High performance Multi-scale Image Processing Framework on Multi-GPUs

with applications for Unbiased Diffeomorphic Atlas Construction

Linh Khanh Ha

February 2011

Outline

1. Motivation
2. Image Processing Framework
3. Combining Probabilistic and Geometries
4. Streaming Out-of-core MIP
5. Conclusion

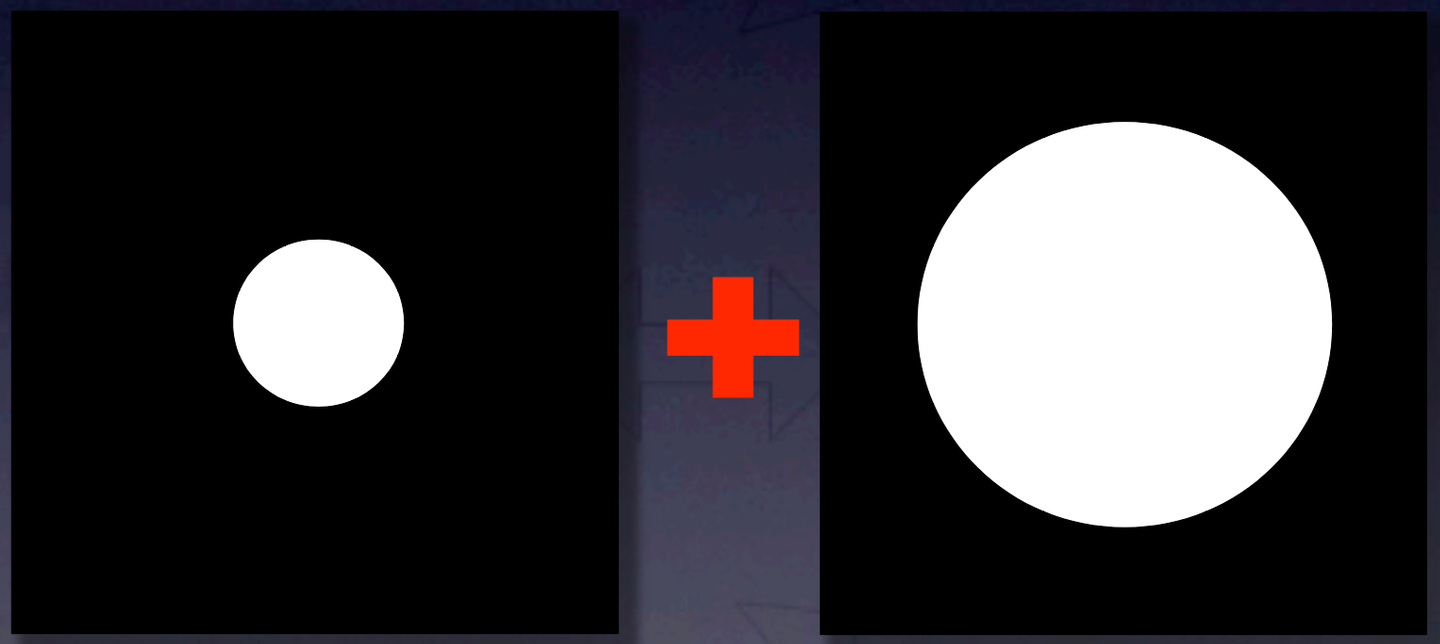
Outline

1. Motivation
2. Image Processing Framework
3. Combining Probabilistic and Geometries
4. Streaming Out-of-core MIP
5. Conclusion

Given a collection of anatomical images,
what is the image of the "Average Anatomy" ?

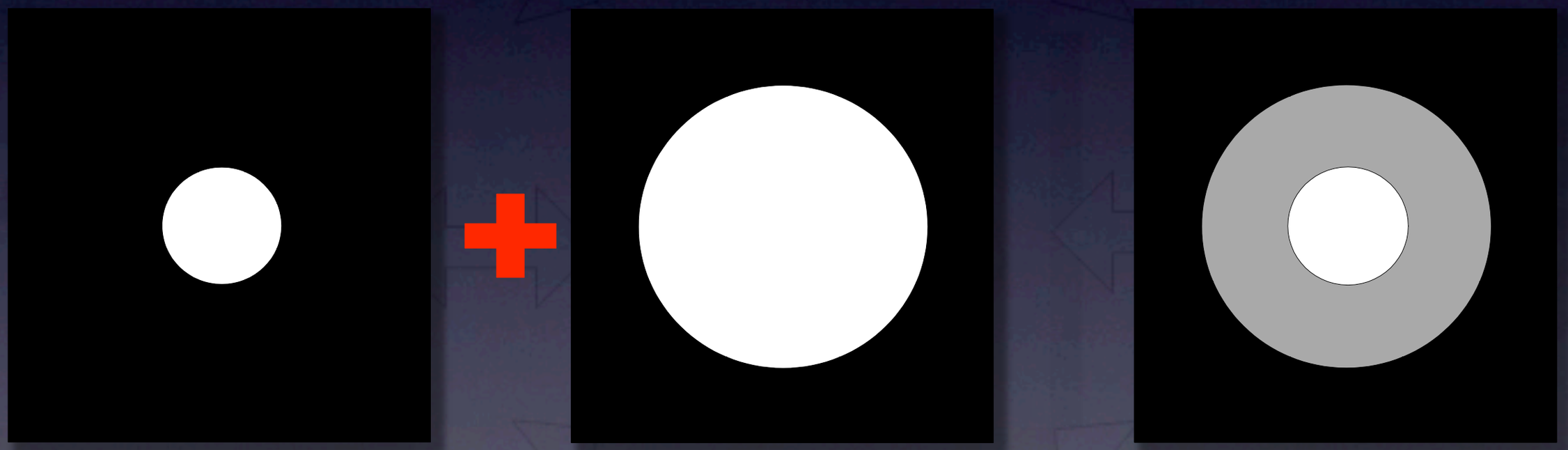
“Average Anatomy”

“An anatomical object that requires least amount of energy in total to deform to each individual of the population”



“Average Anatomy”

“An anatomical object that requires least amount of energy in total to deform to each individual of the population”



Intensity Average

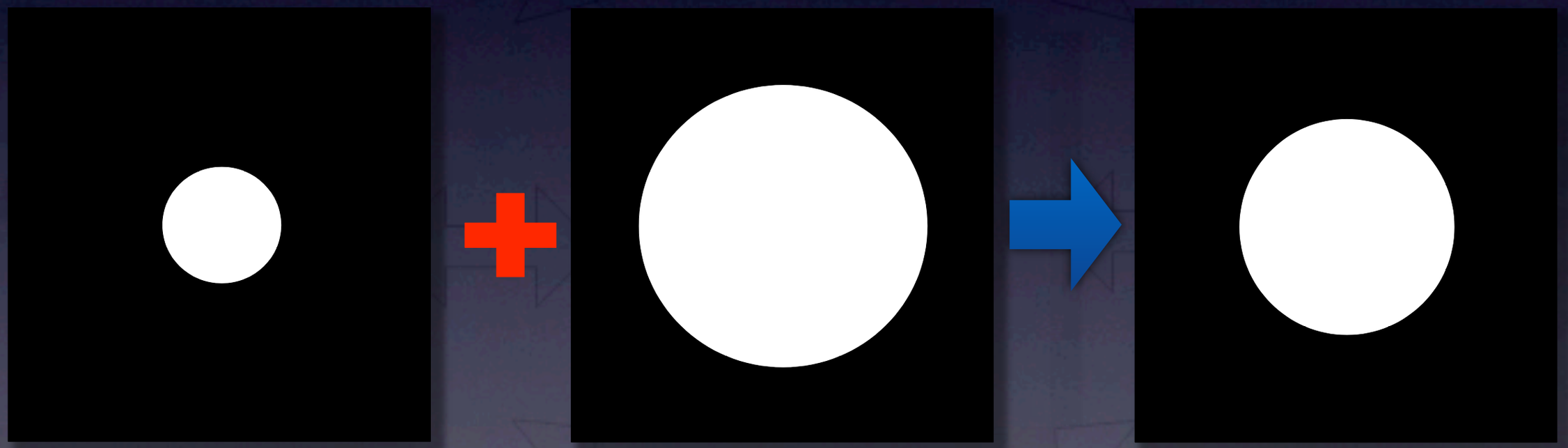
“Average Anatomy”

“An anatomical object that requires least amount of energy in total to deform to each individual of the population”

Intensity Average

“Average Anatomy”

“An anatomical object that requires least amount of energy in total to deform to each individual of the population”



Geometric Average

“Average Anatomy” = Atlas Construction

- A registration problem
  - Find the “optimal” deformation h that deforms one object to the other
    - Desired constraint h is “diffeomorphic”
    - Diffeomorphic mapping - registration on diffeomorphic space

Diffeomorphic mapping

Properties

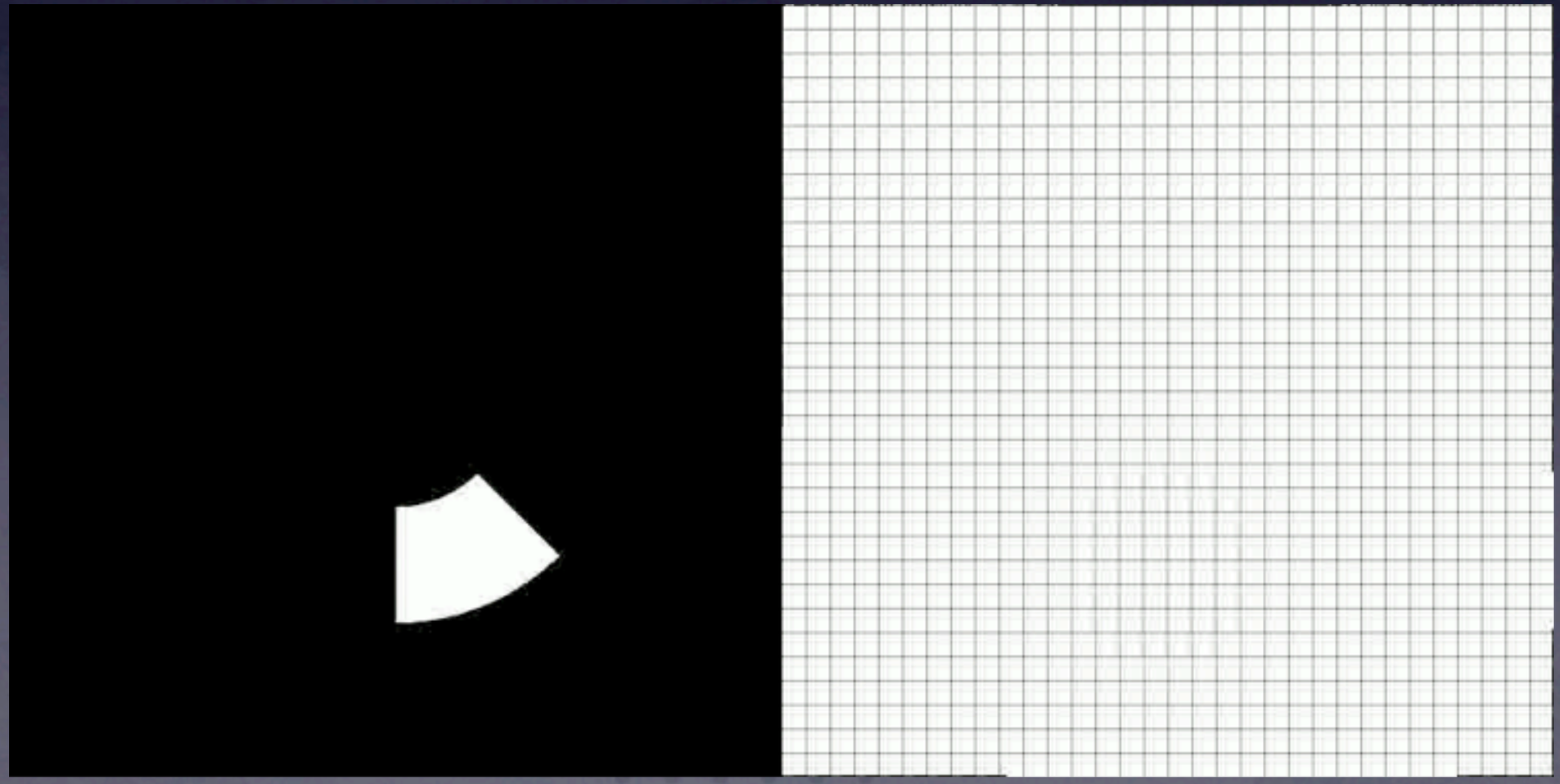
- Continuous
- One to one (invertible)
- Topology is maintained

Non-linear registration

- High registration quality
- Infinite dimension

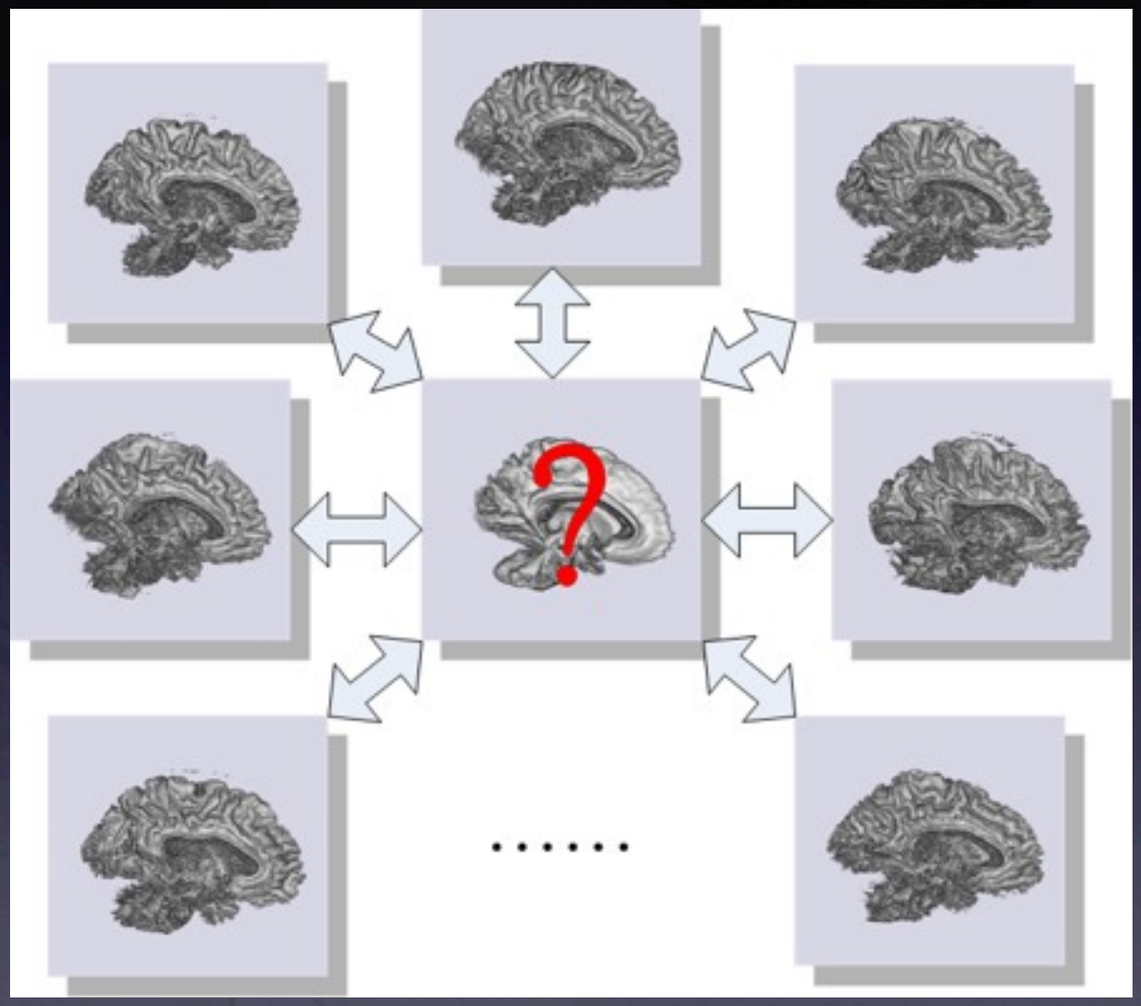
Doing statistic

- Volume, variation of shape



Applications

- Applications:
  - Map population into a common coordinate space
    - Learn about variability
    - Describe difference from normal
  - Development, evolution, disease
  - Automatic segmentation

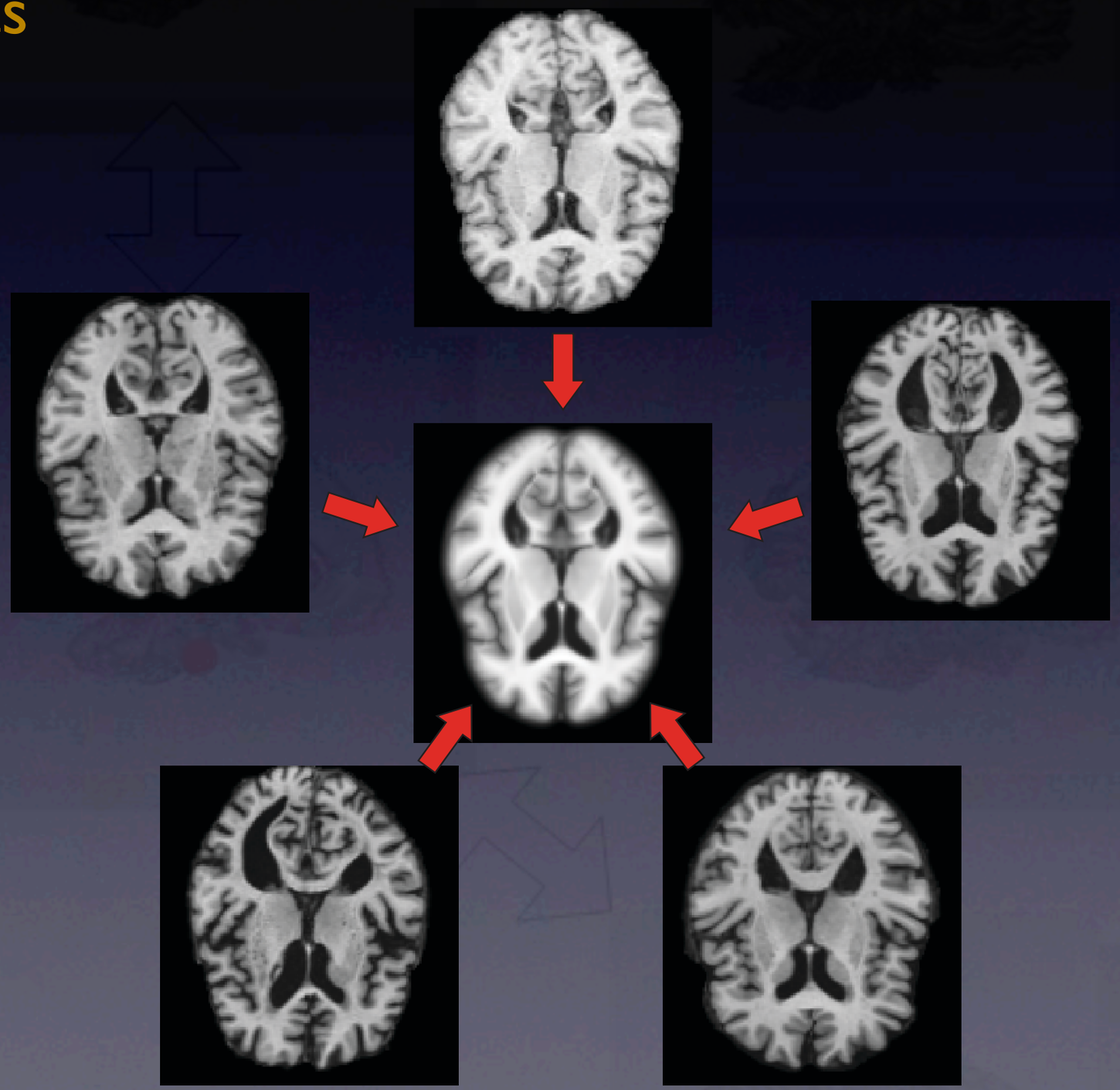


Joshi et al 2004 "Unbiased Diffeomorphic ..."

