

Exploring the Design Space of a Parallel Object Recognition System

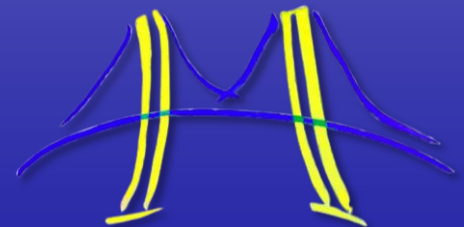
Bor-Yiing Su, subrian@eecs.berkeley.edu

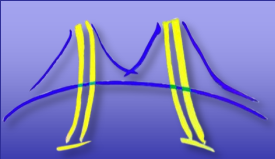
Tasneem G. Brutch, t.brutch@samsung.com

Kurt Keutzer, keutzer@eecs.berkeley.edu

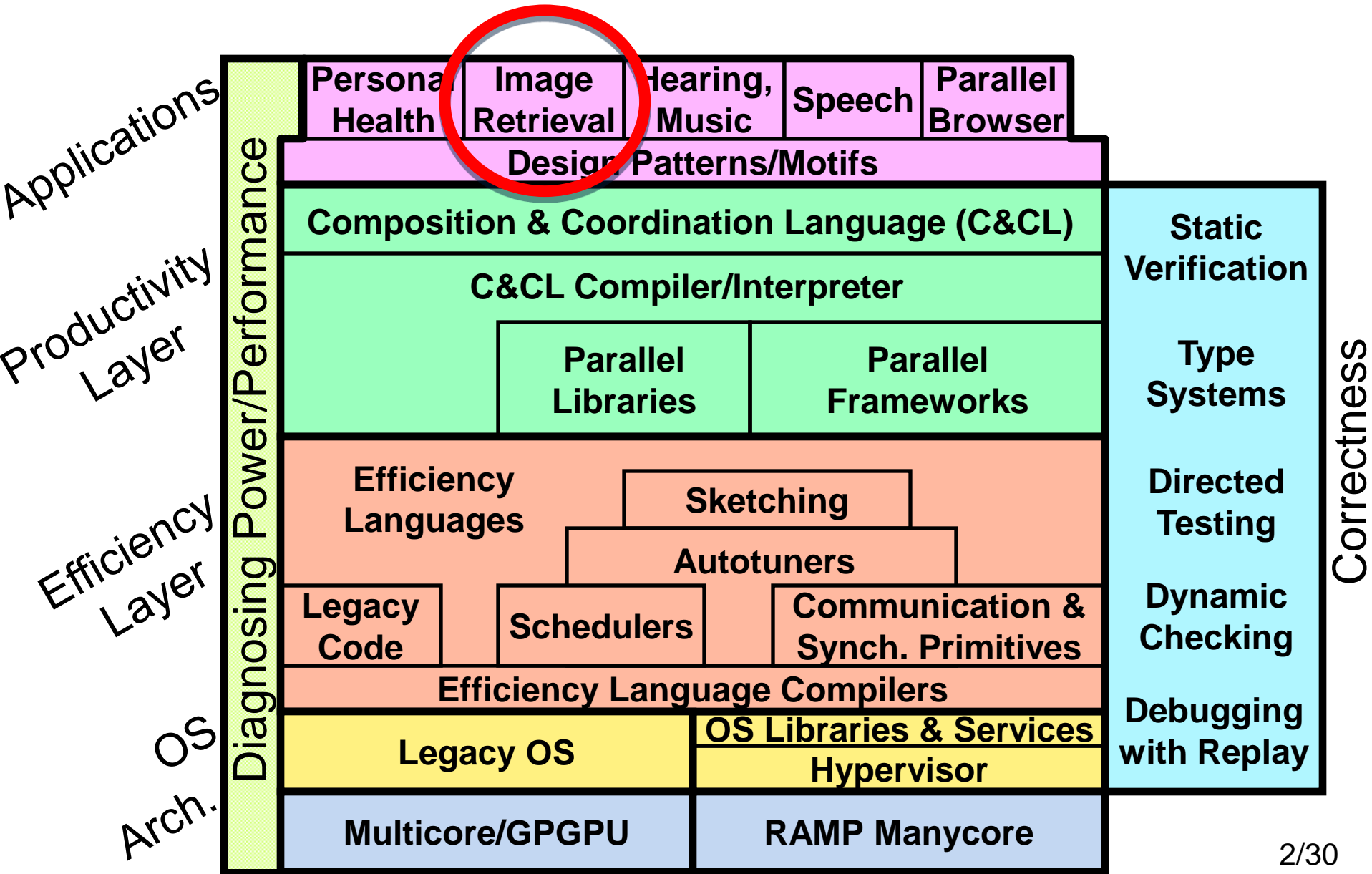


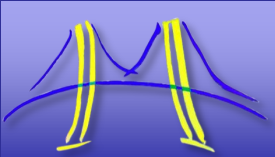
Parallel Computing Lab,
University of California, Berkeley





Category of This Work

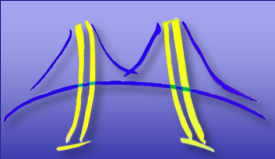




What's New?

- Exploring more design space to further optimize key kernels in the object recognition system
 - Resulting in performance boosts:
 - Training: from 77.8x to 115x
 - Classification: from 72.5x to 119x
- Propose plans of developing frameworks for automating the procedure of design space exploration on object recognition key kernels

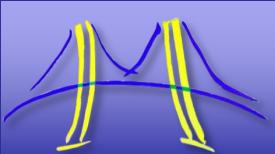
Bor-Yiing Su, Tasneem Brutch, Kurt Keutzer, "A Parallel Region Based Object Recognition System," in *IEEE Workshop on Applications of Computer Vision (WACV 2011)*, Hawaii, January 2011



