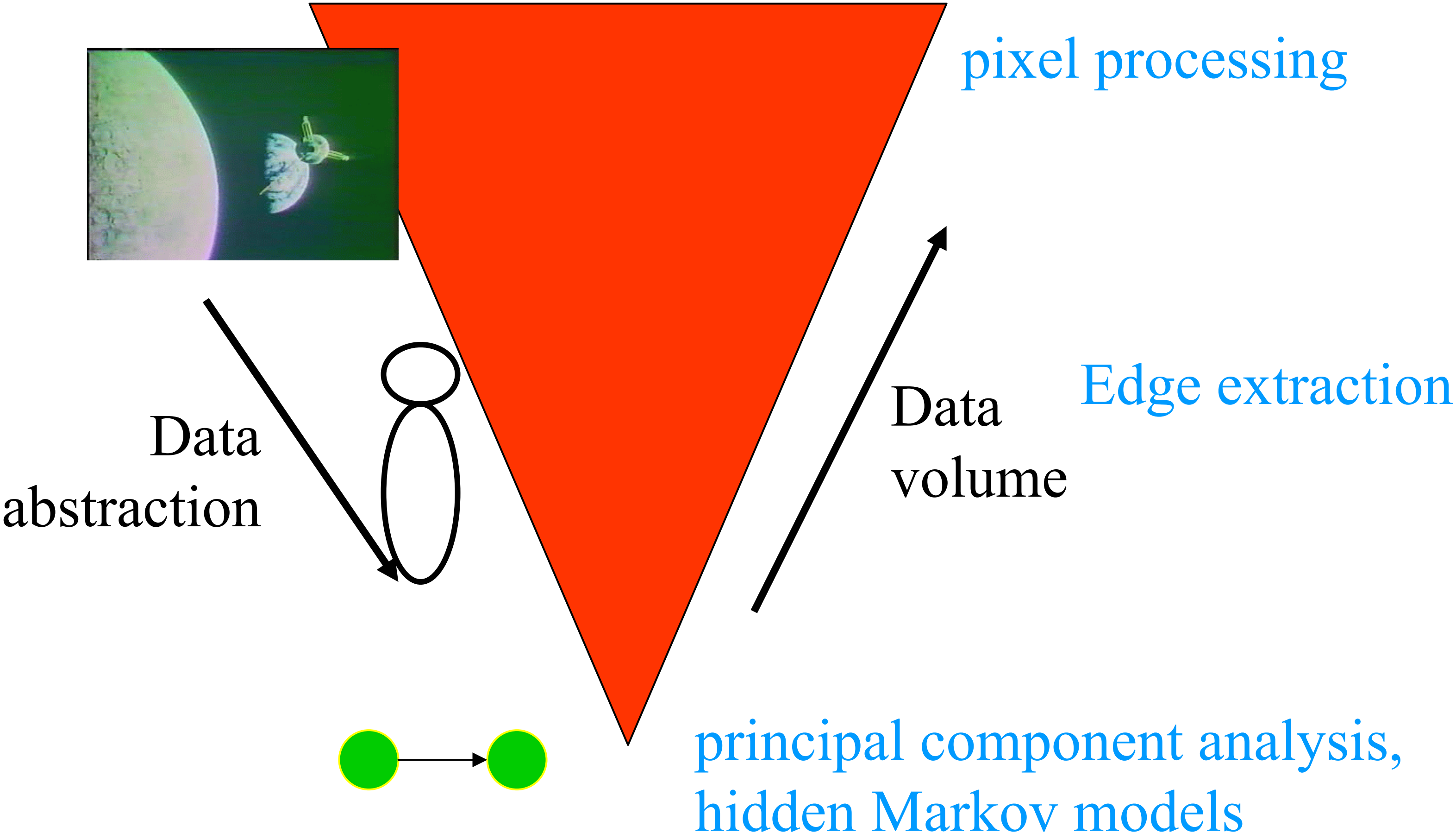
What is video processing?

- Initial steps operate on pixels, are dominated by data.
- Later steps operate on other types of data:
  - Smaller data volumes.
  - Wider variety of data types.
  - More variation in control flow, run time.

Multimedia requirements

- Complex algorithms:
  - multiple phases;
  - data and control.
- Today’s applications: compression.
- Tomorrow’s applications: analysis.

The multimedia processing funnel



pixel processing

Edge extraction

principal component analysis, hidden Markov models

Data abstraction

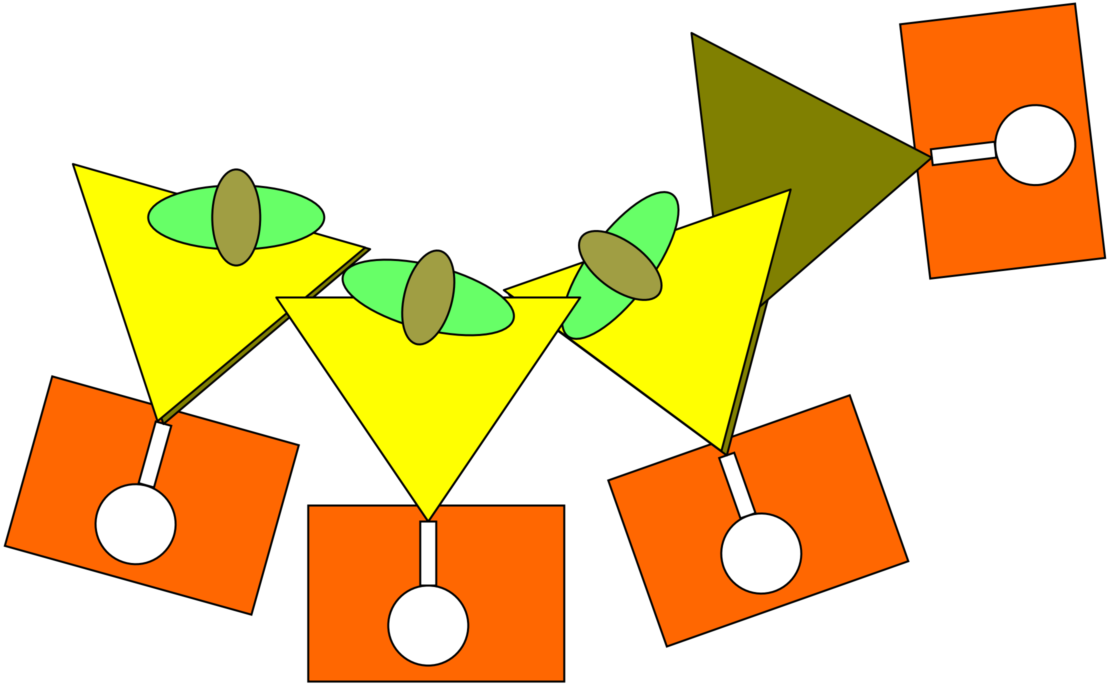
Data volume

Design methodology

- Successive refinement:
  - Matlab algorithms.
  - C on uniprocessor.
  - Custom heterogeneous multiprocessor.
- At each stage:
  - Measure performance.
  - Optimize where possible.
  - Identify optimizations at the next level of abstraction.

Smart cameras for smart rooms

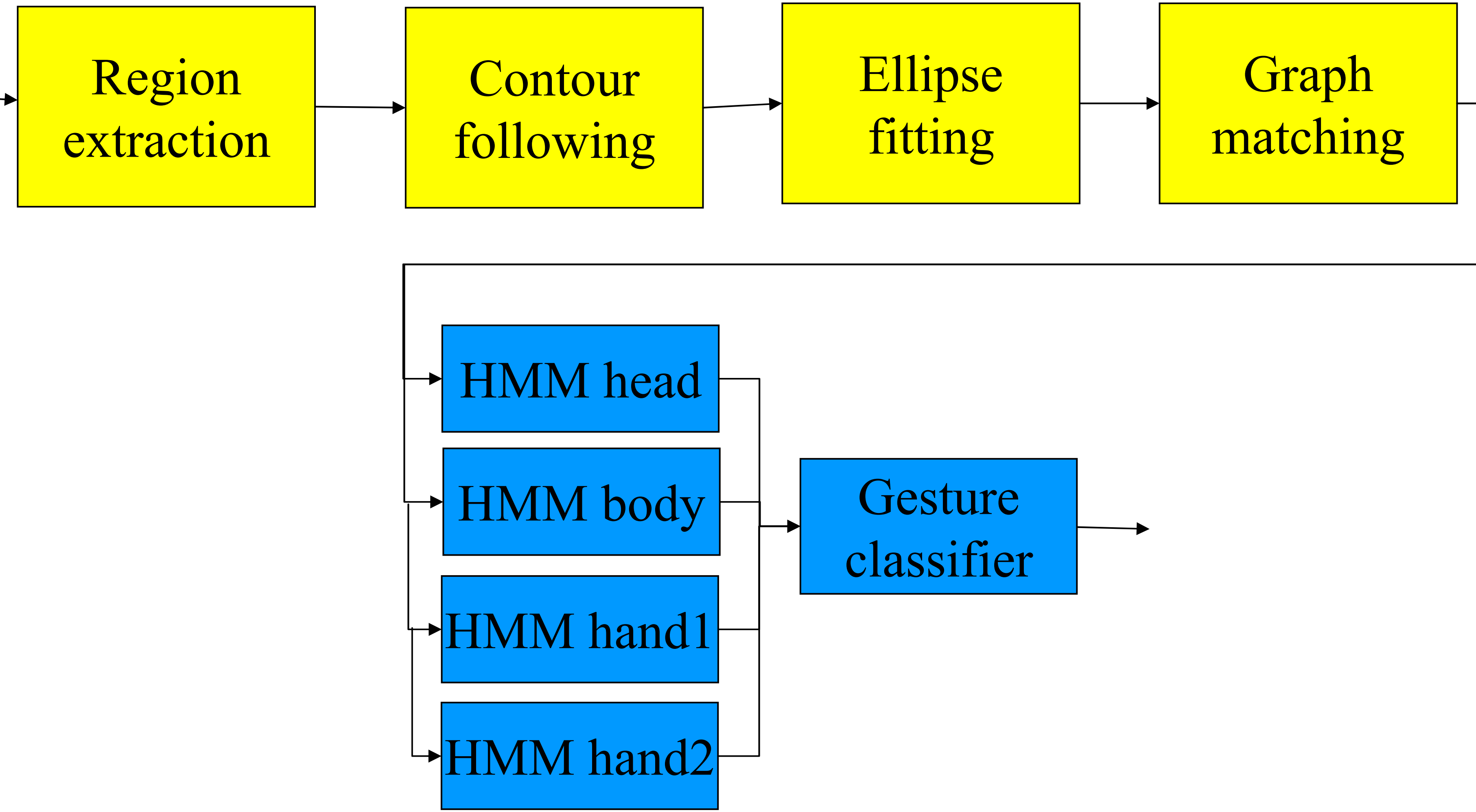
- Coordinated cameras track subject:



