Parallelization of Belief Propagation Method on Embedded Multicore Processors for Stereo Matching

Chi-Hua Lai
Kun-Yuan Hsieh
Shang-Hon Lai
Jenq Kuen Lee

Department of Computer Science
National Tsing-Hua University
Hsinchu, Taiwan

國立清華大學
National Tsing Hua University

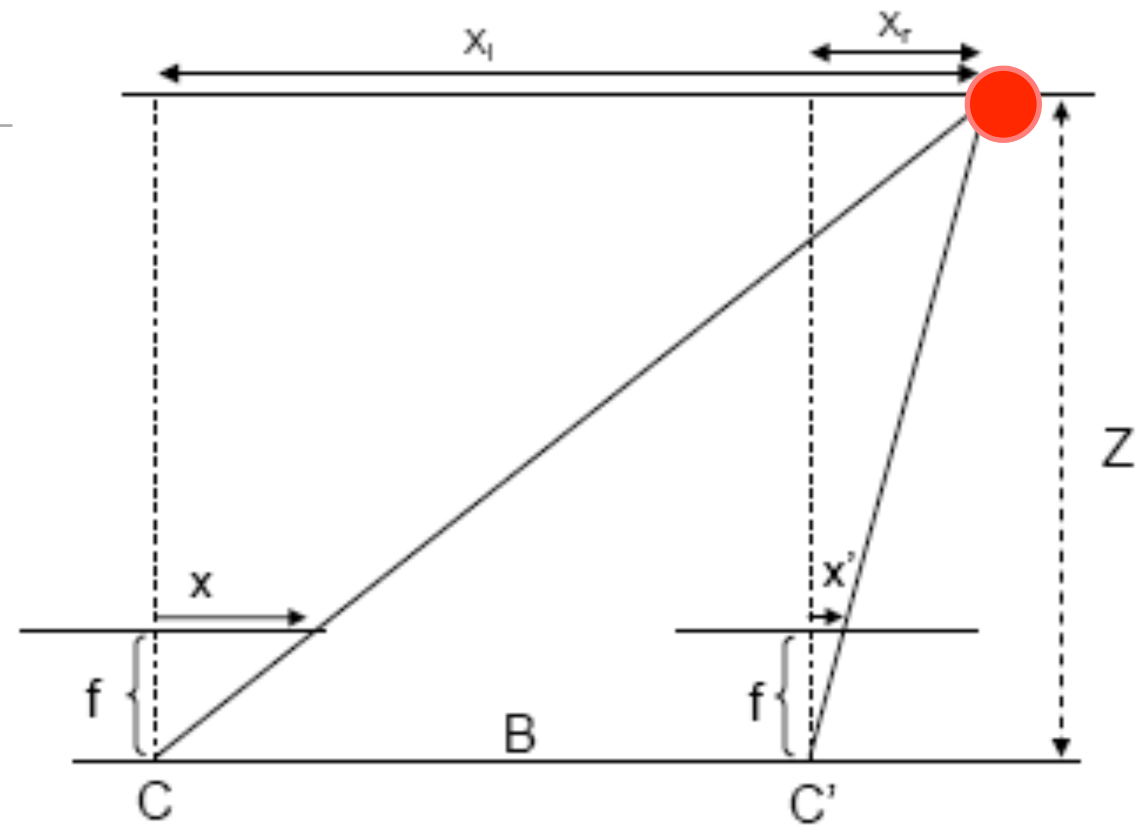
NTHU PL LAB

Outline

- Background and motivation
- Belief propagation(BP) algorithm
- Parallelization opportunities
- Experimental results
- Summary

Stereo Vision

- Compute the depth of 3-D objects through matching 2-D images in the same plane
  - Range information of the environment can help the robot to adapt to the real world
- Human acquires two images of the same objects in different sight to find the distance between the man and the object.
- The geometry model of stereo vision describes the relationship between the eyes and the observed object.



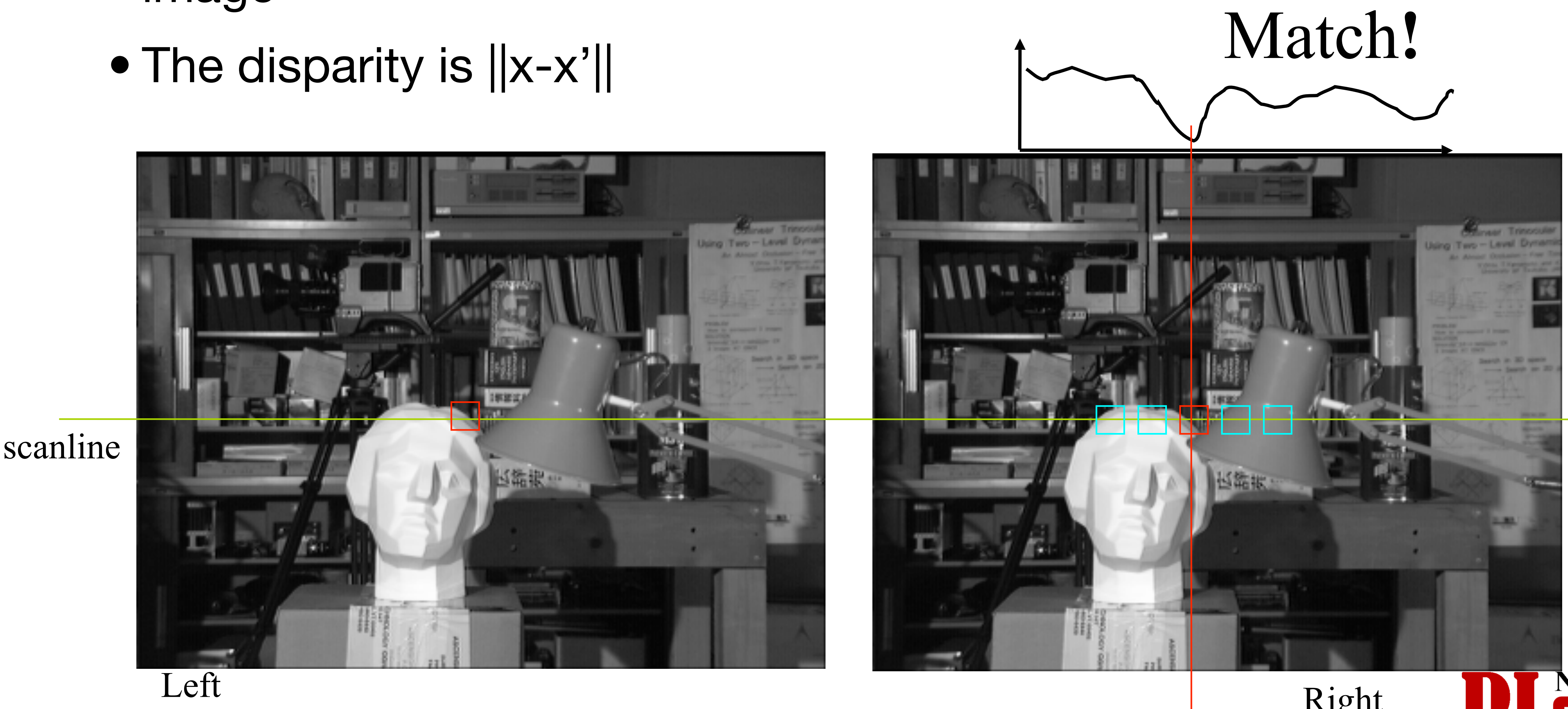
$$\begin{cases} X_l = \frac{x}{f} * Z \\ X_r = \frac{x'}{f} * Z \end{cases}$$

$$X_l - X_r = B$$

$$\Rightarrow Z = \frac{f * B}{(x - x')}$$ (x - x' = disparity)