Outline

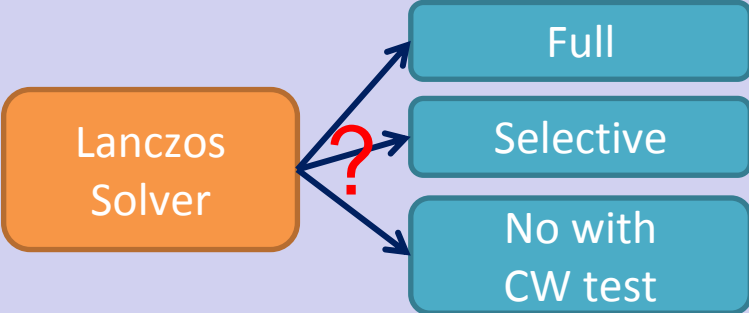
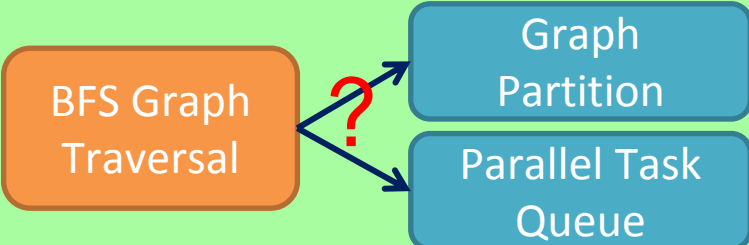
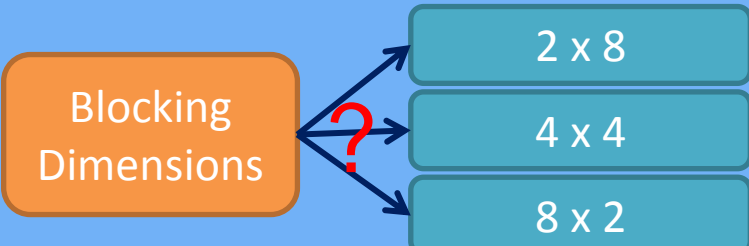


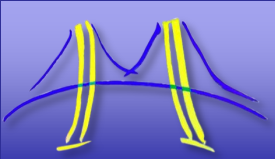
- Design Space
- An Object Recognition System
- Exploring the Design Space of the Object Recognition system
- Future Work



Three Layers of Design Space

- The design space of parallel applications is composed of three layers

Design Space	Explanation	Example
Algorithm Layer	Using different ways to transform same inputs into same or similar outputs	
Parallelization Strategy Layer	Using different strategies to parallelize the same algorithm	
Platform Layer	Using specific hardware features to optimize the same parallelization strategy	



Statements

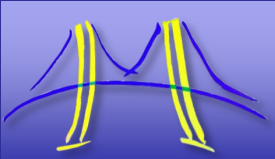
1. **Exploring the design space** is necessary to achieve high performance on a hardware platform of choice
2. **Take advantage of domain knowledge** is necessary to understand trade-offs among different parallelization methods and achieve peak performance

Algorithm Layer

Parallelization
Strategy Layer

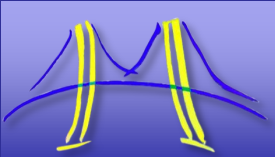
Platform Layer

Design Space



Outline

- Design Space
- An Object Recognition System
- Exploring the Design Space of the Object Recognition system
- Future Work