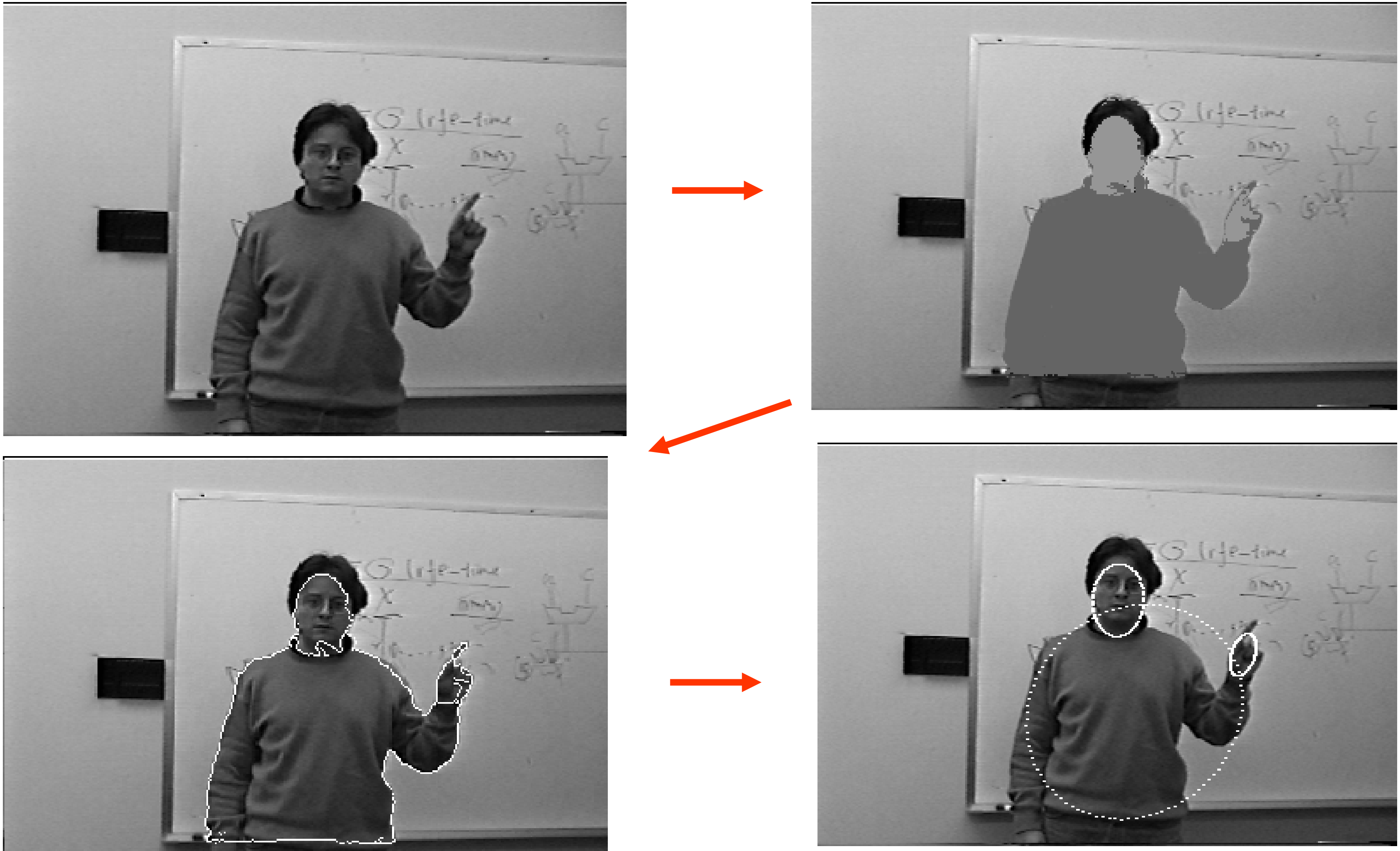
Questions

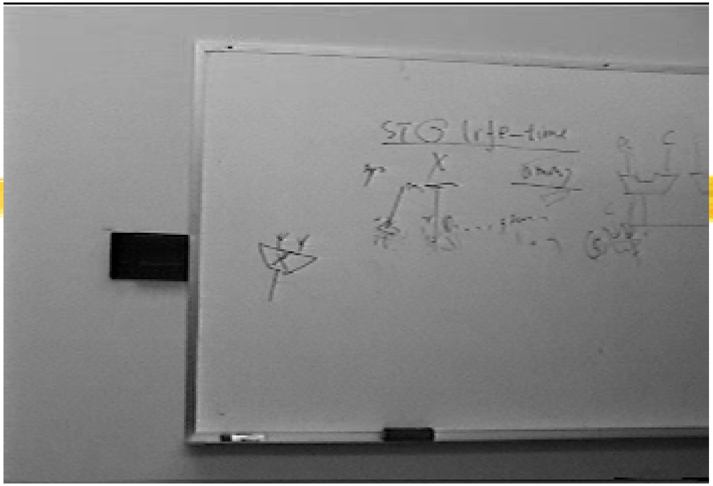
- Measurement:
  - What do we measure?
  - On what implementation do we measure it?
  - How accurate do our measurements have to be?
- Architecture:
  - What uniprocessor architecture is best?
  - Do we need a multiprocessor?
  - How do we balance programmability with other goals?

Ozer et al: human activity recognition algorithm

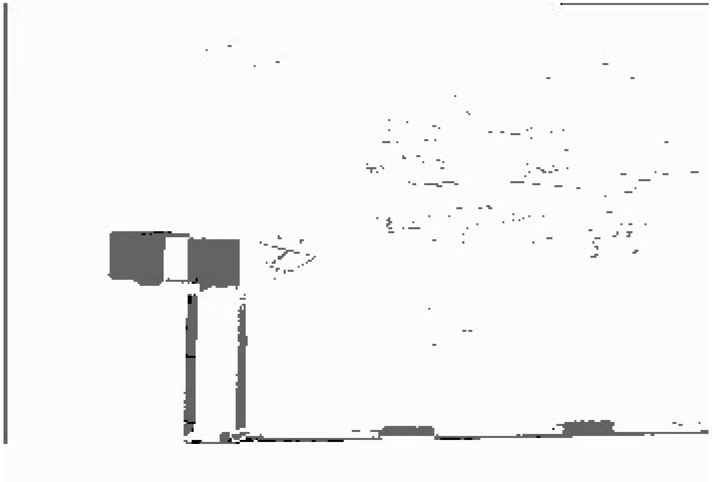


Real-time analysis





Original



Region finding

Ellipse fitting

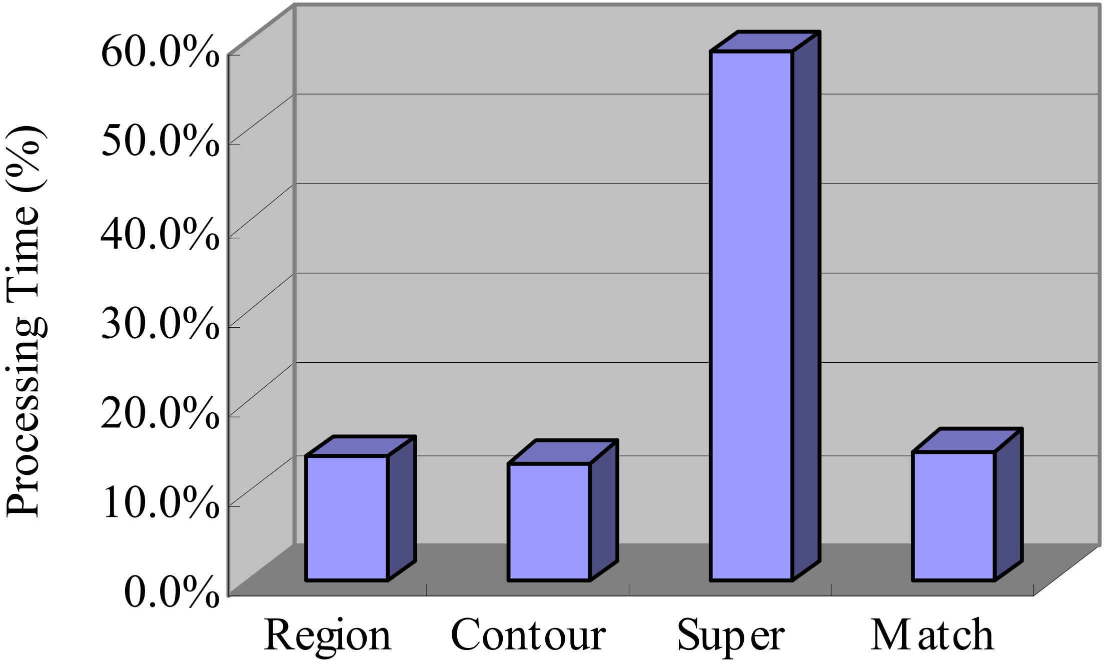
Tuning the smart camera software

- Initial C/Trimedia was direct translation from Matlab.
- Goals:
  - Increase frame rate.
  - Reduce latency.
  - Identify bottlenecks for next-generation architecture.

