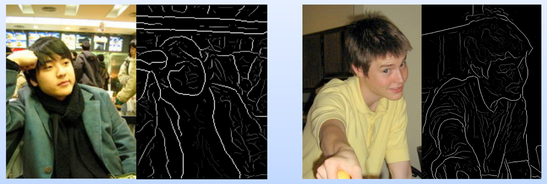
High-Throughput, Accurate Image Contour Detection

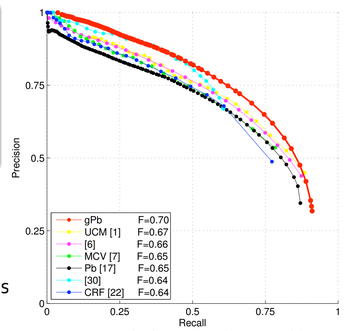


Yunsup Lee and Andrew S. Waterman

Image Contour Detection

- Image contour detection is fundamental to image segmentation and many other computer vision problems

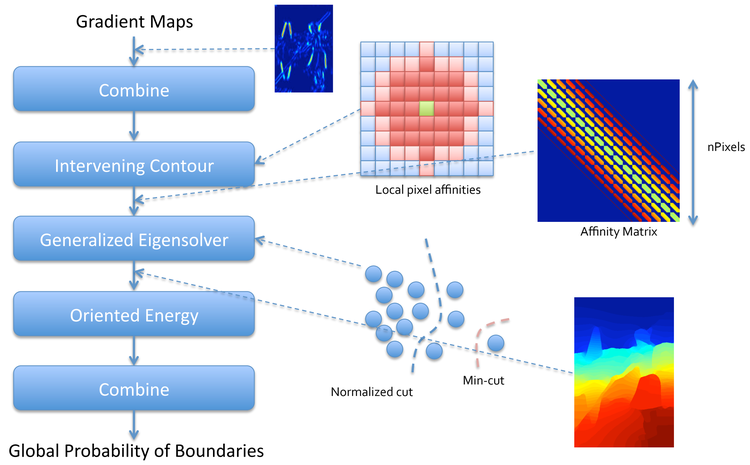
Image | Human Generated Contours | Machine Generated Contours



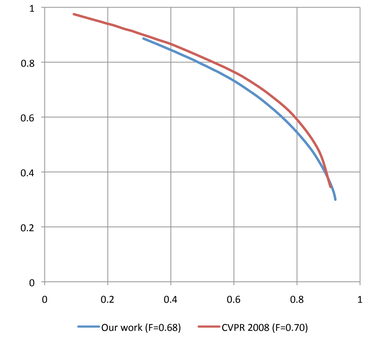
- gPb (**g**lobal **P**robability of **b**oundary) is currently the most accurate detector
- However, it takes 5.8 minutes to process a small image (481 x 321 = 0.15MP)
- This limits its applicability

Maire, Arbelaez, Fowlkes, Malik, CVPR 2008

Gradient Maps to Global Segmentation

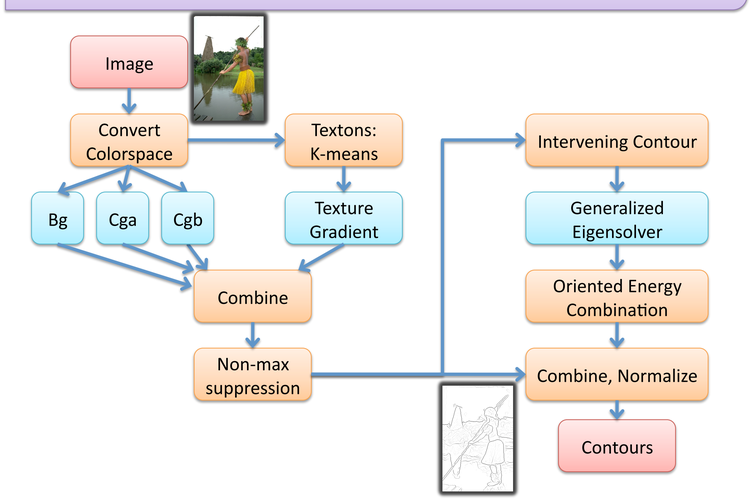


Results & Correctness



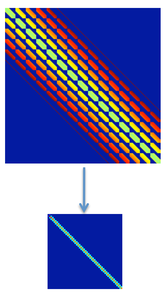
- In order to verify the results, we run the program through the Berkeley Segmentation Dataset (BSDS)