Introduction to Stereo Matching

- Assume the objects in the two image are in the same scanline
- To find the corespondent points of same objects in the images
  - If the point is in position (x, y) in the left image, (x',y) in the other image
  - The disparity is ||x-x'||

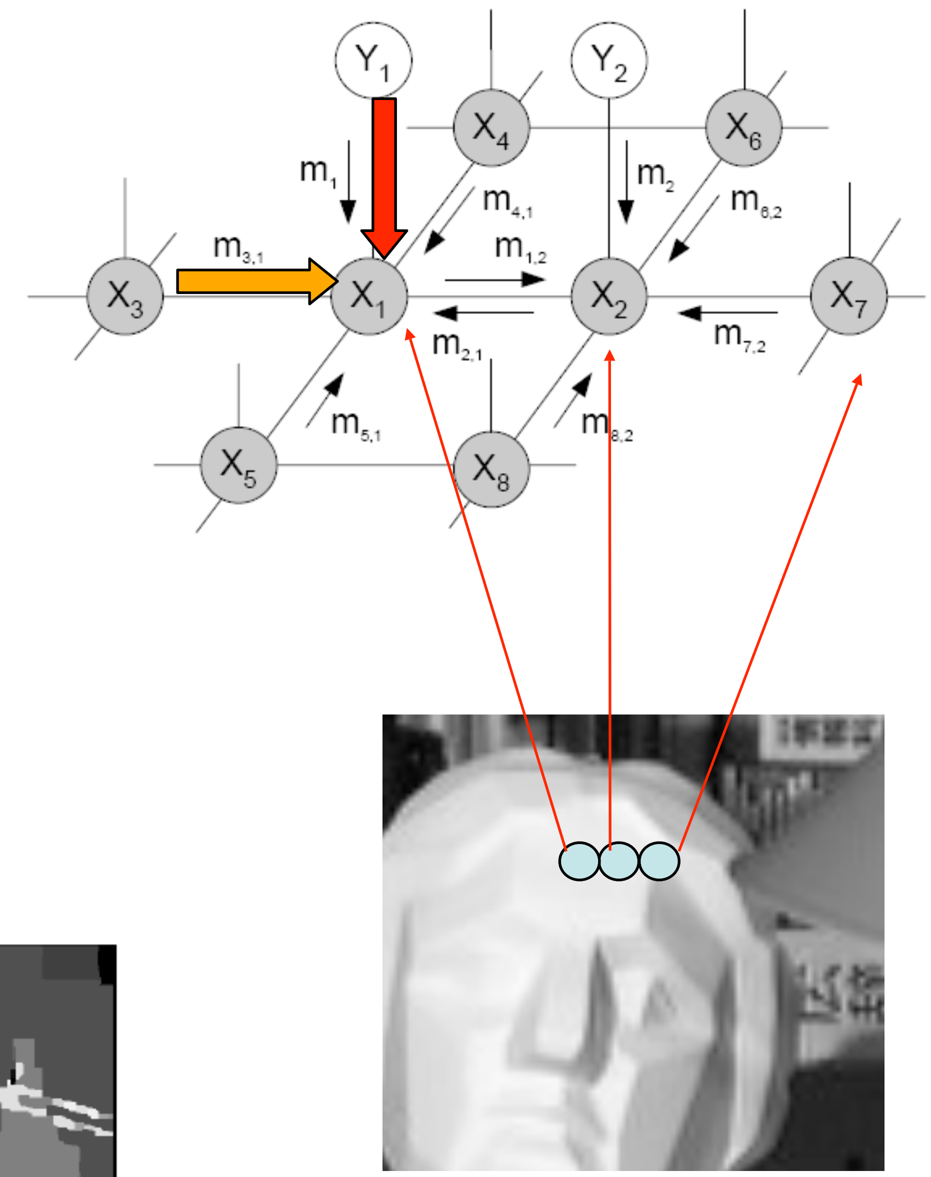


Brief Background: Solving Stereo Matching Problems

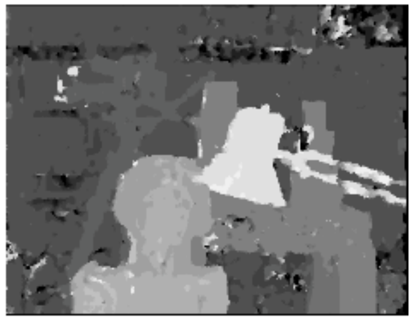
- Two major classes:
  - **Local**
    - Based on correlation, window-based
    - Efficient and suitable for real-time application
    - Have many constraints and results in bad precision
  - **Global**
    - Belief propagation(BP)
    - Retrieving information from the entire image
    - Impressive result, but computation expensive
    - Felzenszwalb proposed a hierarchical method for efficient **BP** method
    - Still not applicable for **real-time** applications

BP: Pixel-labeling Problem

- Pixel-by-pixel
- The goal of labeling is to find minimum of some energy, for stereo problem, it’s the disparity
- Loopy BP, iterative algorithm