Real-time vs. just fast

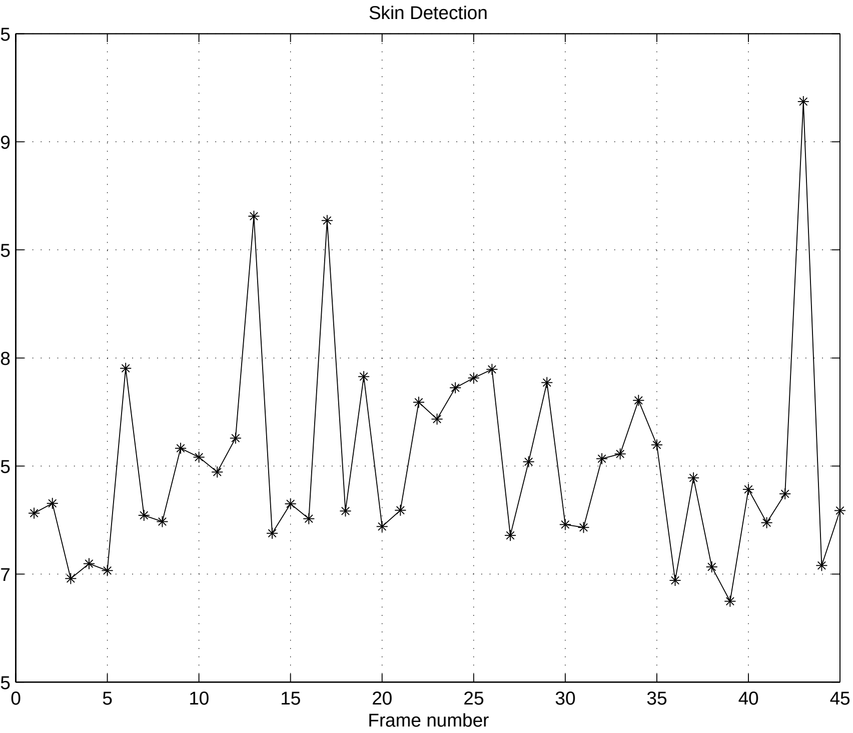
- Real-time computing adheres to constraints:
  - Must perform at a given rate.
  - To satisfy the rate, must minimize variations in processing time.

Stage times before optimization

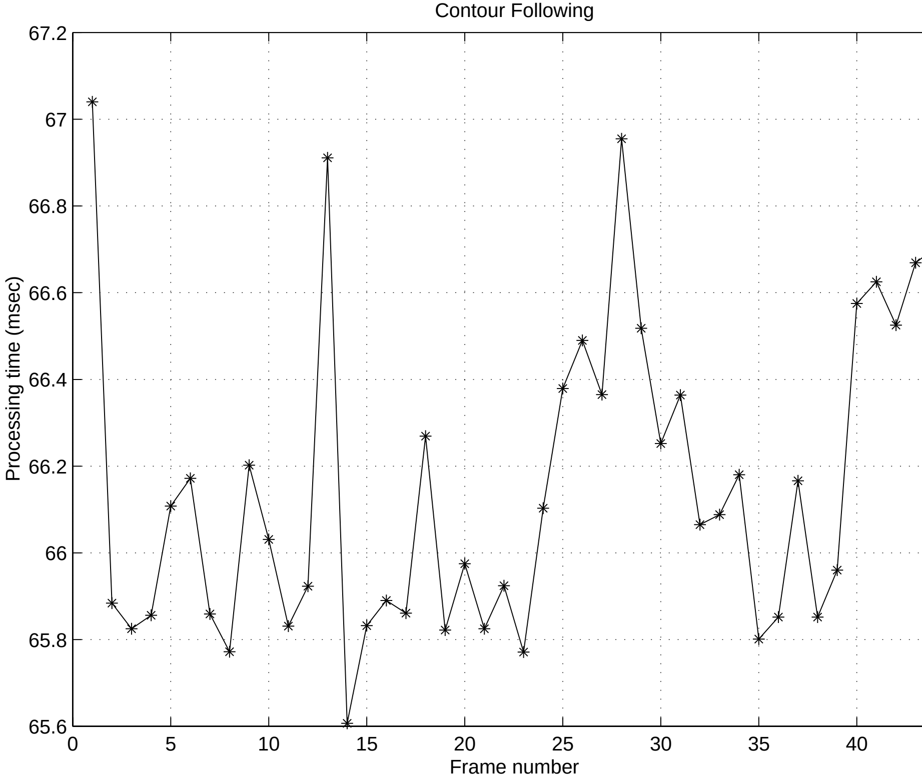


Smart camera CPU times

29.5 ms



67.2 ms

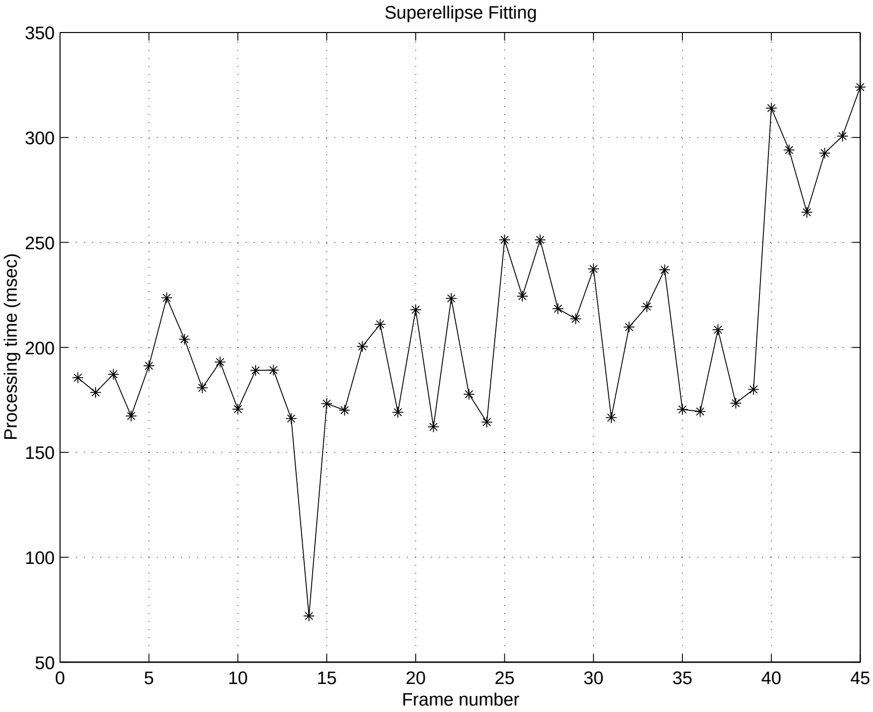


Skin detection

Contour detection

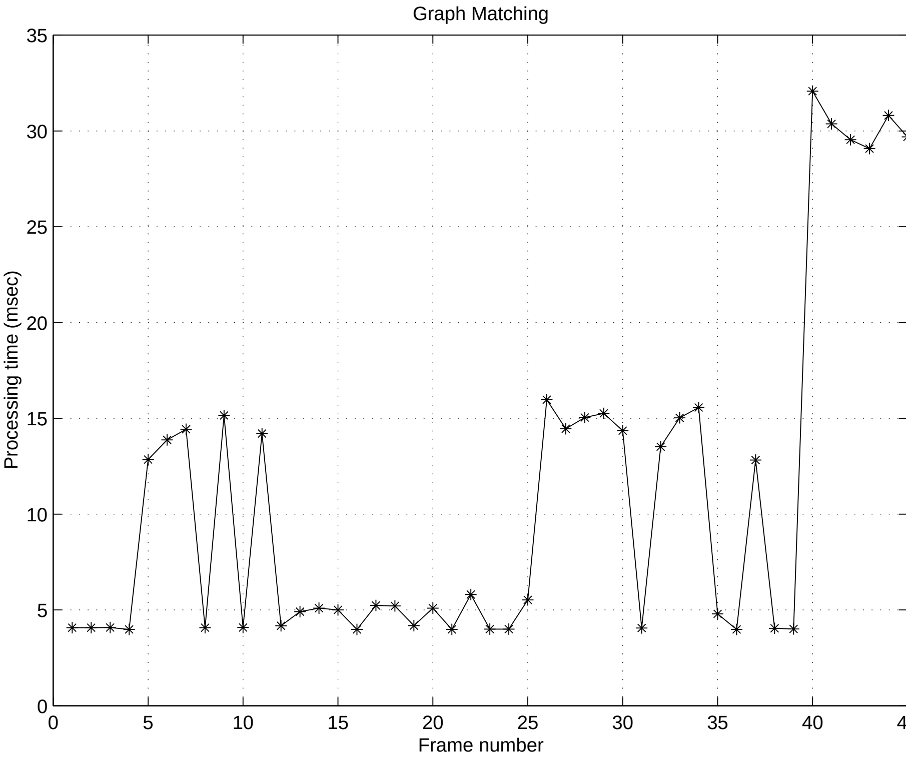
Smart camera CPU times, cont'd.