Unbiased = Consistent + Order independent

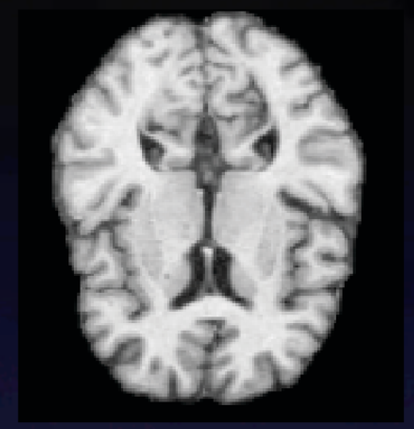
Automatic Segmentation

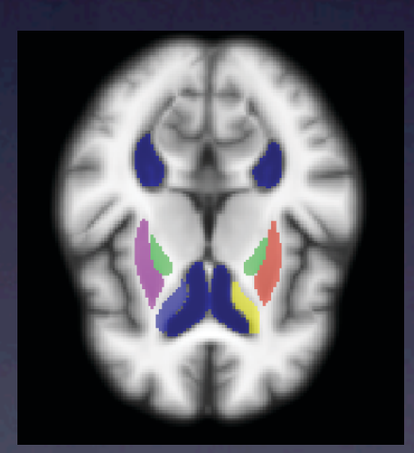
- Compute the atlas

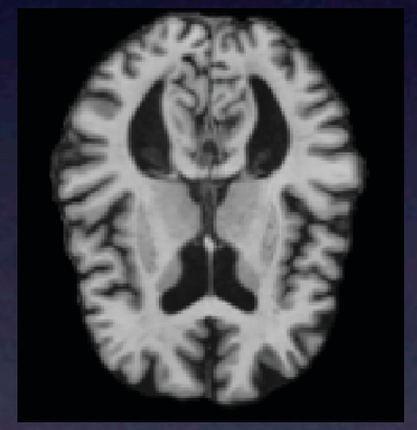


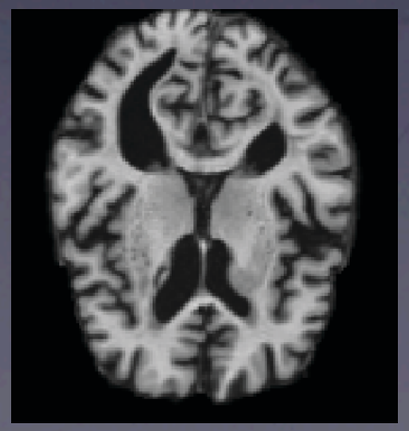
Automatic Segmentation

- Compute the atlas
- Partition the atlas



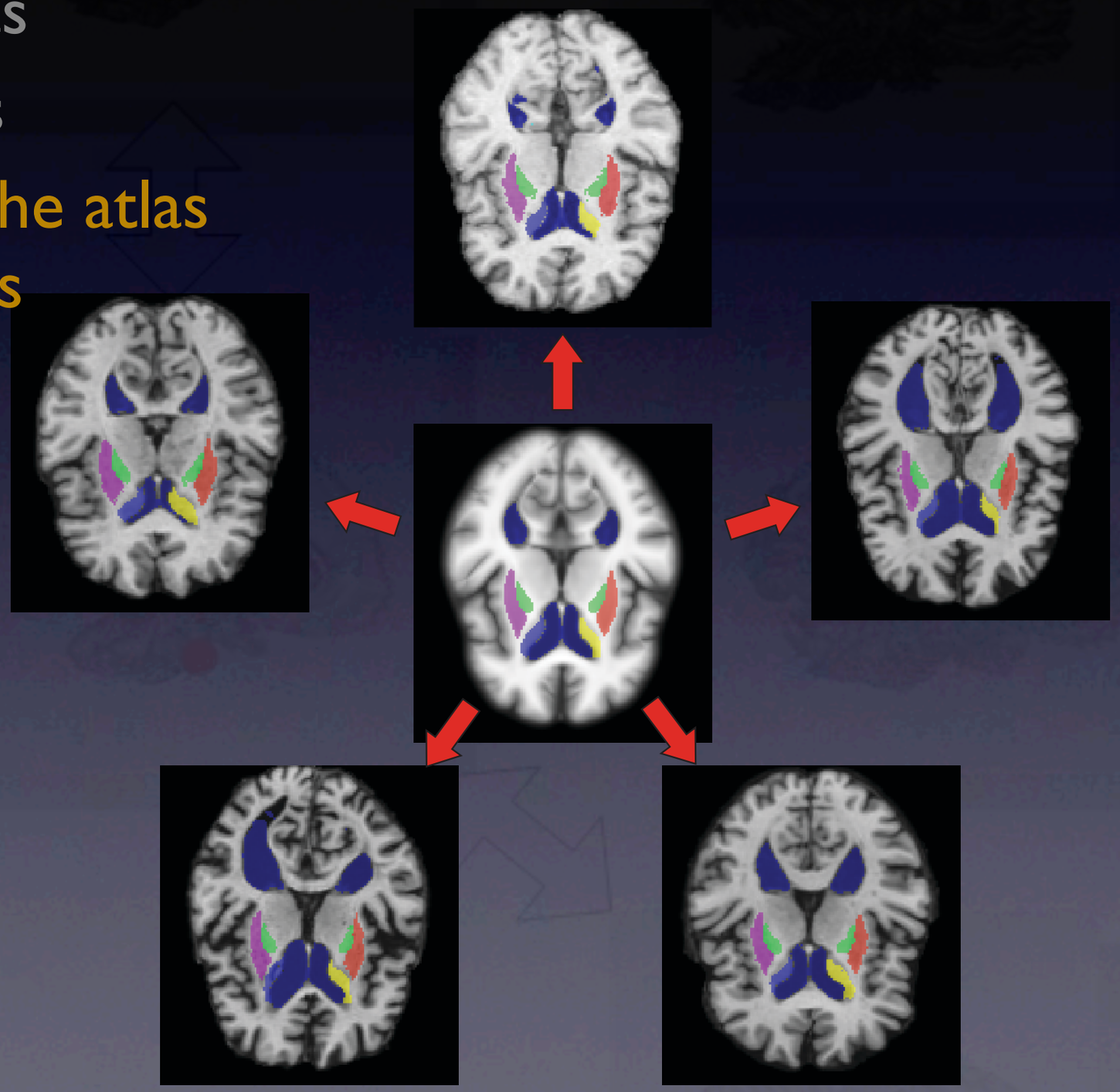




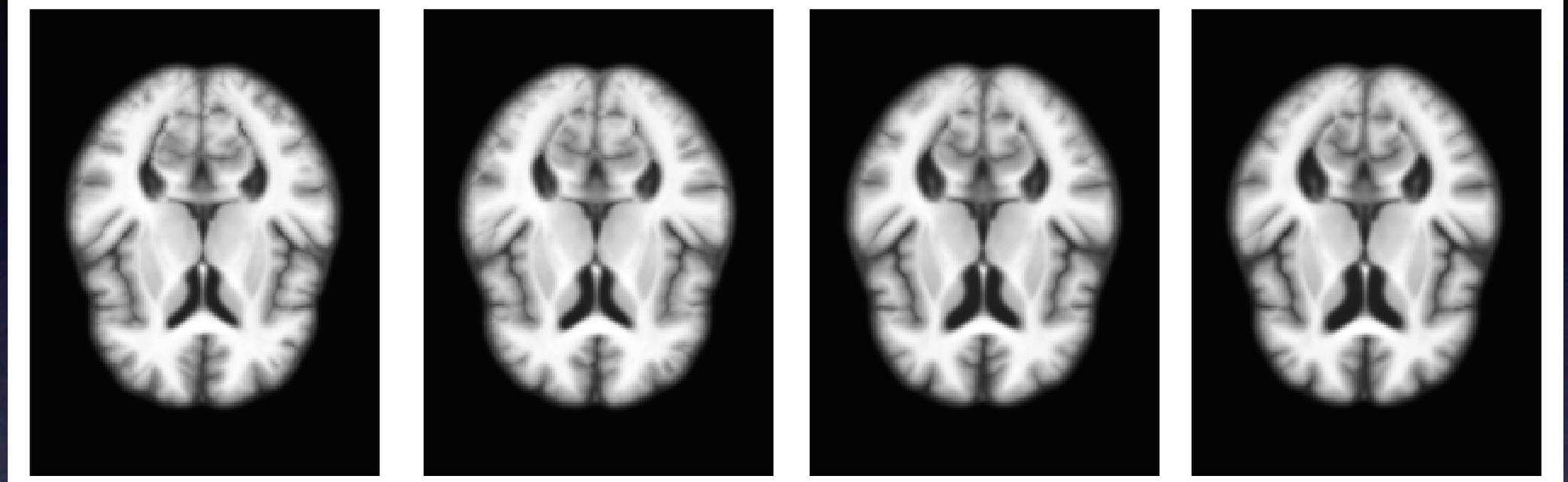


Automatic Segmentation

- Compute the atlas
- Partition the atlas
- Map labels from the atlas back to individuals



Doing analysis on the population

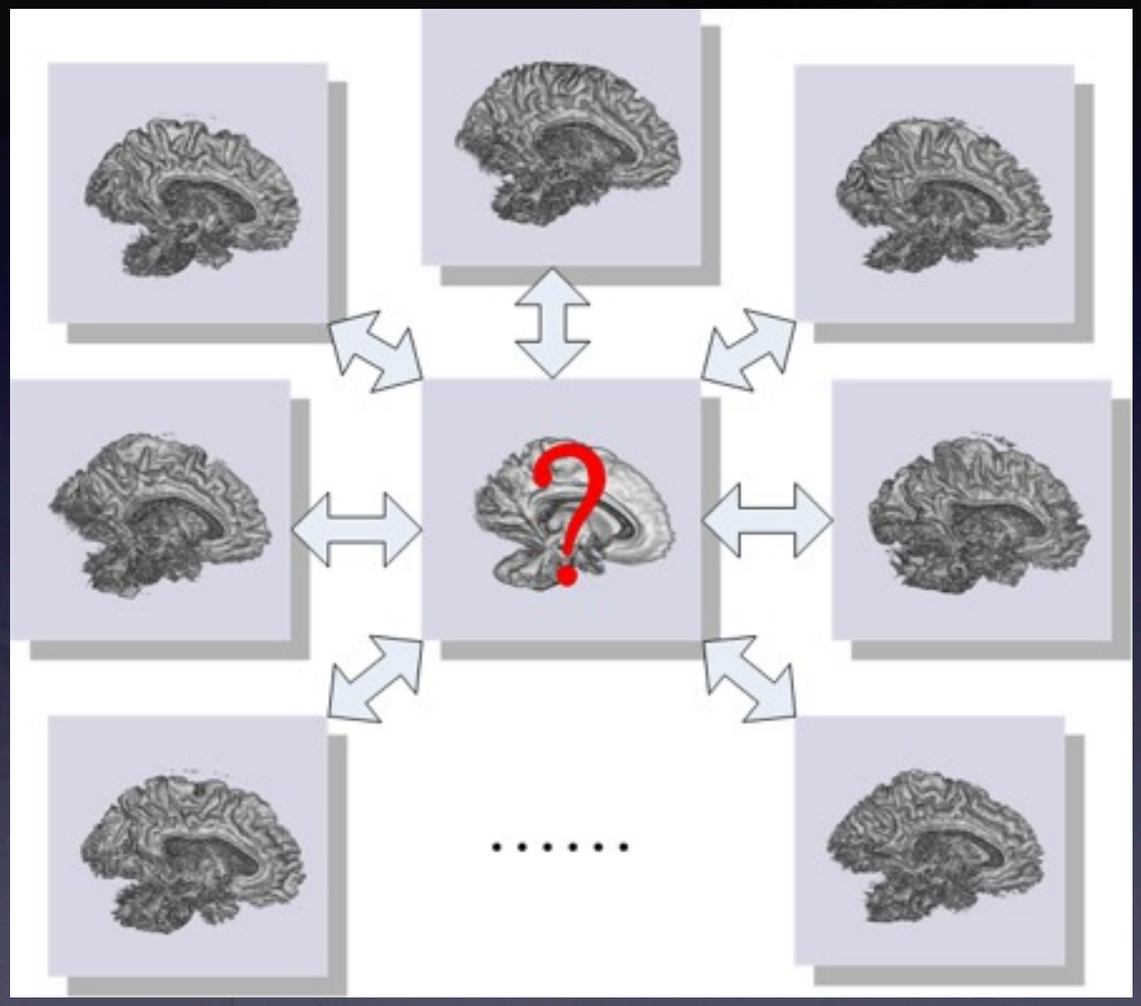


a) Age 65 b) Age 70 c) Age 75 d) Age 80

Age regression analysis on ADNI dataset of 315 brain images

Atlas Construction is challenging

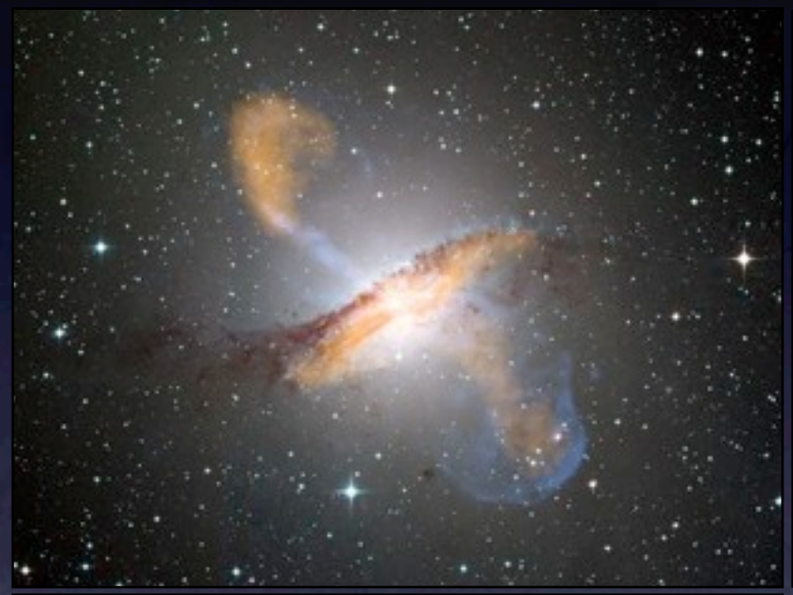
- Energy minimization problem:
  - Requires solving expensive ODEs, and PDEs
    - massive computational power
  - Large memory requirements
    - Hundreds of 3D images
  - Too slow (months) using CPUs



Joshi et al 2004 "Unbiased Diffeomorphic ..."

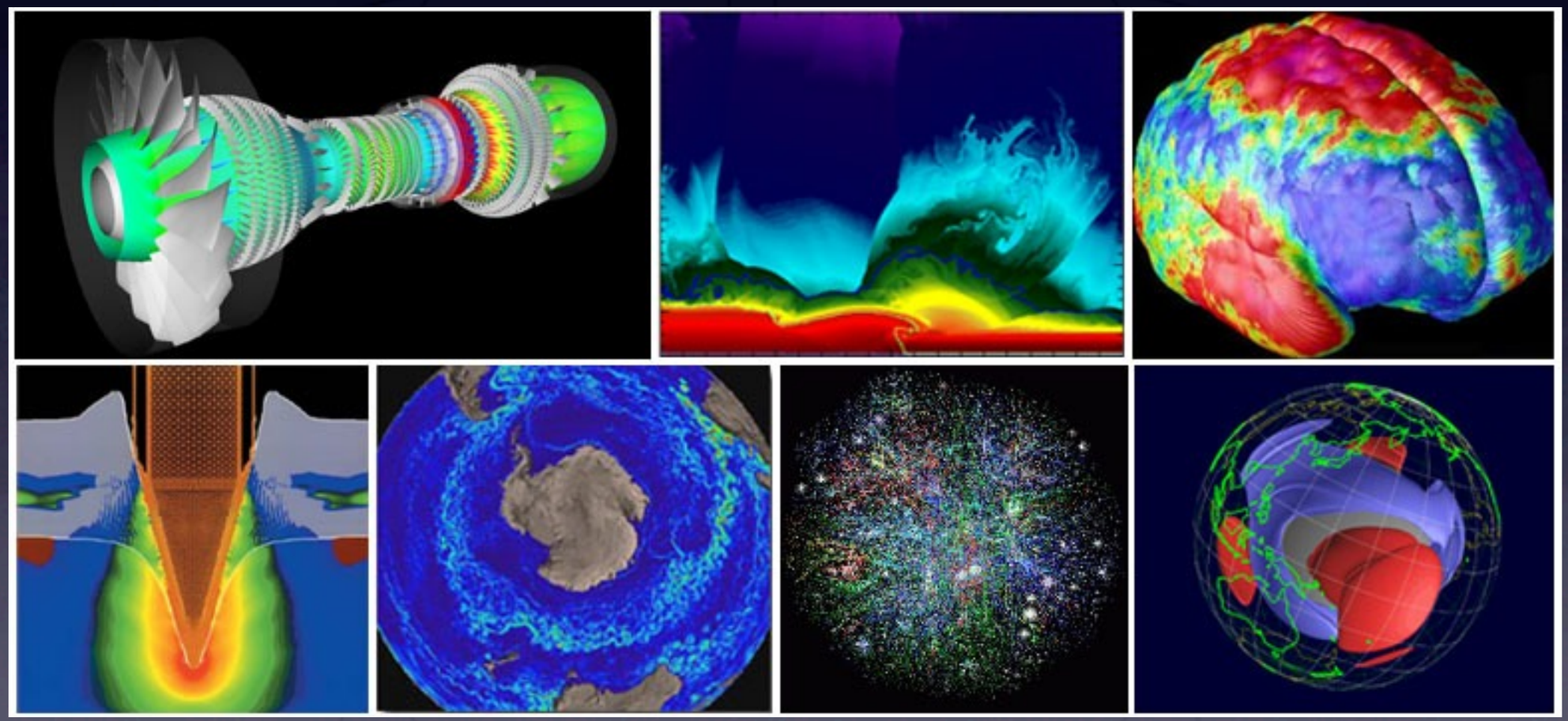
Solution from Parallel Computing

The World around us is massively parallel



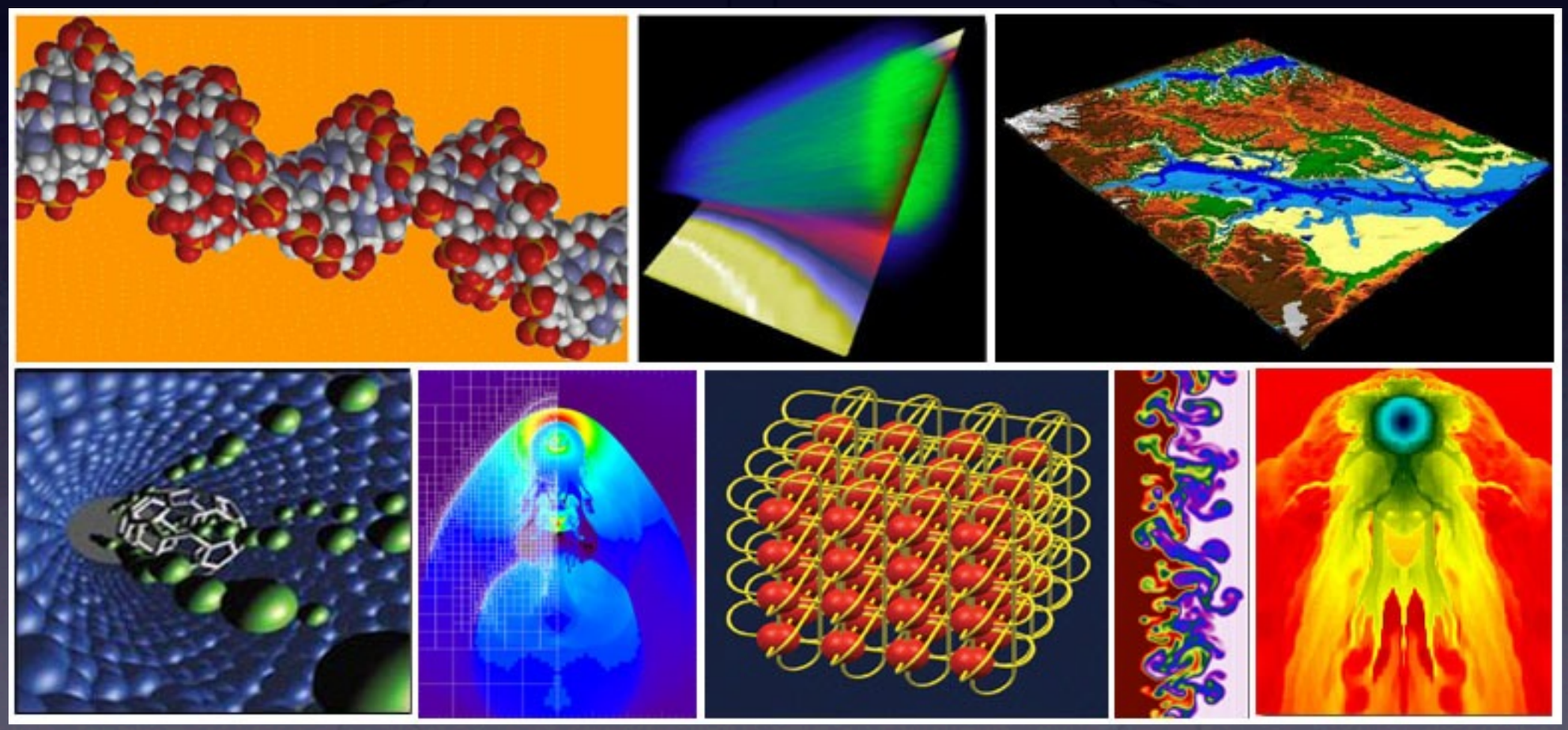


Parallel Computing
Past : high end of computing



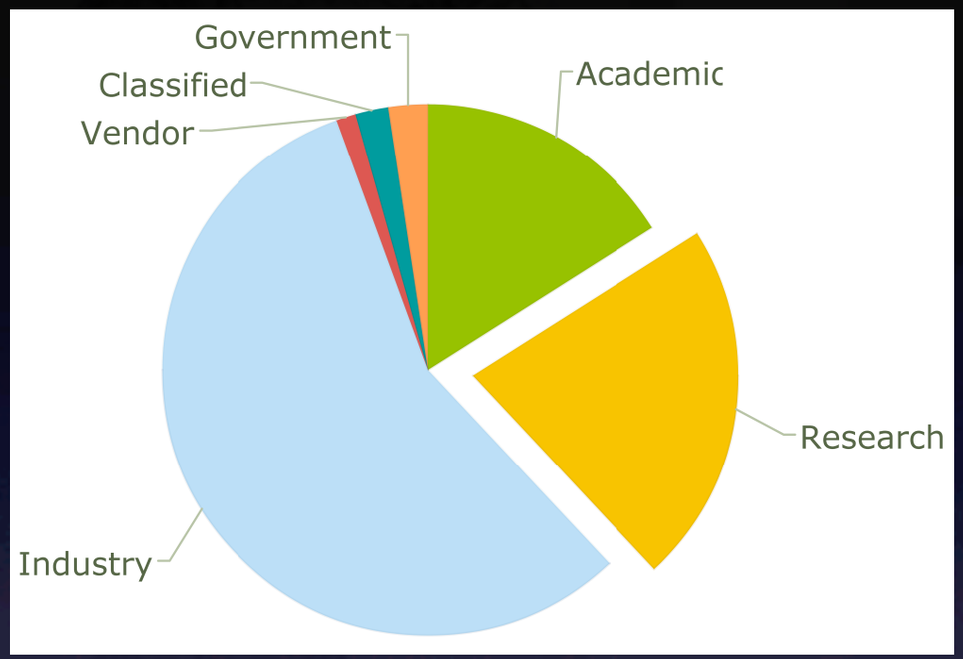
(Blaise Barney "*Introduction to Parallel computing*")

Parallel Computing
Nowadays : used in many research areas



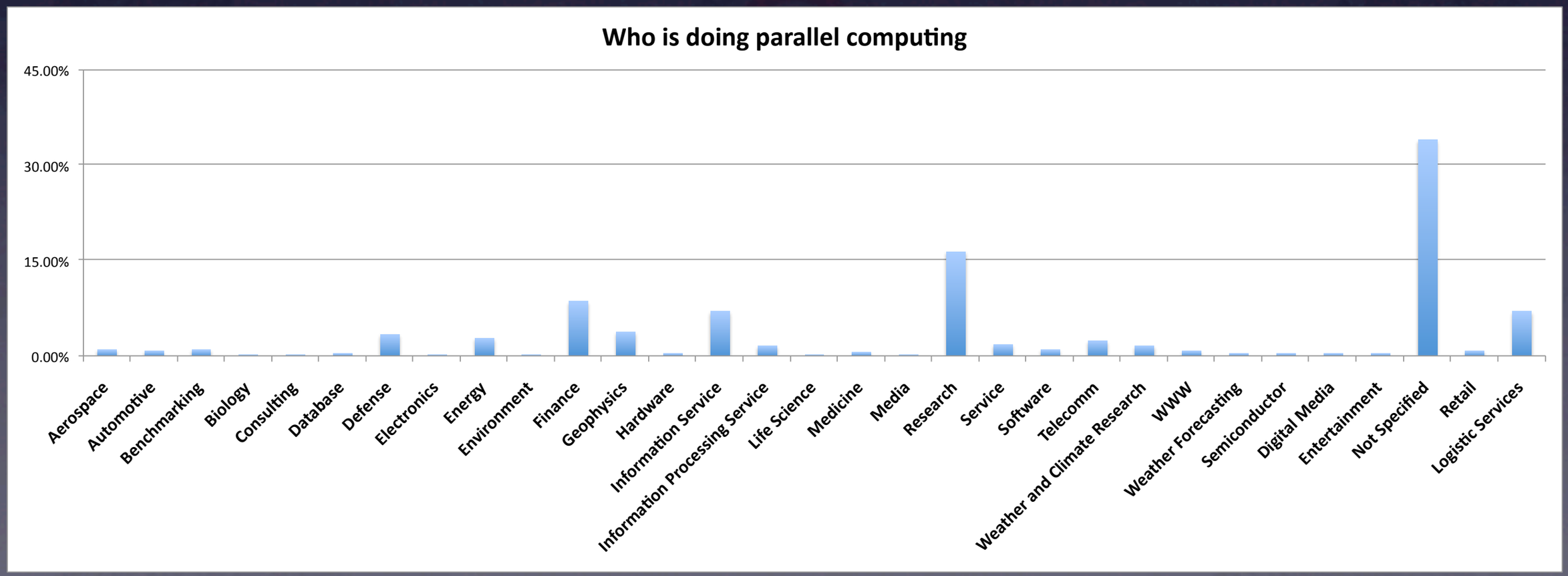
(Blaise Barney "*Introduction to Parallel computing*")

Who is doing parallel computing ?