0 iter


5 iter


20 iter


100 iter

Hierarchical Belief Propagation

Input: two rectified gray-level pictures // left-side and right side

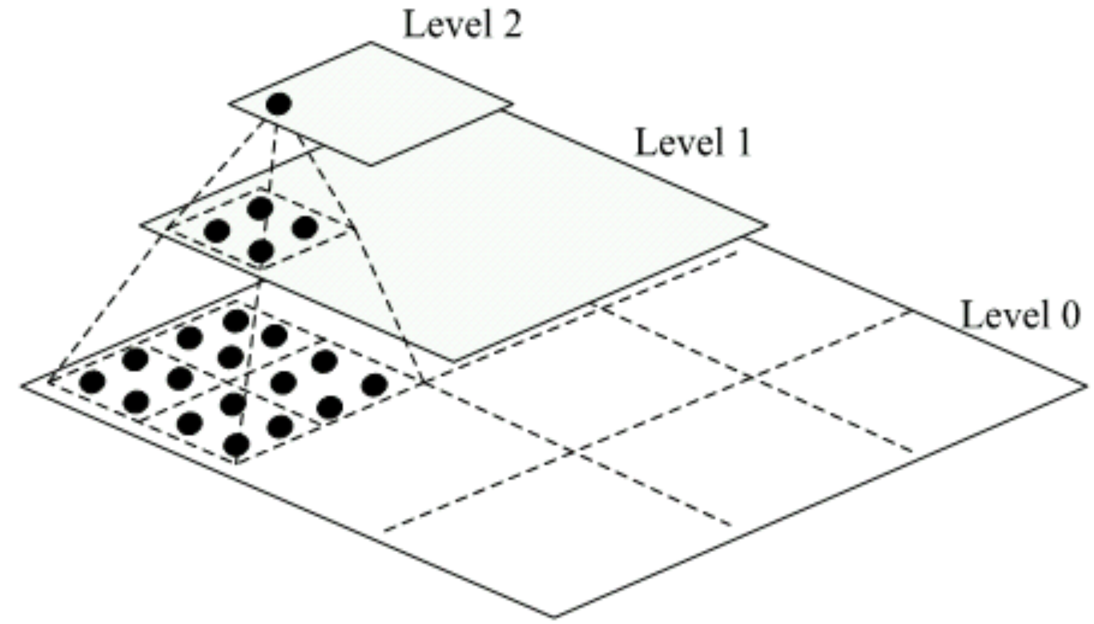
$data^0_{w,h}$ ← difference pixel by pixel between two pictures
// Initial data pyramid

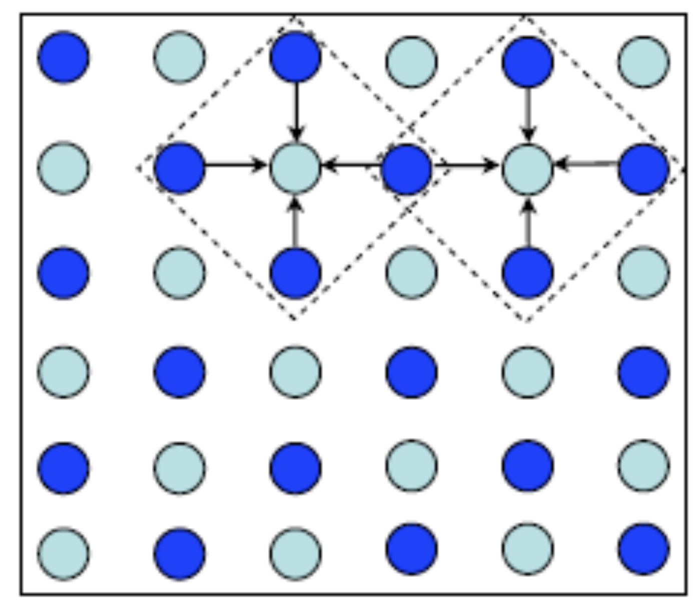
For i ← LEVEL-1 ~ 0 //processing
    If not in top-level
        Initial top-level message layers to 0
    else
        Get message from upper message layers

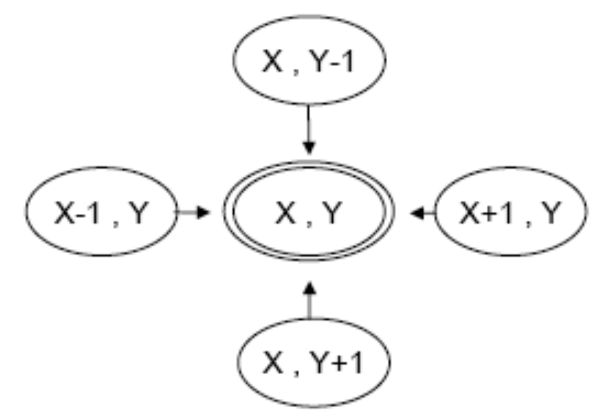
  For t ← 0 ~ ITER-1 //the message deliver iteration
    For y ← 1 ~ height-1
      For x ← (y + t) % 2 ~ width – 1
        Update upward-message of node (x, y)
        Update downward-message of node (x, y)
        Update leftward-message of node (x, y)
        Update rightward-message of node (x, y)
        x=x+2

For y ← 1 ~ height - 1
    For x ← 1 ~ width -1
        accumulate messages delivered by adjacent nodes
        compute disparity of D(x, y)

