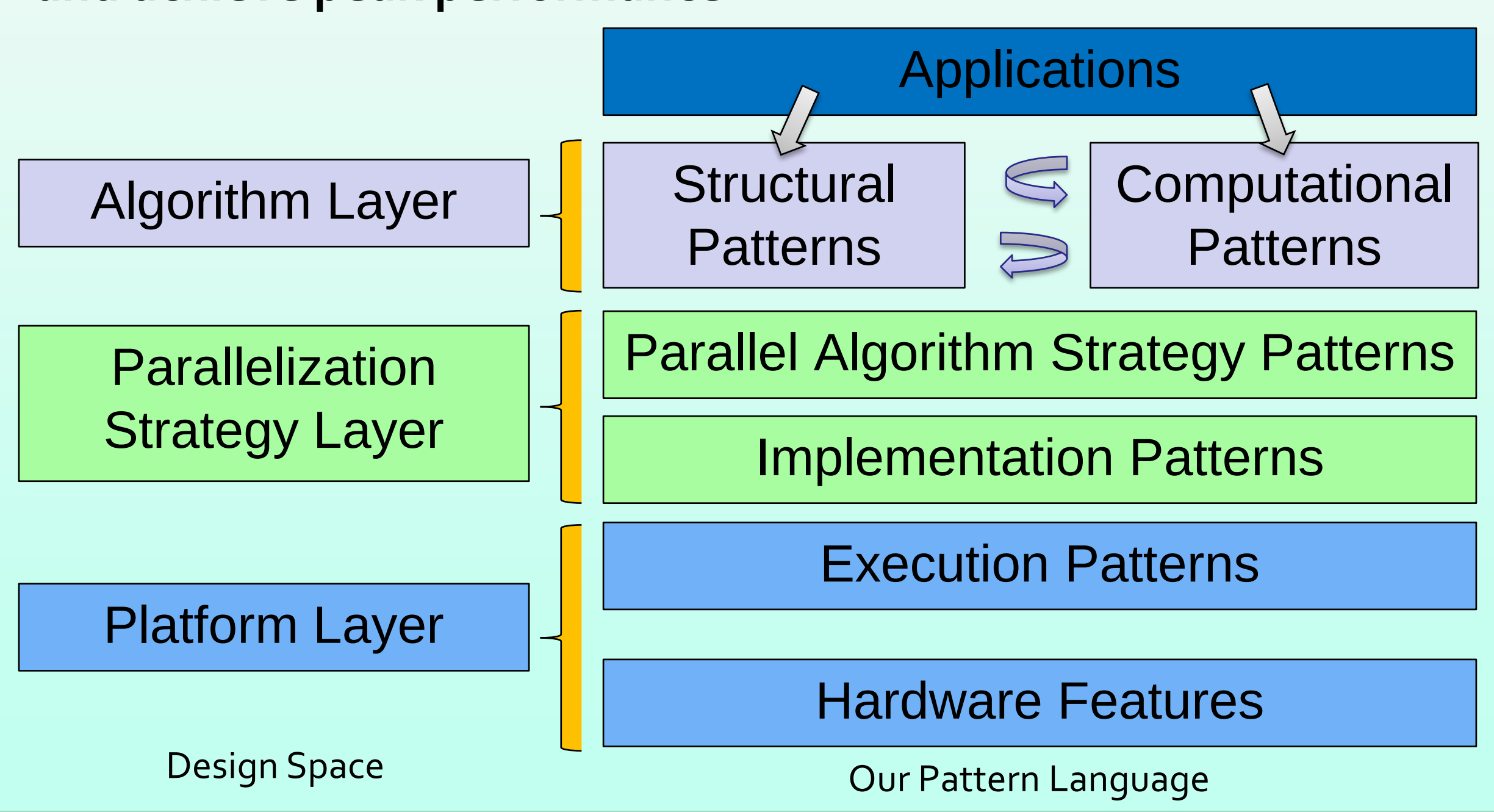
Exploring the Design Space of a Parallel Object Recognition System

