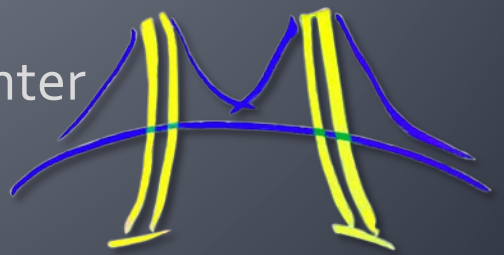
Patterns and Computer Vision

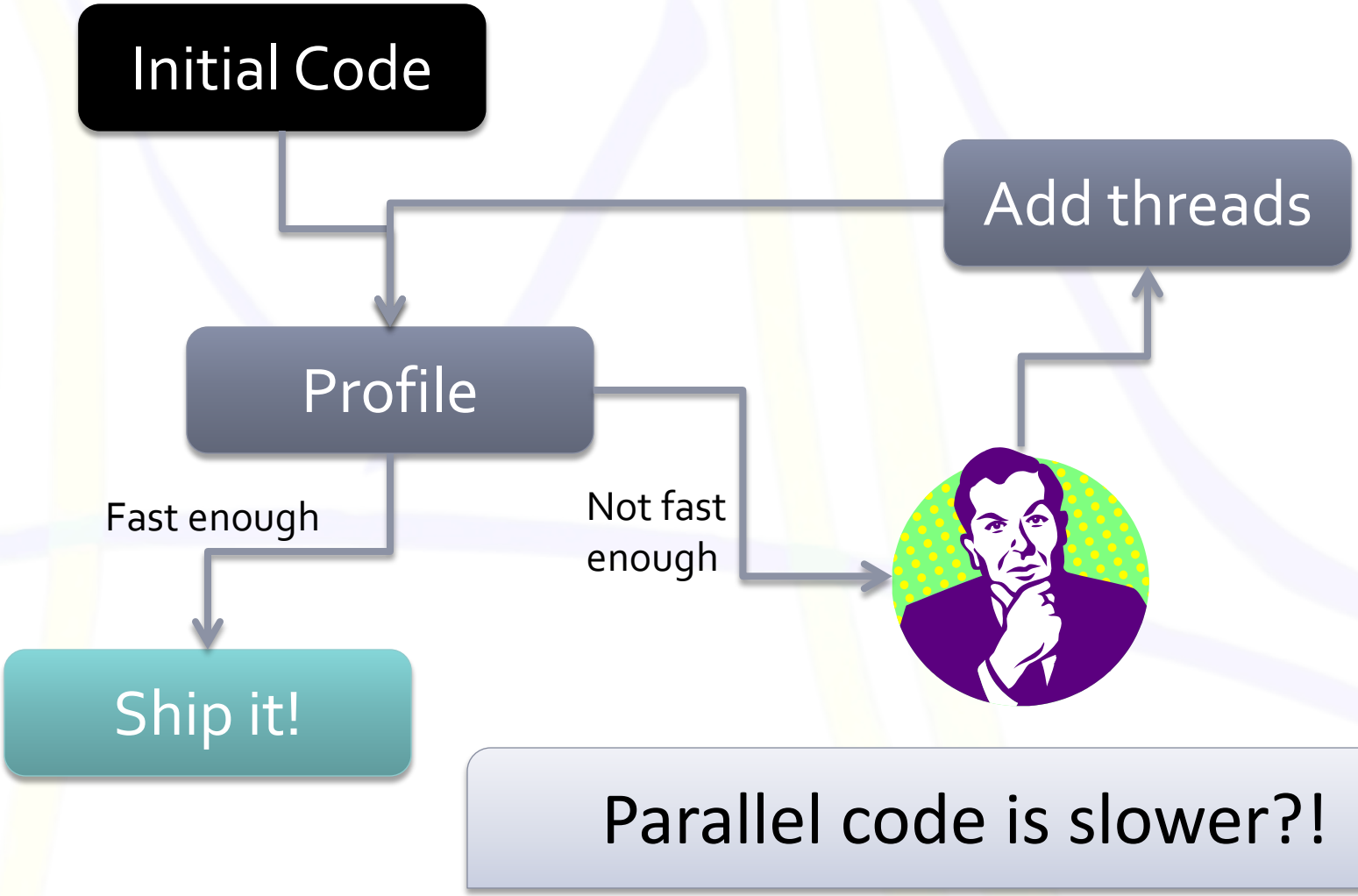
Michael Anderson, **Bryan Catanzaro**, Jike Chong, Katya Gonina, Kurt Keutzer, Tim Mattson, Mark Murphy, David Sheffield, Bor-Yiing Su, Narayanan Sundaram