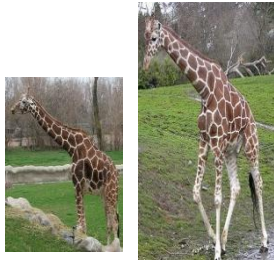
Object Recognition System

Trained Categories

Bottles



Giraffes



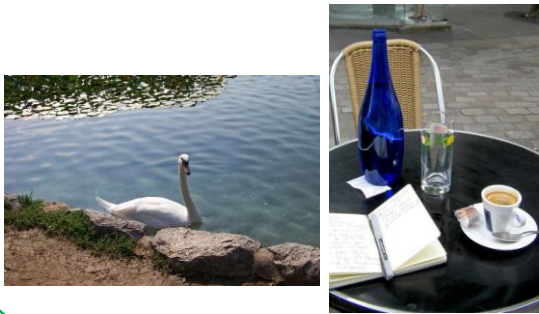
Mugs



Swans

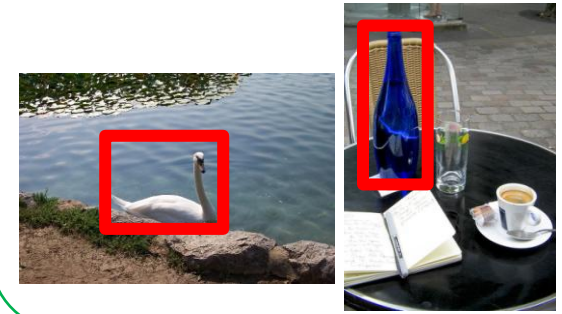


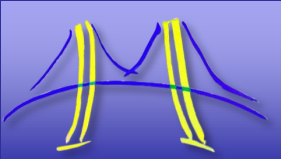
Image Queries



Object
Recognition
System

Outputs

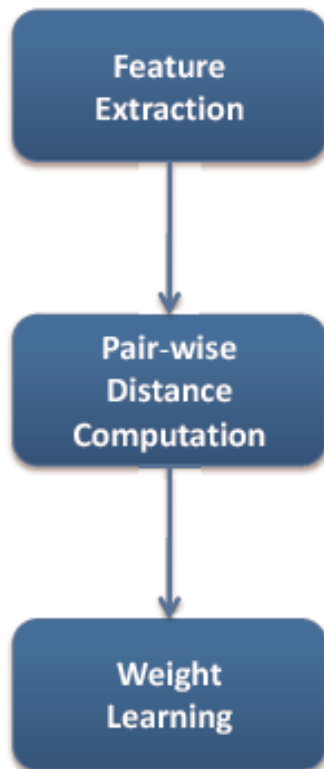




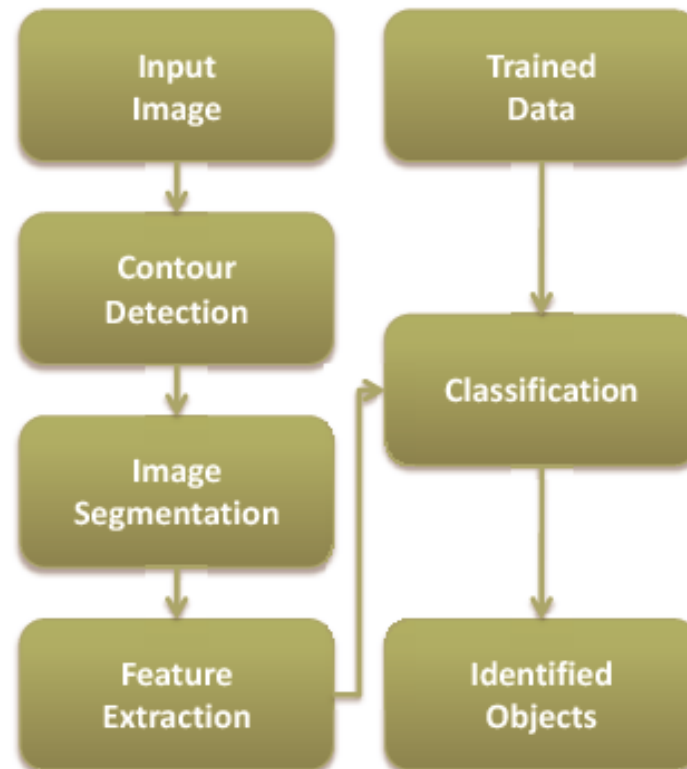
Targeting Object Recognition System

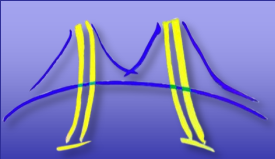
- C. Gu, J. Lim, P. Arbeláez, and J. Malik. Recognition using regions. In *CVPR*, 2009

Training Flow



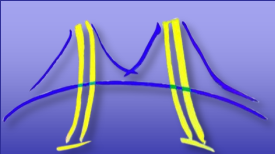
Classification Flow





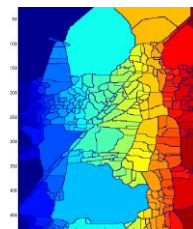
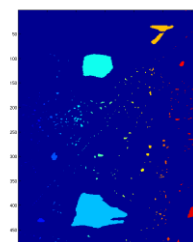
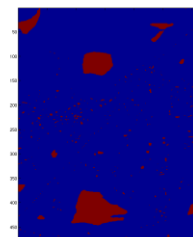
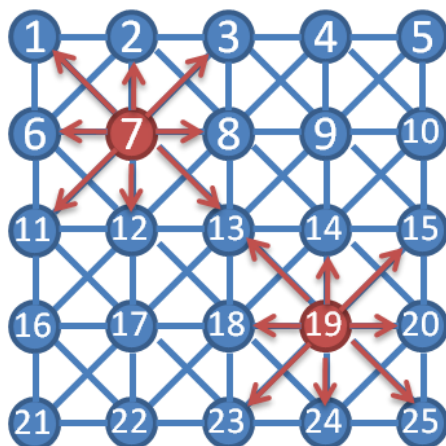
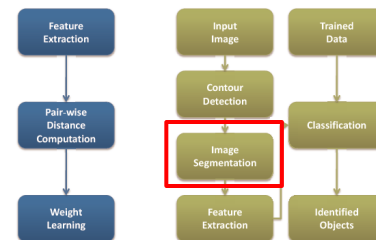
Outline

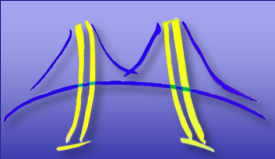
- Design Space
- An Object Recognition System
- ➡ ■ Exploring the Design Space of the Object Recognition system
 - Breadth First Search (BFS) Graph Traversal Kernel
 - Histogram Kernel
 - Pair-wise Distance Kernel
 - Overall Performance
 - Demo
- Future Work



BFS Graph Traversal Kernel

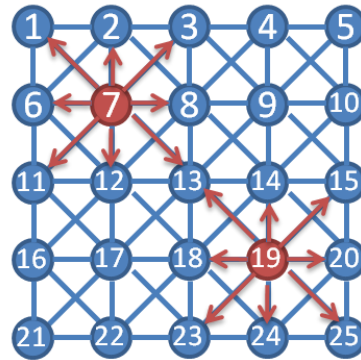
- The image segmentation component heavily relies on the BFS graph traversal kernel
- Image Graph:
 - Nodes represent image pixels
 - Edges represent neighboring relationships
- BFS graph traversal kernel: propagate information from some pixels to other pixels



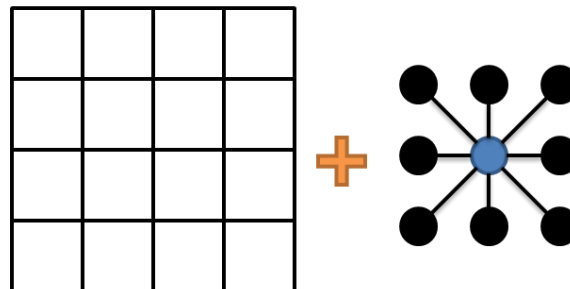


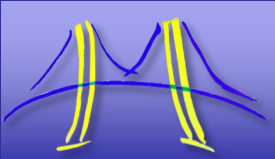
Exploring the Algorithm Layer

- Direct algorithm: Each source node propagates information to its neighbors
 - Traditional BFS graph traversal algorithm