Bor-Yiing Su, Tasneem Brutch, Kurt Keutzer

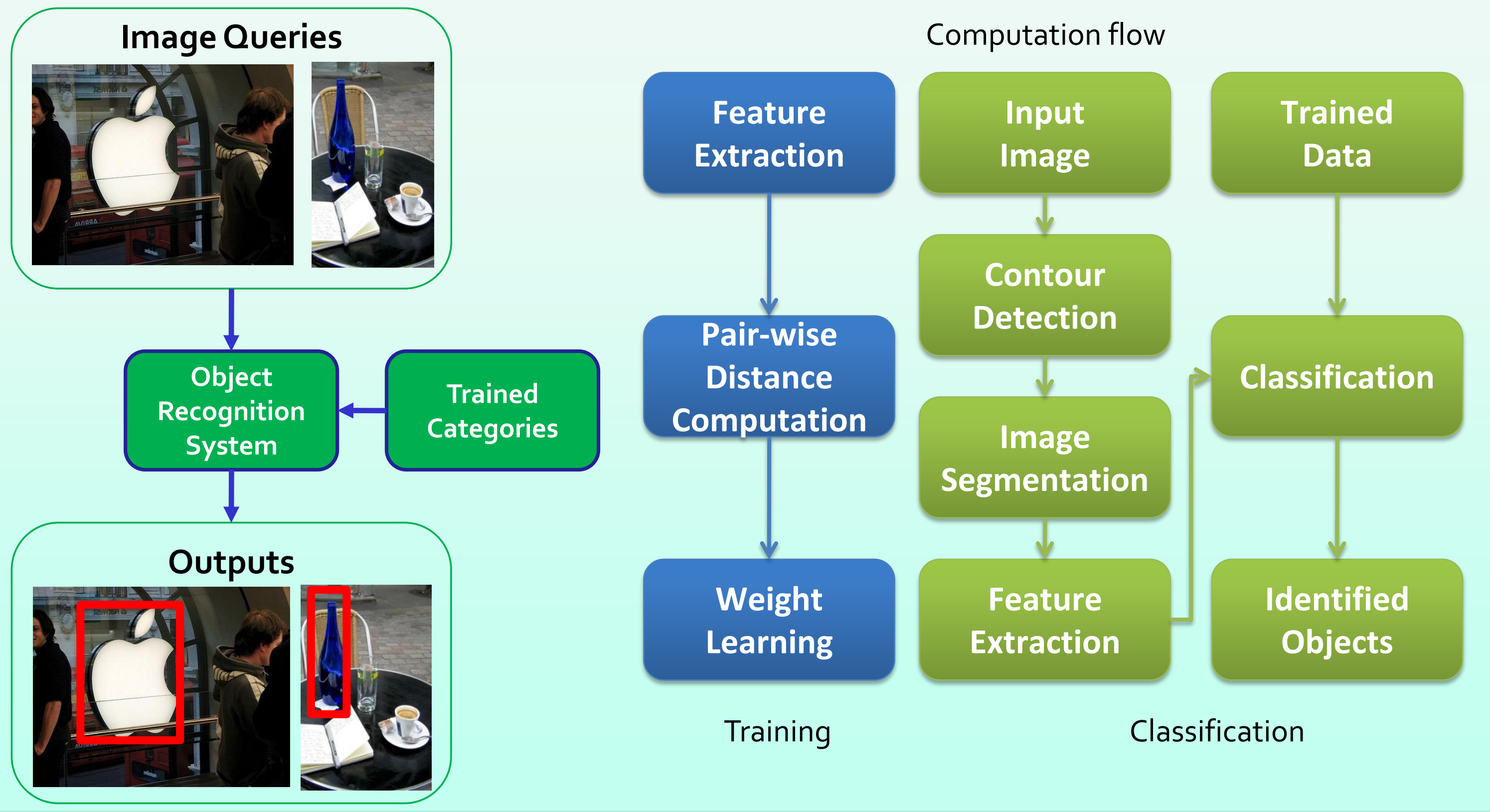
Statements

1. **Exploring the design space** is necessary to achieve high performance on a hardware platform of choice
2. **Take advantage of domain knowledge** is necessary to understand trade-offs among different parallelization methods and achieve peak performance