Top 500 statistic page

http://www.top500.org/stats



Why GPU Computing

Economical reason

- High performance
- Low power consumption
- Smaller footprint

Mass product support

- Appear in many products
- Constant technical improvement
- Game and medical research community

Technical reason

- Fast for computational purpose
  - High computational power Teraflops/card
  - High memory bandwidth
- High scalability, unified programming model (CUDA, Open CL)
- 100x faster in many applications

Recent success

- The active HPC research area in the past 10 years
  - Double performance every year
  - GPGPU, Unified computing model, Fermi, GPU cluster ...
- Fastest computing devices

GPU Computing is challenging

New programming concept

- Porting code is non trivial
- Rethinking about data structure and algorithms

Optimization is hard

- Code specific to systems
- Low level optimization
- Expert knowledge about the architecture

Lack of development tools

- Hard to debug and maintain
- Learning curve is high
- Time to product is high

But it is changing to give more pleasant programming experience.

In this thesis, we provide :

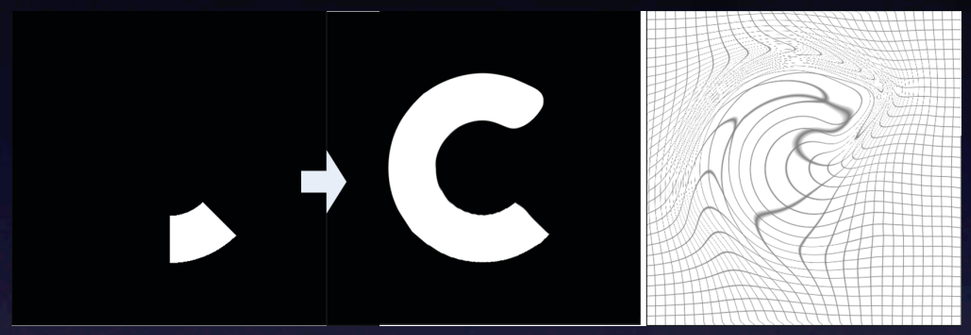
Essential tools for researchers to harness GPU processing power !

A GPU framework

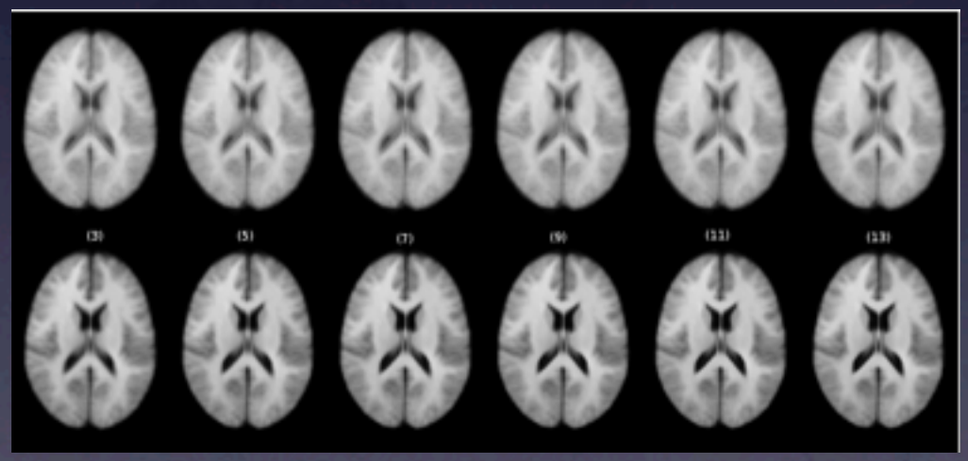
Outline

1. Motivation
2. Image Processing Framework
3. Combining Probabilistic and Geometries
4. Streaming Out-of-core MIP
5. Conclusion

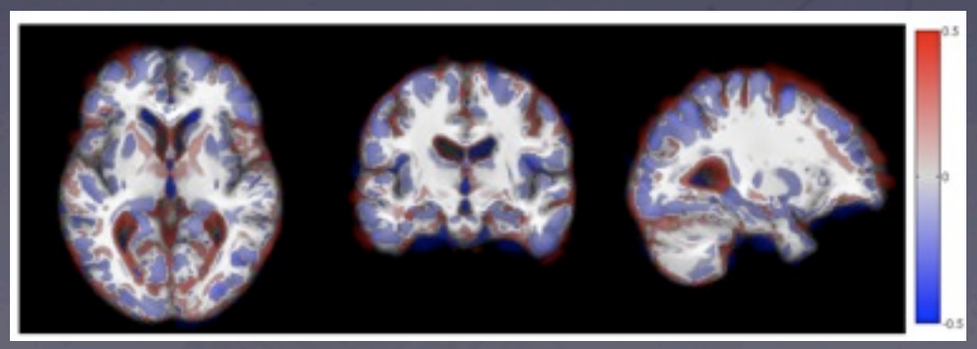
High performance Image Processing framework on GPU



Fast Parallel Unbiased Diffeomorphic Atlas Construction on Multi-Graphics Processing Units
Eurographic Symposium on Parallel Graphic and Visualization 2009
L. Ha, Jens Kruger and Claudio T. Silva



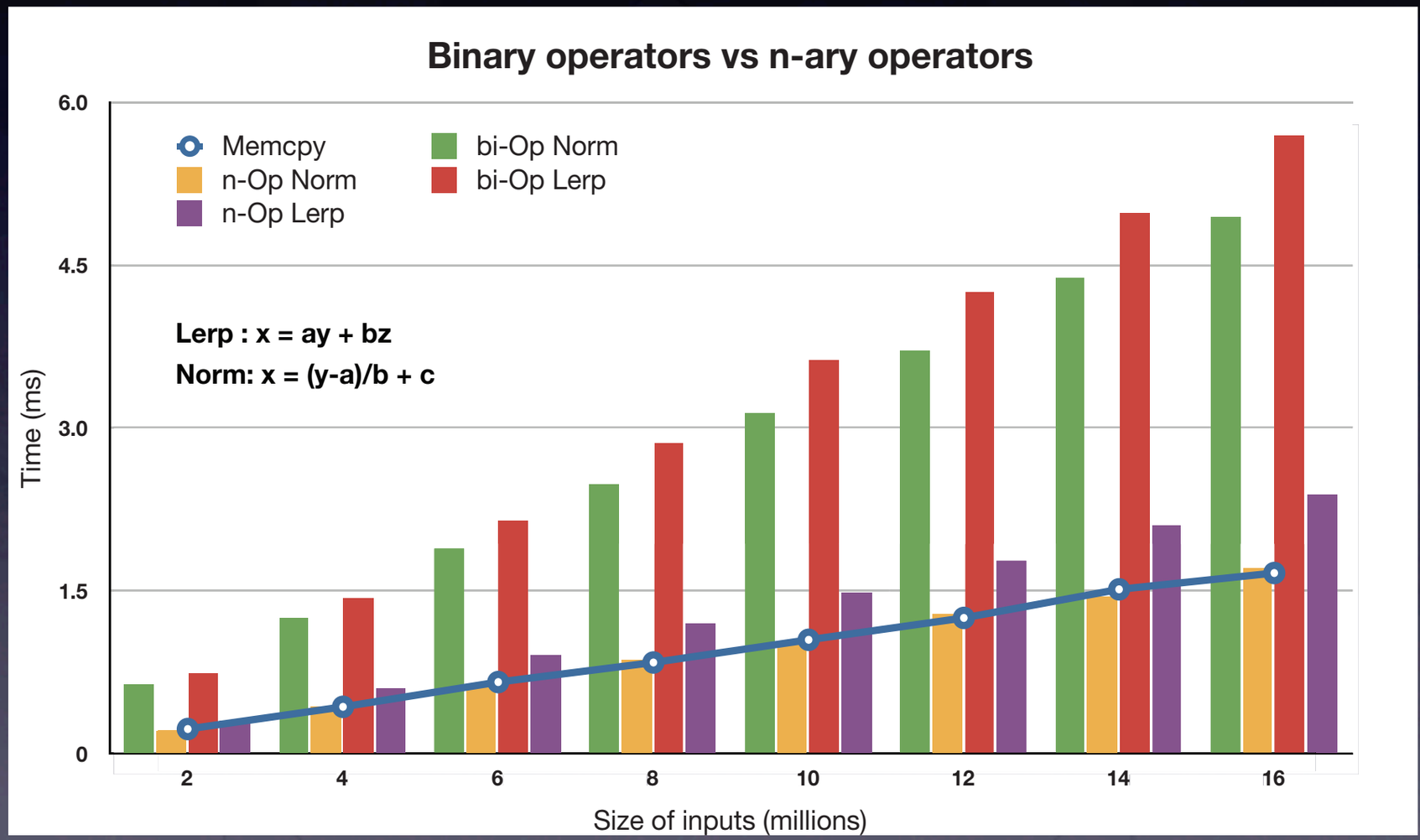
Multi-scale Unbiased Diffeomorphic Atlas Construction on Multi-GPUs
GPU Computing Gems Volume I
L. Ha, Jens Kruger, Sarang Joshi and Claudio T. Silva



Multivariate Statistical Analysis of Deformation Momenta Relating Anatomical Shape to Neuropsychological Measures
MICCAI 2010
Nikhil Singh, P. Thomas Fletcher, J. Samuel Preston, Linh Ha, Richard King, J. Stephen Marron, Michael Wiener and Sarang Joshi

Advantages

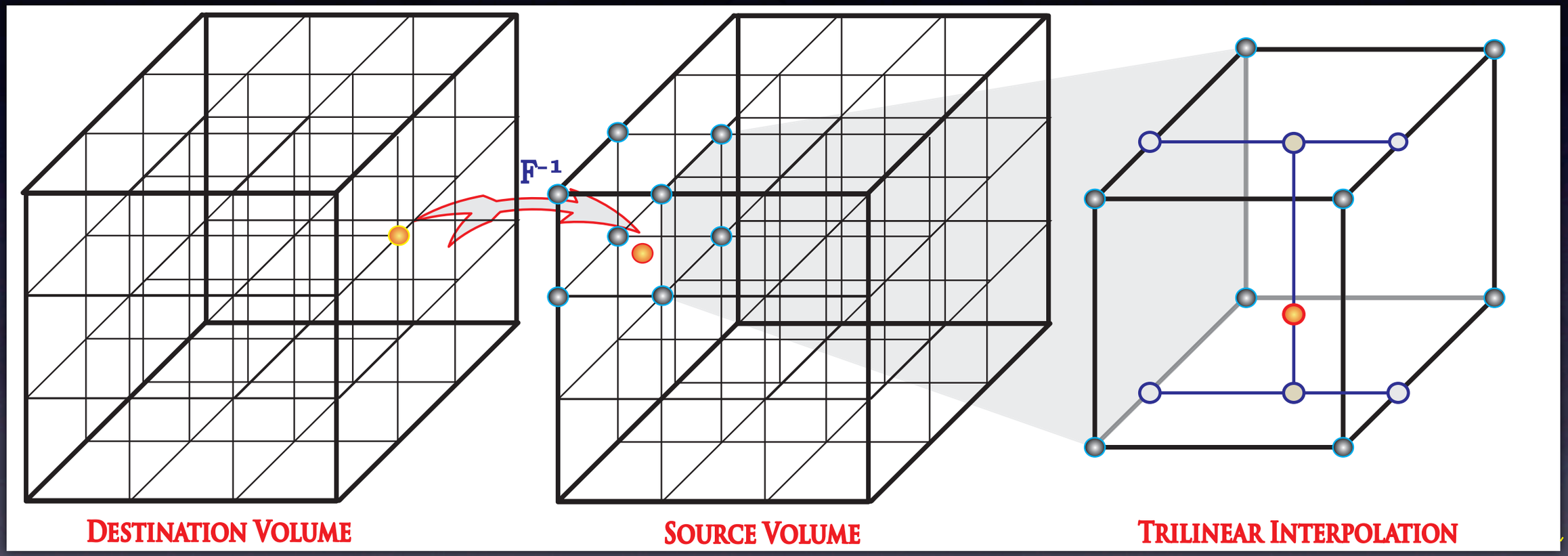
- High performance basic operations



We found optimal approach for basic image processing functions

Advanced functions

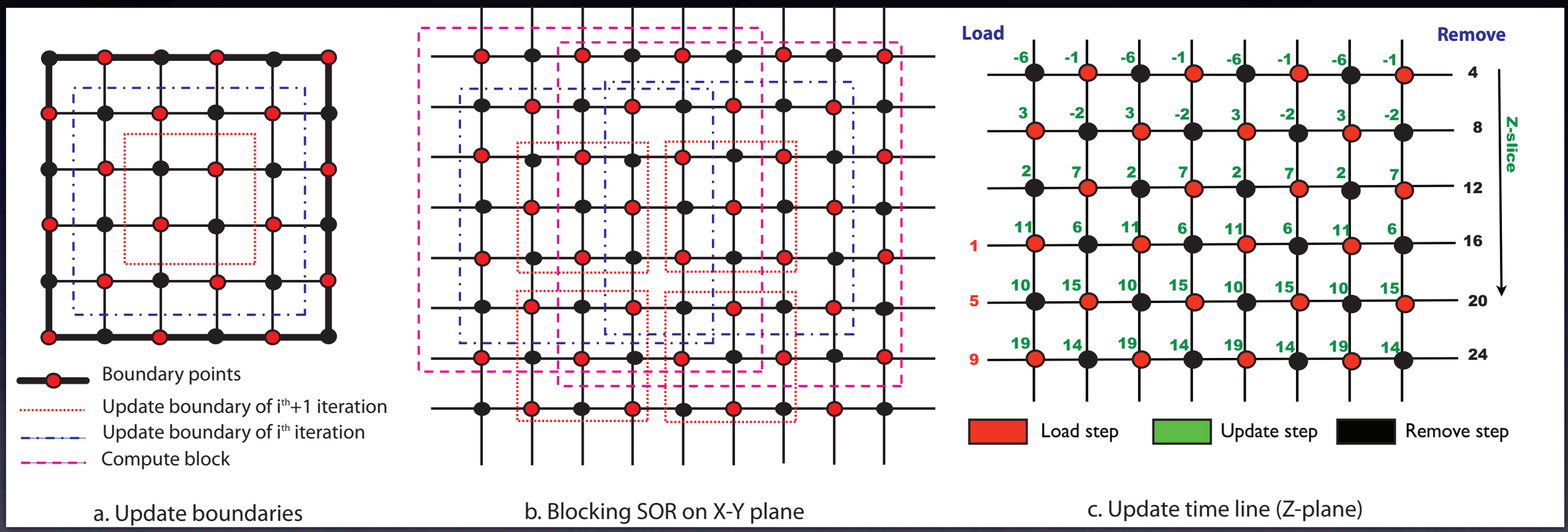
- ODE Integration $h_i^{k+1} = h_i^k(x + \epsilon v_i^k(x))$



GPU hardware tri-linear interpolation using 3D texture
(50x time faster)

Advanced functions

- PDE Solvers $(\alpha\nabla^2 + \beta\nabla + \gamma)v(x) = F(x)$



a. Update boundaries

b. Blocking SOR on X-Y plane

c. Update time line (Z-plane)