Output: disparity graph D





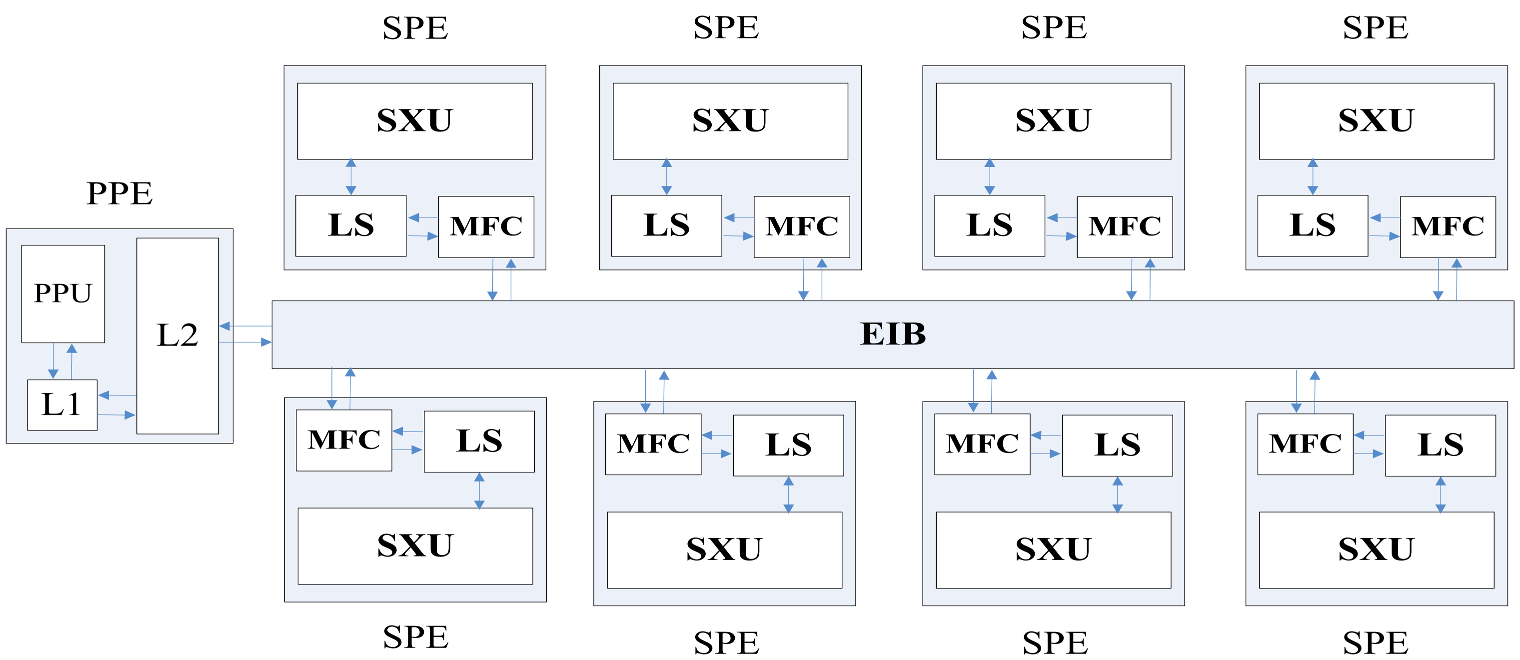


Efficient Belief Propagation for Early Vision. Pedro F. Felzenszwalb and Daniel P. Huttenlocher. International Journal of Computer Vision, Vol. 70, No. 1, October 2006.

NTHU

Hardware Support Parallelism of Cell BE

- 1 PPE and 8 SPE
- DMA for data transferring between PPE and SPE
- SIMD instructions in PPE and SPE
  - Exploit data parallelism
  - Intrinsic provided for the programmers



Hierarchical Belief Propagation

Input: two rectified gray-level pictures // left-side and right side

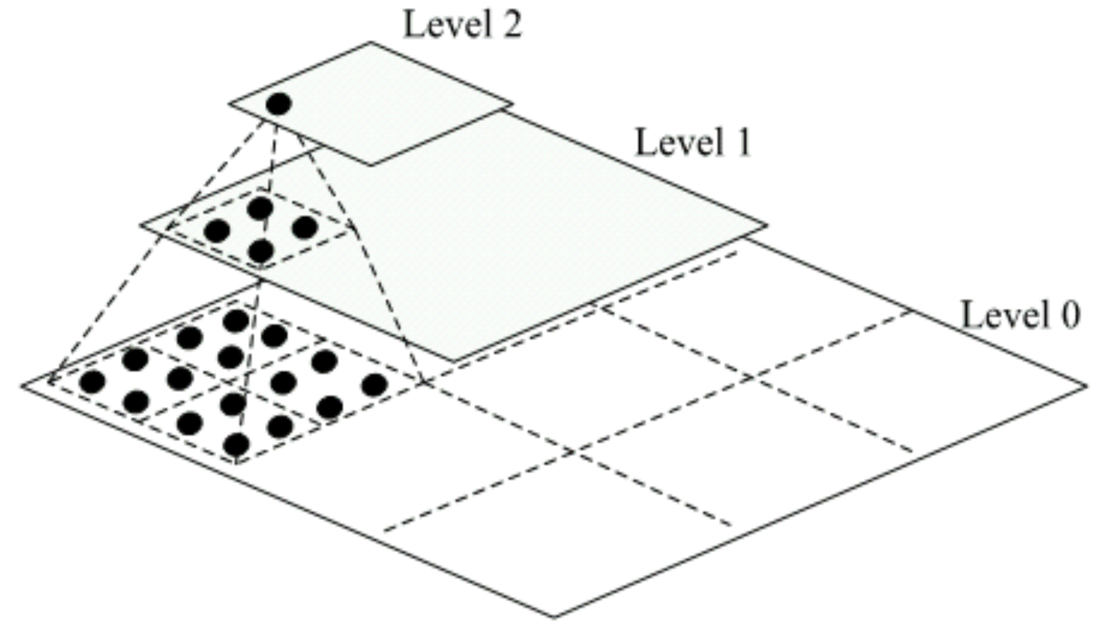
$data^{0}_{w,h}$ ← difference pixel by pixel between two pictures
// Initial data pyramid

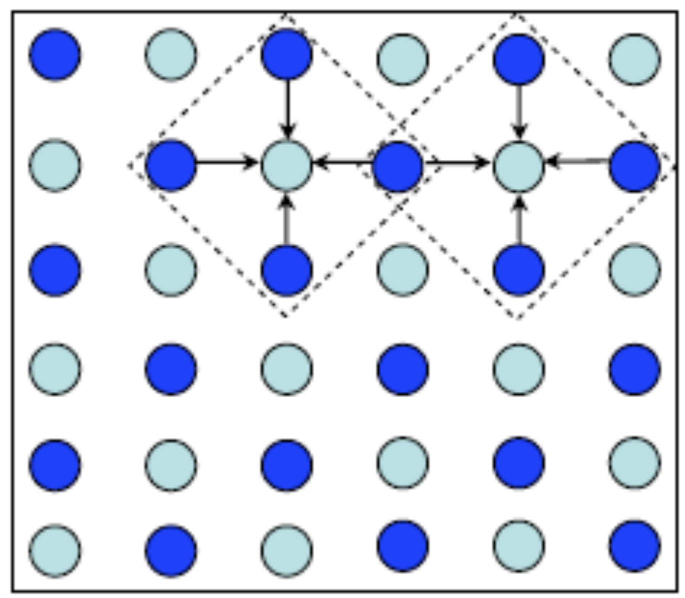
For i ← LEVEL-1 ~ 0 //processing
If not in top-level
Initial top-level message layers to 0
else
Get message from upper message layers

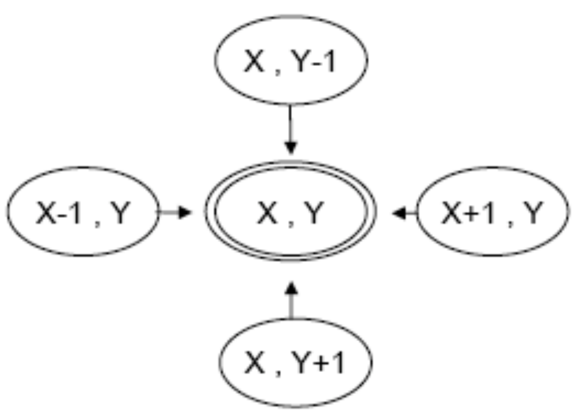
For t ← 0 ~ ITER-1 //the message deliver iteration
For y ← 1 ~ height-1
For x ← (y + t) % 2 ~ width – 1
Update upward-message of node (x, y)
Update downward-message of node (x, y)
Update leftward-message of node (x, y)
Update rightward-message of node (x, y)
x=x+2

For y ← 1 ~ height - 1
For x ← 1 ~ width -1
accumulate messages delivered by adjacent nodes
compute disparity of D(x, y)