250 ms



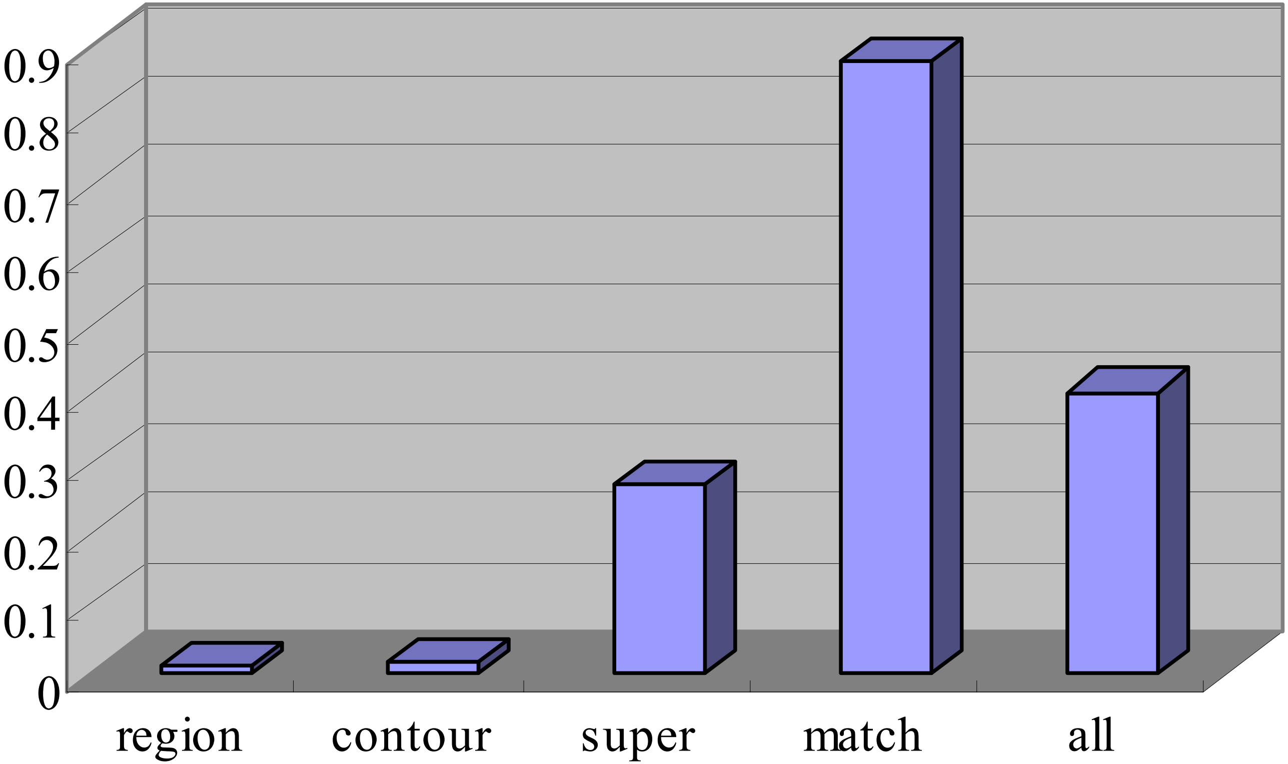
Superellipse fitting

35 ms



Graph matching

Normalized standard deviation of stage times



Optimizations

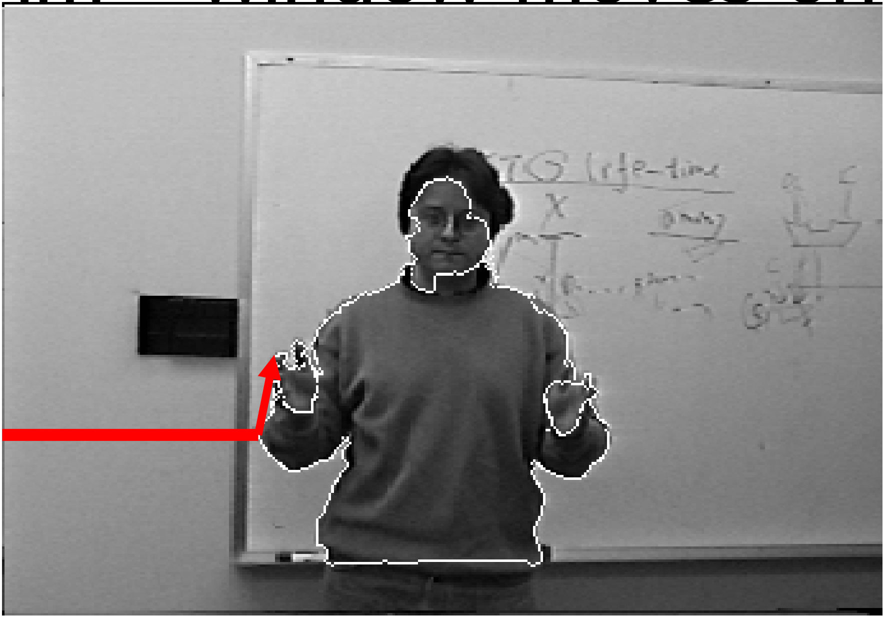
- Change the algorithm.
- Change the program structure.
- Change the instructions.

Algorithmic changes

- Superellipses were expensive to fit and overkill.
  - Replaced with ellipse fitting.
- Improved adjacency algorithm.

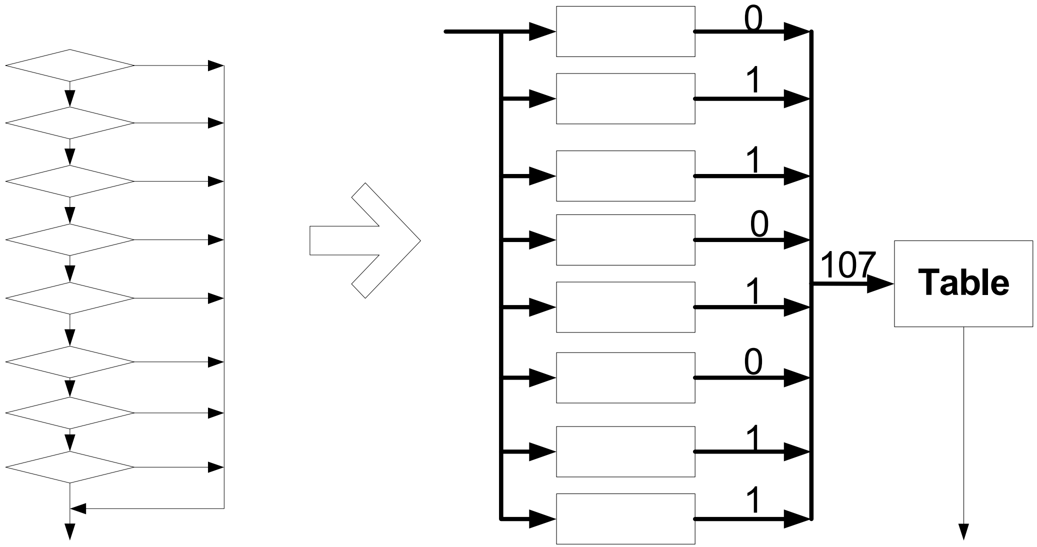
Region finding

- Operates on 3 x 3 window.
- Roughly linear in frame size.
- Sequential algorithm---window moves one pixel per step.



Program changes

- Contour fitting is very control intensive:
  - Compares local configurations of bits.
- Transformed into data-parallel operations for VLIW:



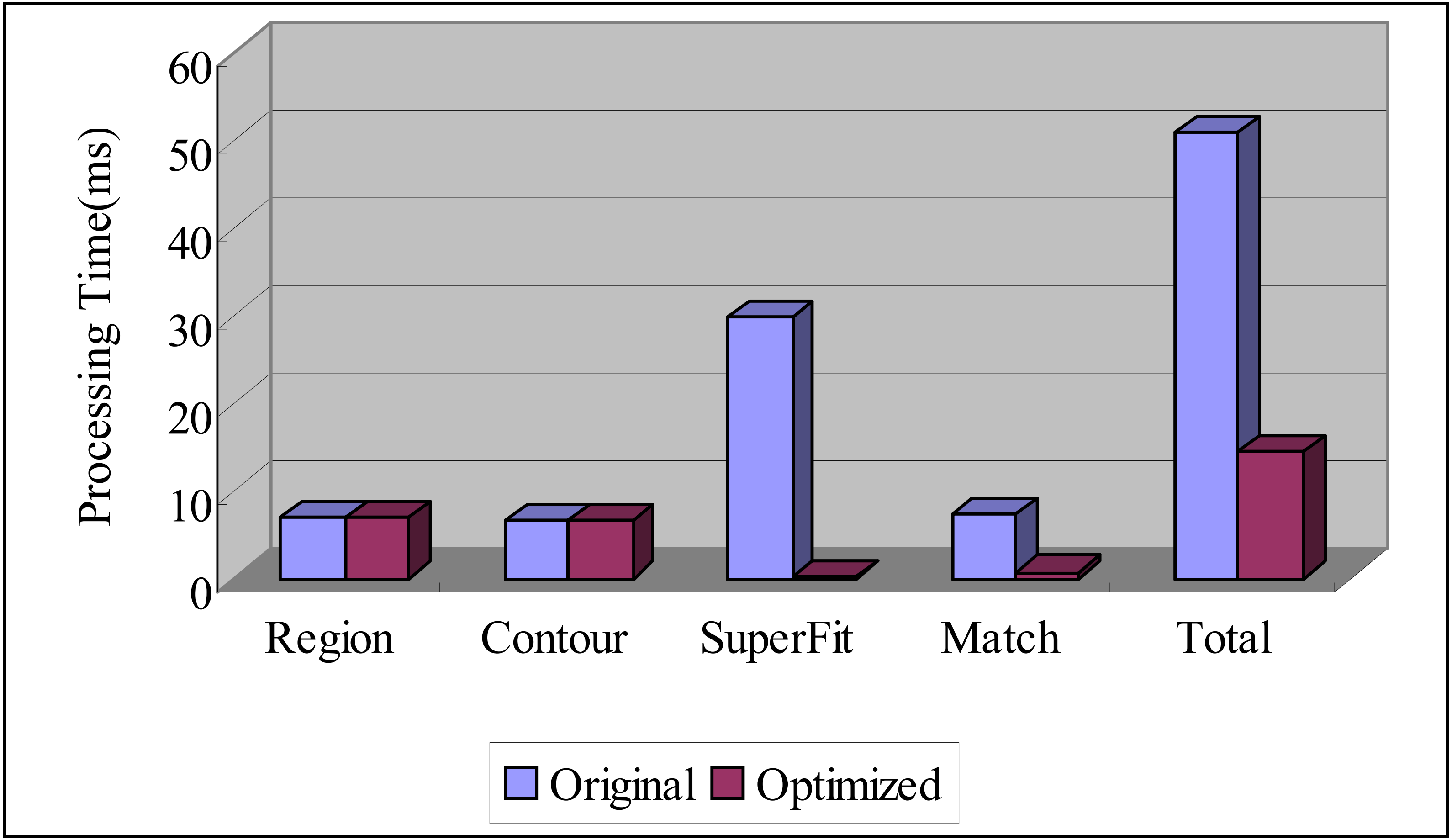
Control-oriented

Data-oriented

Instruction changes

- Trimedia provides library of intrinsic functions that map onto Trimedia instruction sequences.
- Goal: eliminate branches.
  - Special instructions.
  - Loop unrolling.

Before and after stage times



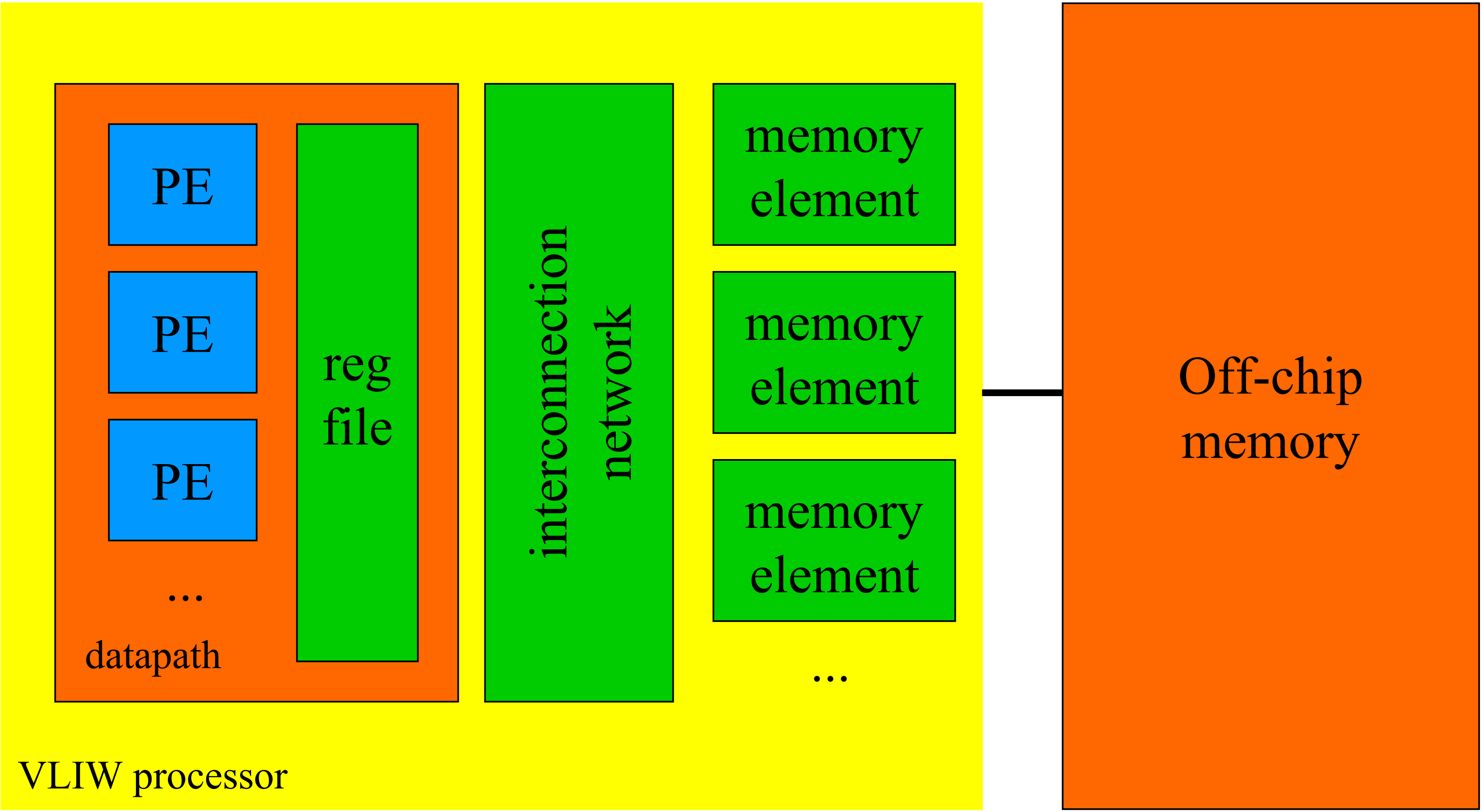
Results

- Before: 5 frames/sec.
- After: 31 frames/sec w/o HMM, 25 frames/sec with HMM.
- Latency approx. 100 ms.
- Smaller variation in frame processing time.

Architectural experiments

- Fritts/Wolf:
  - characterize applications;
  - compare architectural styles (VLIW, superscalar);
  - evaluate architectural parameters (clock rate, pipelining, etc.).

VLIW processor model



Workload characteristics experiments

- Goal: compare media workload characteristics to general-purpose load.
- Used MediaBench benchmarks.
- Compiled on Impact compiler, measured with with Impact simulator.

Basic characteristics

- Comparison of operation frequencies with SPEC
  - (ALU, mem, branch, shift, FP, mult) => (4, 2, 1, 1, 1, 1)
  - Lower frequency of memory and floating-point operations
  - More arithmetic operations
  - Larger variation in memory usage
- Basic block statistics
  - Average of 5.5 operations per basic block
  - Need global scheduling techniques to extract ILP

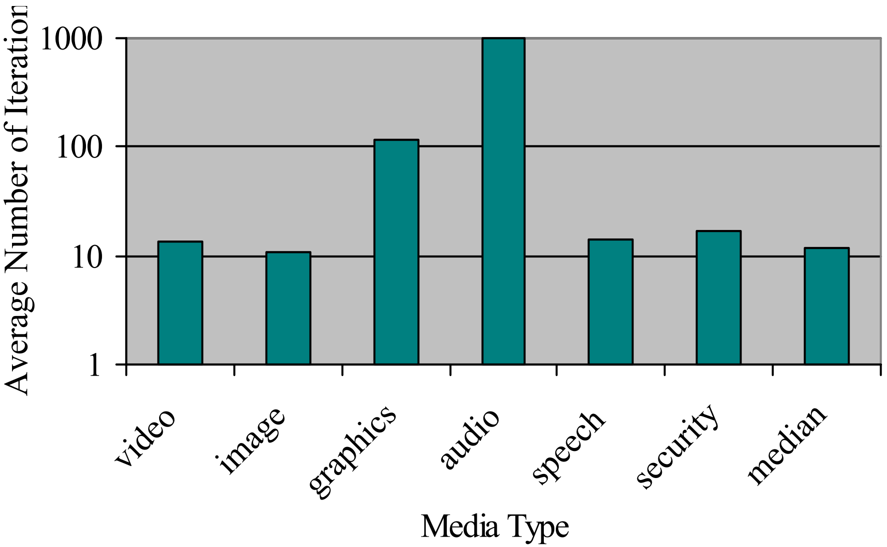
Basic characteristics, cont’d

- Static branch prediction
  - Average of 89.5% static branch prediction on training input
  - Average of 85.9% static branch prediction on evaluation input
- Data types and sizes
  - Nearly 70% of all instructions require only 8 or 16 bit data types