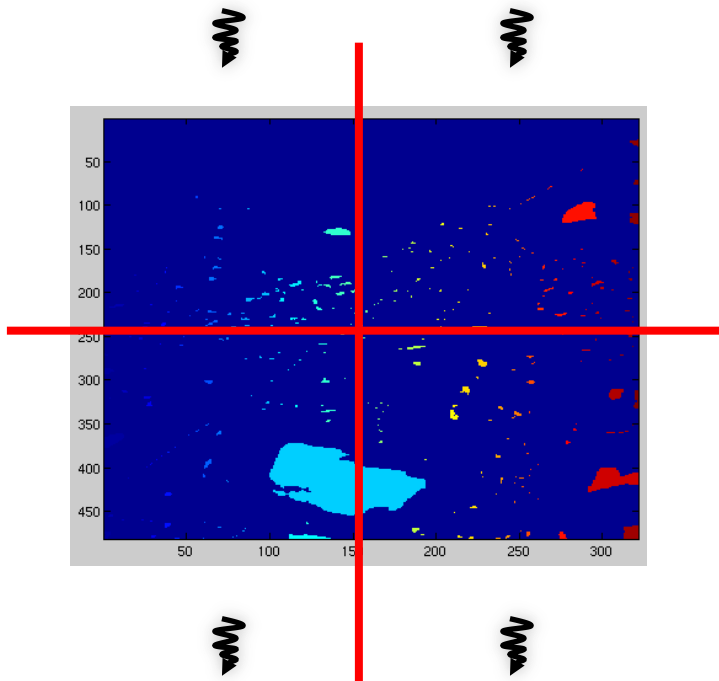
- Reverse algorithm: Each node checks whether it can be updated by one or more neighboring nodes
 - Structured grid computation



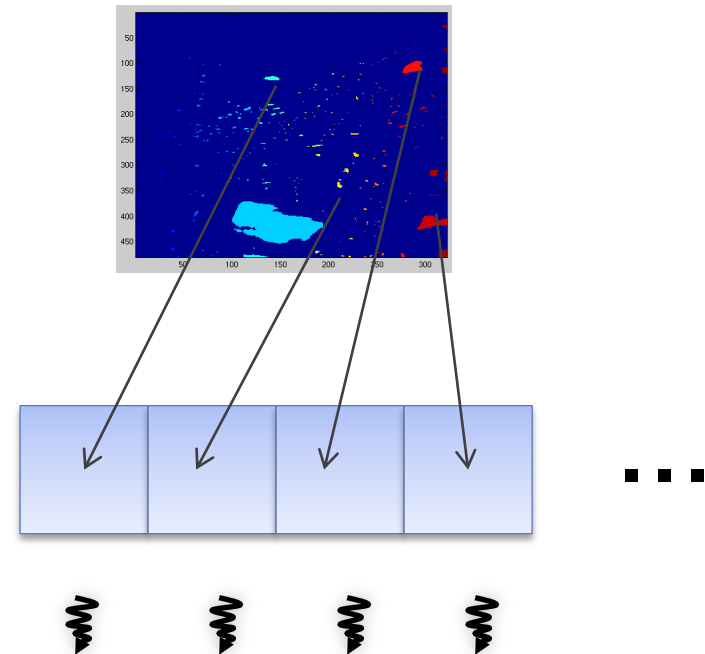


Exploring the Parallelization Strategy Layer

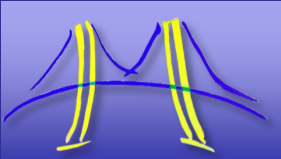
- Two strategies can be used to parallelize the traditional BFS graph traversal algorithm
 - Graph partition
 - Parallel task queue



Graph Partition

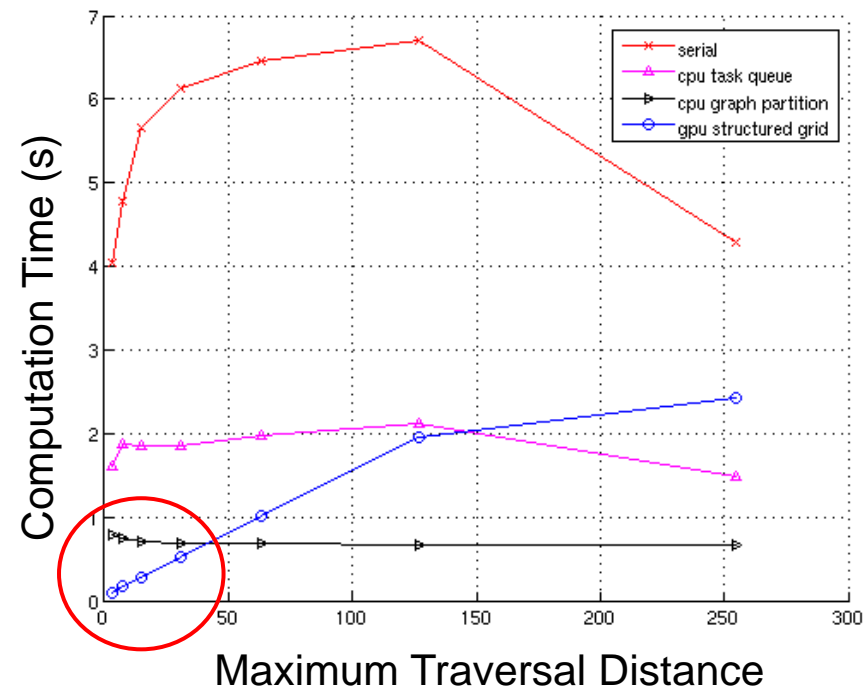
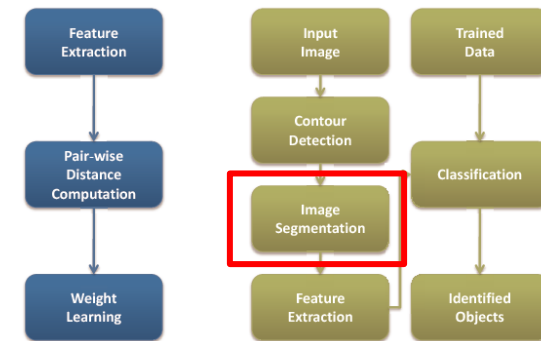


Task Queue



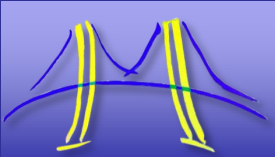
Associating Design Space Exploration with Input Data Properties