- FFT-based solver
- Iterative solvers
  - Block SOR
  - CG template solver

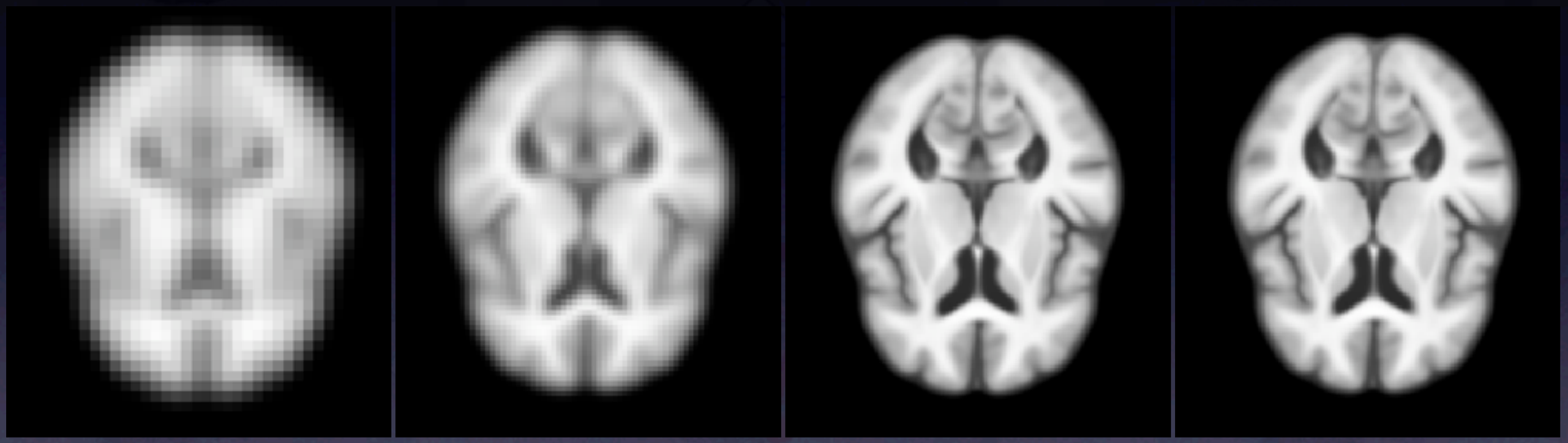
Multi-scale framework

Level 2

Level 1

Level 0

Final result



Increase details

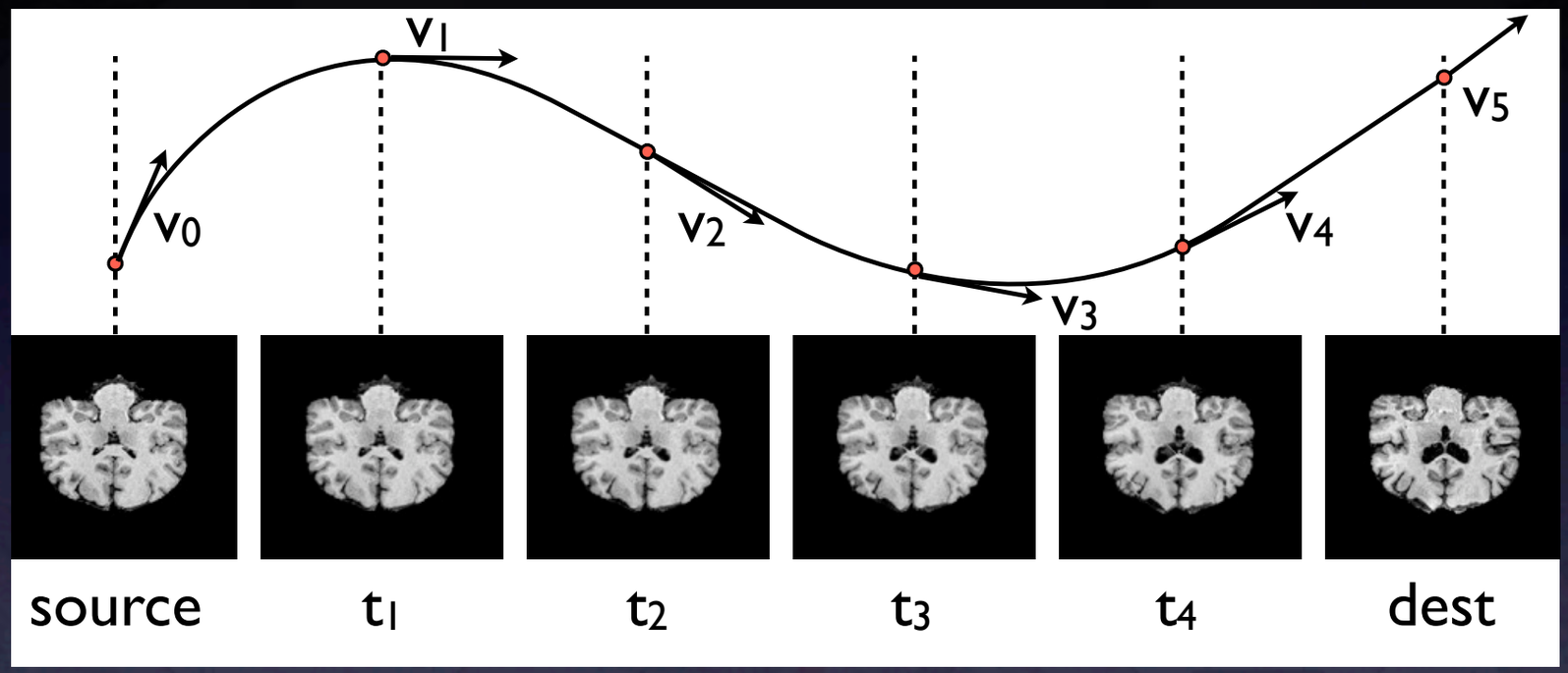
Increase quality

Speed up convergence

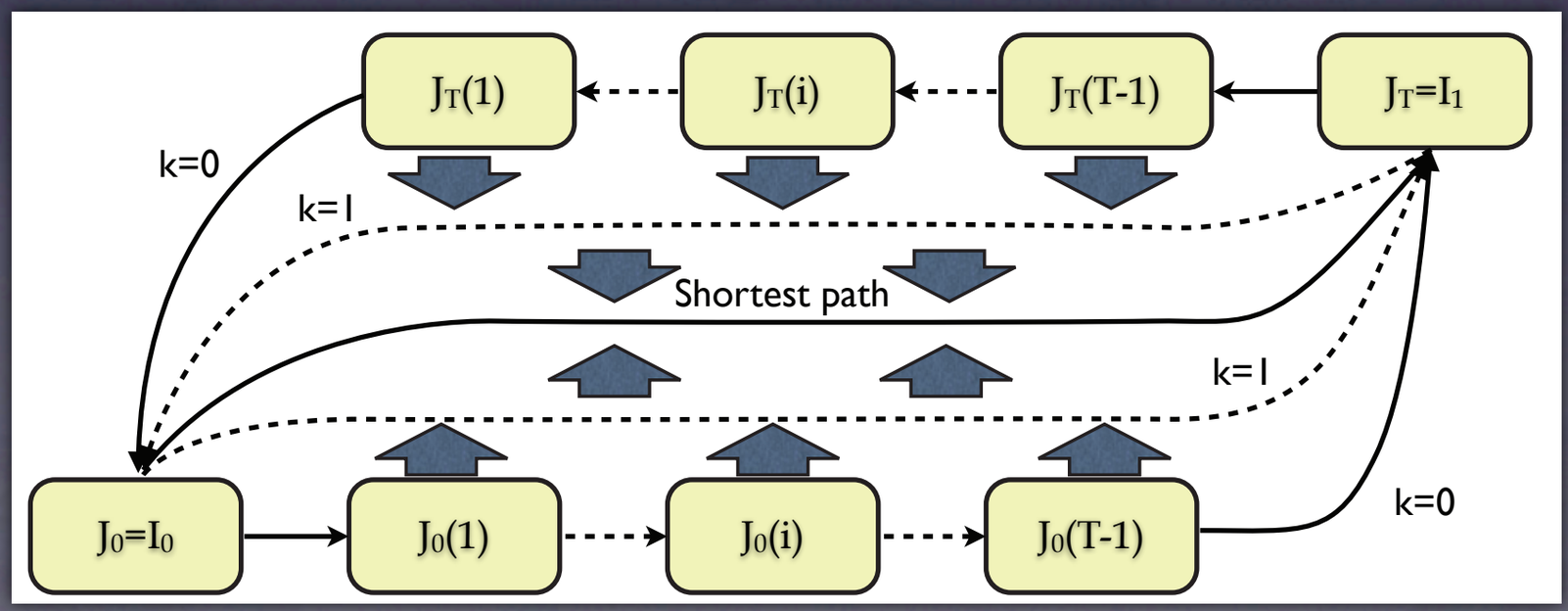
Decrease computation

Different registration algorithms

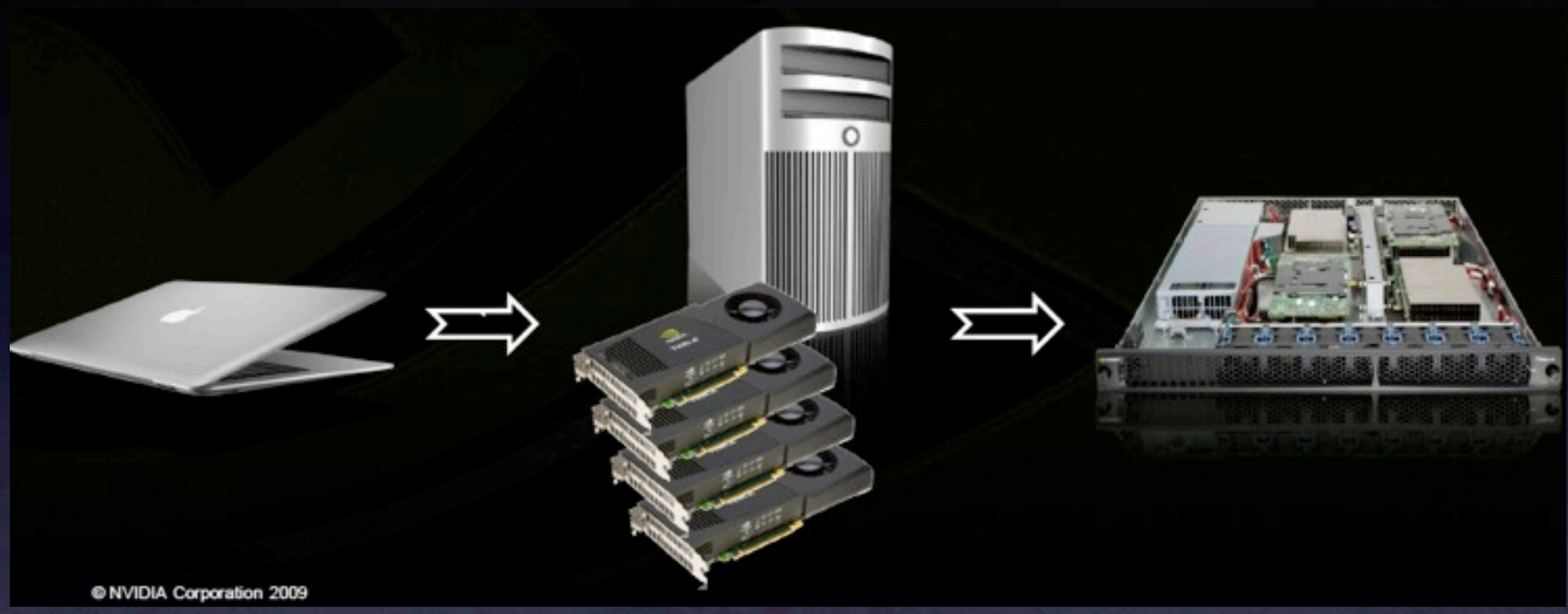
- Greedy Iterative algorithm

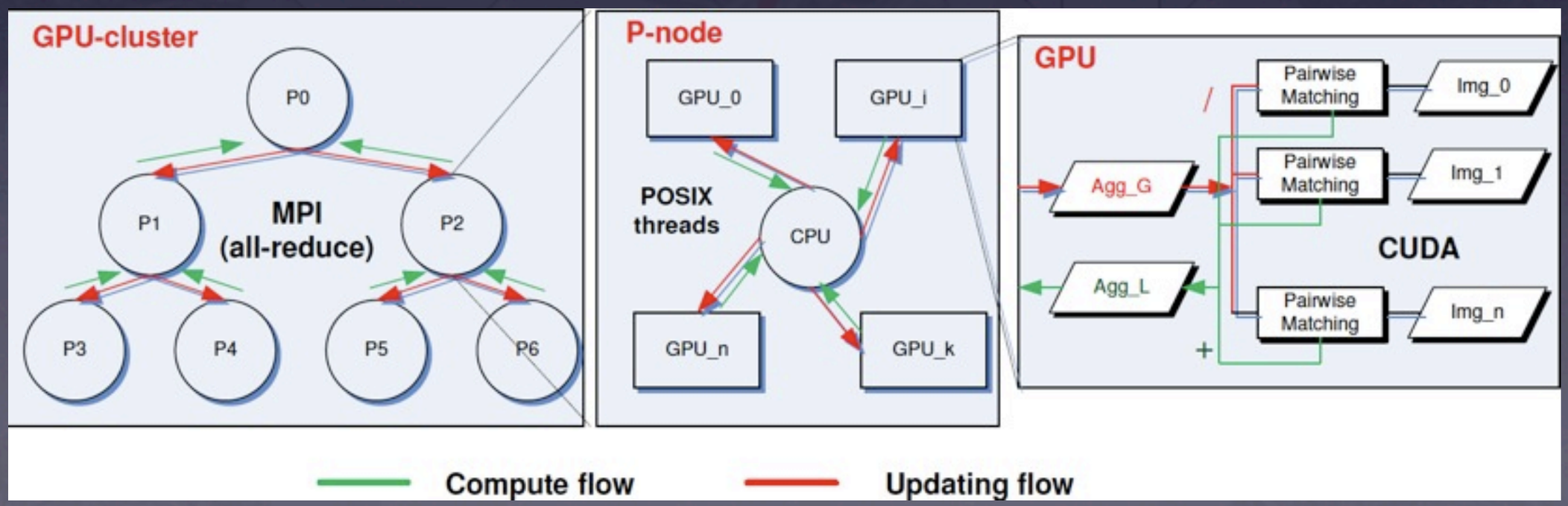


- LDDMM algorithm

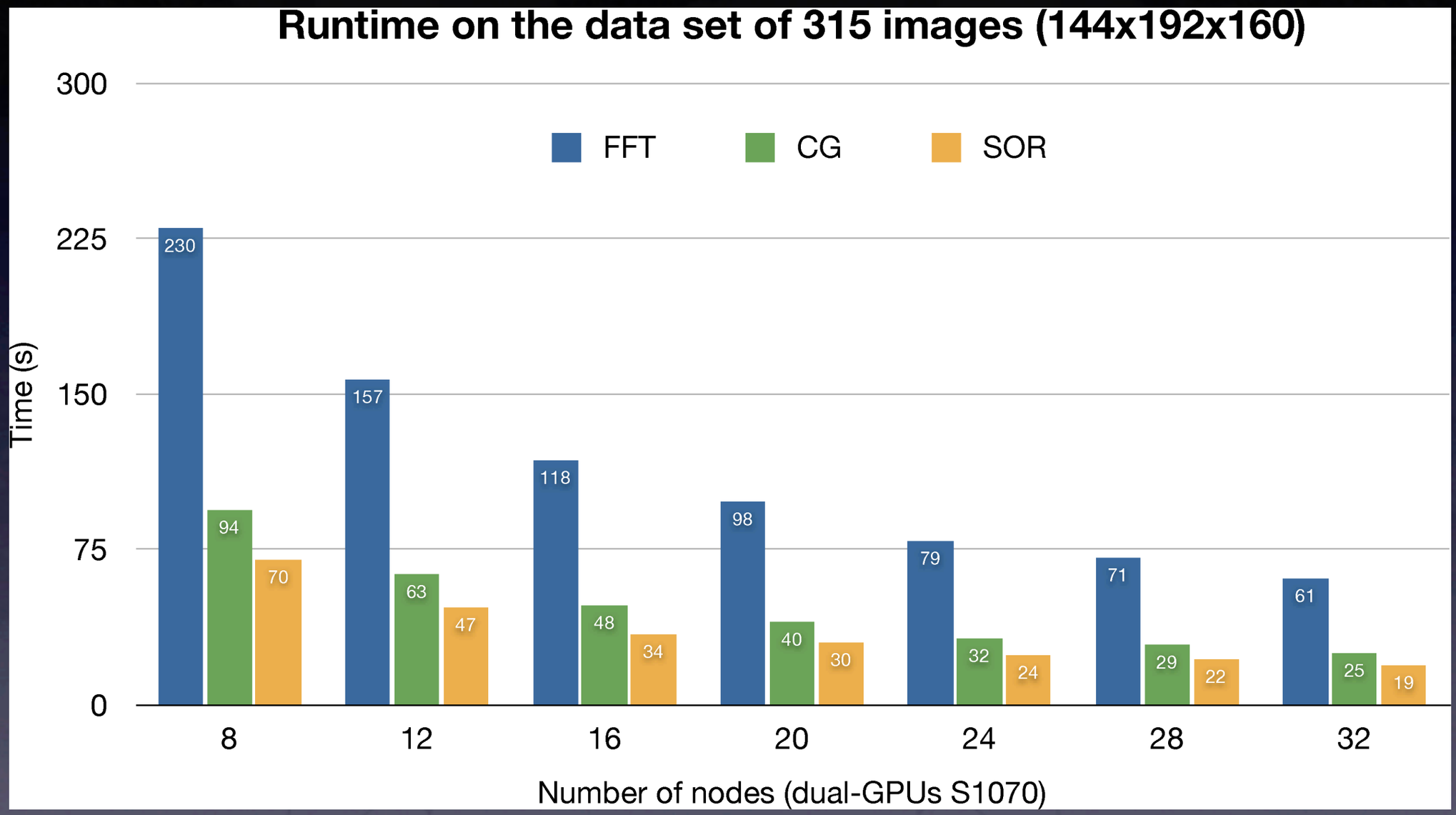


Single GPU - Multi-GPUs - GPU clusters





Performance and Scalability



Part 1 : Conclusion

- In part 1 we present
  - A general image processing framework
    - Basic/Advanced functions
    - Multi-scale framework
    - Unbiased Atlas Construction implementation
      - Multi-GPU
      - GPU clusters
- Next:
  - A registration challenge
  - Efficient computation on irregular grid

Outline

1. Motivation
2. Image Processing Framework
3. Combining Probabilistic and Geometries
4. Streaming Out-of-core MIP
5. Conclusion

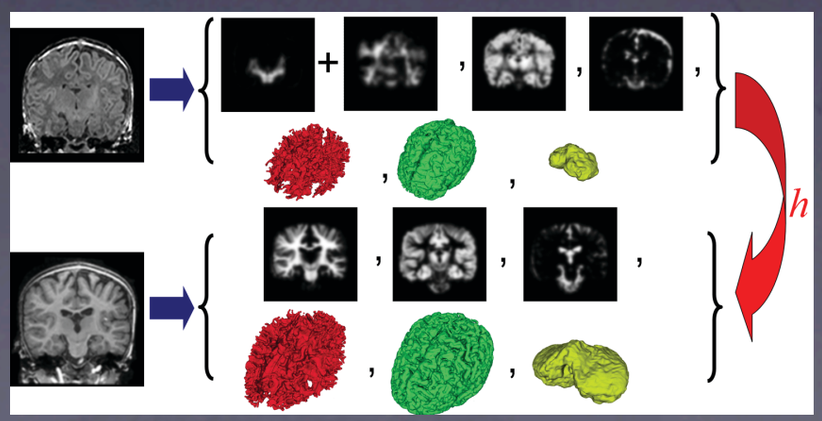


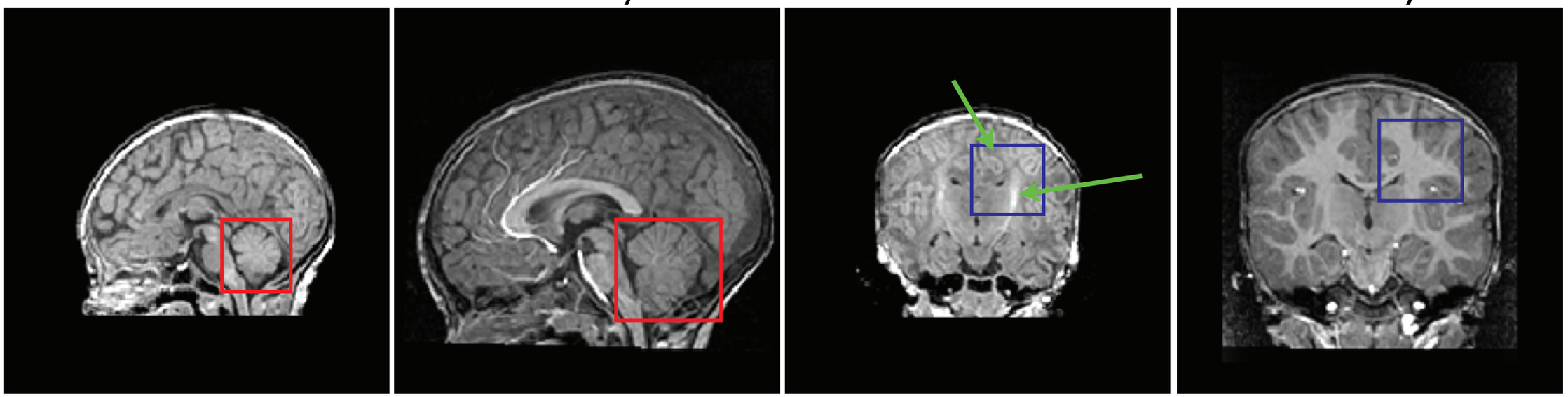
Image Registration Driven by Combined Probabilistic and Geometric Descriptors
MICCAI 2010
Linh Ha, Marcel Prastawa, Guido Gerig, John H. Gilmore, Claudio T. Silva and Sarang Joshi

How to incorporate extra information (surfaces, lines...) to improve registration quality ?

Early brain development
Registration challenge

Subjects

180 - two weeks | 180 - two years | 180 - two weeks | 180 - two years



- Total volume grows 115%
- Cerebellum grows 235%

a. Large-scale deformation

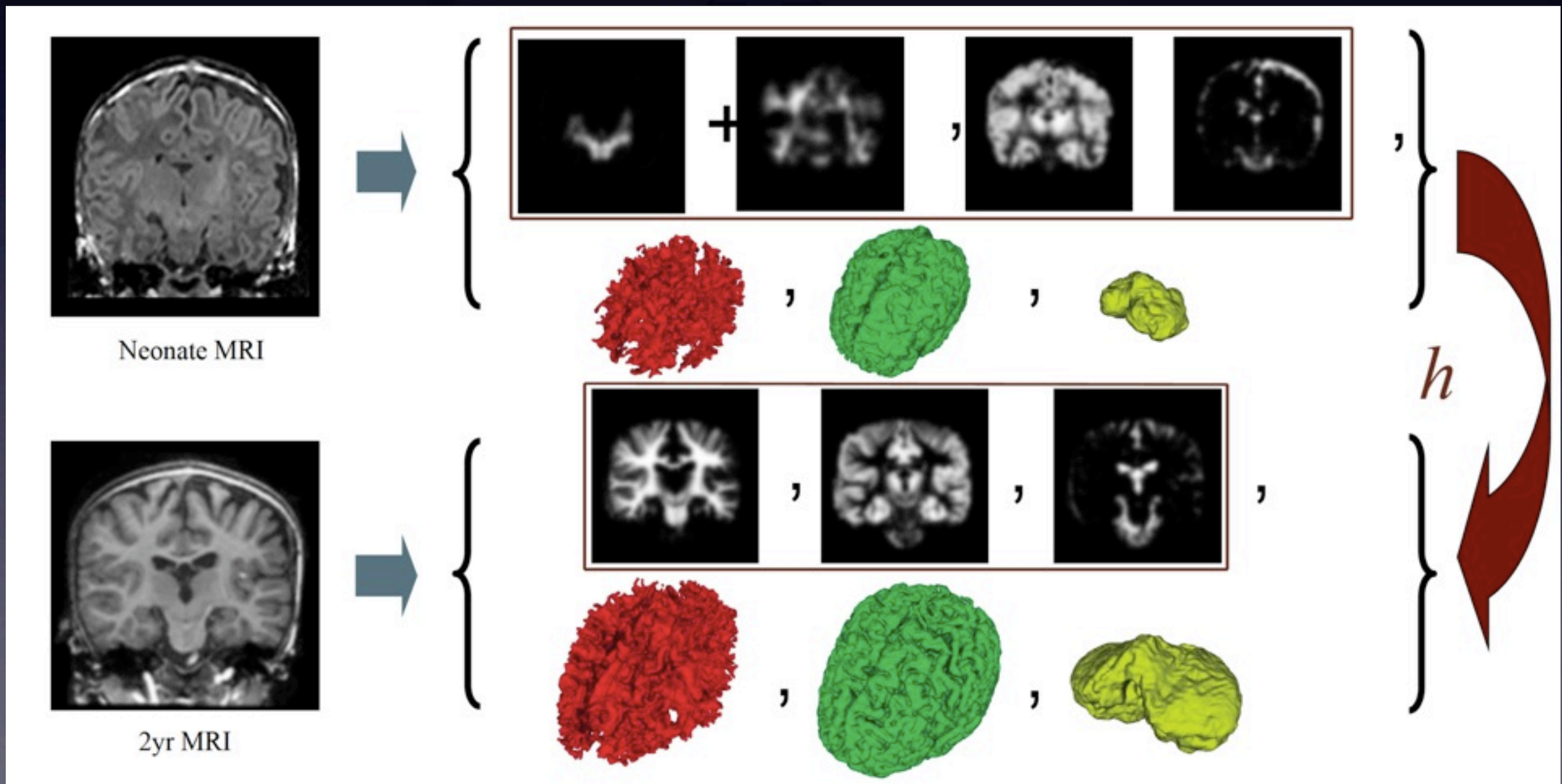
Two-intensity distribution of wm | One intensity distribution of wm

b. Intensity distribution change

- We need robust registration
  - Handle large deformation
  - Don't rely on raw intensity measurement

Multi-compartment Registration

- Represent brain anatomies using "Multi-compartment model"



Currents Norm

Given two anatomies $\mathcal{A}_1, \mathcal{A}_2$ find a transformation *h* that minimizes

$$\hat{h} = \arg\min_h E(h \cdot \mathcal{A}_1, \mathcal{A}_2)^2 + D(h, e)^2$$

Metric between anatomies

$$\sum_{c=1}^{N_c} \|p_{1,c}(x) - p_{2,c}(x)\|_k^{L^2} + \sum_{j=1}^{N_s} \|[\mathcal{M}_{1,j}(2) - \mathcal{M}_{2,j}(2)]\|_k^2$$

Metric between surfaces by using the "currents norm"

$$\|[\mathcal{M}(2)]\|_k^2 = \sum_{f=1}^{N_f} \sum_{f'=1}^{N_f} \langle \eta(f), \eta(f') \rangle \, k(c(f), c(f'))$$

$$\|[\mathcal{M}_{1,j}(2) - \mathcal{M}_{2,j}(2)]\|_k = \|[\mathcal{M}_1(2) \cup (-\mathcal{M}_2(2))]\|_k$$

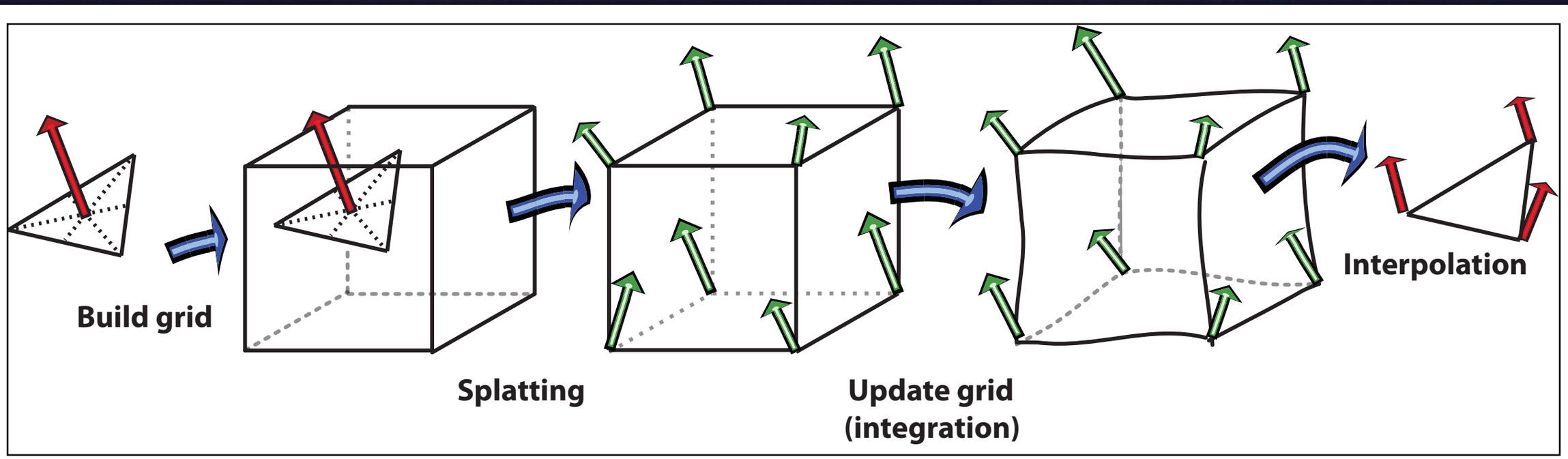
Particle Mesh Computation

PM computation

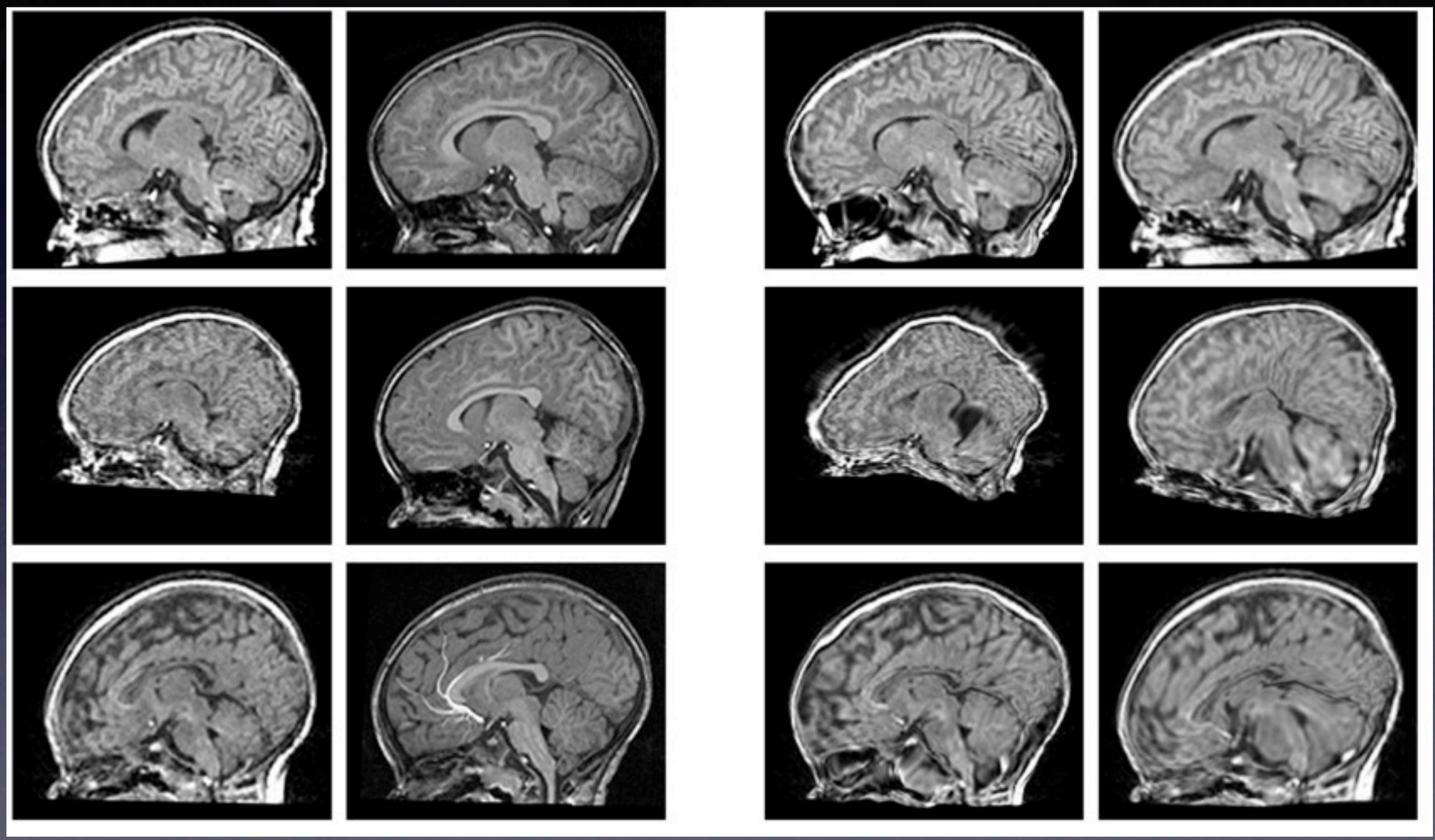
- O(N log N)
- GPU friendly - Fast

Brute force

- O(N_f x N_f)
- Memory, computation intensive - Slow



Registration result



Registration results of neonates mapped to two-year olds. From left to right: (a) neonatal T1 image after affine registration, (b) reference T1 image at 2-years, followed by (c) neonatal T1 after deformable mutual information registration using B-splines, and (d) our method.