Multimedia looping characteristics

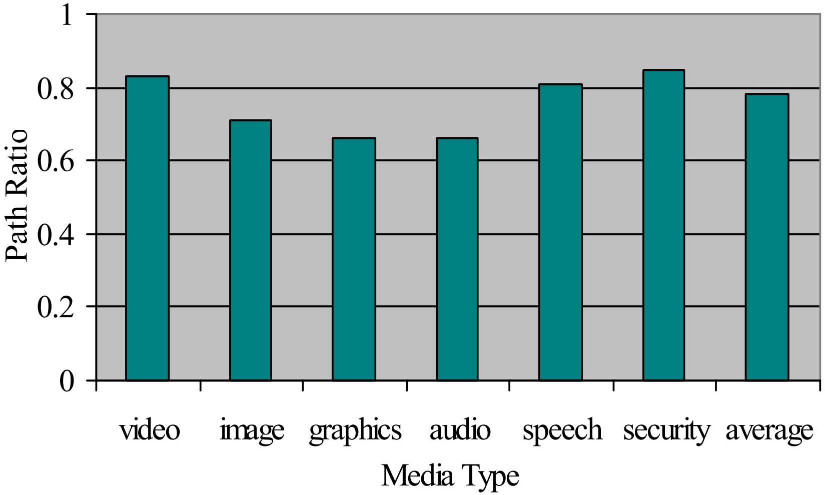
- Highly loop centric
  - 95% of CPU time in two innermost loop levels
  - Significant processing regularity
  - About 10 iterations per loop on average
- Complex loop control
  - = average # of instructions executed per loop invocation/total # of loop instructions
  - Average path ratio of 78%--high complexity

Average iterations per loop and path ratio



- average number of loop iterations

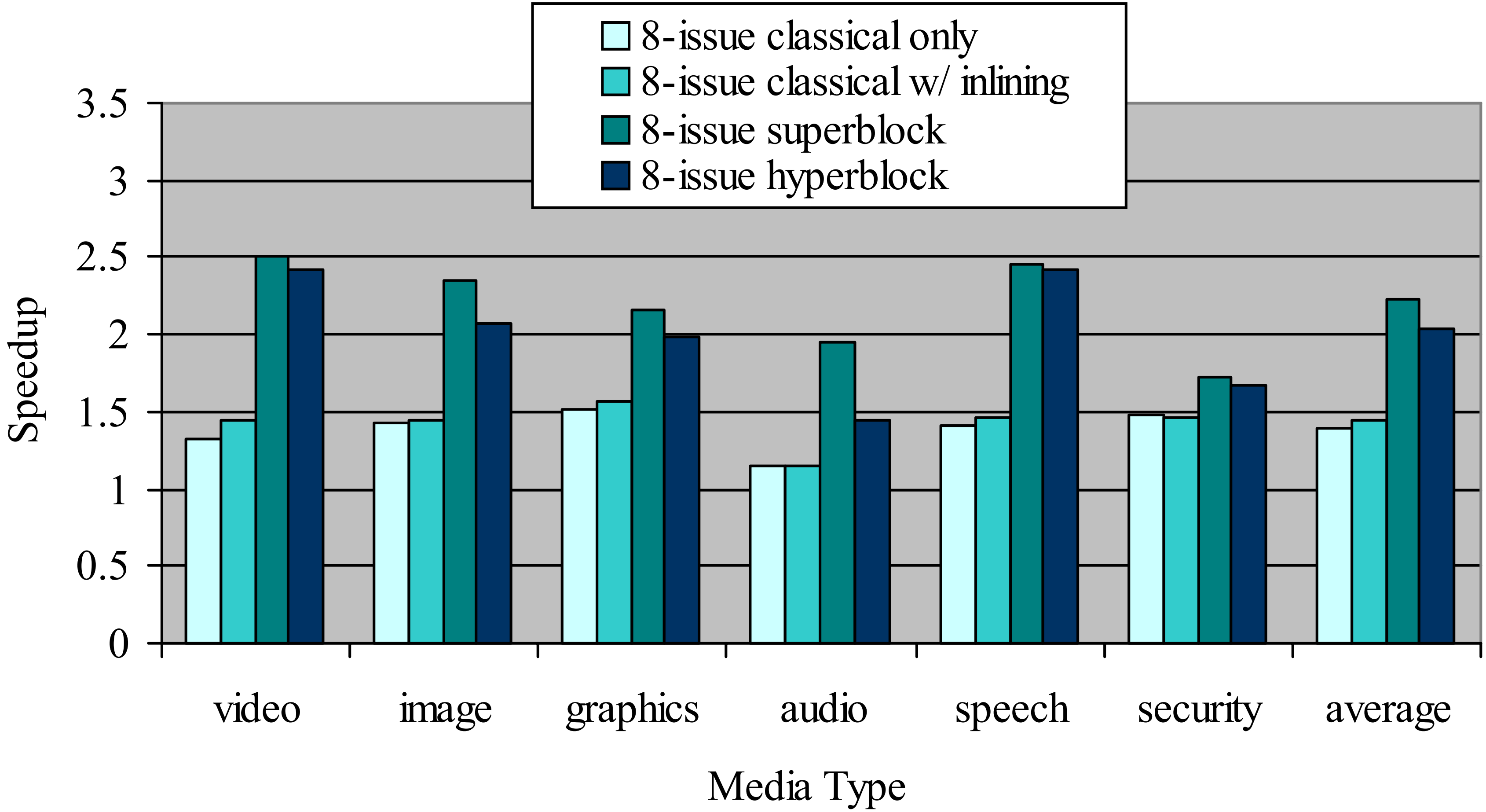
- average path ratio



Instruction level parallelism

- Instruction level parallelism
  - base model: single issue using classical optimizations only
  - parallel model: 8-issue
- Explores only parallel scheduling performance
  - assumes an ideal processor model
  - no performance penalties from branches, cache misses, etc.

ILP results


8-issue classical only
8-issue classical w/ inlining
8-issue superblock
8-issue hyperblock
3.5
3
2.5
2
1.5
1
0.5
0
Speedup
video
image
graphics
audio
speech
security
average
Media Type

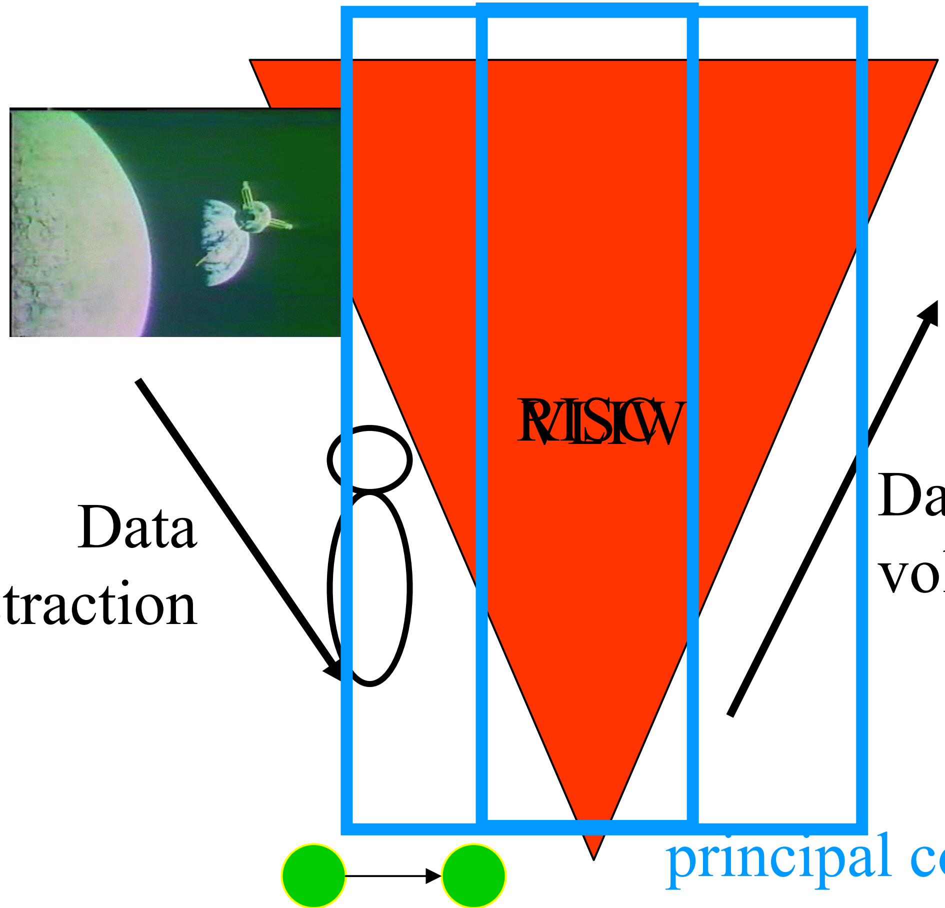
Multiprocessor architectures for video

- Interested in high-speed video processing.
  - 150 frames/sec.
- Want reasonably low-power operation for pervasive applications.

High-speed smart cameras

- High frame rates provide better motion capture.
- Frame rate of 150 frames/sec is considered desirable.
- Stanford CMOS camera can digitize at 10,000 frames/sec.