- 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



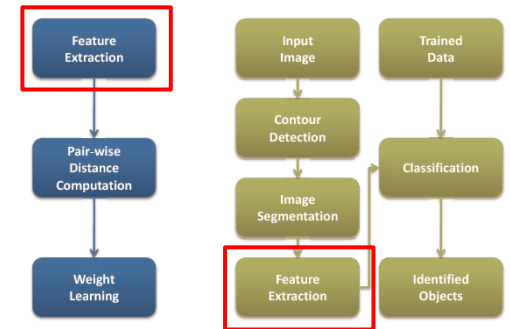
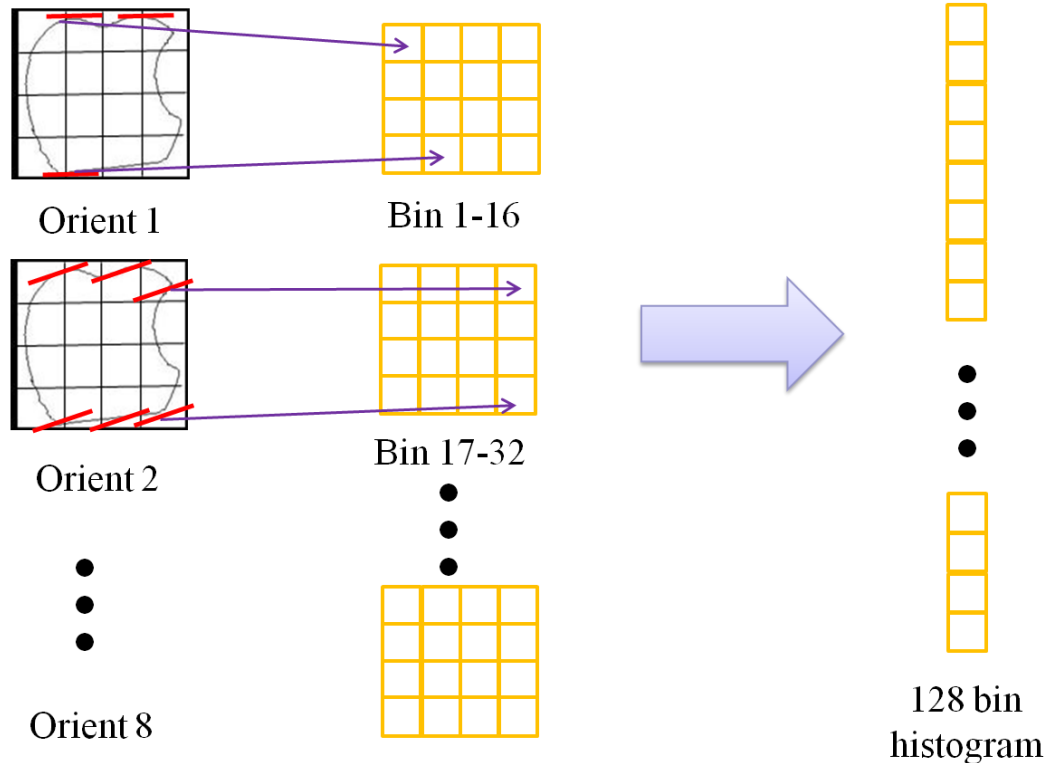
Conclusion:

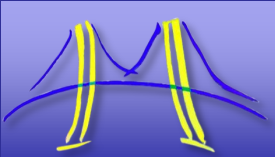
- Use the **structured grid method on a GPU** in our system



Histogram Kernel

- Each image region is represented by its contour features
- The contour feature of a region is represented by a 128-bin histogram





Exploring the Algorithm Layer

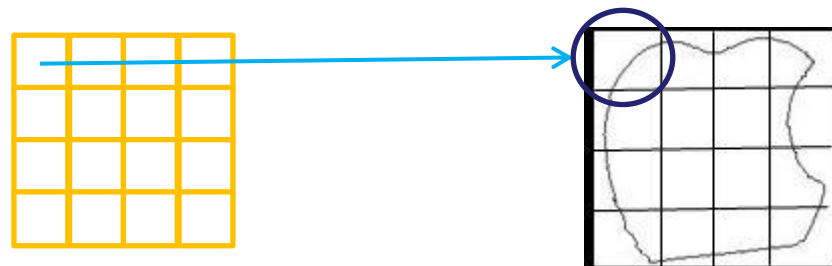
- Data to bins algorithm:
 - Each data point atomically accumulate itself into the corresponding histogram bin

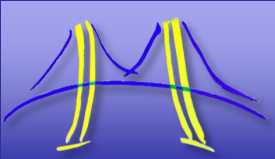
```
foreach pixel  $p$   
  accumulate  $p$  into bin  $b$ 
```



- Bins to data algorithm:
 - Each bin process its responsible data points

```
foreach histogram bin  $b$   
  process pixels  $p_1 \dots p_n$ 
```



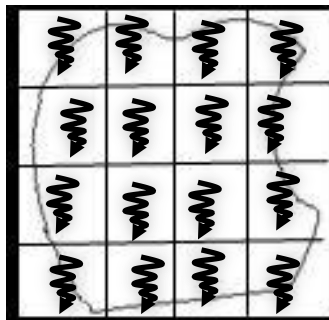


Exploring the Parallelization Strategy Layer

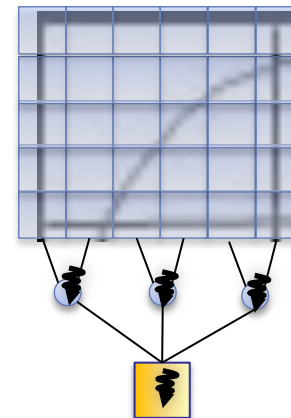
- Process each region in parallel



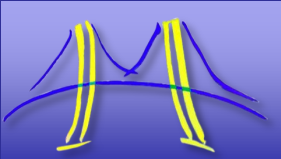
- When dealing with one region, two strategies can be used to parallelize the bins to data algorithm
 - Geometric decomposition: Process each histogram bin in parallel
 - Parallel reduction: For a histogram bin, accumulate its corresponding data points by parallel reduction