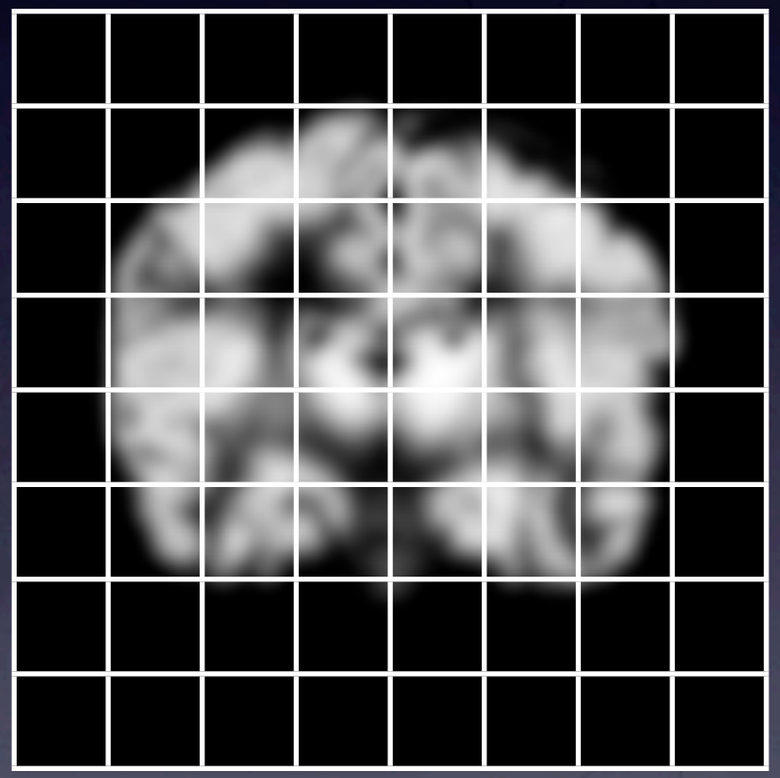
Quantitative performance

$$Overlap(h.S0, S2) = \frac{|h.S0 \cap S2|}{|S2|}$$

	Subject	0012	0102	0106	0121	0130	0146	0156	0174	0177	0180
White matter	MI	0.329	0.155	0.281	0.384	0.379	0.230	0.257	0.300	0.350	0.301
	P+G	0.497	0.397	0.442	0.453	0.482	0.414	0.461	0.440	0.478	0.442
Cerebellum	MI	0.755	0.212	0.588	0.515	0.732	0.820	0.713	0.569	0.631	0.777
	P+G	0.881	0.821	0.875	0.878	0.858	0.899	0.907	0.885	0.896	0.892

Dual Computation

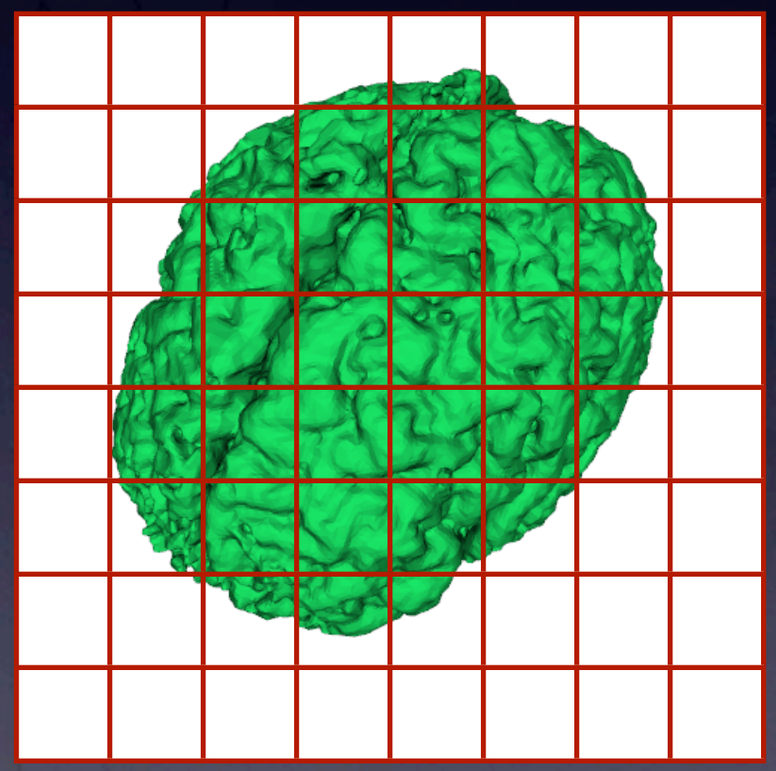
Regular Grid



Interpolation

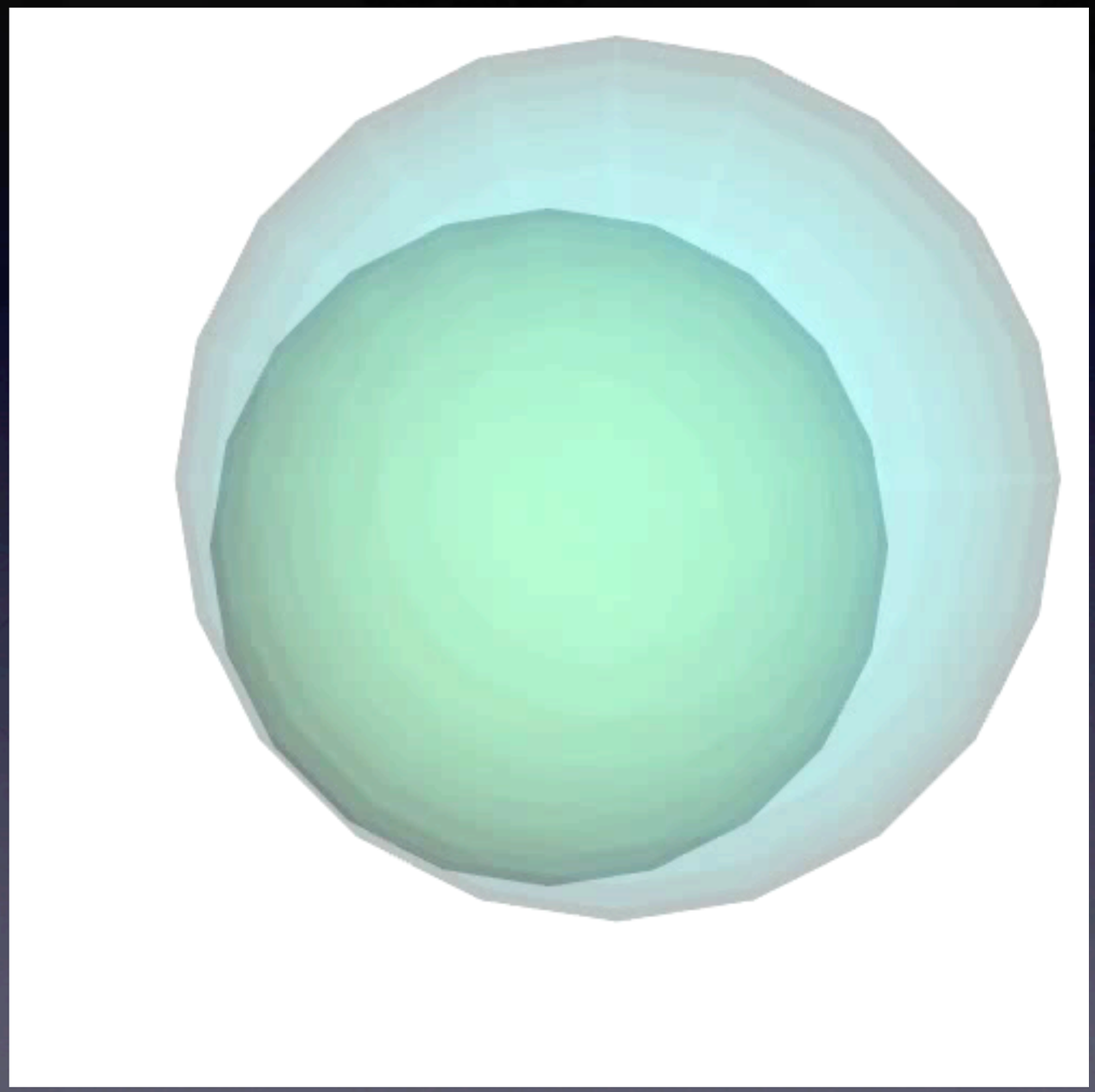
Splatting

Irregular Grid

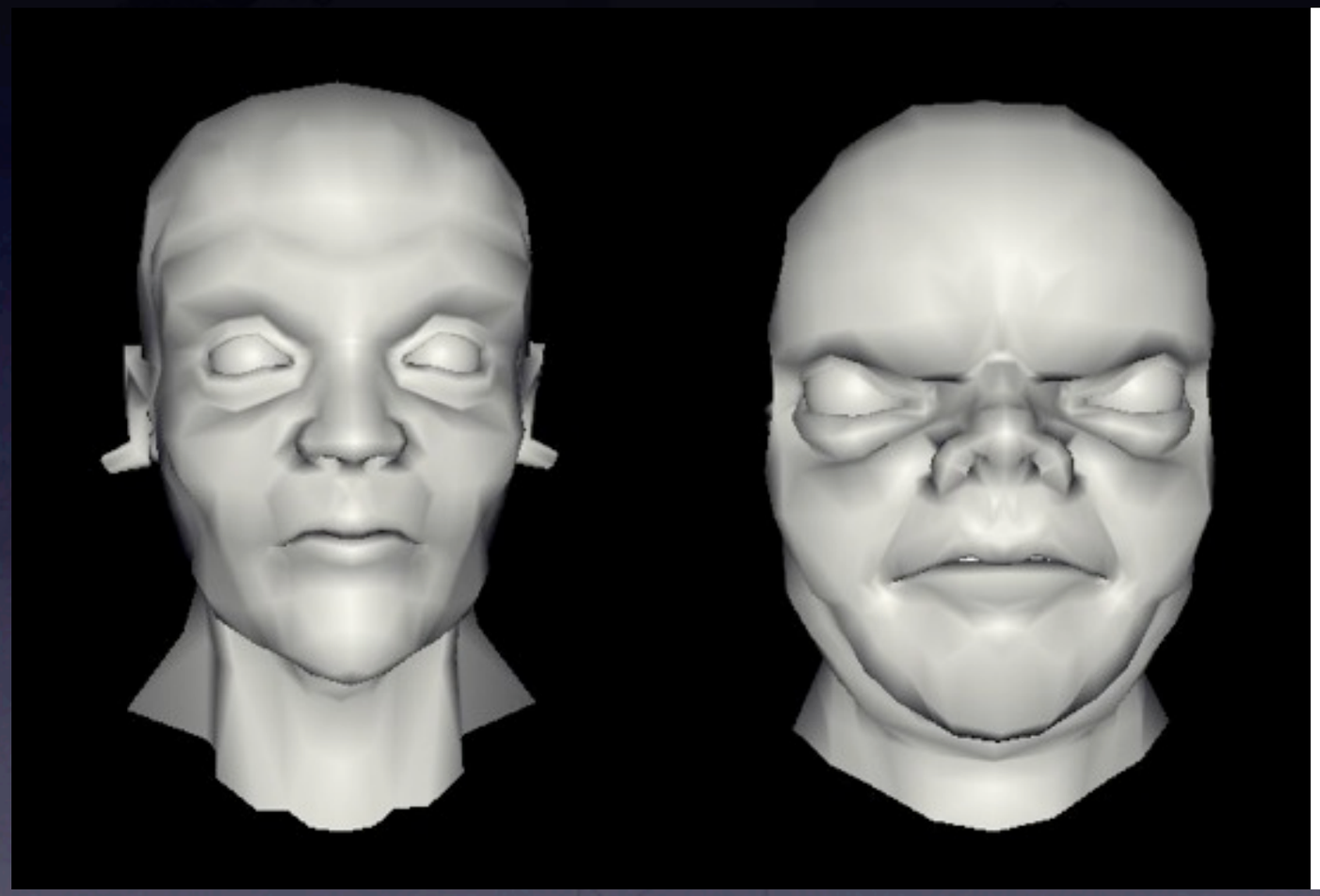


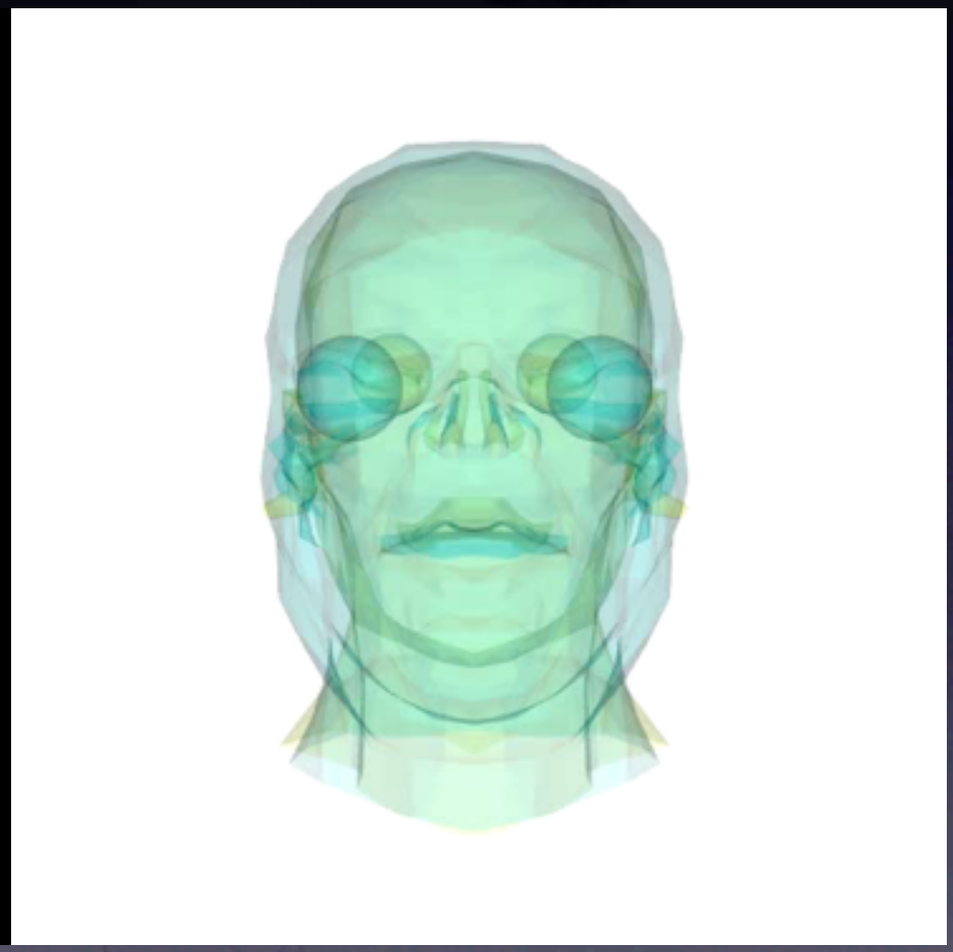
GPU-friendly domain

Demo:
Registration using geometry



Demo:
Registration using geometry





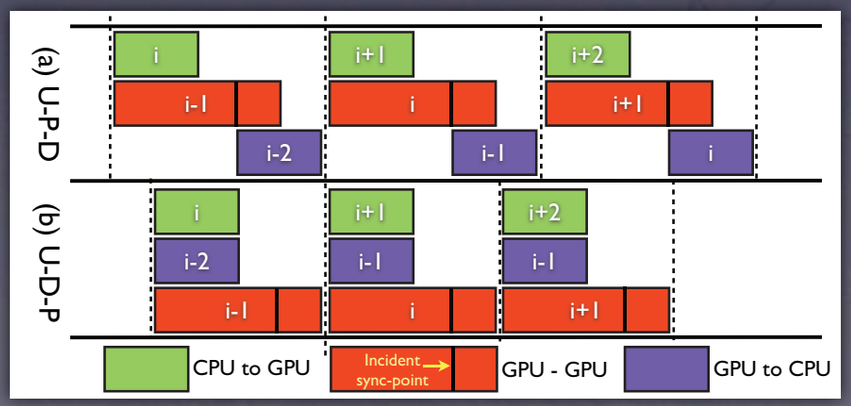
Part 2 : Conclusion

- What have been done so far
  - High performance processing framework
  - Solution for a registration challenge
    - Efficient computation on irregular domain
- Next
  - Define multi-image processing (MIP) operations
  - How to perform Out-of-Core MIP on GPUs
    - Asynchronous processing
    - Degrading challenge and solution

Outline

1. Motivation
2. Image Processing Framework
3. Combining Probabilistic and Geometries
4. Streaming Out-of-core MIP
5. Conclusion

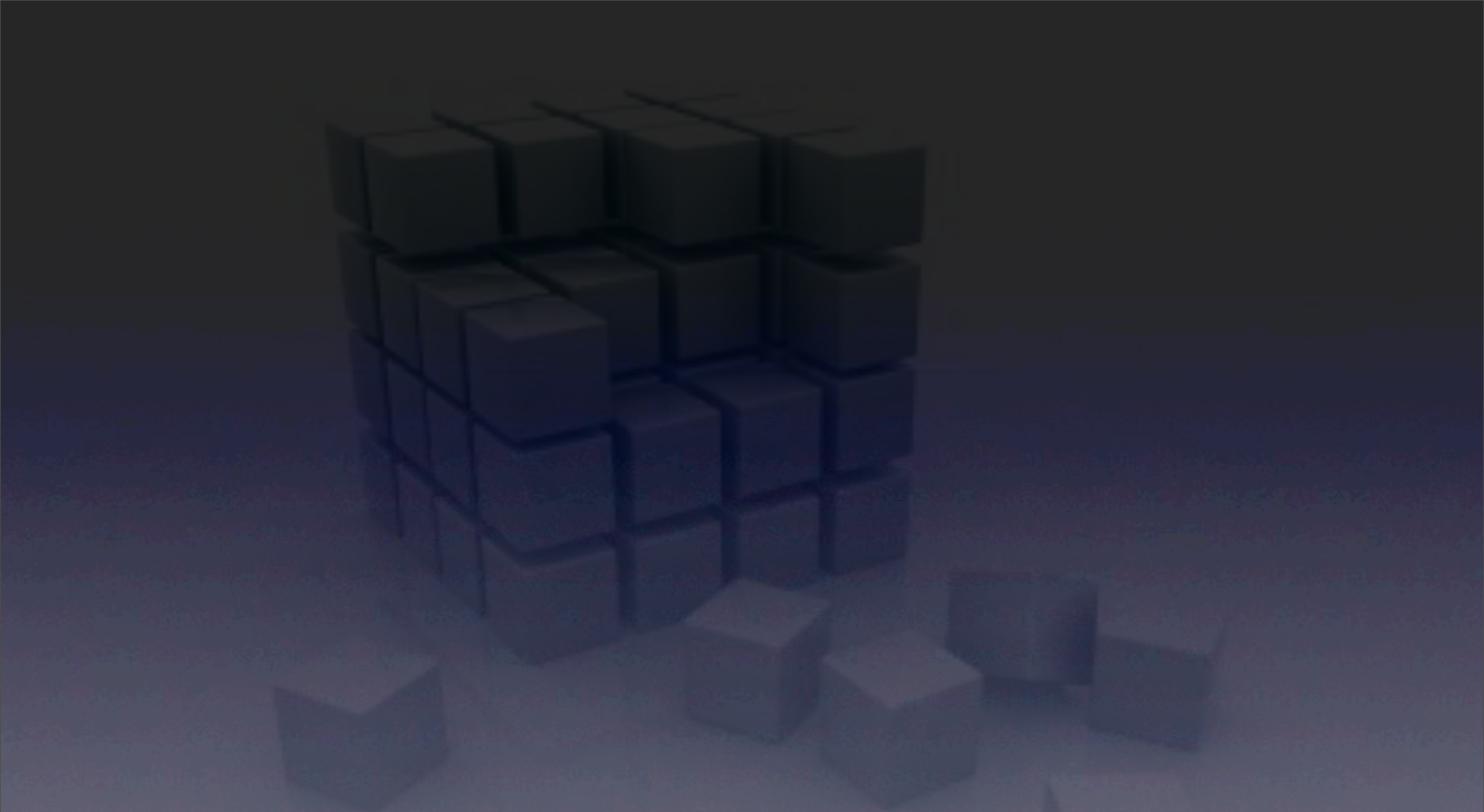
MIP = Multi-Image Processing



Optimal Multi-Image Processing Streaming Framework on Parallel Heterogeneous Systems
Submitted to EGPGV 2011
Linh Ha, Jens Kruger, Joao Comba, Sarang Joshi and Claudio T. Silva

Why we need Out-Of-Core ?

Computing Level	Resources	Accessibility	Applications
GPU Cluster	Abundant	Restricted	Few
Multi-GPU Workstation	Restricted	Feasible	Many
Single GPU desktop	Limited	Widely available	Plenty

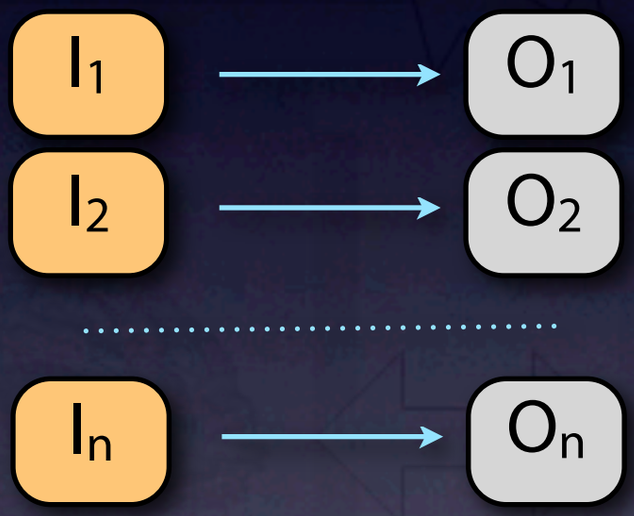


What are the building blocks of a multi-image processing framework ?