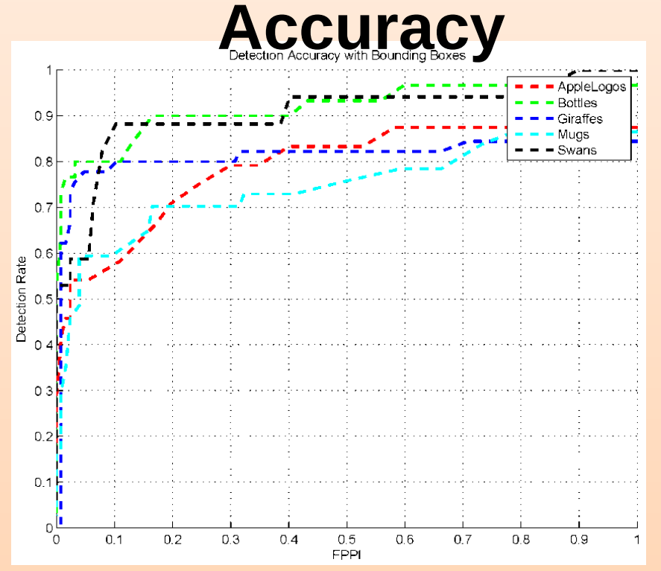
Targeting Object Recognition System

C. Gu, J. Lim, P. Arbelaez, and J. Malik, "Recognition Using Regions," Conference on Computer Vision and Pattern Recognition (CVPR'09), Miami, FL, 2009.

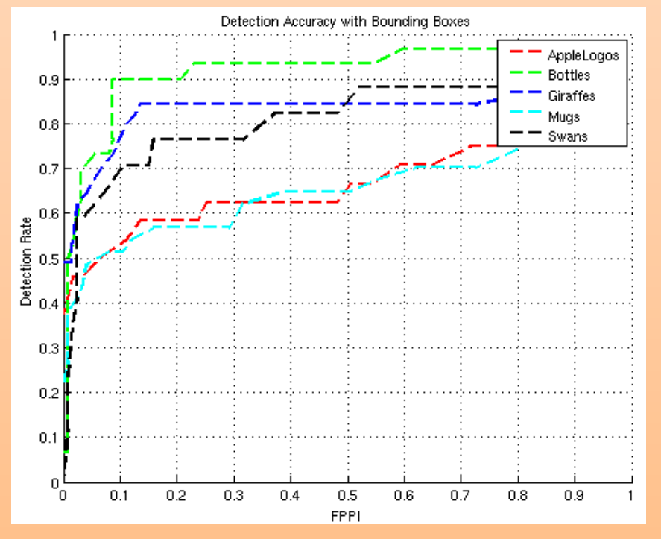


Overall Performance

Accuracy



Original Serial Algorithm



Parallel Algorithm

Speedups

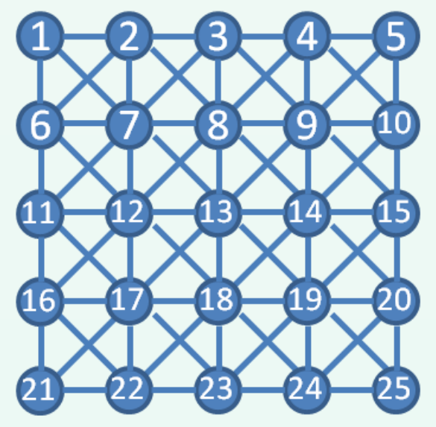
Computation	Computation time (s)		Speedup
	Serial	Parallel	
Feature	543	15.97	34x
Distance	1732	2.9	597x
Weight	57	1.41	40x
Total	2332	20.28	115x

Training on 127 images

Computation	Computation time (s)		Speedup
	Serial	Parallel	
Contour	236.7	1.58	150x
Segmentation	2.27	0.357	6.36x
Feature	7.97	0.065	123x
Hough Voting	84.13	0.779	108x
Total	331.07	2.781	119x