…and the rest of the PALLAS group

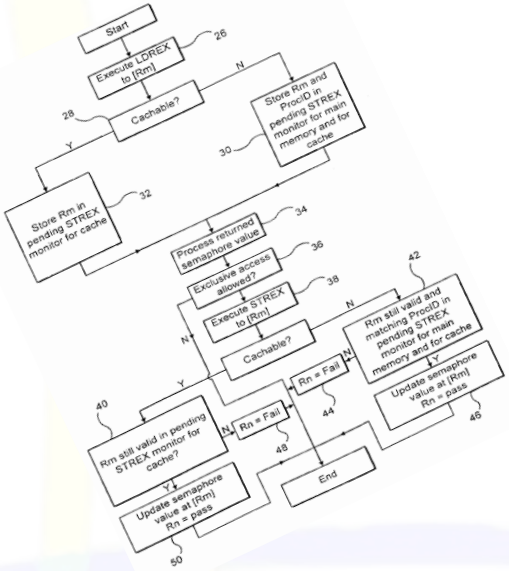
Universal Parallel Computing Research Center
University of California, Berkeley



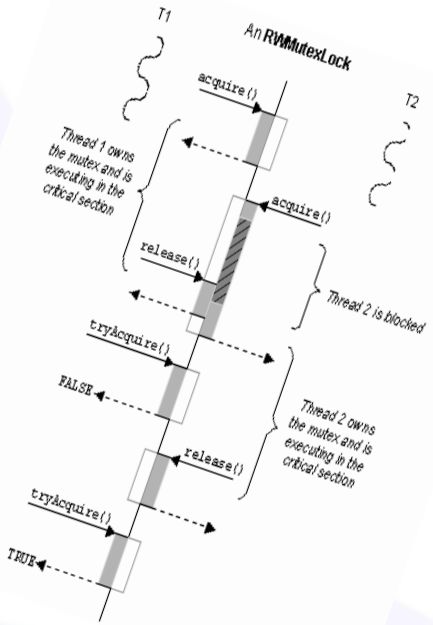
How NOT to write parallel programs



What is this person thinking of?