Why heterogeneous architectures make sense



pixel processing

RISC
VLIW

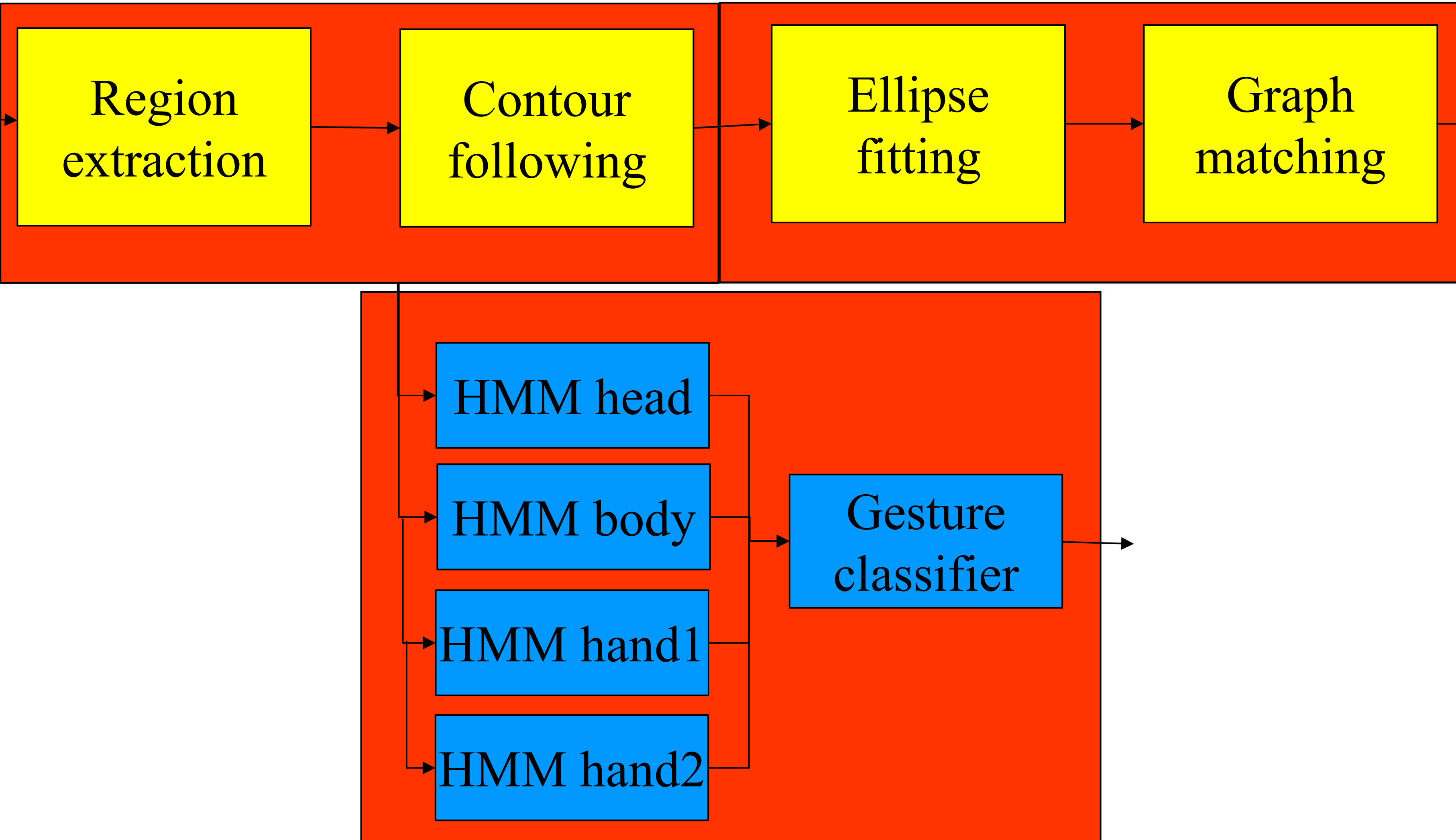
Data abstraction

Data volume

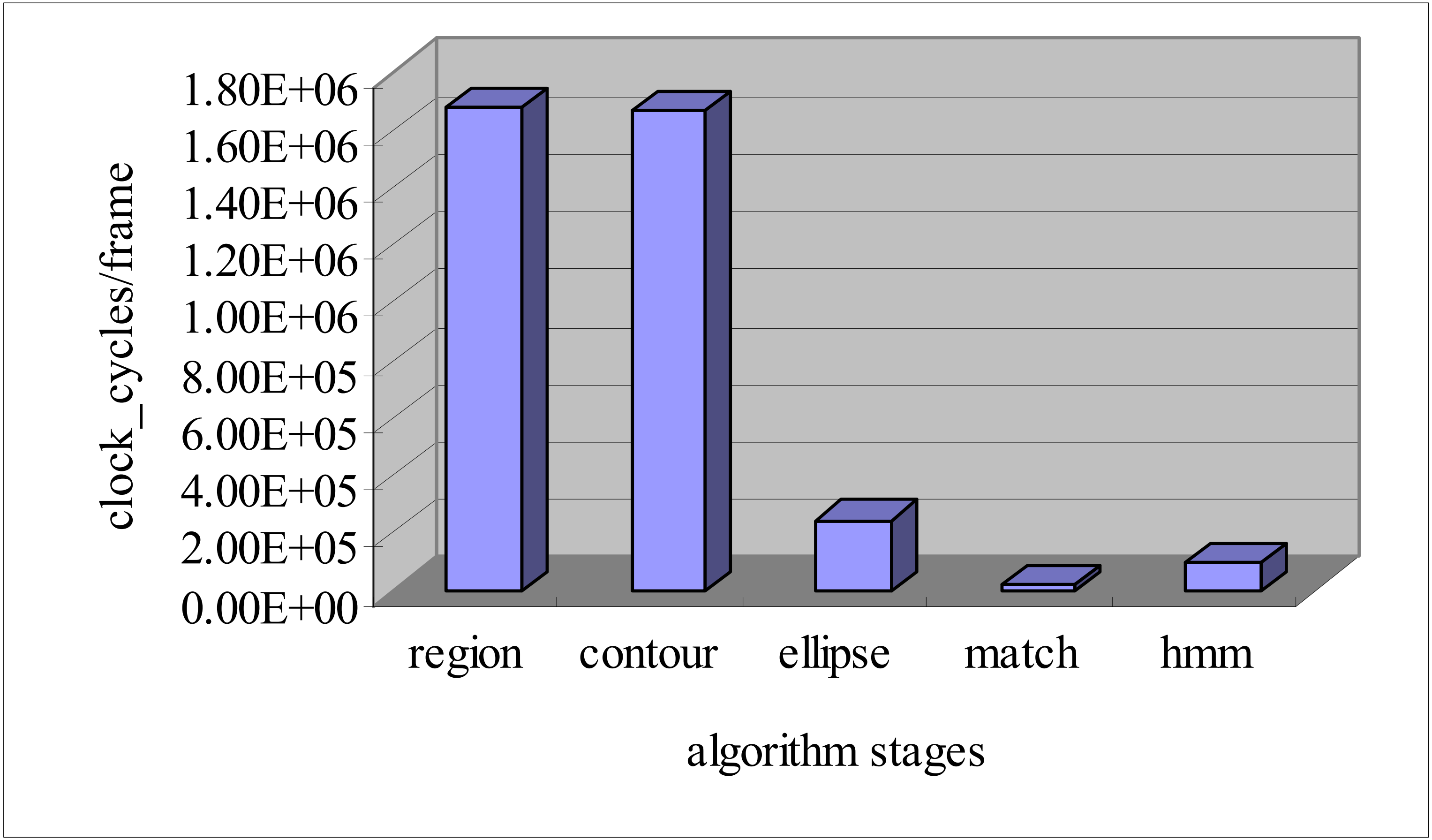
Edge extraction

principal component analysis, hidden Markov models

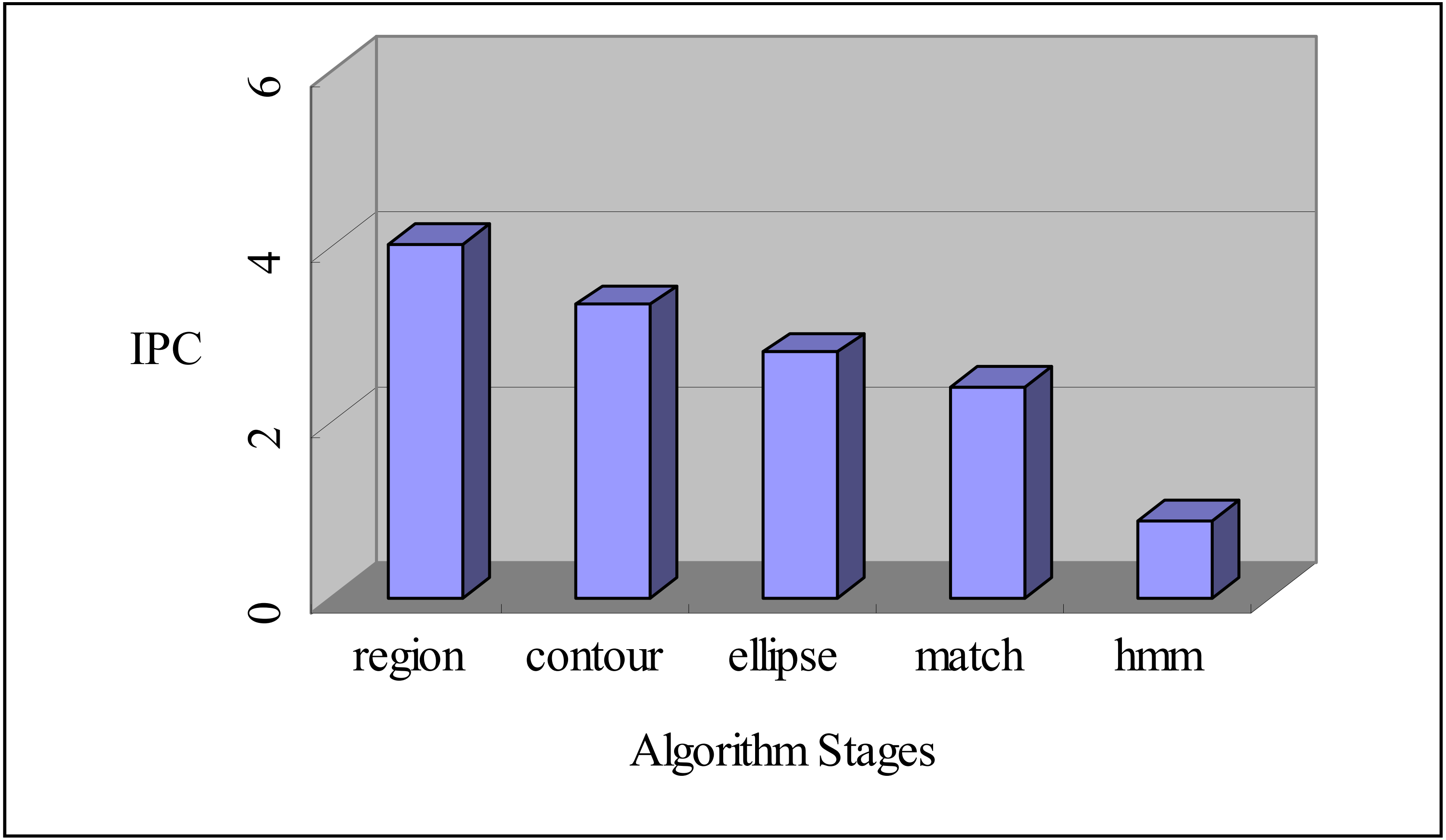
Algorithm flow



Average processing time by stage



Average IPC by stage

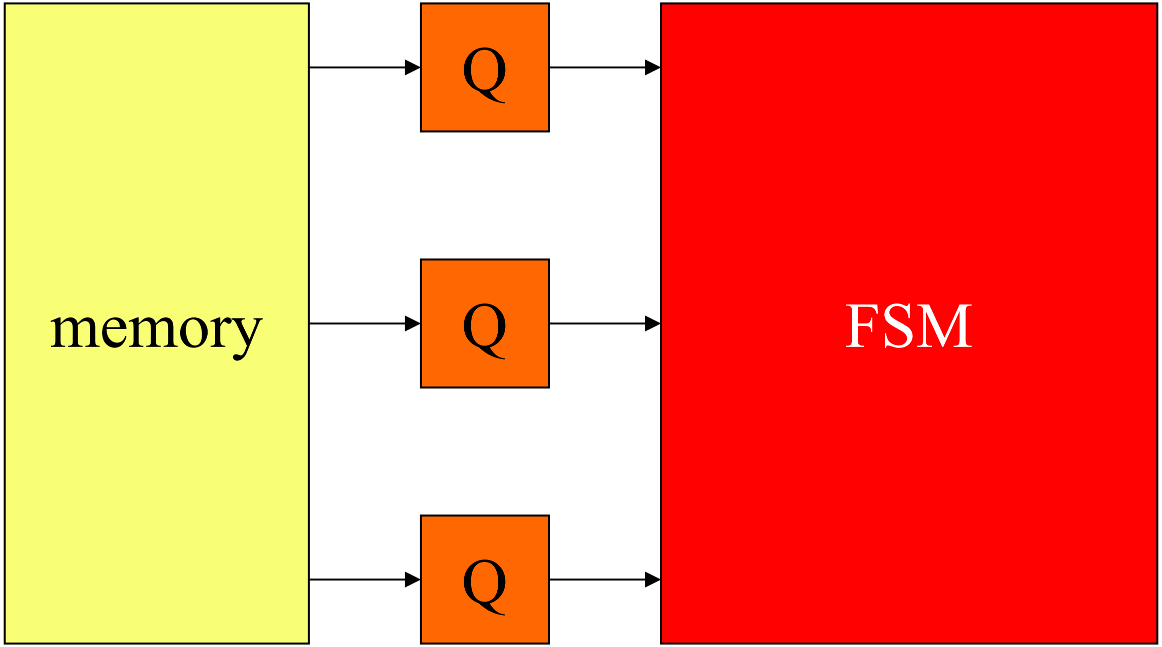


Tiehan's VLIW implementation

- Unroll loop to perform multiple comparisons in parallel.
- Pack results into bit vector to address results table.
- Register file, cache provide for reuse of pixel values.

Contour crawler machine

- Hardware implementation of VLIW code:



Crawler and memory

- Crawler performance depends on memory system.
- Access patterns vary in 2 dimensions:

1	2	3
8	X	4
7	6	5

Memory system design

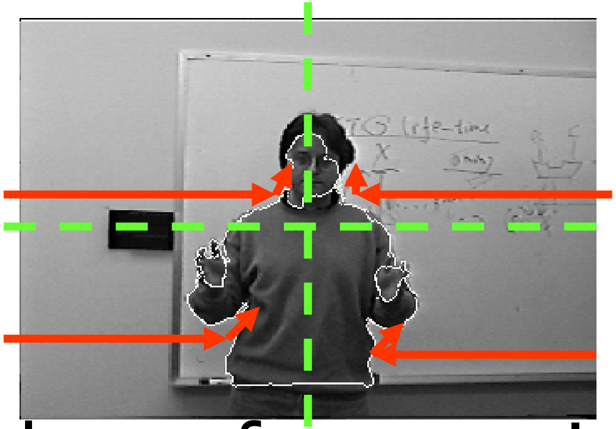
- Want to minimize number of partitions to reduce row/column overhead.
- Only memory organization that allows for all parallel accesses is one-word partition.
- Assume we fetch one row or column at a time---3 fetches/cycle.

Single contour crawler

- Assuming row/column access pattern, crawler is faster than VLIW by a relatively small constant.

Multiple crawlers

- Assuming we can patch together contours, we can start multiple crawlers.