Output: disparity graph D

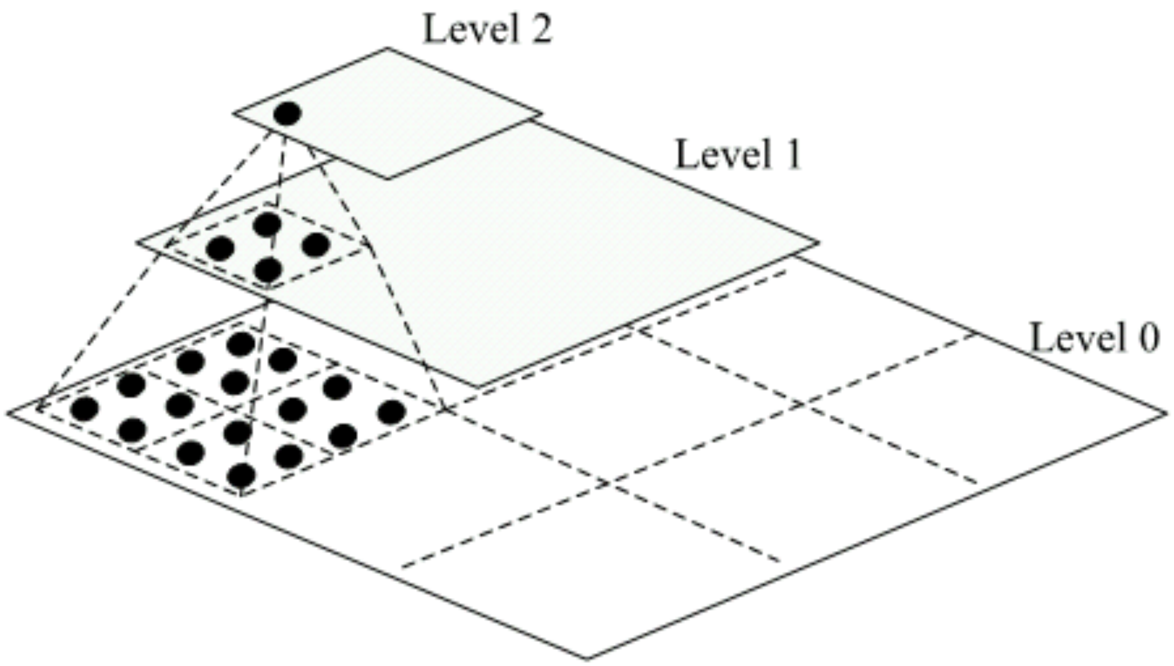






Efficient Belief Propagation for Early Vision. Pedro F. Felzenszwalb and Daniel P. Huttenlocher. International Journal of Computer Vision, Vol. 70, No. 1, October 2006.