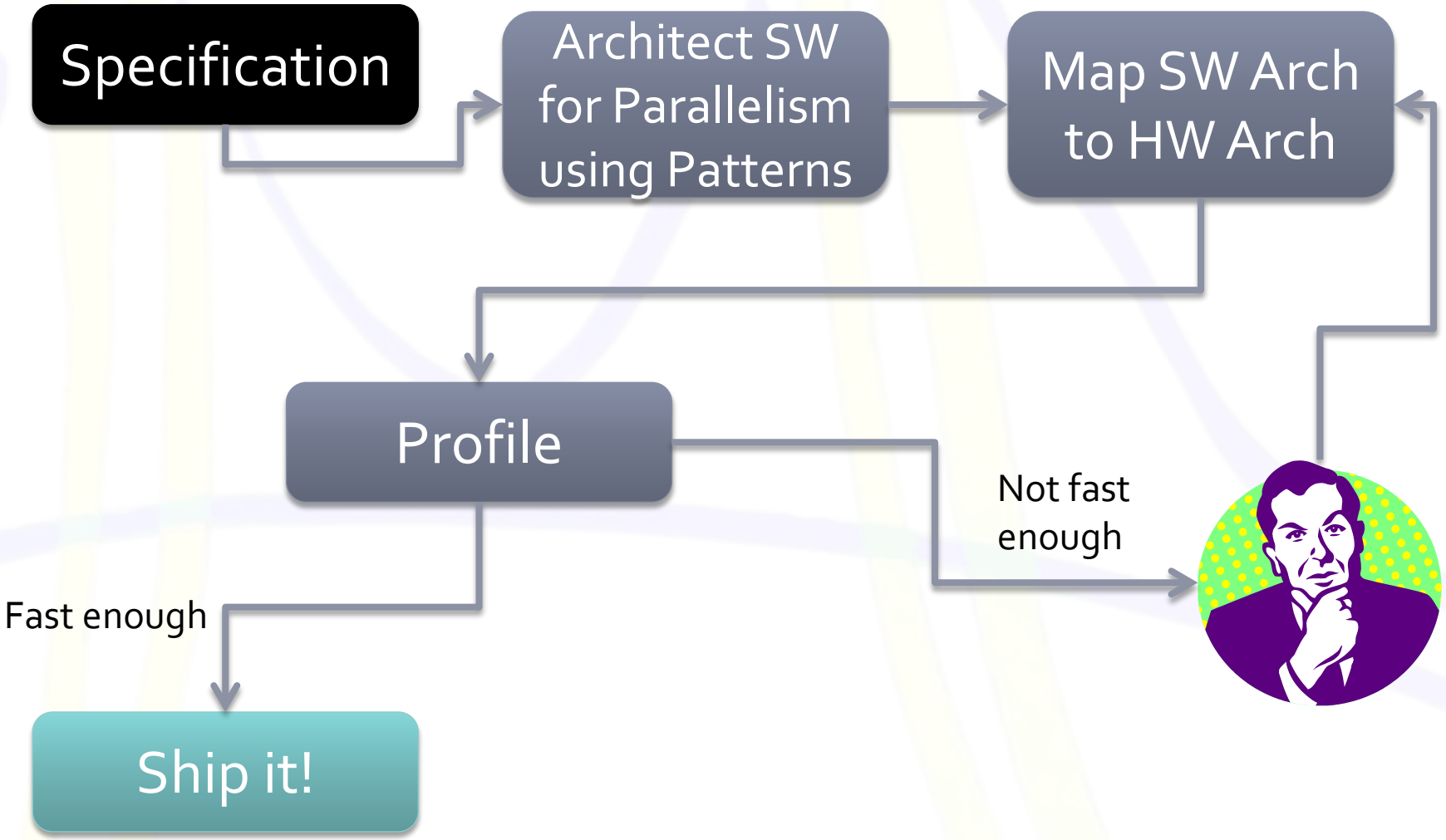
Add threads



Edward Lee, "The Problem with Threads"