Classification

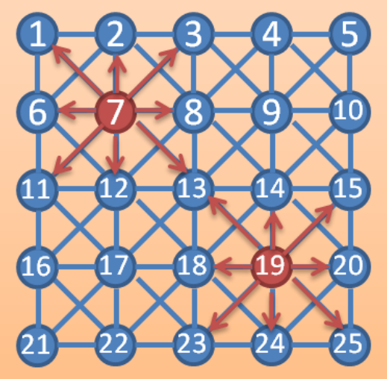
BFS Graph Traversal Kernel

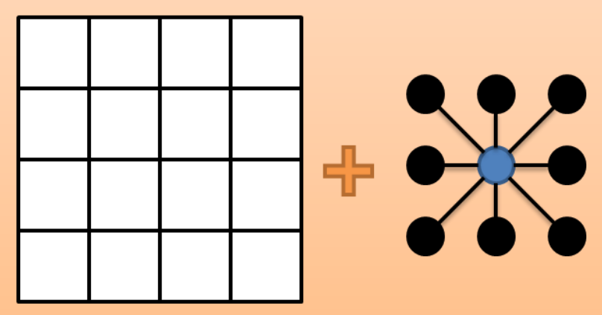


- Graph representation of an image
  - Each pixel is represented by a node
  - Neighborhood relationship between pixels represented by edges
- BFS Graph traversal on an image
  - Propagate information from some pixels to other pixels

Exploring the Algorithm Layer

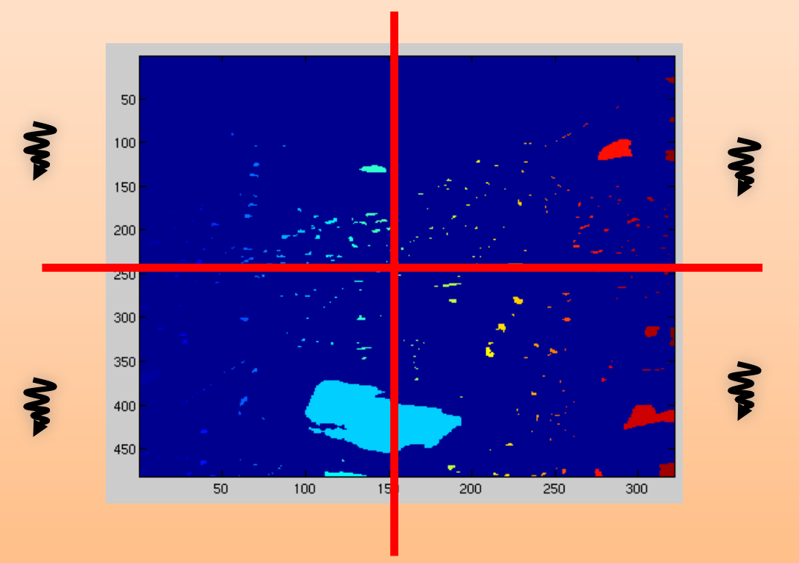
- Traditional BFS graph traversal algorithm
  - Propagate information to nearby neighbors
- Structured grid algorithm
  - Gather information from nearby neighbors