- Multiple crawler performance is limited by memory.
  - Multiple crawlers’ memory accesses can conflict.

Full-frame SIMD

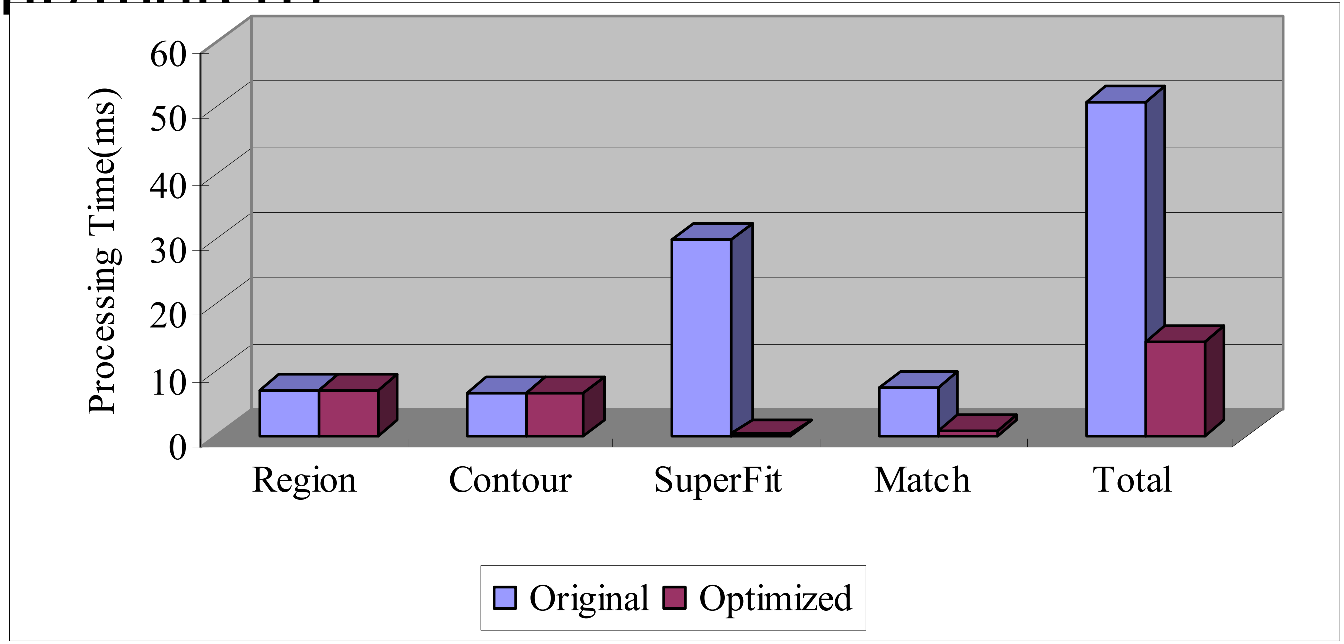
- Can build a large SIMD array with one processor per pixel.
- Area*delay:
  - Speed is roughly constant.
  - PE is probably about the same size as the crawler.
  - Not clear it is worth the silicon.

Heterogeneous system

- Region:
  - Stream processor with current algorithm.
  - Stream processor + RISC for others.
- Contour:
  - Crawler.
- Ellipse:
  - Superscalar/RISC.
- Graph:
  - RISC.

Stage pipelining

⌘ Stages are fairly well balanced (region/contour, superfit/match):



Heterogeneous vs. VLIW

- VLIW:
  - Off-the-shelf IP.
  - Easy to program.
  - 10 mm^2 in 0.13 micron.
- Heterogeneous:
  - Requires more design of blocks, memory.
  - Pipelineable for 2.3X speed-up.

Heterogeneous multiprocessor size

stage	PE	area (mm^2)
background	MIPS32 4Km	0.9
contour	custom	0.001
ellipse, graph	MIPS64 5Kf	5
total frame processor		5.901
classification	MIPS64 5Kf	5
number of frame processors		3
grand total		22.703

Summary

- Multimedia applications are already more complex and will become more so:
  - multiple algorithms;
  - complex control and data.
- Instruction-level parallelism helps, but isn’t everything.