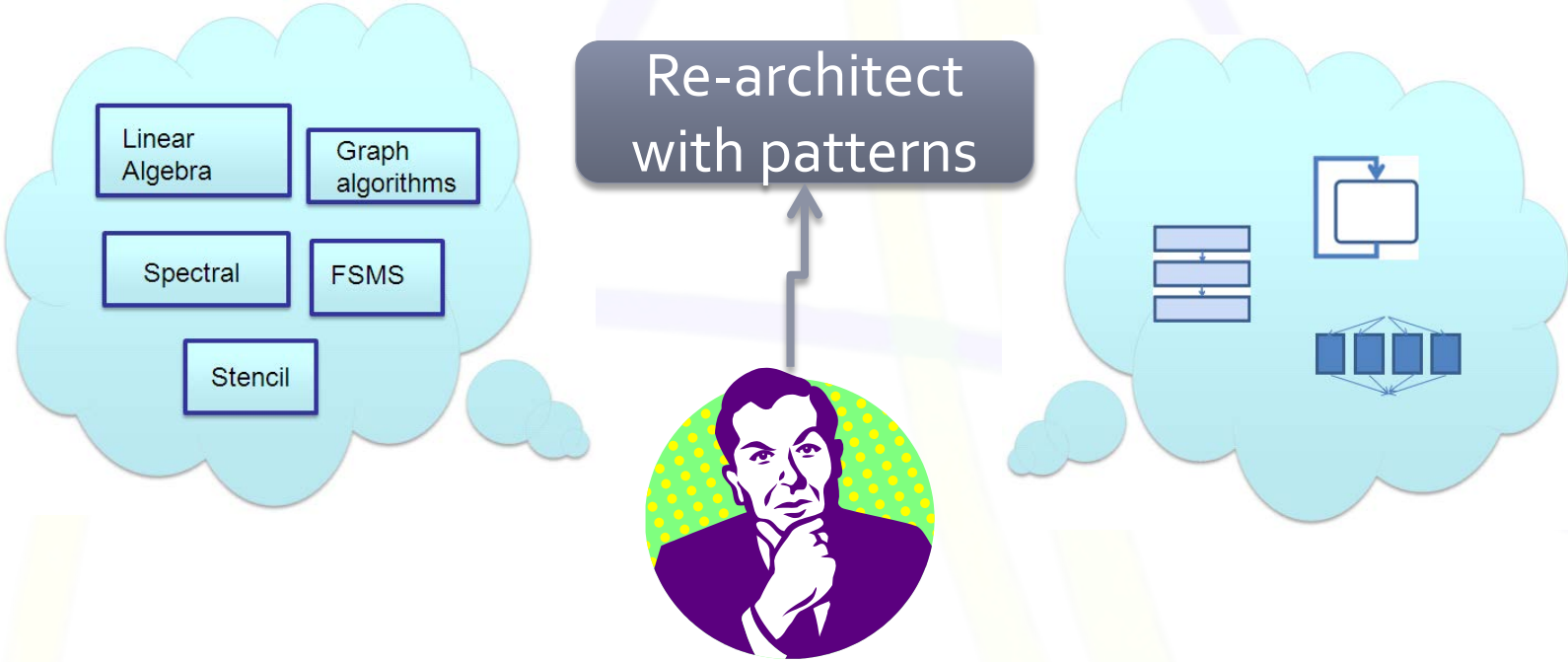
Threads, locks, semaphores, mutexes

Architecting parallel software



What is this person thinking of?

- Today's code is serial, but the world is parallel
  - Find the inherent parallelism, express it in software, map it to parallel hardware



Applying this methodology

- We developed a pattern language to help people learn how to successfully architect parallel programs
- We applied this methodology in several areas
  - Computer Vision
    - Support Vector Machine Training & Classification
    - **High-Quality Image Contour Detection**
    - Object Recognition
    - Optical Flow & Point Tracking
    - Poselet based recognition and pose estimation
  - **MRI Reconstruction for Compressed Sensing**
  - Speech Recognition
  - Computational Finance

Parallelism in action

- Berkeley professor Jitendra Malik approached us:

> “ My new contour detector is awesome, but people can’t use it: it’s too slow

- A collaborative effort at the ParLab made it 100X faster
  - Rethink algorithms using patterns
  - Efficient parallel implementation

3.7 mins to 1.8 seconds

Image Contour Detection

- High quality contours
  - Provide high quality segments
  - And high quality classification
- But they are expensive to compute

Image

Image Contours

Image Segments

Image Contours

- Contours are subjective – they depend on personal perspective
- Surprise: Humans agree (more or less)
- We can compare against human "ground truth"