Multi-image Operations

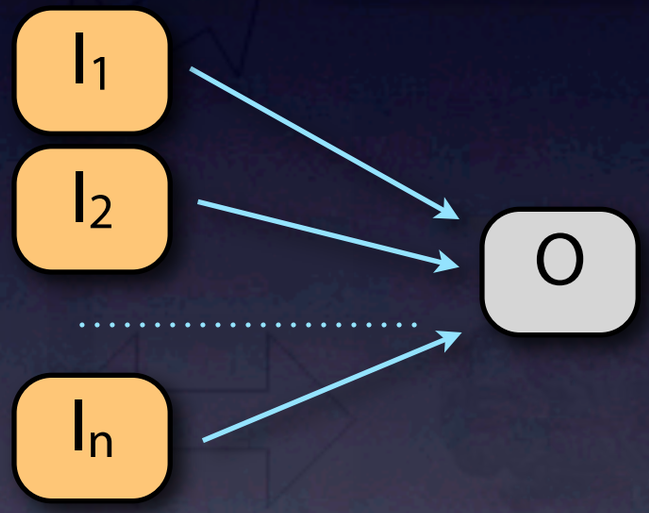
(Flynn's Taxonomy classification)

Multi-Input Multi-Output (MIMO)



- add, mul, sub, divide, normalized
- convolution, filter

Multi-Input Single-Output (MISO)

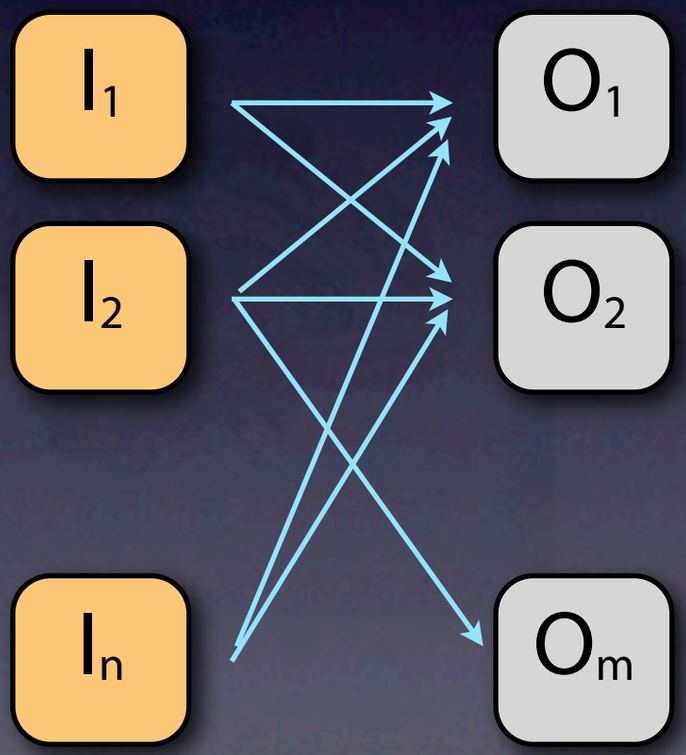


- max, min, range
- average, accumulate

All multi-image operators are presented either by MIMO or MISO

Complex MIMO

Complex
n-inputs & m-outputs

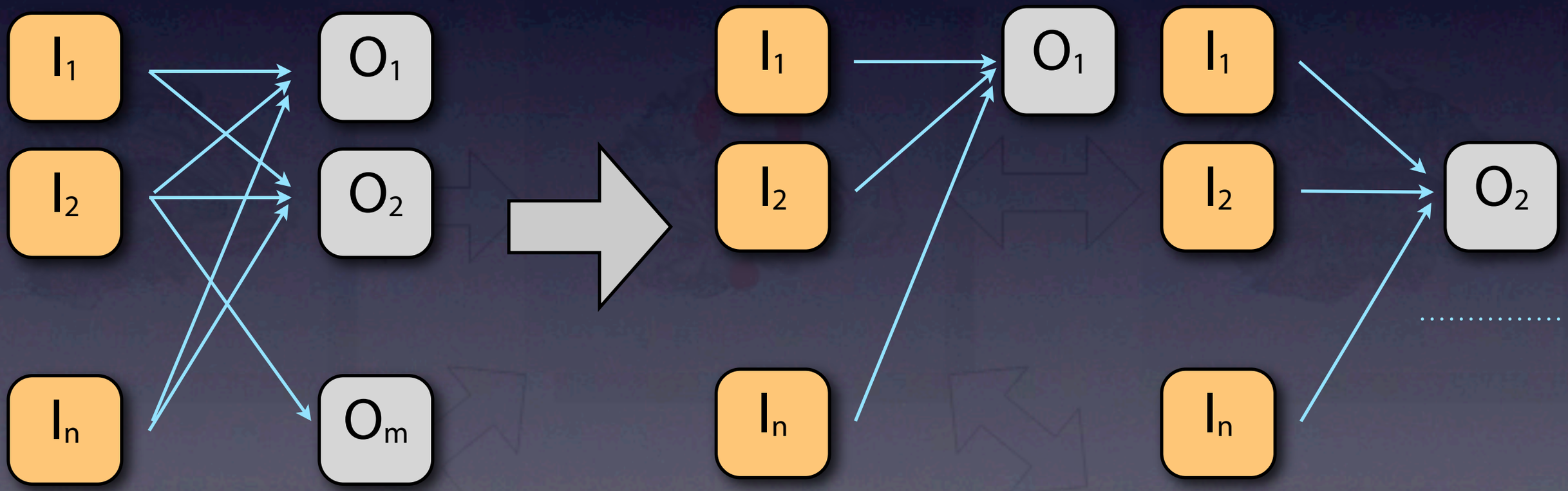


Complex MIMO

Complex
n-inputs & m-outputs

Decomposition

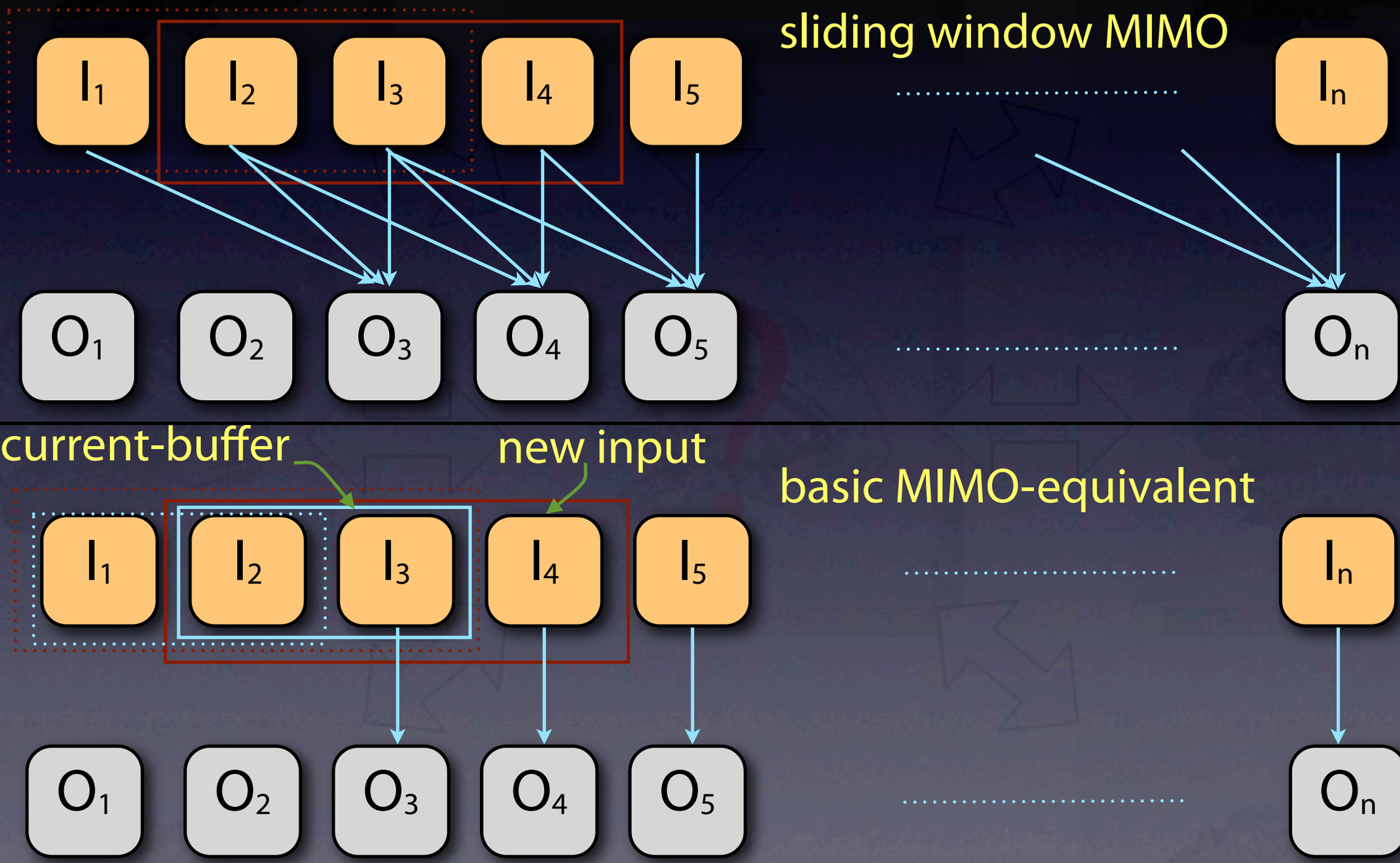
Basic
MIMO & MISO



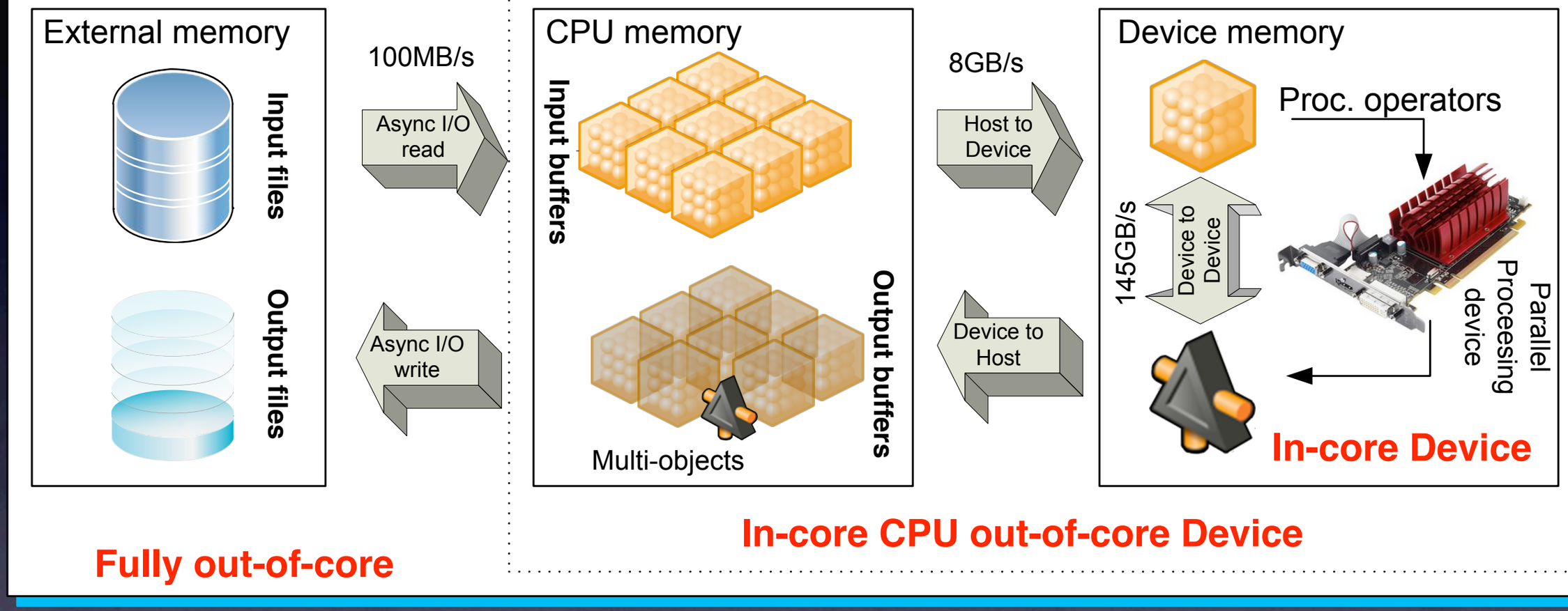
m instances of MISO

Sliding window MIMO complex

(Functions with memory stages)



Out-of-Core MIP Challenges



	External	CPU system memory	GPU global memory
Mem Size	n-TB	10-256GB	512MB-6GB
Bandwidth	100MB/s	3-8 GB/s	140-200GB/s

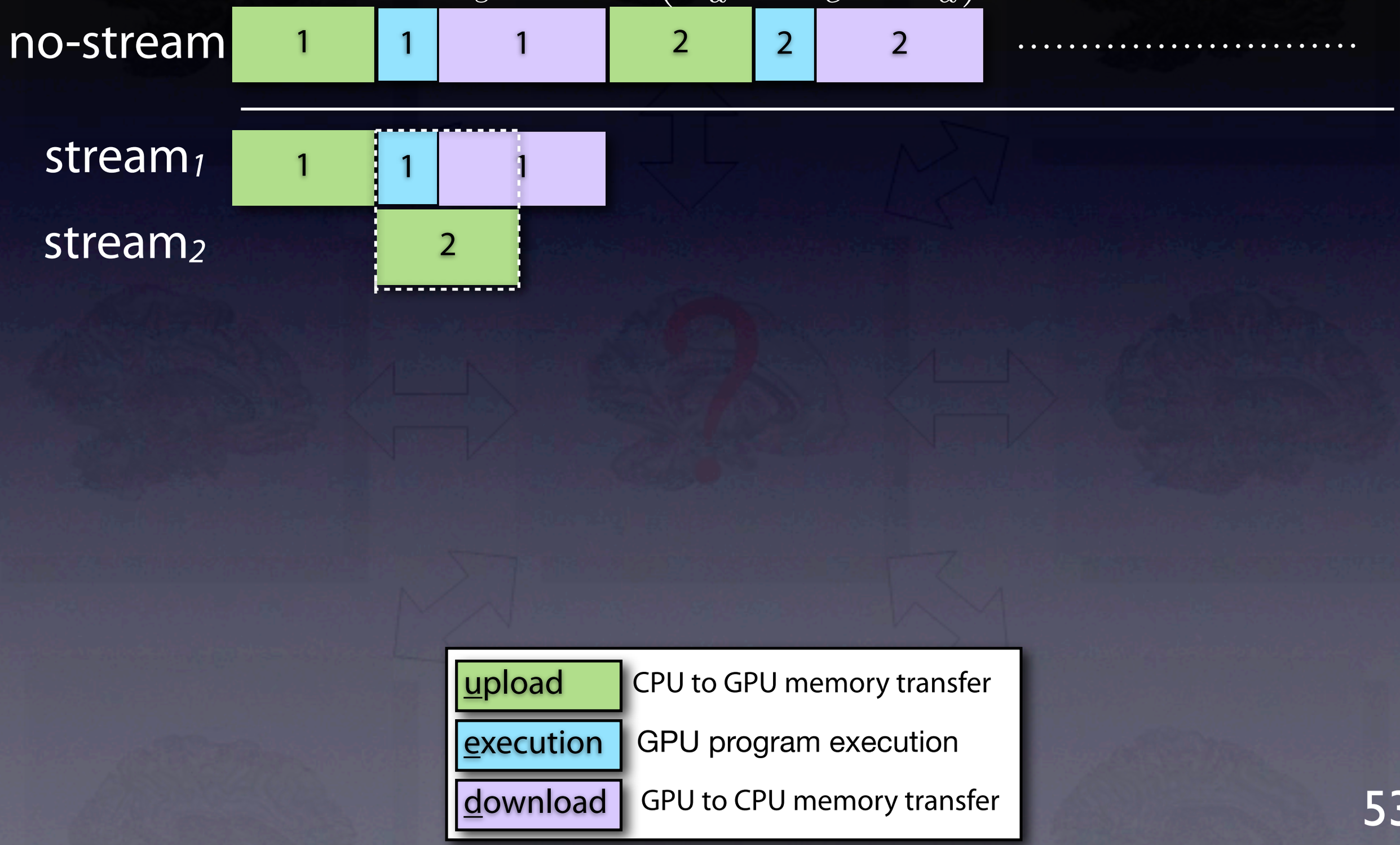
1-mag difference

2-mags difference

Data transfer becomes bottleneck ! Solution: Asynchronous processing

Asynchronous versus Synchronous

$$T_s = n \times (T_u + T_e + T_d)$$



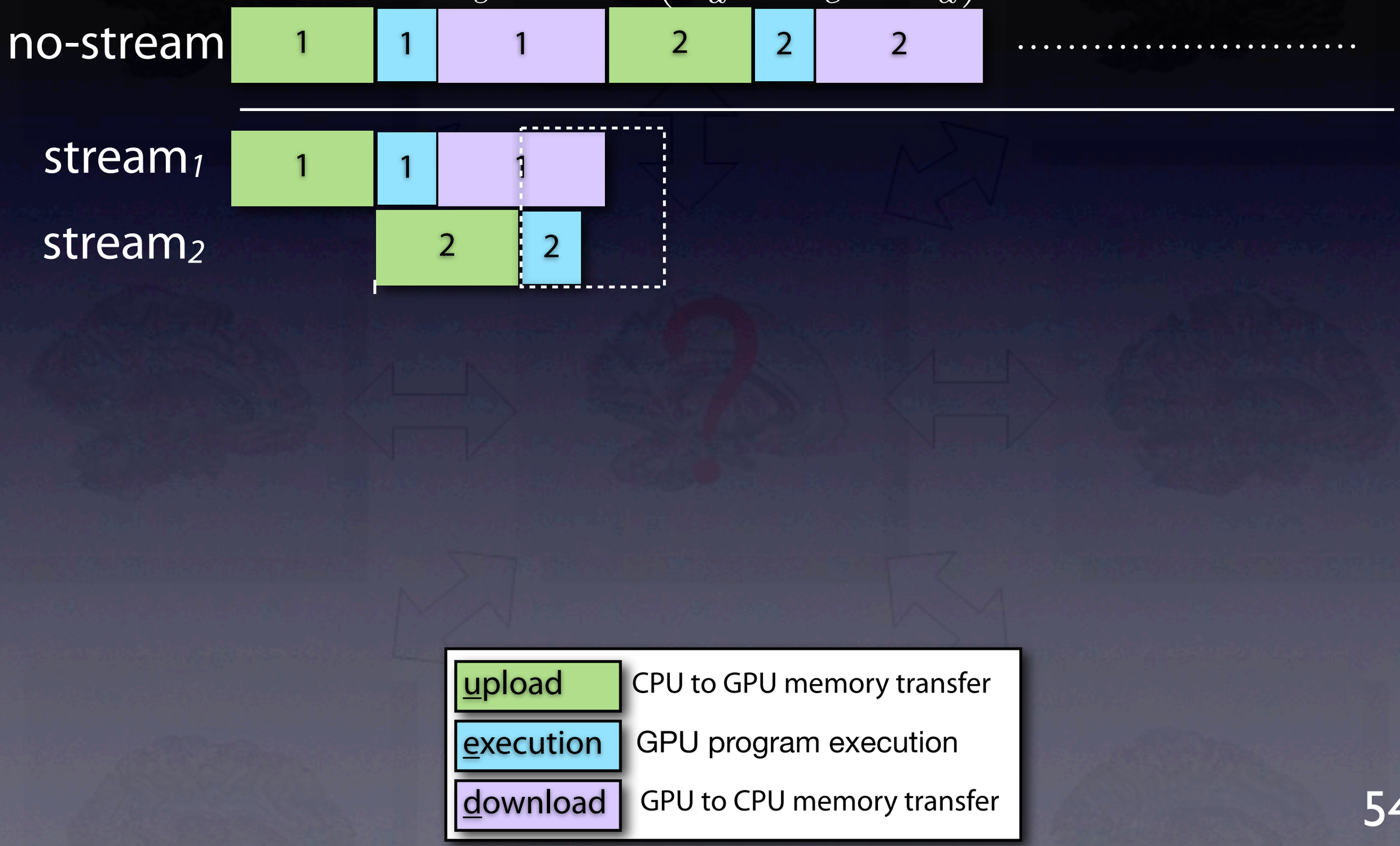
upload — CPU to GPU memory transfer

execution — GPU program execution

download — GPU to CPU memory transfer

Asynchronous versus Synchronous

$$T_s = n \times (T_u + T_e + T_d)$$



upload — CPU to GPU memory transfer

execution — GPU program execution

download — GPU to CPU memory transfer

Asynchronous versus Synchronous

$$T_s = n \times (T_u + T_e + T_d)$$

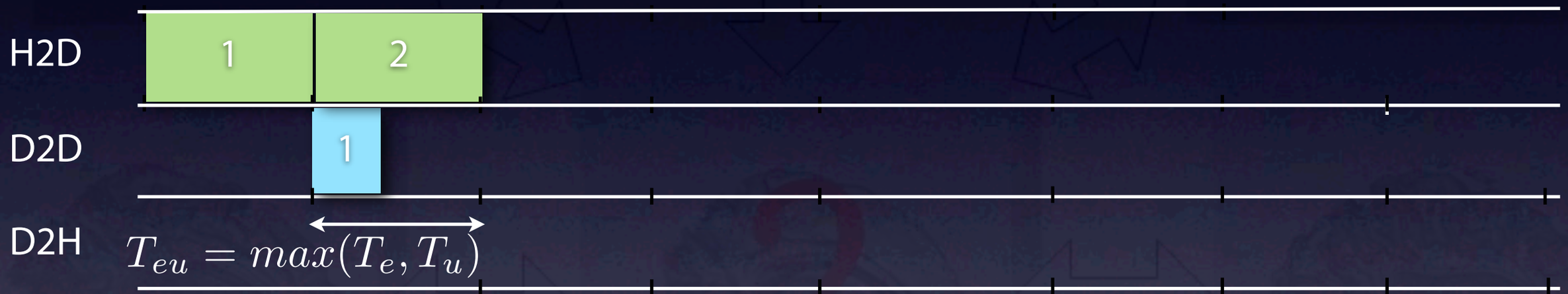
$$\leftarrow \quad T_a = n \times T_{max} + (T_u + T_e + T_d - T_{max}) \quad \rightarrow$$

$$T_{max} = max(T_u, T_e, T_d)$$

upload	CPU to GPU memory transfer
execution	GPU program execution
download	GPU to CPU memory transfer

Hardware-aware Execution

Stream is assigned after the hardware execution unit



upload	CPU to GPU memory transfer
execution	GPU program execution
download	GPU to CPU memory transfer

The system with two data transfer + one execution units

Hardware-aware Execution

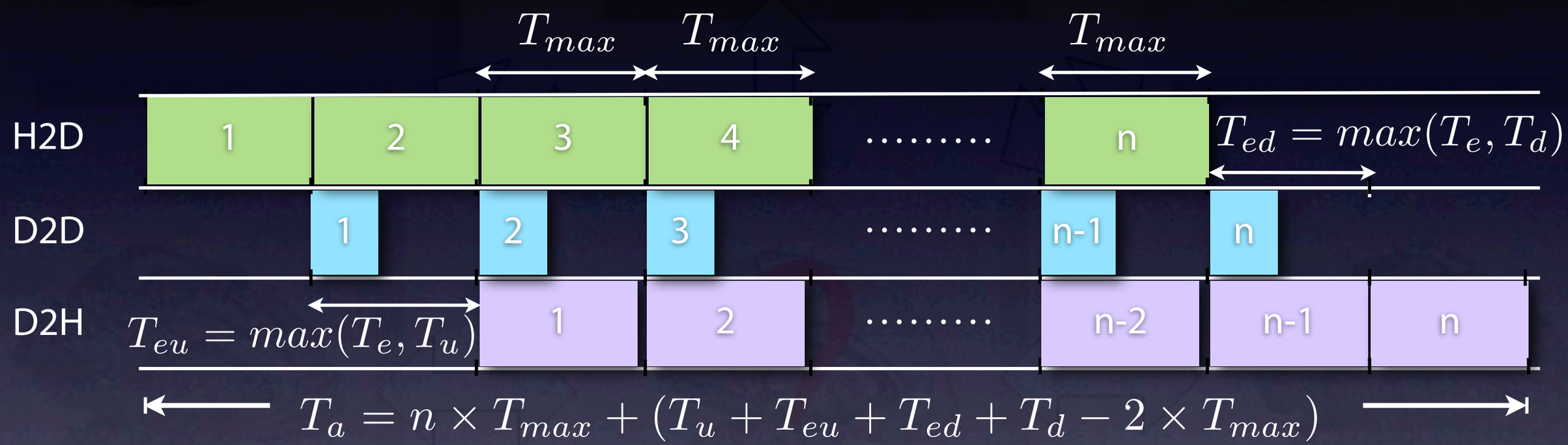
Stream is assigned after the hardware execution unit

upload	CPU to GPU memory transfer
execution	GPU program execution
download	GPU to CPU memory transfer