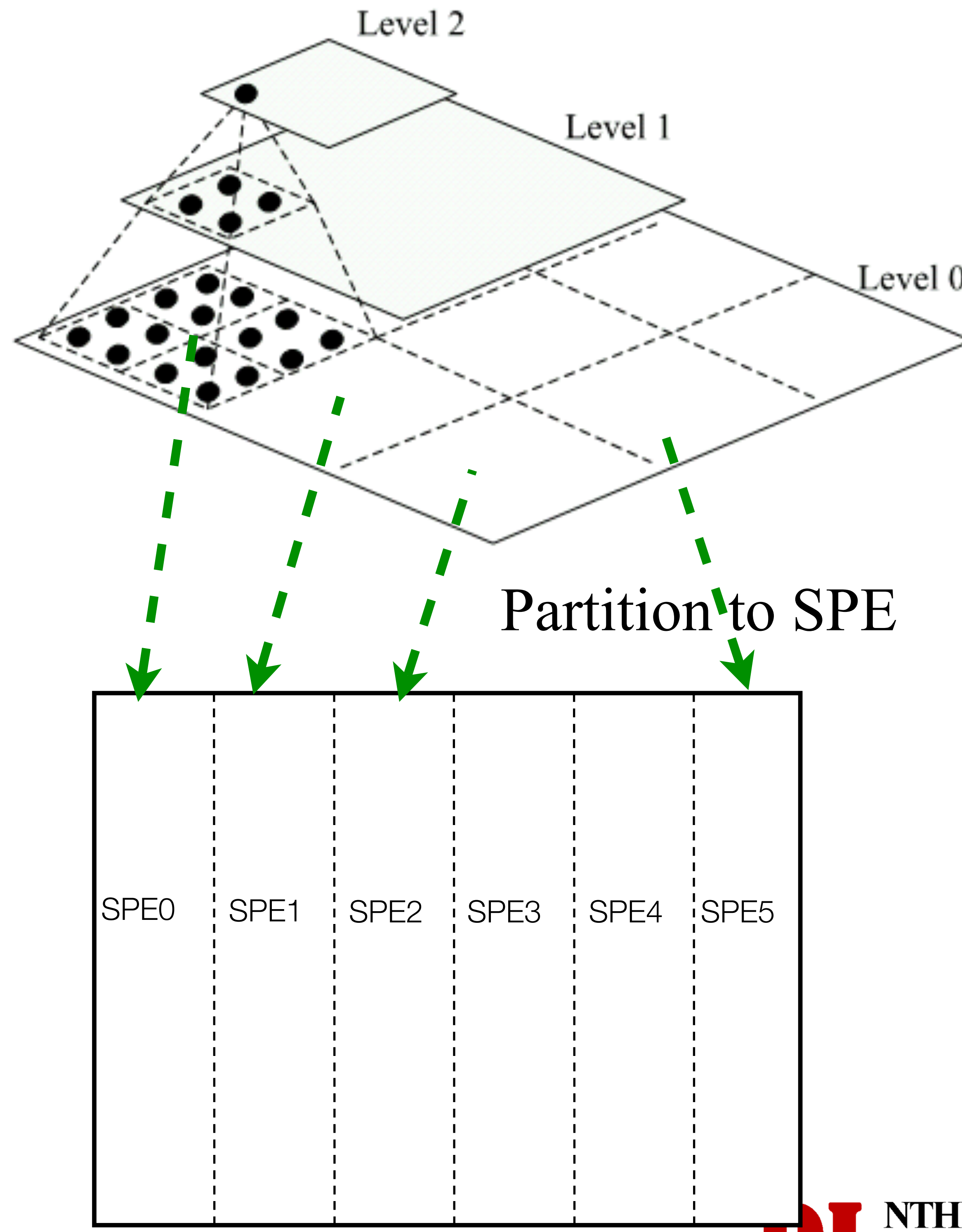
Initialization: Building Data Pyramid



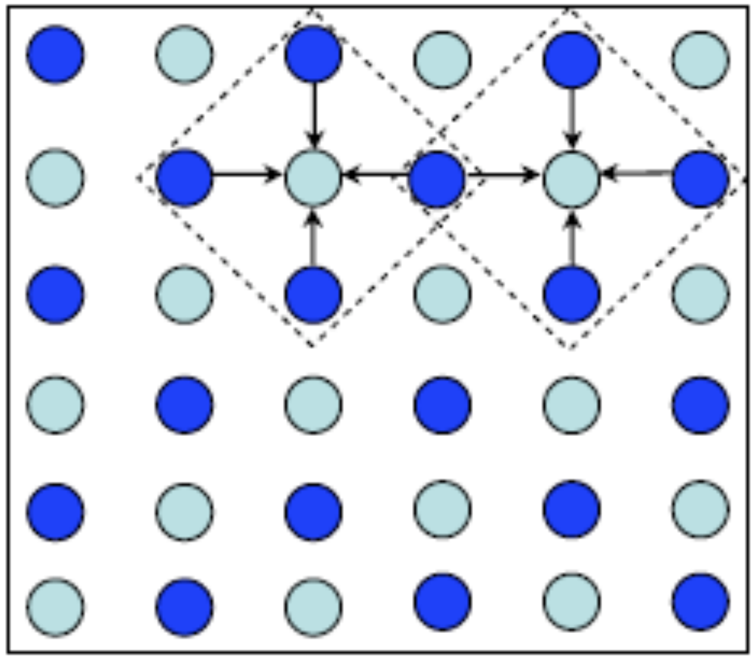
- Constructing the data layer from the input images
- Taking four nodes from the fine-grain layer to build the node to coarser-grain layer
- Considering the generation of each node in the coarsest-grain layer
  - Independent computation
  - **Data parallelism**
- Strategy I:
  - Building each sub data pyramid in parallel
  - Executing on SPEs

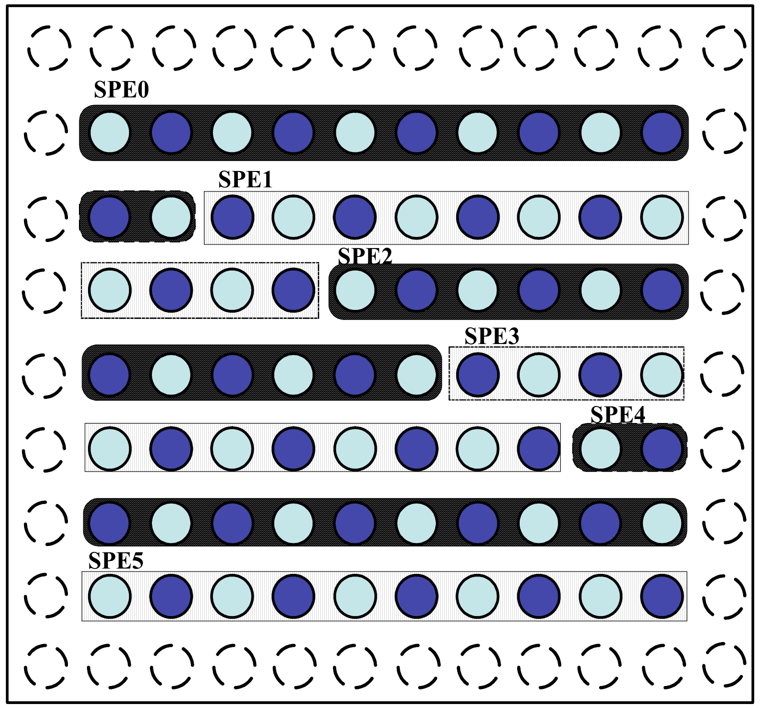
Initialization: Building Data Pyramid

- Constructing the data layer from the input images
- Taking four nodes from the fine-grain layer to build the node to coarser-grain layer
- Considering the generation of each node in the coarsest-grain layer
  - Independent computation
  - **Data parallelism**
- Strategy I:
  - Building each sub data pyramid in parallel
  - Executing on SPEs



Labeling, Message Updating
Finalizing, Calculating the Disparity





- 90% of the computation workload
- Iteratively updating
- Message updating
  - Required the result(message) of the last iteration from the adjacent nodes
  - To update the message to the adjacent nodes
- The updating processes for each node, the four directions, are each iteration is independent
- Strategies:
  - Updating the message in each direction in the SPEs -> bad data reuse
  - Grouping the nodes for each SPE

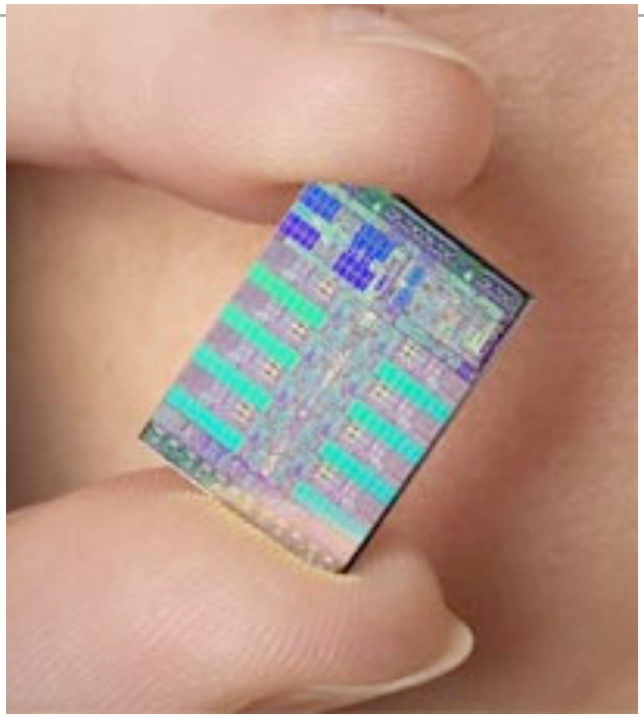
Exploiting Hardware Features

- SIMD
  - Process four data elements simultaneously
- Using DMA by multi-buffering
  - To overlap communication and computation
  - Hiding the data transferring overhead
- Configuring the data layout
  - Aligned to 16 bytes
  - For DMA transferring
  - For SIMD vector operations