Image

Human Generated Contours

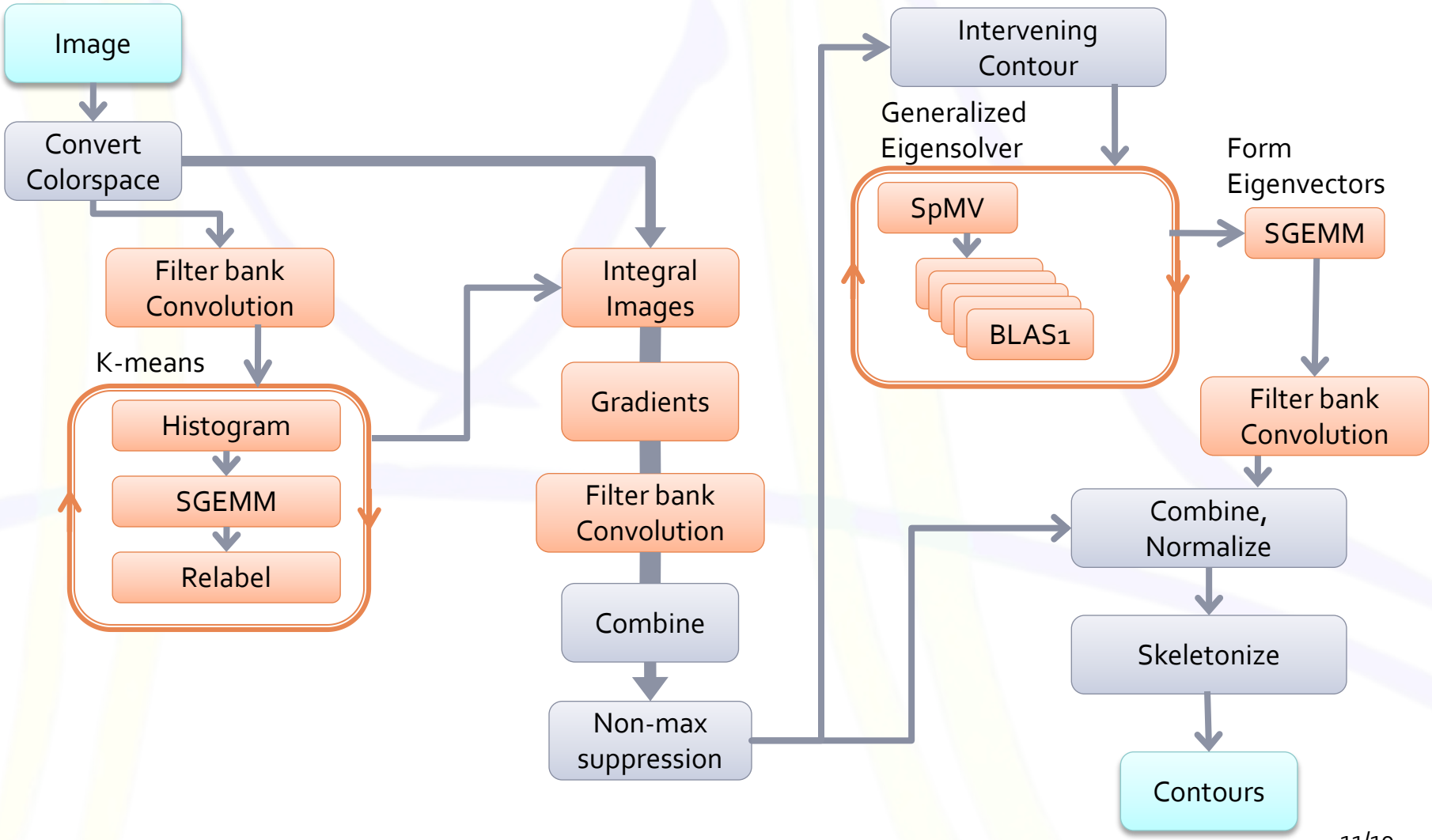
Machine Generated Contours

gPb Algorithm

- **g**lobal **P**robability of **b**oundary
- Currently, the most accurate image contour detector
- 3.7 mins per small image (0.15 MP) limits applicability
  - N billion images on web
  - Large computer cluster would take several years to find their contours
  - By then we'd have another 3N images to process…