The system with two data transfer + one execution units

Hardware-aware Execution

Stream is assigned after the hardware execution unit

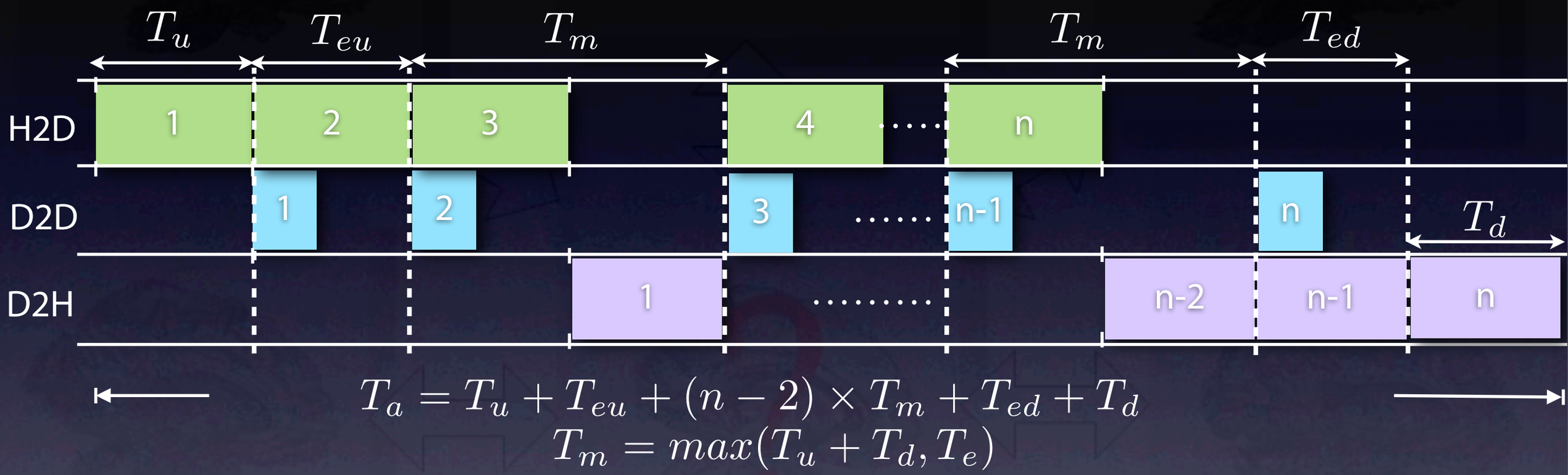


upload	CPU to GPU memory transfer
execution	GPU program execution
download	GPU to CPU memory transfer

The system with two data transfer + one execution units

Hardware-independent Execution

Stream is assigned after the function stage

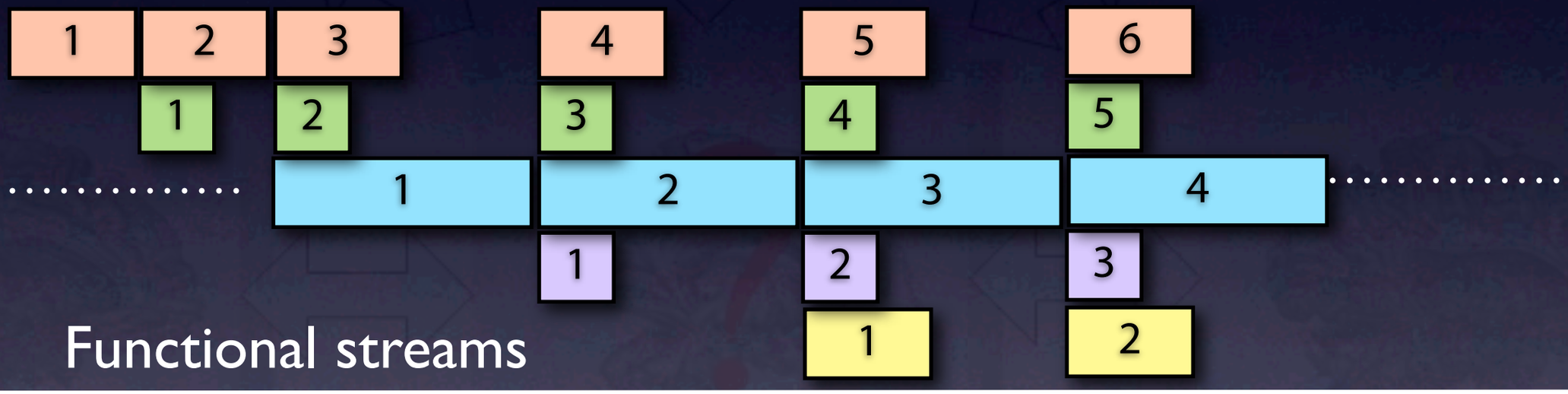


upload	CPU to GPU memory transfer
execution	GPU program execution
download	GPU to CPU memory transfer

Hardware-independent model is proven to be optimal (EGPGV 2011)

Extension to Full Out-of-Core MIP

Full Out-Of-Core = Out-Of-Core GPU In-core CPU + Out-Of-Core CPU transfers

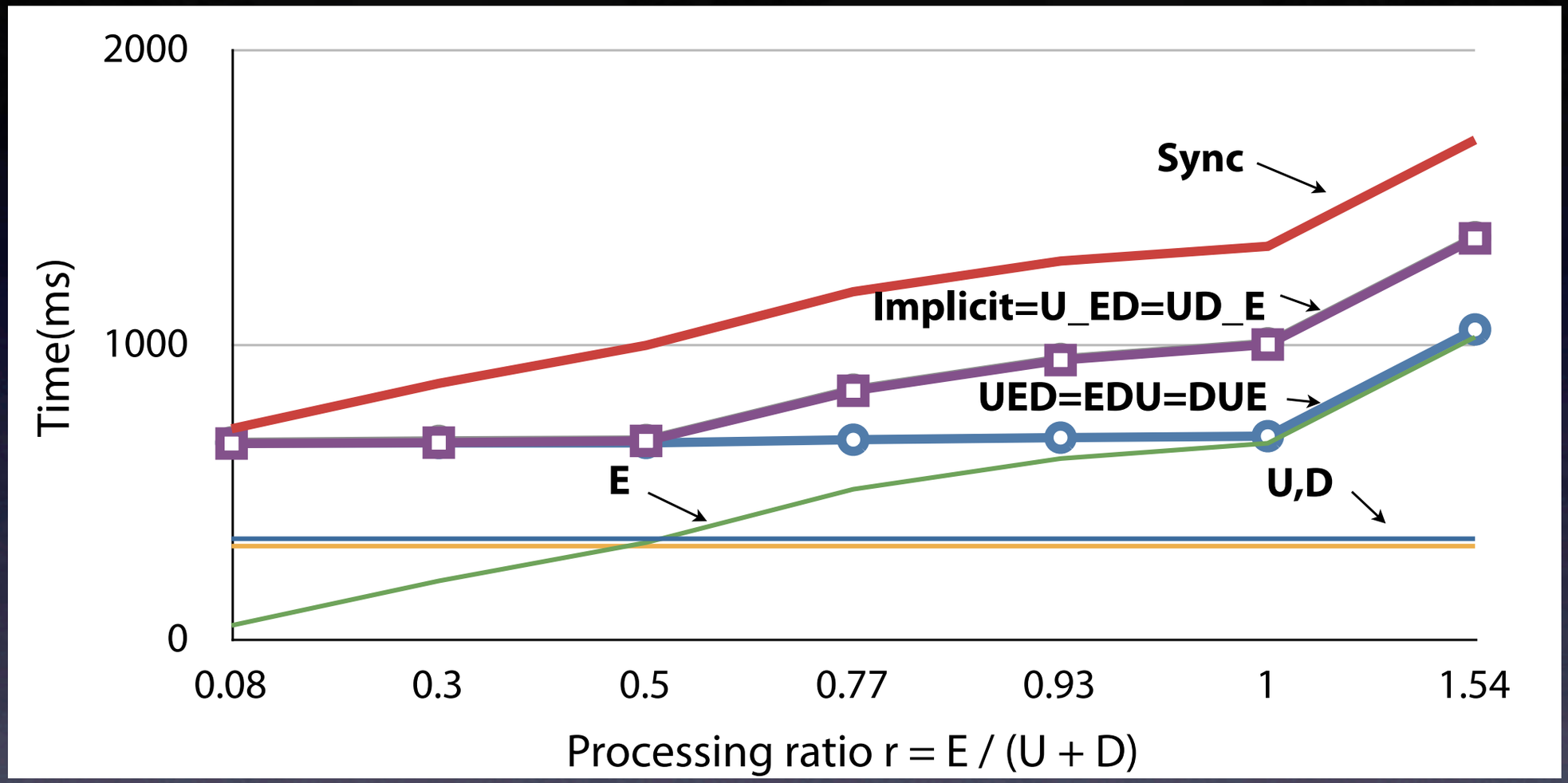


Functional streams

upload	CPU to GPU memory transfer	d-upload	disk to CPU memory transfer
execution	GPU program execution	d-download	CPU to disk memory transfer
download	GPU to CPU memory transfer		

The hardware independent extension to full out-of-core MIP is proven to be optimal

Ideal condition runtime

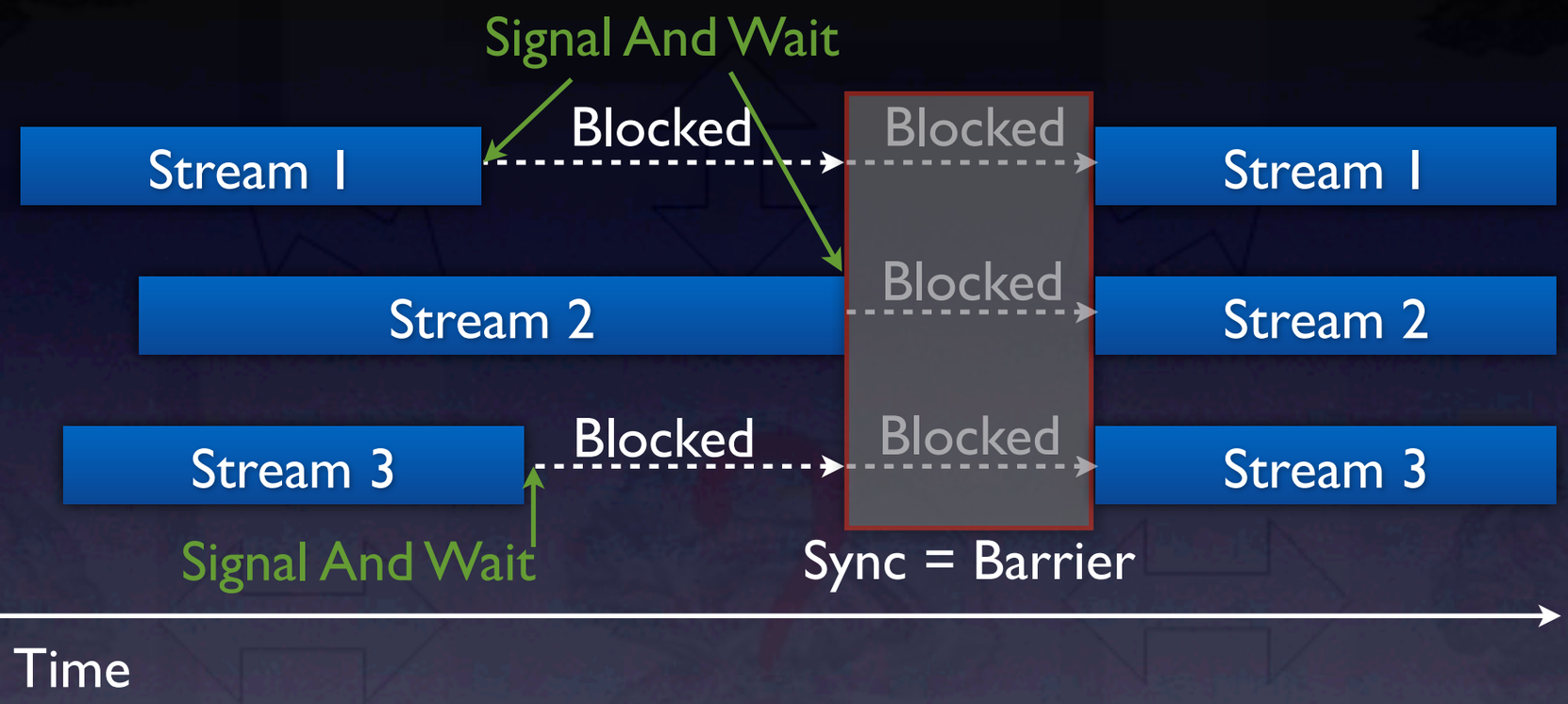


r = E/(U+D)	<<1	1	>>1
Function type	Transfer dominant	Balance	Processing dominent
Speed up benefit	Low	High (2)	Low

Asynchronous Processing in Practice: Degrading Conditions

Forced synchronization

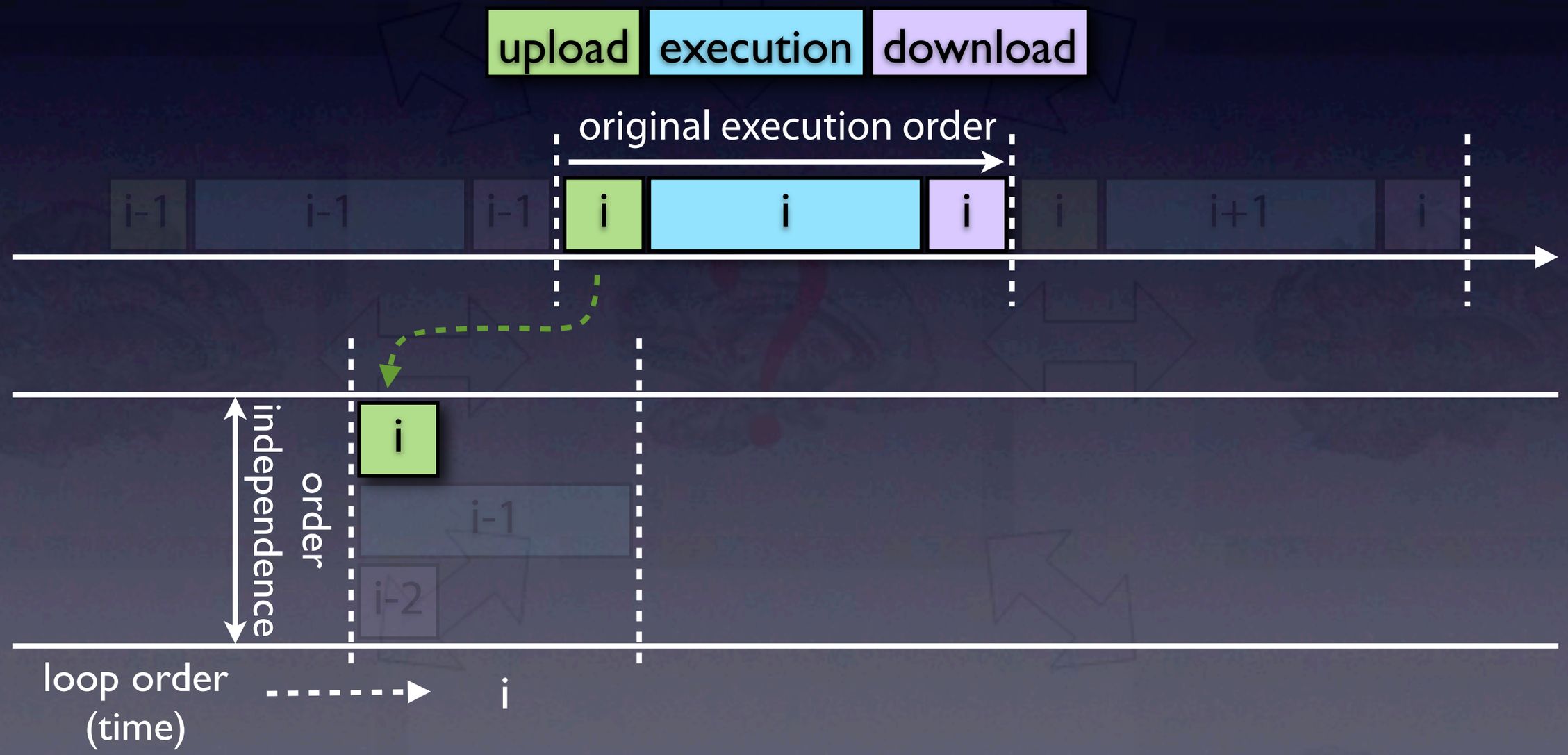
required to preserve semantic order of a program



- Reasons
  - Synchronization calls
  - Call to external functions without asynchronous support
  - Asynchronous mismatch
  - Cross-stream functions (in-stream synchronization)