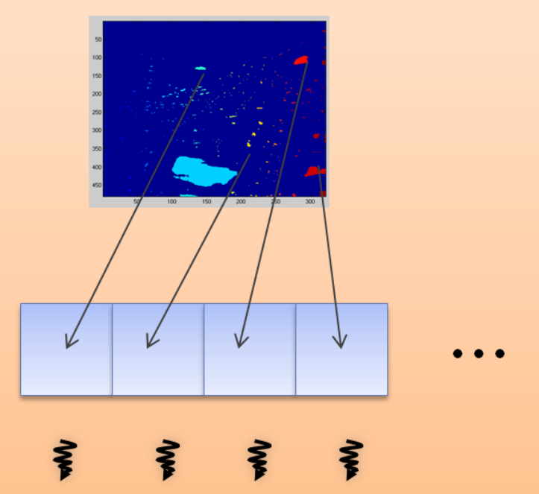




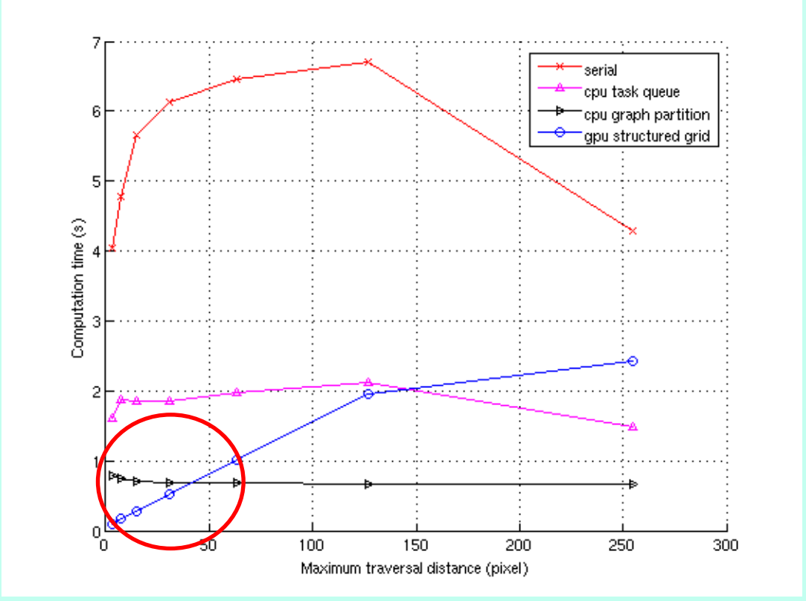
Exploring the Parallelization Strategy Layer

- Graph Partition
- Parallel Task Queue



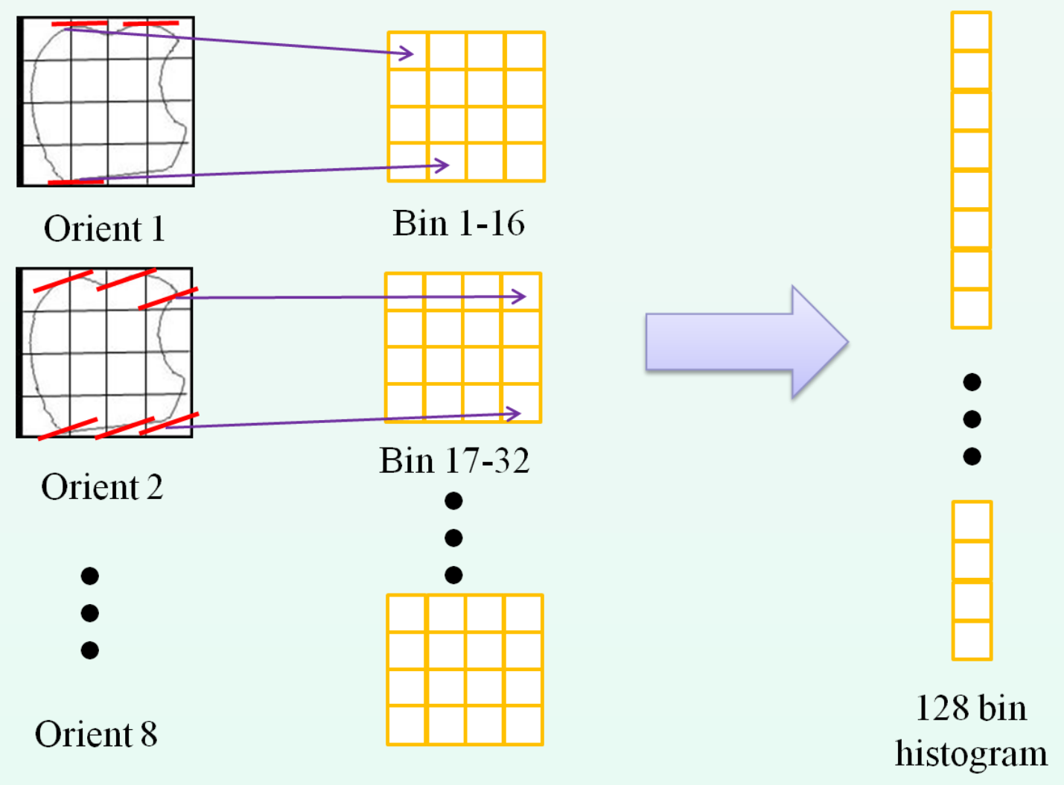


- Explored Design Space
  - Parallel Task queue on Intel Core i7 using OpenMP with 8 threads
  - Graph partition on Intel Core i7 using OpenMP with 8 threads
  - Structured grid on Nvidia GTX 480