Maire, Arbelaez, Fowlkes, Malik, CVPR 2008

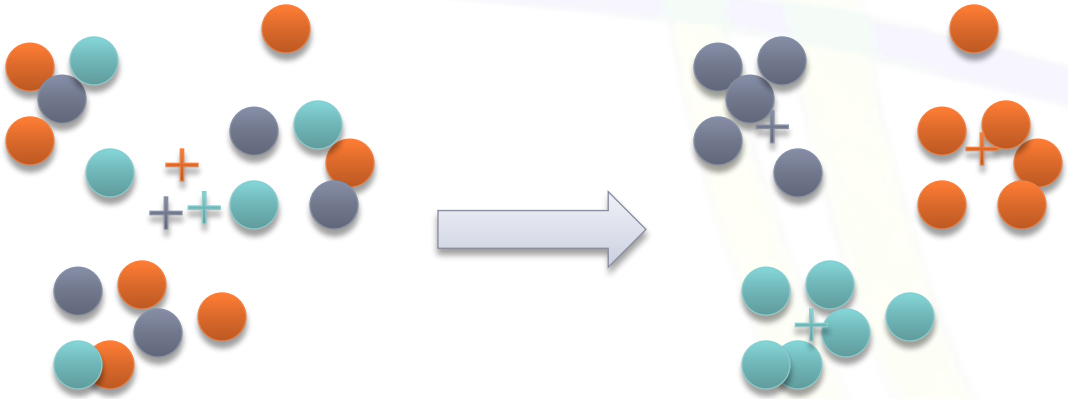
gPb Computation Outline



Textons: Kmeans

- Textures are analyzed in the image by finding textons
- The image is convolved with a filter bank
- Responses to the filter bank are clustered using k-means clustering

- **Parallel implementation: use BLAS, parallelize histogram construction**



Gradients

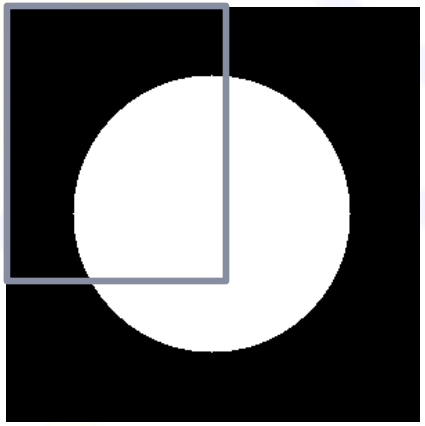
- An edge detector designed to filter out noise
- For every pixel, and for a given scale and orientation, compare half-discs centered at that pixel
- If half-discs are different, response is high (edge detected)

- **Algorithmic transformation: Use integral images to accelerate the analysis of all the half discs**