Geometric Decomposition



Parallel Reduction

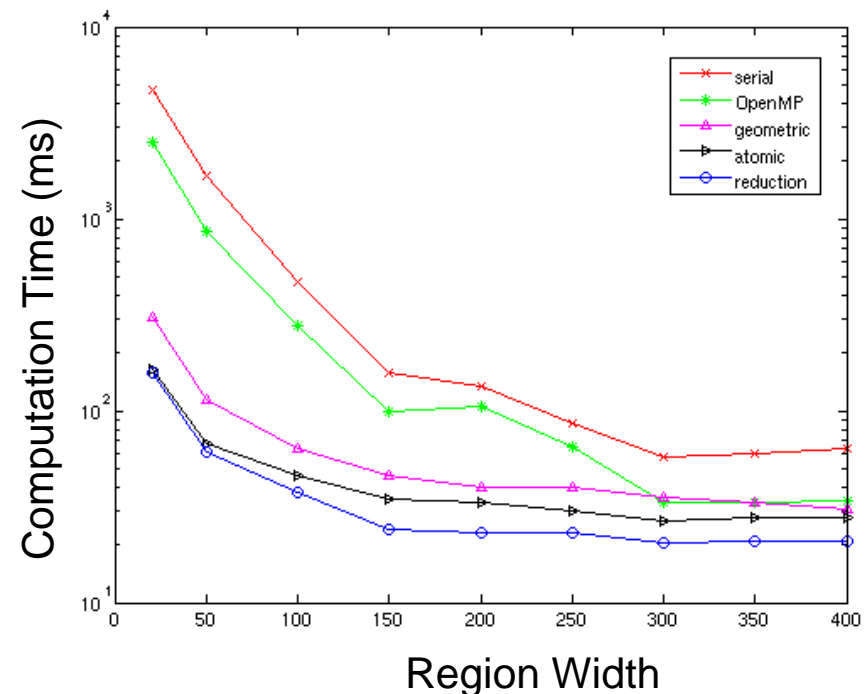
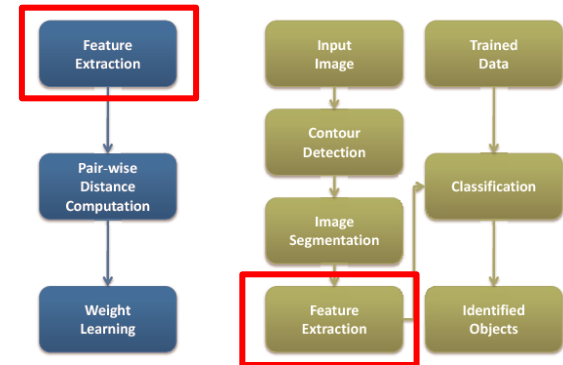


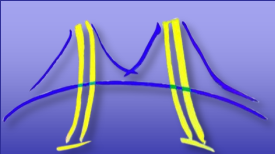
Associating Design Space Exploration with Input Data Properties

- 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

Conclusion:

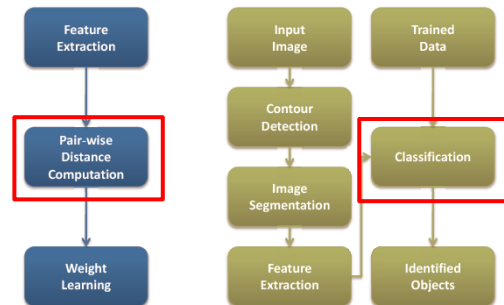
- Use the **parallel reduction method on a GPU** in our system





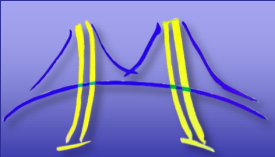
Pair-wise χ^2 Distance Kernel

- In both the training stage and the classification stage, we need to compute the pair-wise distance between two region sets
 - Similar regions have shorter distance
 - Different regions have longer distance
- It is a matrix matrix multiplication computation
 - Replacing dot product into χ^2 distance



Definition of the χ^2 distance:

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



Exploring the Design Space

■ Algorithm Layer

■ Inner χ^2 distance

Algorithm: Inner χ^2

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

■ Outer χ^2 distance

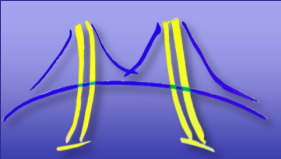
Algorithm: Outer χ^2

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

■ Platform Layer

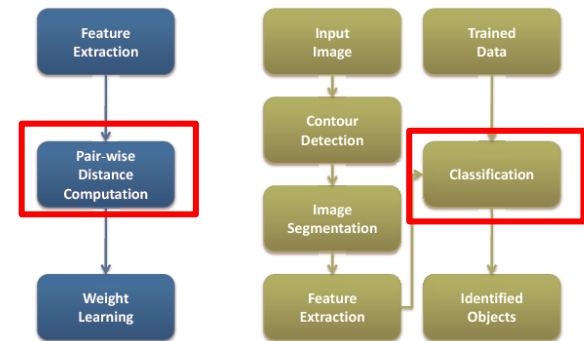
■ Cache Mechanisms

- No Cache
- Hardware Controlled Cache (Texture memory on GPU)
- Software Controlled Cache (Shared memory on GPU)



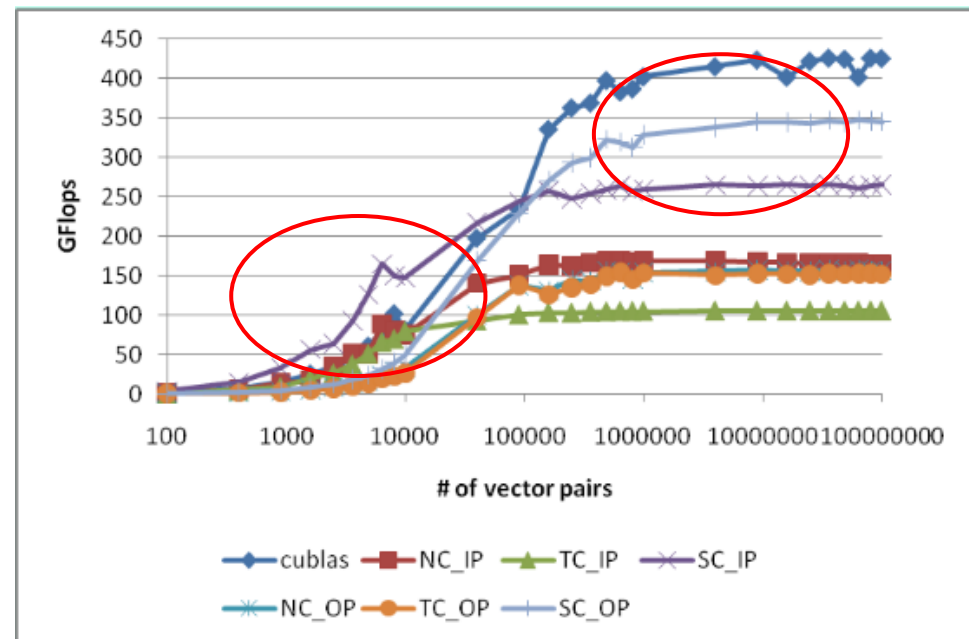
Associating Design Space Exploration with Input Data Properties