gPb Breakdown: Pipe & Filter

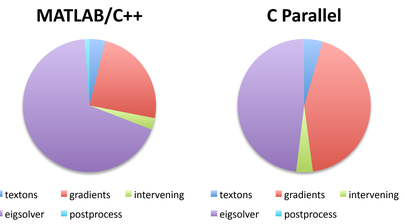


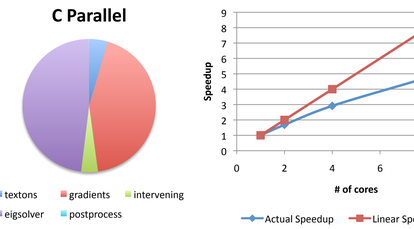
Hybrid Lanczos Eigensolver

- Our problem, then, is to find the *k* smallest eigenvalues with their eigenvectors from a large, symmetric, real sparse matrix (*n=width*height*=154401 for a 321 x 481 image)
- The Lanczos Algorithm is well suited for this problem
- Exterior eigenvalues converge quickly
  - We only need a few of the smallest eigenvalues
- Inside the Lanczos iteration loop
  - SpMV routine; NUMA aware
  - BLAS1 routines; saxpy,sdot,snrm2,sscal
- Calculating the eigenvalues of the small symmetric matrix
  - LAPACK routines; dstebz
- Calculating the eigenvectors
  - LAPACK routines; dstein
  - BLAS3 routines; sgemm



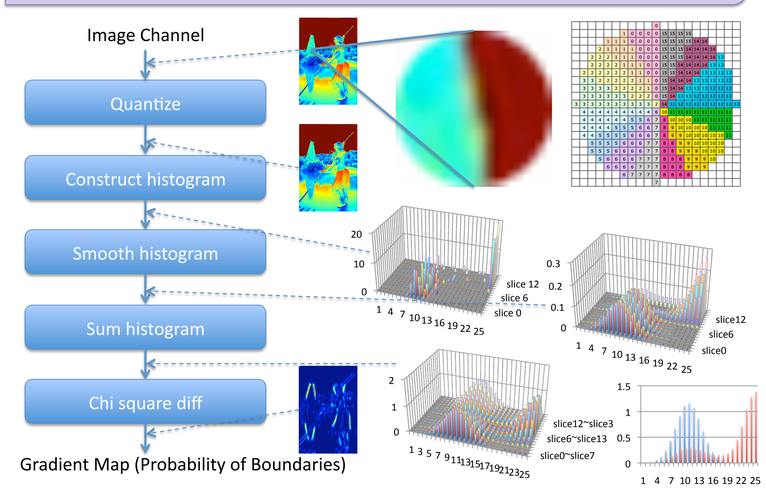
Time Breakdown & Speedup





Kernel	MATLAB/C++	C Serial	C Parallel (8)	Speedup
Textons	8.6	7.29	1.35	6.38x
Gradients	53.8	79.98	12.92	4.17x
Intervening	6.3	8.63	1.21	5.23x
Eigensolver	151.0	45.82	14.29	10.57x
Postprocess	2.7	0.23	0.03	86.54x
Total	**222.4**	**141.95**	**29.79**	**7.47x**

Image Channels to Gradient Maps



Optimizations

- Parallelization using HardThreads API
  - Lightweight exploitation of data parallelism
  - Bare-metal access to hardware resources
    - on x86+Linux, we emulate this by pinning threads to HW contexts
    - on RAMP, we implemented first-class HW support for this mechanism
- Tuned synchronization primitives (atomics, barriers)
- Algorithmic transformations
  - Generalized eigensolver => Lanczos + Cullum-Willoughby
- NUMA-aware memory allocation
  - Significant latency reduction for multi-socket systems
- Loop unrolling, cache blocking for SpMV
  - Little improvement on x86 (dynamic scheduling overlaps latencies)
  - Lower inst. count improves in-order SPARC performance considerably

Conclusions & Future Work

- We achieved **7.5x** speedup against the original MATLAB/C++ implementation
- We achieve qualitatively similar accuracy across the BSDS test suite of 100 test images
- Our parallel implementation executes portably on x86 and RAMP
- This motivating real-world application will help drive parallel computer architecture research via the RAMP port