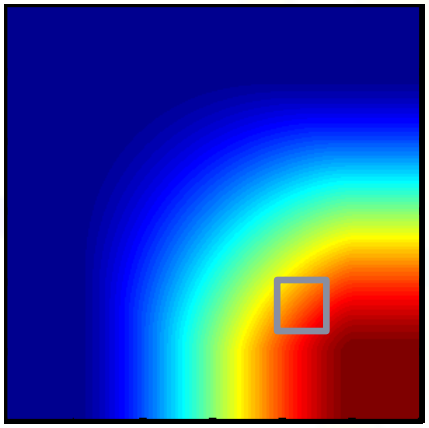
Integral Images

- Integral images are used to reduce algorithmic complexity

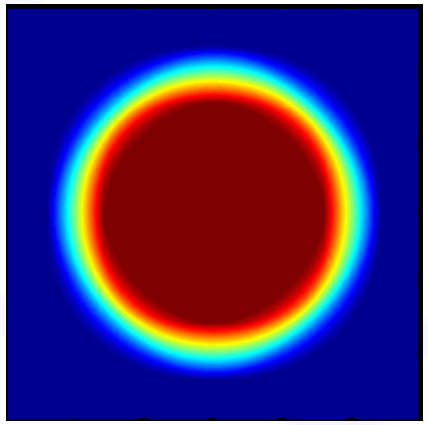
Input image



Integral image
$O(n^2)$



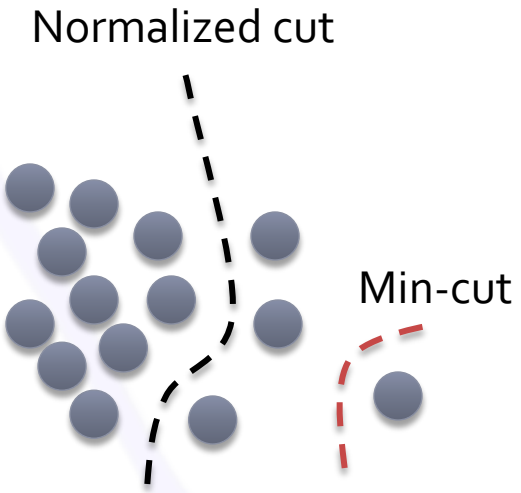
Input convolved with constant rectangle
$O(n^2r^2)$ ➔ $O(n^2)$



- Saves redundant summation in gradient calculation
- Implement using parallel-prefix sums (scans)

Spectral Graph Partitioning

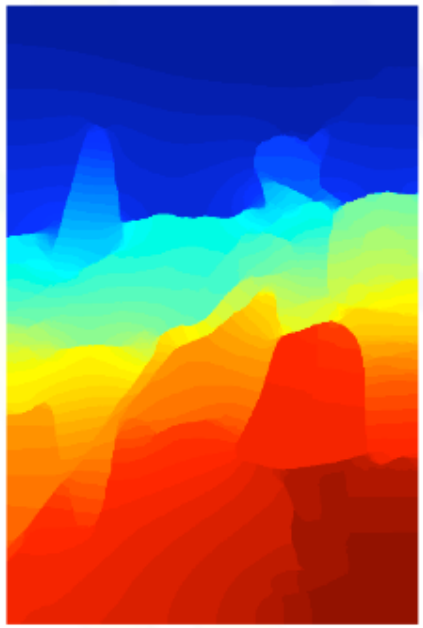
- Spectral methods avoid creating small, isolated regions
- An affinity matrix links each pixel to its local neighbors
- Interconnected local neighborhoods make a global system
- This step was the most computationally dominant for the serial implementation



Chainmail

Spectral Graph Partitioning, *cont.*

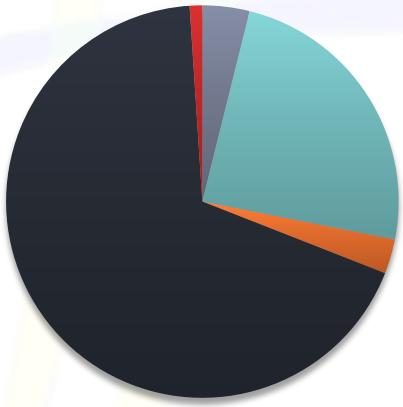
- Eigenproblem: we only need the smallest *k* eigenvectors
- **Algorithmic transformation, suggested by ParLab faculty Jim Demmel**:
  - Lanczos algorithm with the Cullum-Willoughby test



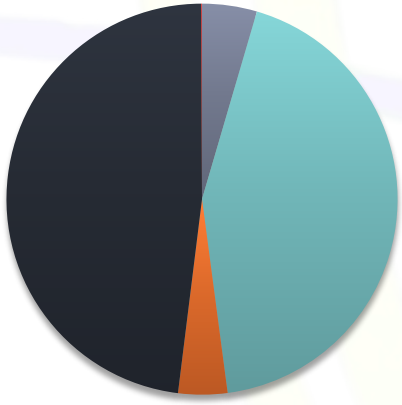
Performance Results

Computation	Original C++	C + Pthreads (8 threads, 2 sockets)	Damascene (GTX280)
Textons	8.6	1.35	0.152
Gradients	53.8	12.92	0.75
Intervening Contour	6.3	1.21	0.03
Eigensolver	151.0	14.29	0.81
Overall	222 seconds	29.79 seconds	1.8 seconds