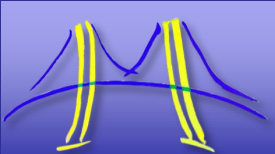
- Explored Design Space
 - The combination of two algorithms and three cache mechanisms on Nvidia GTX 480



Conclusion:

- Use the **outer χ^2 distance method** with software controlled cache in the **training stage**
- Use the **inner χ^2 distance method** with software controlled cache in the **classification stage**





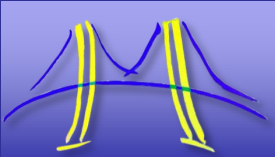
Overall Performance: Speedups

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

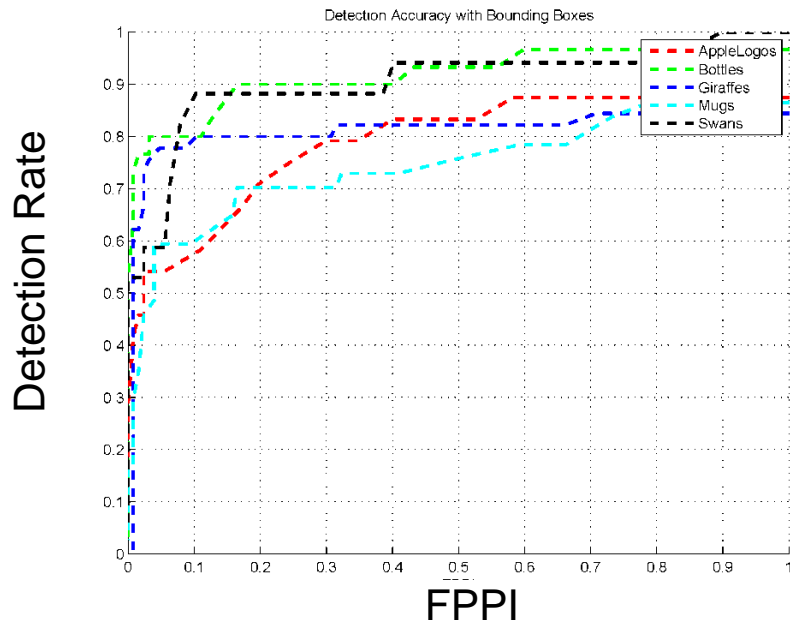
Training

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

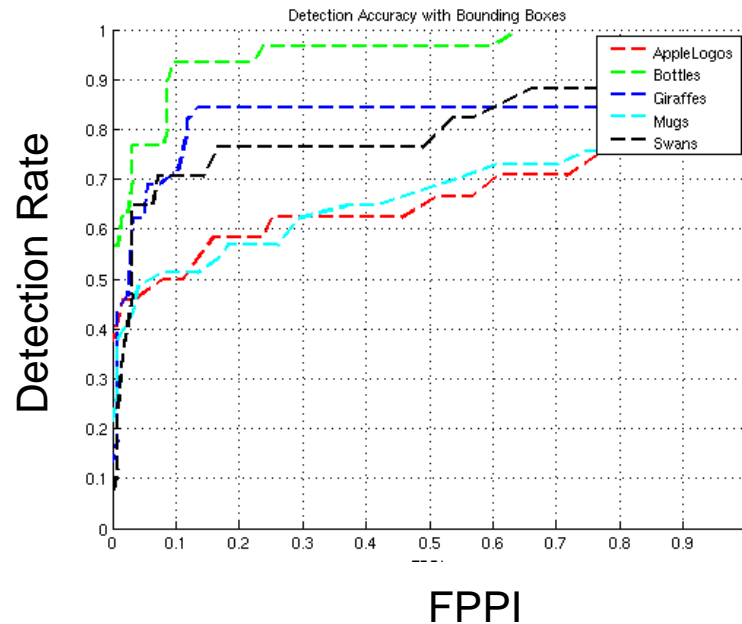
Classification



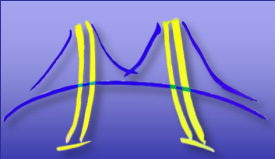
Overall Performance: Detection Accuracy



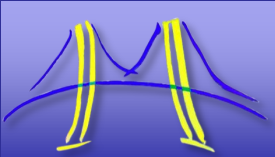
Serial



Parallel



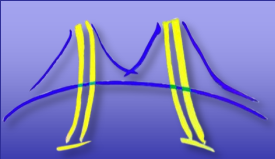
Demo



Conclusion

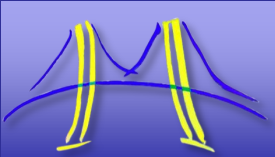
- Exploring the design space is necessary to achieve high performance on a hardware platform of choice
- Take advantage of domain knowledge is necessary to understand trade-offs among different parallelization methods and achieve peak performance
- We have developed a parallel object recognition system with comparable detection accuracy while achieving 110x-120x times speedup
- Work presented at Workshop on Applications of Computer Vision 2011

Bor-Yiing Su, Tasneem Brutch, Kurt Keutzer, "A Parallel Region Based Object Recognition System," in *IEEE Workshop on Applications of Computer Vision (WACV 2011)*, Hawaii, January 2011



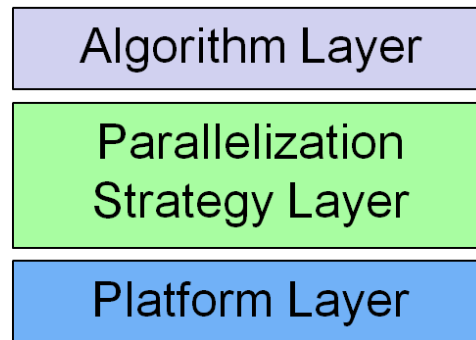
Outline

- Design Space
- An Object Recognition System
- Exploring the Design Space of the Object Recognition system
- ➡ ■ Future Work
 - Develop Frameworks for Object Recognition Key Computations
 - Integration with the Par Lab stack



Frameworks for Computer Vision

- Design space exploration is a very time consuming procedure
 - We need to develop frameworks to automate design space exploration



- What frameworks to develop?