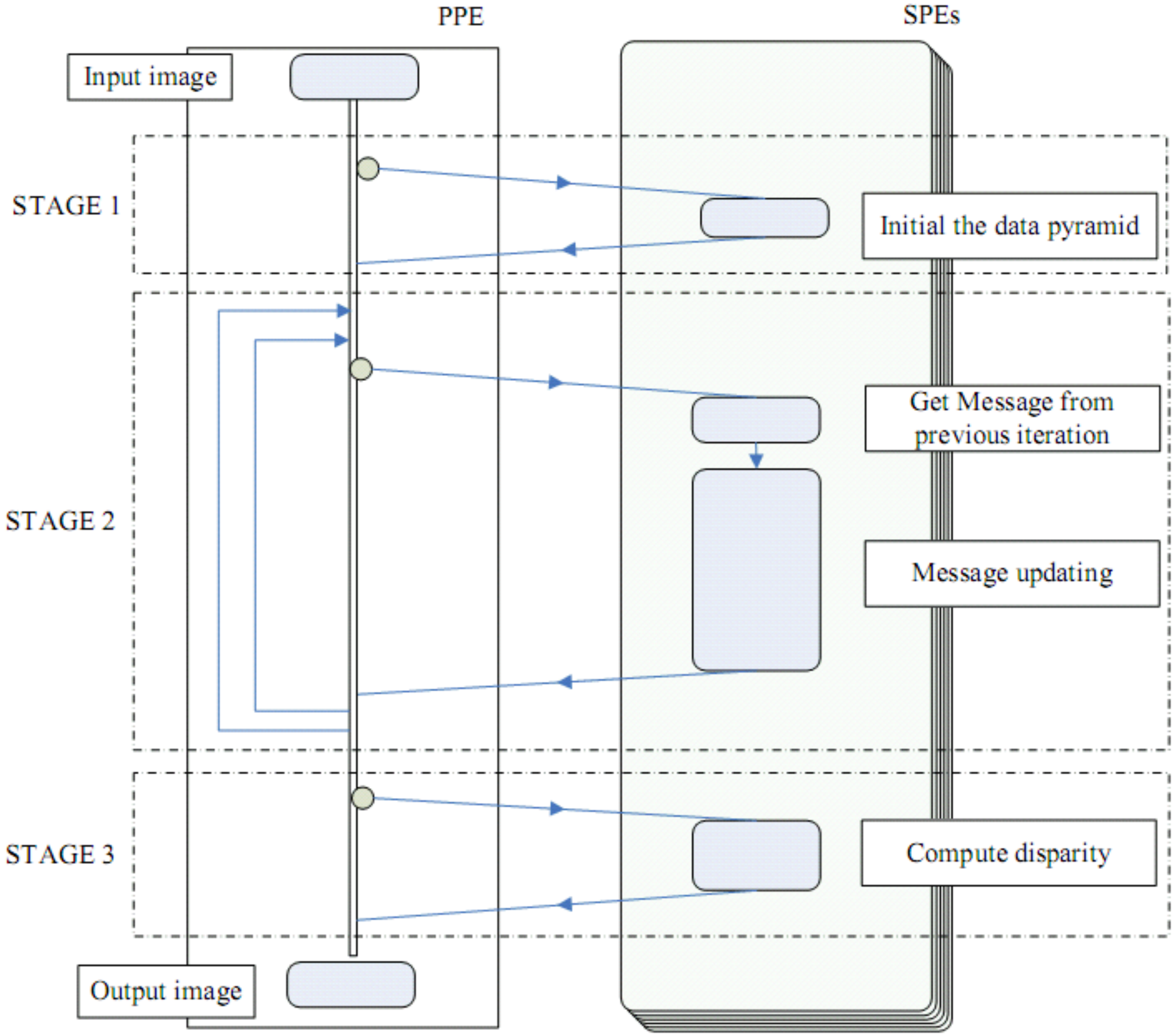
```
floatVec f __attribute__ ((aligned (16)));
vector float *vc = (vector float *)&(f.vec[0]);;
vector float fconst = (vector float ) { 1.0 , 1.0 , 1.0 , 1.0 } ;
vector float a0, a1, a2, a3;
...
/* f i s seperated by vc [0] , vc [1] , vc [2] , vc [ 3] */
a0 = spu add( fconst , vc [0] ) ;
a1 = spu add( fconst , vc [1] ) ;
a2 = spu add( fconst , vc [2] ) ;
a3 = spu add( fconst , vc [3] ) ;
```

Experiment Environment

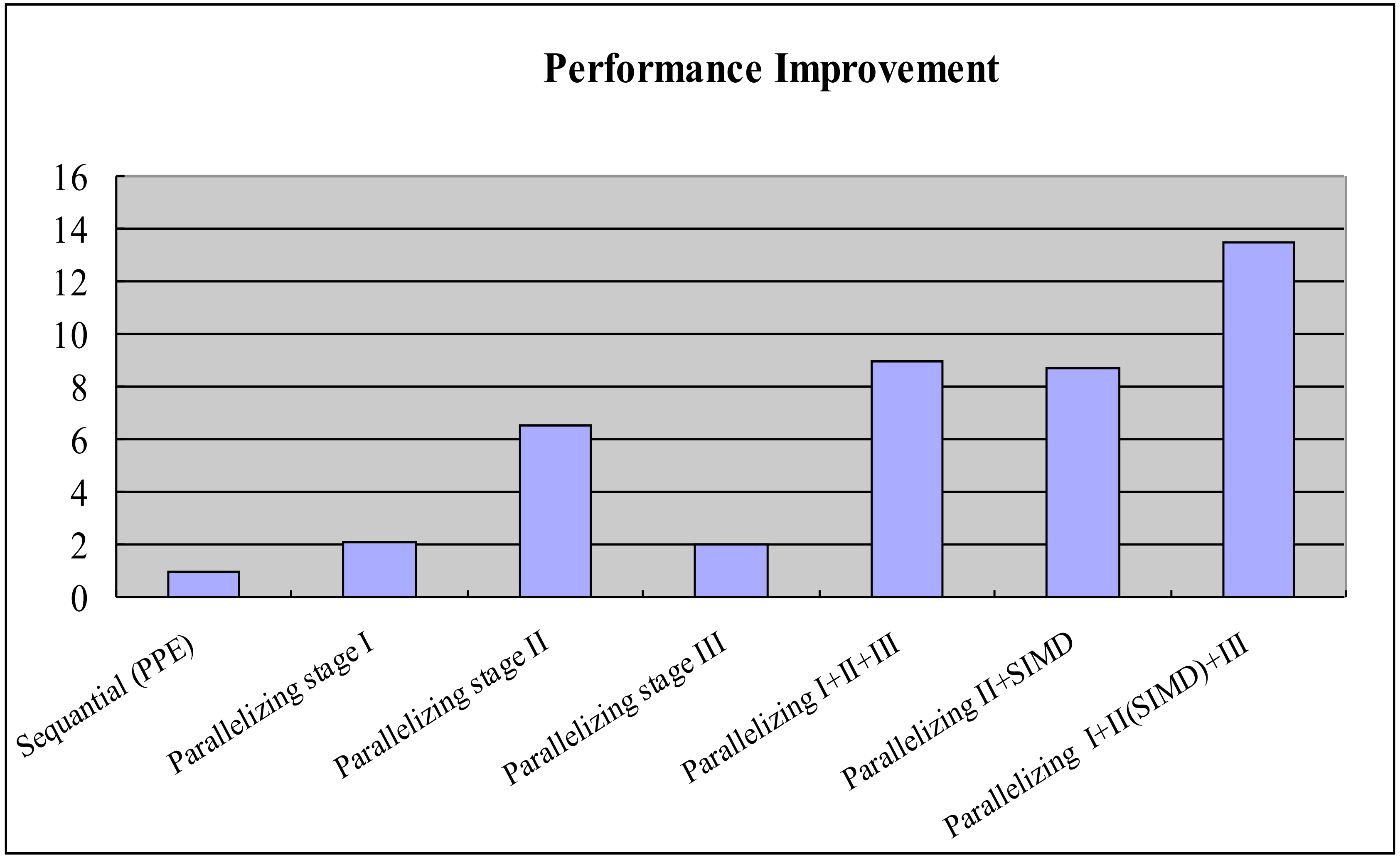
- CPU
  - Cell B.E. processor
  - One 64-bit PowerPC architecture core (PPE)
  - 8SPEs, but only 6 SPEs accessible
- Memory
  - 256 MB XDR DRAM
- OS
  - Linux (Fedora 6)
  - Kernel 2.6.25.4
- Cell programming environment
  - gcc 4.1.1-57 for PPU
  - gcc 4.1.1-107 for SPE
  - binutils-2.17.50-32 for PPU
  - binutils-2.17.50-33 for SPE
  - libspe 2.1
  - newlib 1.5.0-7 for SPE
- Input images of 384X288
  - **5** iterations, **6** disparity levels