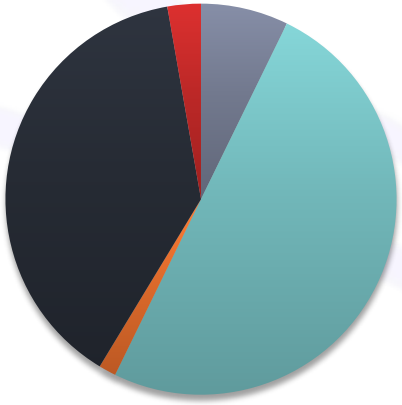
gPb: CVPR 2008



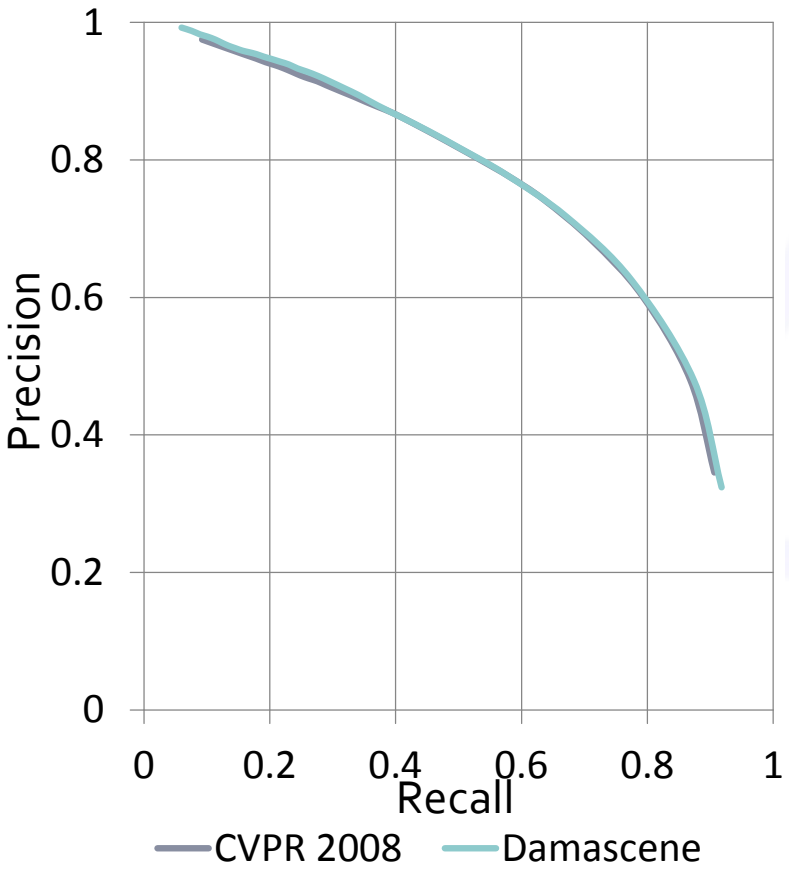
Pthreads



GTX280



Accuracy & Summary



- Our results just as accurate
- 1.8 seconds instead of 3.7 minutes
- Speedup came from:
  - Rethinking algorithms
  - Parallel implementation
- 3500 downloads
- Further work built on this to create image classifiers

Wrapping up

- Contour detection is only one example of the many computations we successfully parallelized
- Today we'll also mention
  - compressed sensing MRI reconstruction
  - Gaussian mixture modeling (PyCASP)
  - Optical flow
- …and there are many more
- Rearchitecting applications using patterns helped us make parallelism practically useful

Parallelism enables new applications