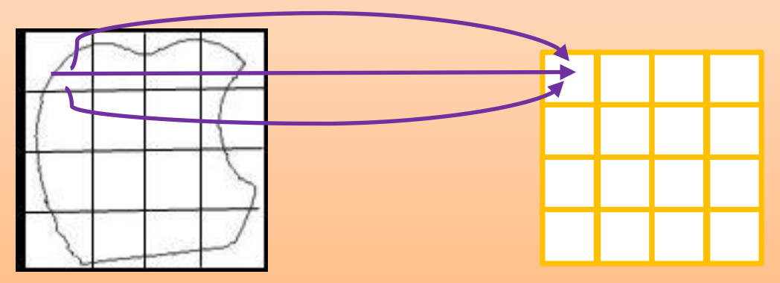
Histogram Kernel

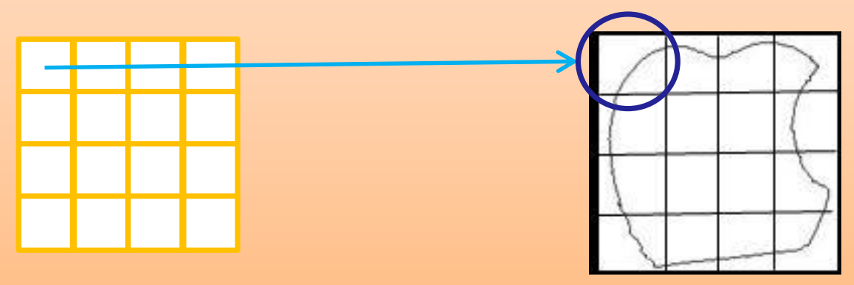
- Use a 128-bin histogram to represent the contour feature of a region
  - The location information is discretized into a 4 x 4 grid
  - The orientation information is discretized into 8 orientations



Exploring the Algorithm Layer

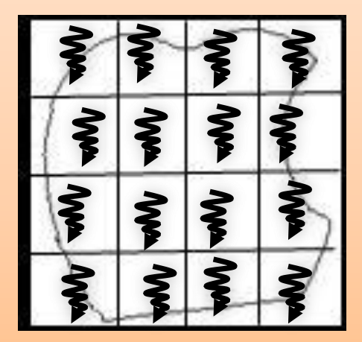
- Data to bins algorithm
  - Each data point atomically accumulate itself into the corresponding histogram bin
- Bins to data algorithm
  - Each bin process its responsible data points