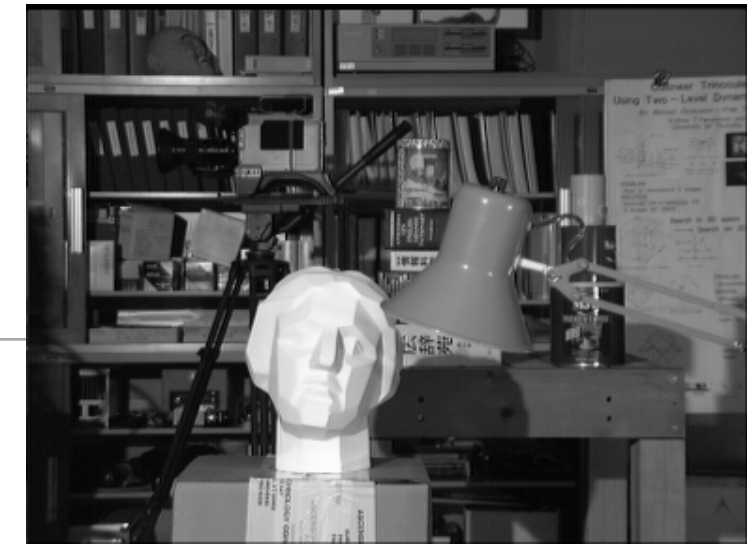


The Execution Flow on CellBE



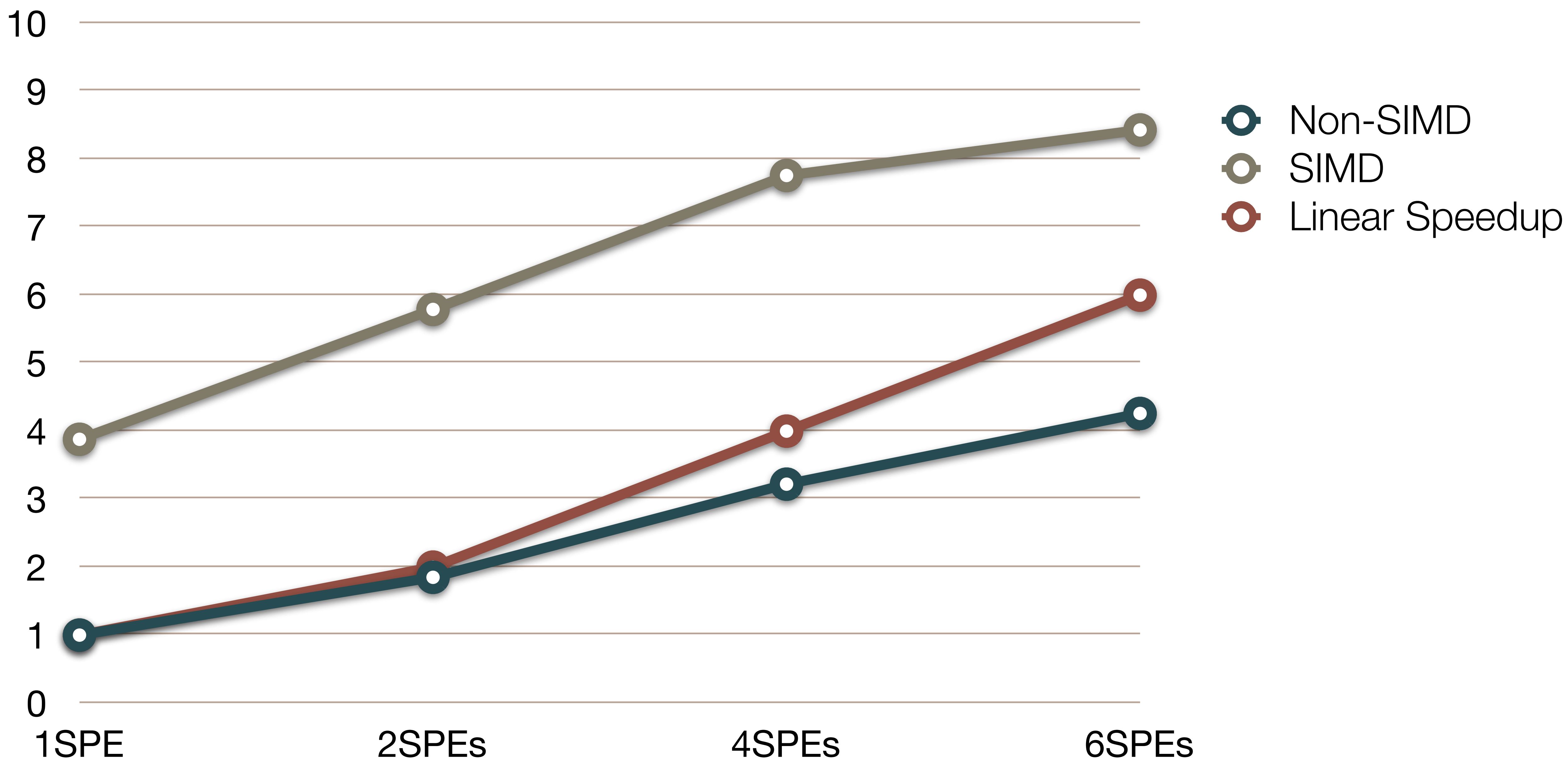
Performance Results





	Sequential	Sequential	Parallelized BP
Platform	3G Pentium 4	PPE on Cell BE	Cell BE
Performance(seconds)	1.195	2.34	0.175
Frame Per Second	0.84	0.43	5.71

Performance Scaling



Conclusion

- Analyzig and examing the parallelization of a belief propagation algorithm on the multicore processors
- Exploiting the opportunities for real-time application
- With careful analysis and parallelizing, the implementation is able to produce a good result

Thank you!

國立清華大學
National Tsing Hua University

NTHU PL LAB