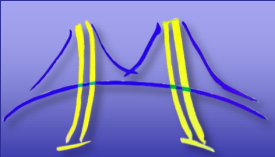
Object Recognition Patterns

Structures

Window Sliding
Pyramid Image Scaling
Region-Based Processing
Image-Based Processing
Pair-wise Vector
Processing

Computations

K-means	Histogram	Hough Transform
Mean-Shift	Accumulation	Eigen Decomposition
Agglomerative	Convolution	Quadratic
Vector Distance	Pixel-wise Graph	Programming
	Traversal	



Framework of Pair-wise Distance

User Customization

Defining the distance between two vectors

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

Parallelize Computation

Exploring the design space automatically

Inner χ^2

Outer χ^2

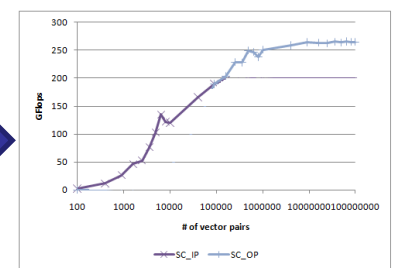
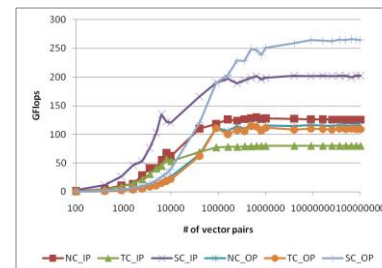
Transpose Matrix

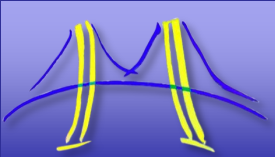
Blocking Dimension

Cache Mechanisms

Optimize Computation

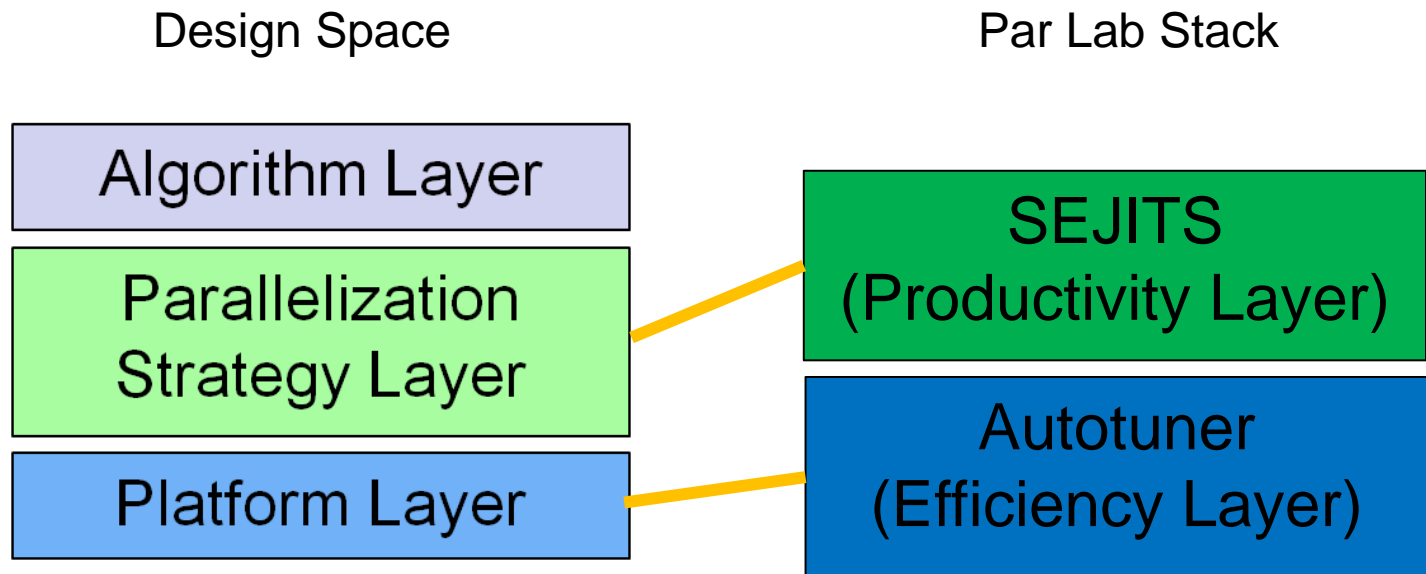
Generating the customized library

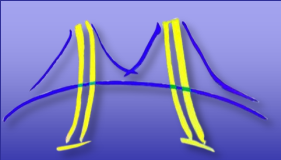




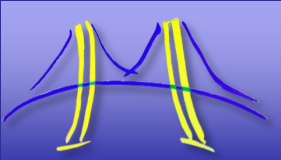
Integration with the Par Lab Stack

- Use the Par Lab stack to develop frameworks for object recognition
 - Use SEJITS from the productivity layer to efficiently express different parallelization strategies
 - Use autotuner from the efficiency layer to explore the design space of the platform layer

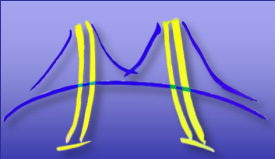




Questions



Backup Slides



Relationship with Our Pattern Language (OPL)

Design Space

Our Pattern Language