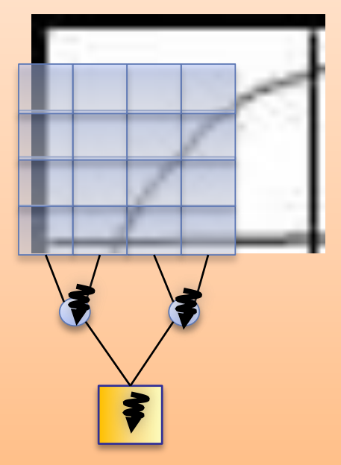




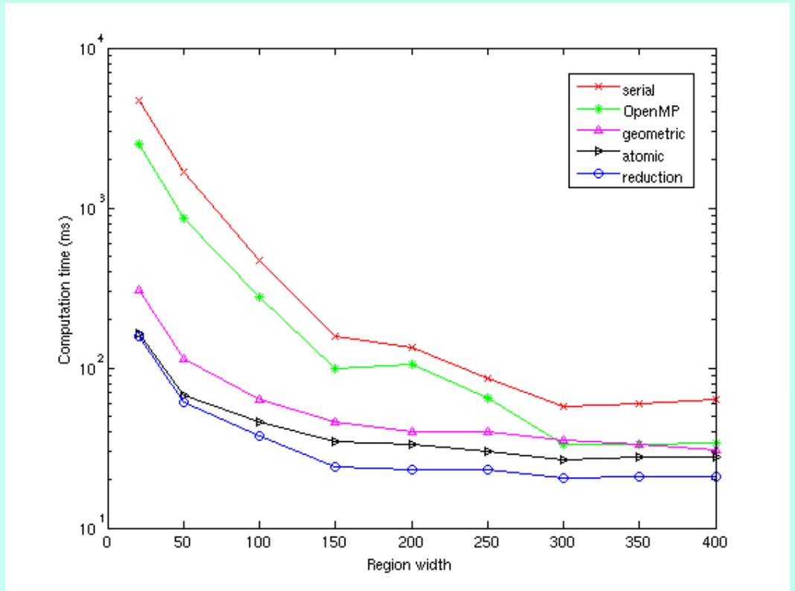
Exploring the Parallelization Strategy Layer

- Geometric Decomposition
- Parallel Reduction



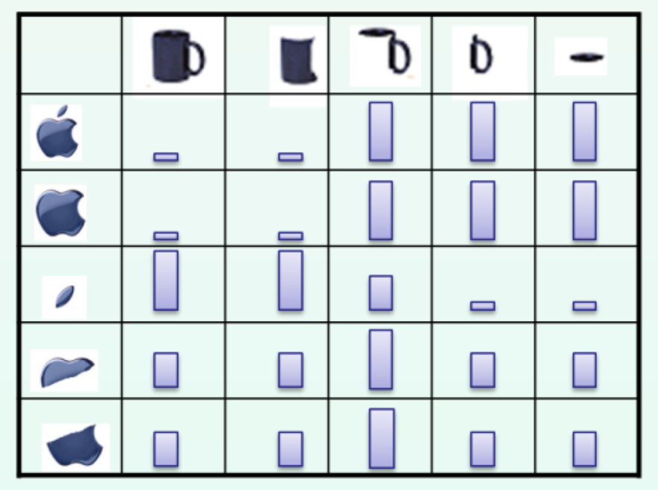


- Explored Design Space
  - Process each region in parallel on Intel Core i7 using OpenMP with 8 threads
  - Geometric decomposition on Nvidia GTX 480
  - Atomic accumulation algorithm on Nvidia GTX 480
  - Parallel reduction on Nvidia GTX 480



Pair-wise χ^2 Distance Kernel

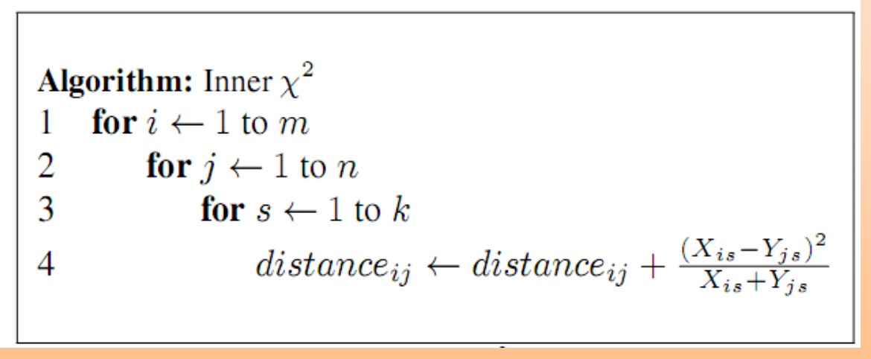
- Compute χ^2 Distance between each pair of regions
  - Similar region pairs have shorter distances
  - Different region pairs have longer distances



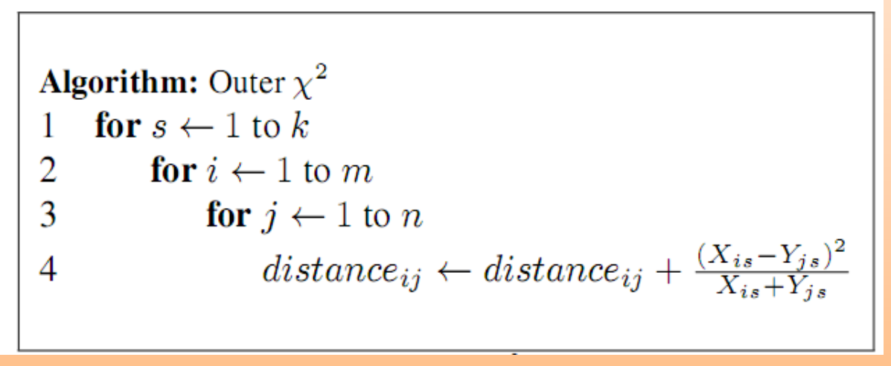
$$\chi^2(x,y) = \frac{1}{2}\sum_i \frac{(x_i - y_i)^2}{x_i + y_i}$$

Exploring the Algorithm Layer

- Inner χ^2 Distance
- Outer χ^2 Distance



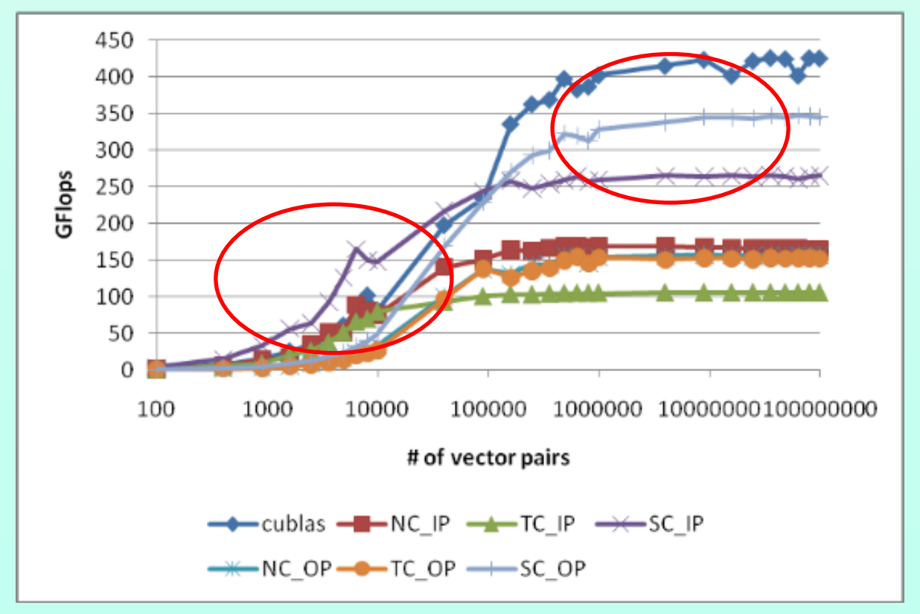
```
Algorithm: Inner $\chi^2$
1  for $i \leftarrow 1$ to $m$
2      for $j \leftarrow 1$ to $n$
3          for $s \leftarrow 1$ to $k$
4              $distance_{ij} \leftarrow distance_{ij} + \frac{(X_{is}-Y_{js})^2}{X_{is}+Y_{js}}$
```



```
Algorithm: Outer $\chi^2$
1  for $s \leftarrow 1$ to $k$
2      for $i \leftarrow 1$ to $m$
3          for $j \leftarrow 1$ to $n$
4              $distance_{ij} \leftarrow distance_{ij} + \frac{(X_{is}-Y_{js})^2}{X_{is}+Y_{js}}$
```

Exploring the Platform Layer

- Cache Mechanisms
  - No Cache
  - Hardware Controlled Cache (Texture Memory)
  - Software Controlled Cache (Shared Memory)



Future Work

- Develop Frameworks for Object Recognition Key Computations
  - Automate the design space exploration procedure
- Integration with the Par Lab stack
  - Use SEJITS to explore the parallelization strategy layer
  - Use autotuners to explore the platform layer