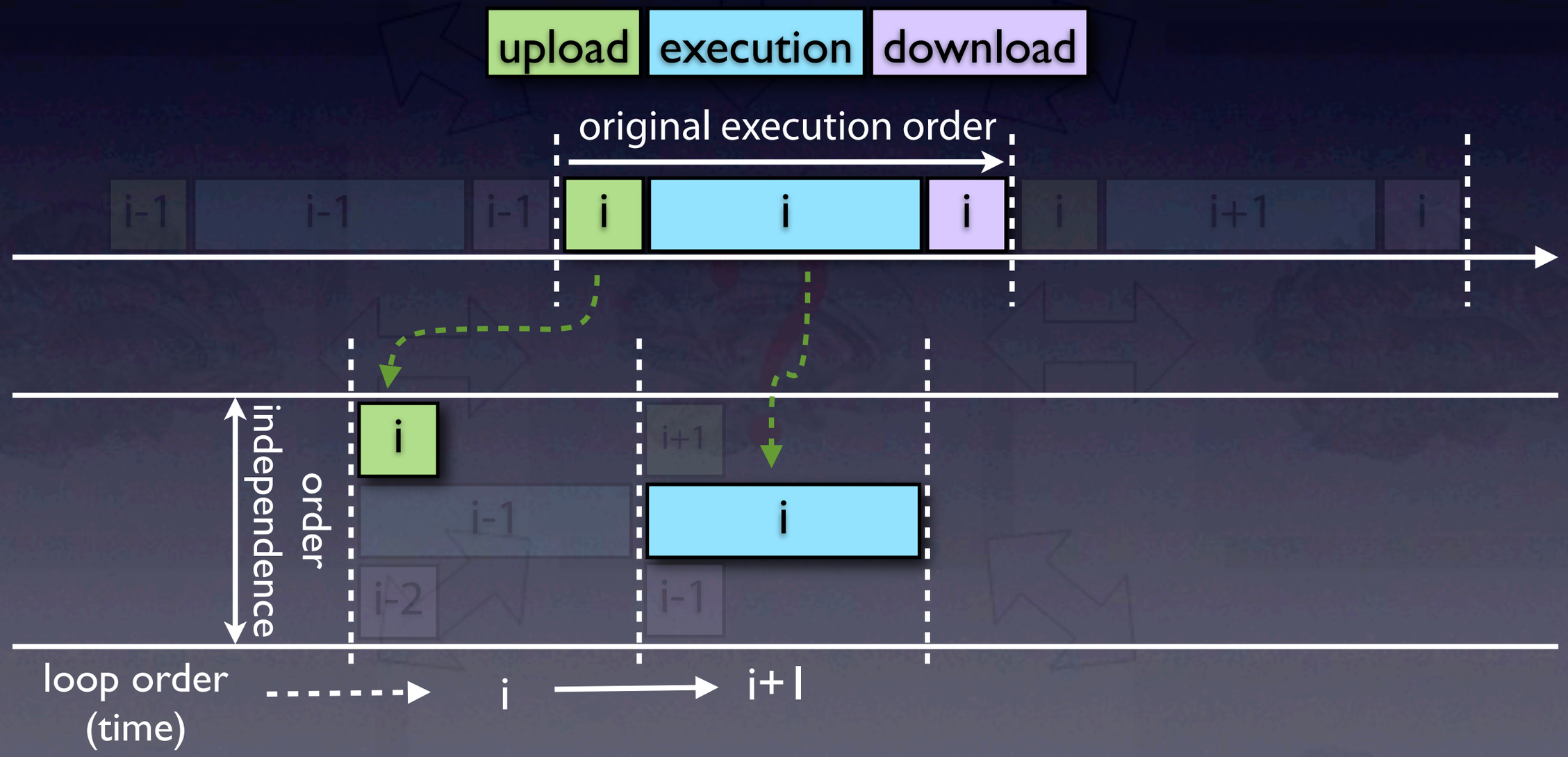
Order-independent Streaming Mode

- Reordering is available for stages of different images



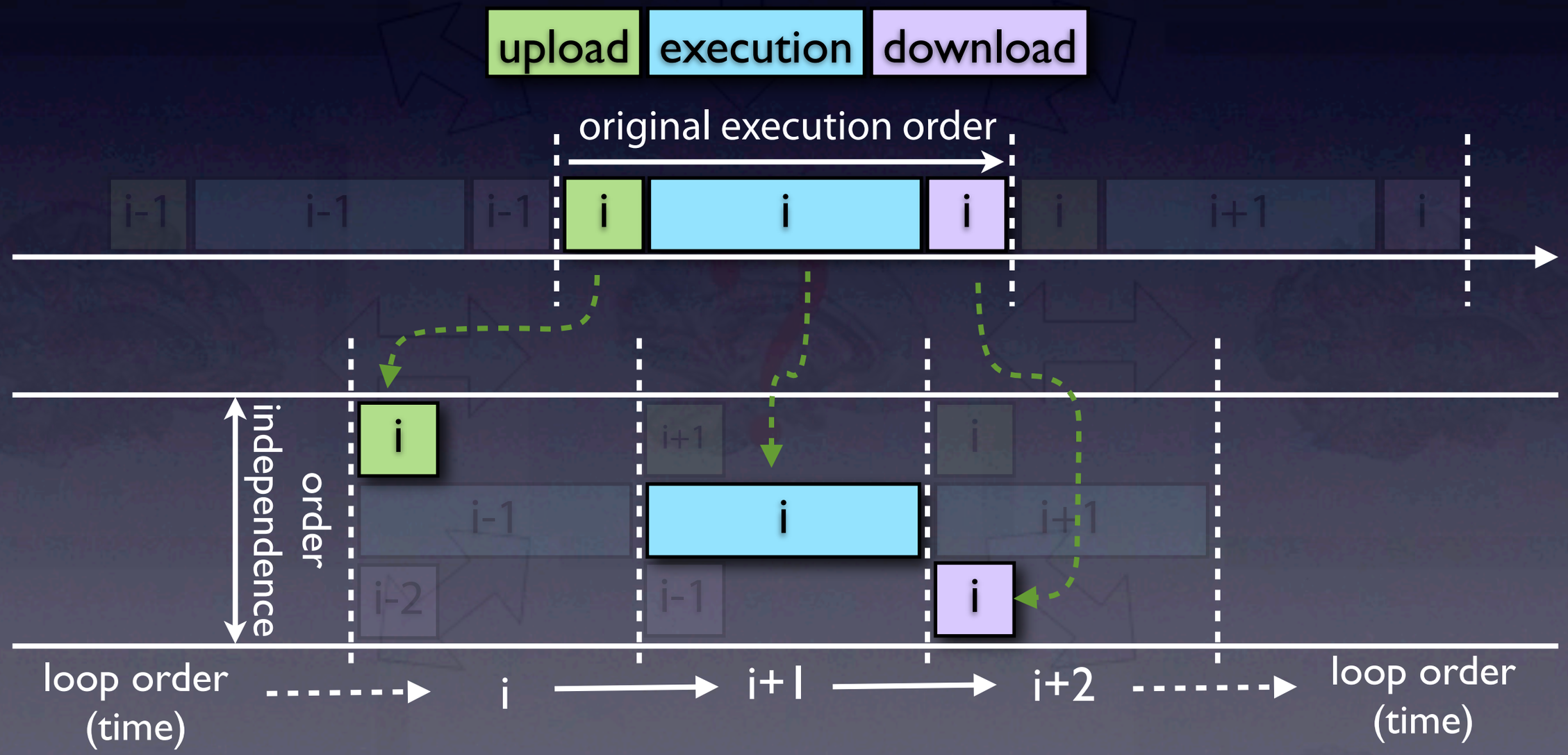
Order-independent Streaming Mode

- Reordering is available for stages of different images



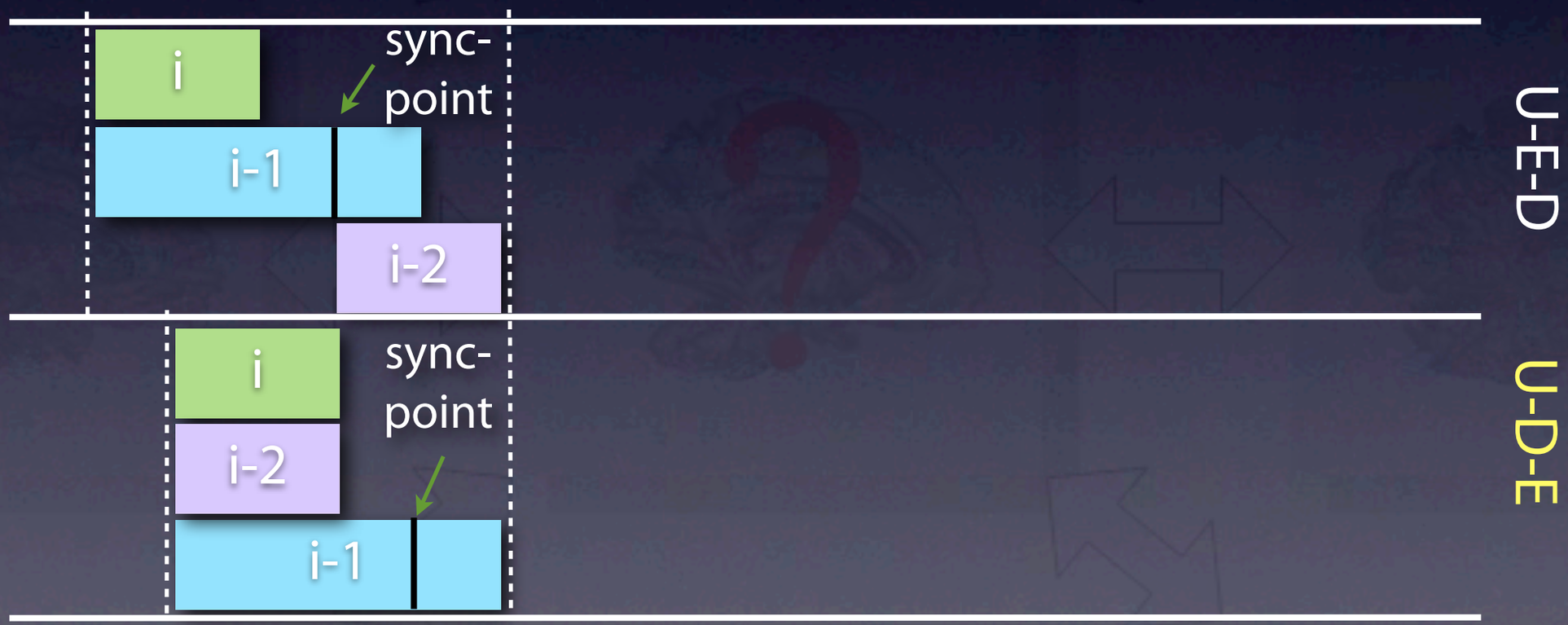
Order-independent Streaming Mode

- Reordering is available for stages of different images



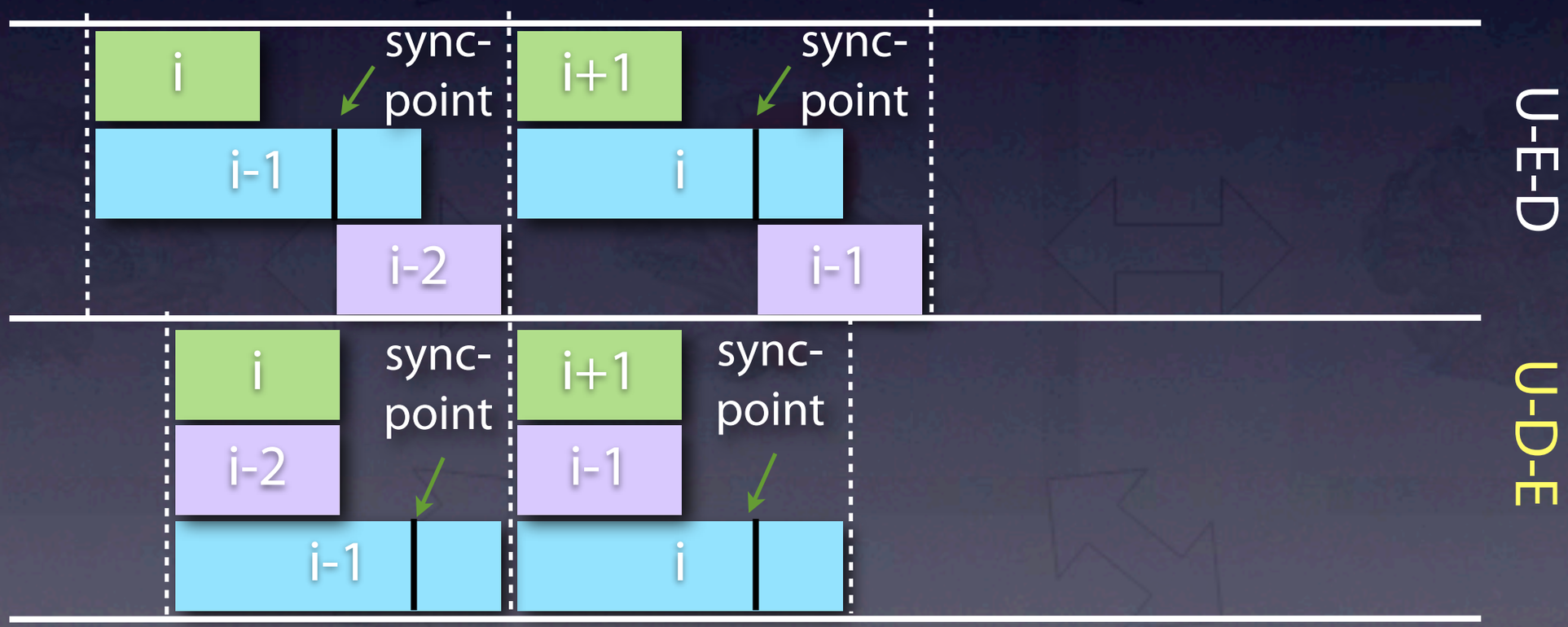
Solution for forced synchronization

- Reordering reduces penalty of forced synchronization in execution stage



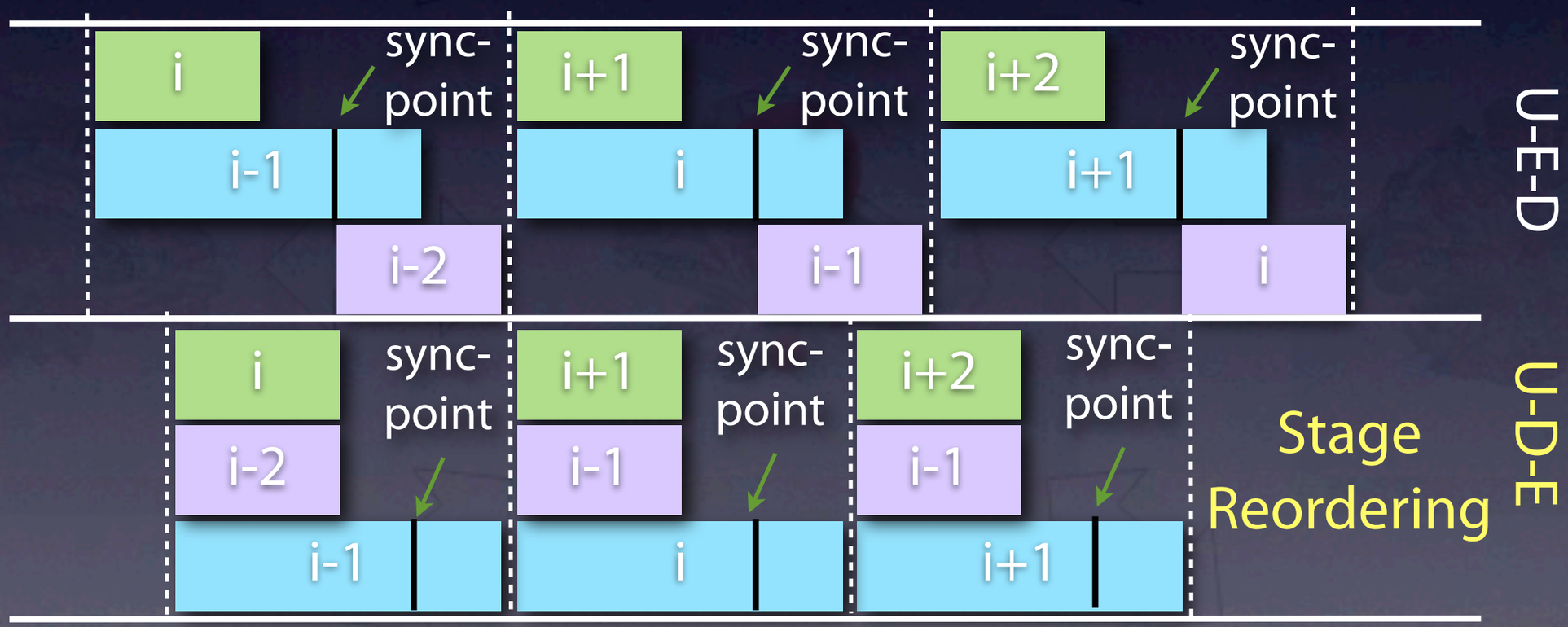
Solution for forced synchronization

- Reordering reduces penalty of forced synchronization in execution stage

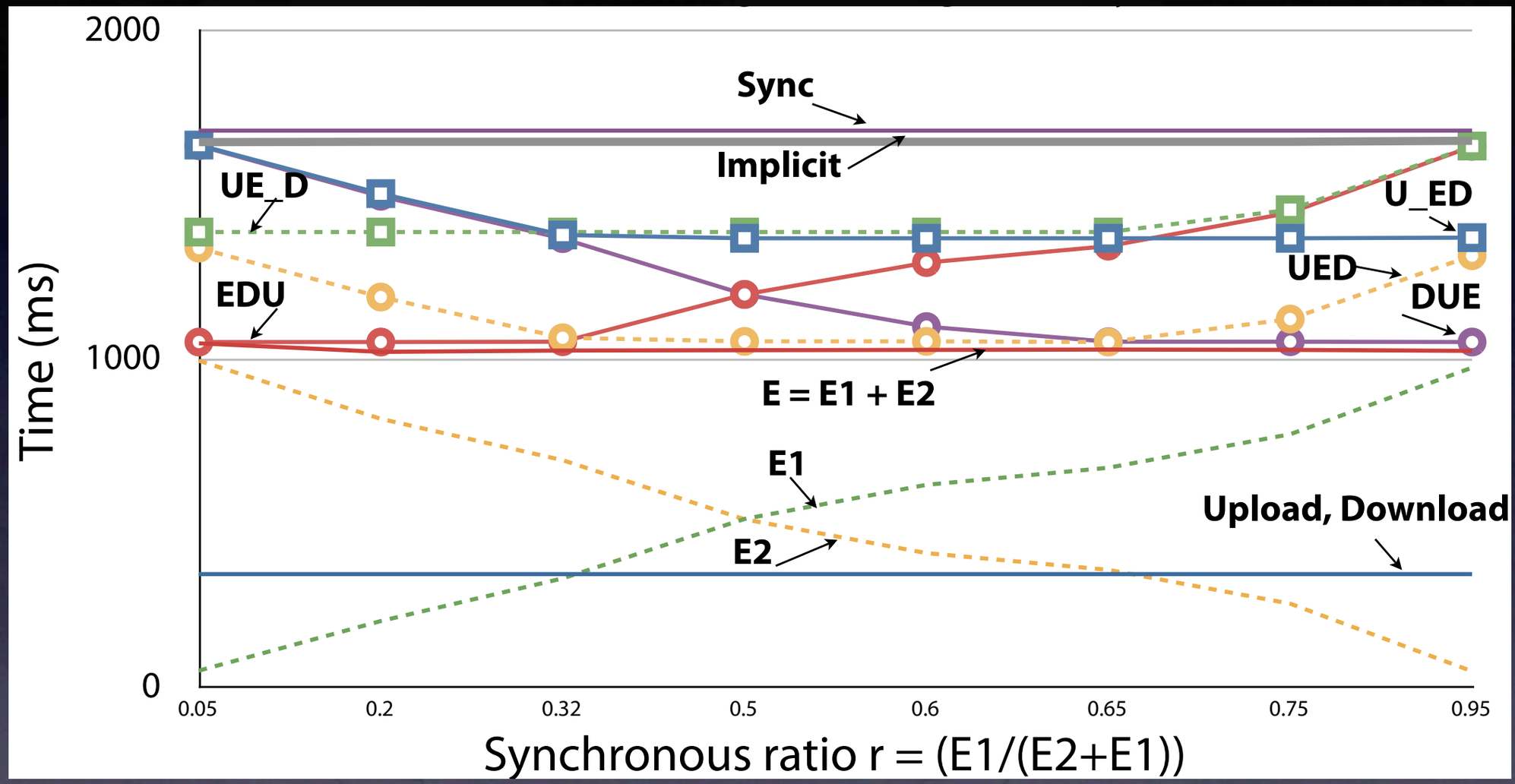


Solution for forced synchronization

- Reordering reduces penalty of forced synchronization in execution stage



Degrading condition runtime

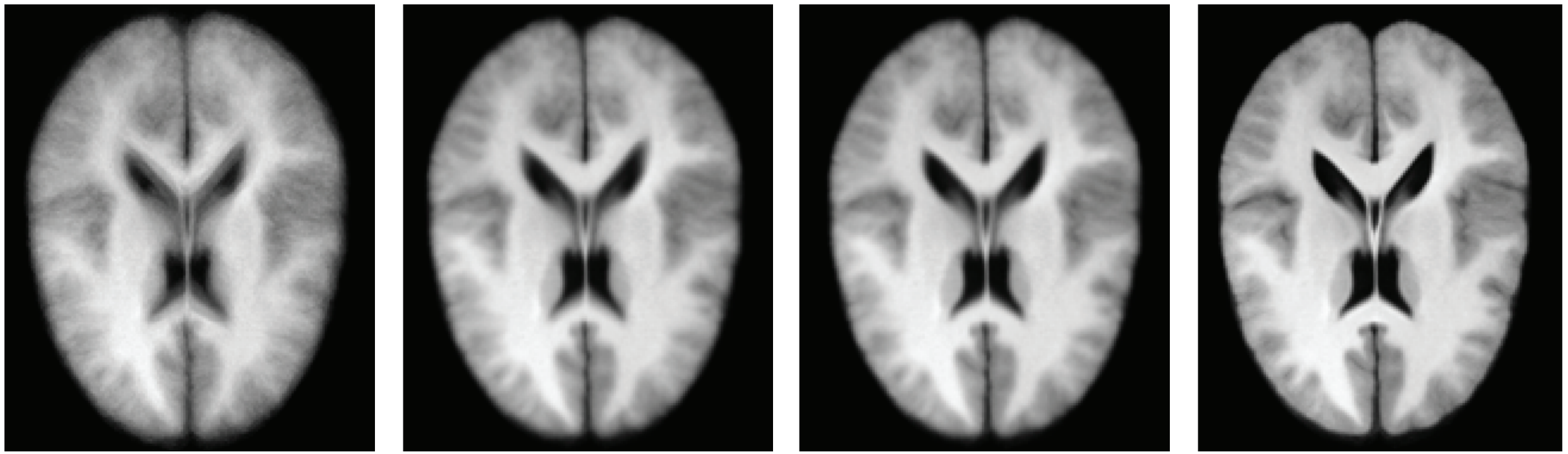


Functions in practice

Function	U	E	D	Sync	Impl	Hrd-aware	Hrd-indp
Max	347	13	0	360	349	349	349
Energy	692	20	0	710	698	700	698
Averaging	347	20	11	378	360	363	361
Normalization	347	28	322	694	696	687	677
Gaussian	347	431	322	1099	735	770	678
Atlas	201446	213423	1359583	555204	NA	372567	340356

Note: Full Out-Of-Core Atlas Construction takes 1h40 minutes with Async. execution and 2 hours with Sync. execution to finish

Approximate Atlas versus Complete Atlas

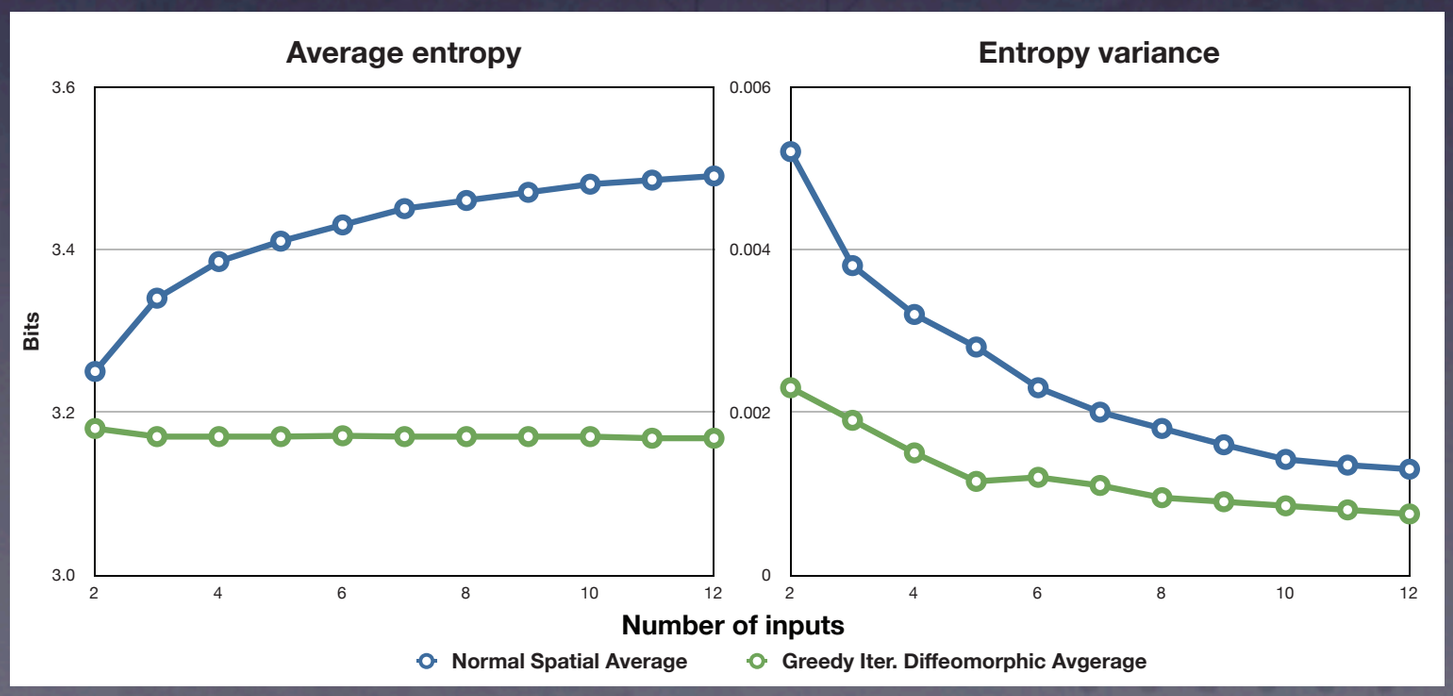


a. Intensity average image

b. Random-permuation regression atlas construction with cohort = 3

c. Random-permuation regression atlas construction with cohort = 9

d. Full-diffeomorphic atlas construction Number of images 156



Permutation regression

- Number of tests = 100
- Cohort = 3 takes 40 minutes
- Cohort = 9 takes 2 hours

Exact out-of-core computation takes 6 minutes

Conclusion

- The thesis presents
  - A high performance Image Processing Framework
    - Multi-scale implementation
    - Multi-GPUs and GPU-cluster implementation
  - A robust registration method combines probabilistic and geometries
    - Efficient computation on irregular domain
  - An optimal out-of-core MIP framework
- Future work
  - Extend diffeomorphic framework to handle cracks
  - Extend computation to irregular domain
  - Out-of-core processing for general data structure

Software package is available to download at

http://www.sci.utah.edu/software/13/370-atlaswerks.html

Acknowledgment

- Grants:
  - 5R01EB007688, MH064065, P41 RR023953 and NSF grant CNS-07511152
  - Vietnam Education Foundation
- Thank
  - Advisor: Cláudio T. Silva
  - Co-advisors: Sarang Joshi, Jens Krüger
  - Committee members: Joao Comba, P.Thomas Fletcher
  - Mentors at IBM, Exxon Mobile: James Klosowski, Wagner Correa, Mark Dobin, Dominique G.Gillard
  - VGC group: Huy, Emanuel, Erik, Steven, Carlos, John, Tiago, ...
  - Collaborators: Marcel Prastawa, Guido Gerig, Sam Preston, Thomas Fogal ...
  - People at SCI, School of Computing

Thank you for attending my talk

Question and Answer

1: **Input** : N volume inputs
2: **Output**: Template atlas volume
3: **for** $k = 1$ to max_iters **do**
4: Fix images I_i^k, compute the template $\hat{I^k} = \frac{1}{N} \frac{\sum_{i=1}^{N} I_i^k w_i}{\sum_{i=1}^{N} w_i}$
5: **for** $i = 1$ to N **do** {loop over the images}
6: Fix the template $\hat{I^k}$, solve pairwise-matching problem between I_i^k and $\hat{I^k}$
7: Update deformed image I_i^k with current velocity
8: **end for**
9: **end for**

Algorithm 1: Atlas construction framework

1: **Input** : N input images
2: **Output**: N processed output images
3: **for** $k = 1$ to N **do**
4: Upload the k-th image from the storage device to the processing device
5: Process the input in-core on the processing device
6: Download the output image back to the storage device
7: **end for**

Algorithm 2: Synchronous out-of-core MIMO operators

1: **Input** : N input volumes, device input buffers d_i [3] and device input buffers d_o [3]
2: **Output**: N processed output volumes
3: **for** $k = 1$ to $N + 2$ **do**
4: **if** $k <= N$ **then**
5: Load the data $iImg$ [k] from storage device to device buffer d_i [$k\%3$] on the *H2D* stream
6: **end if**
7: **if** $k > 1$ and $k - 1 <= N$ **then**
8: Apply the operator on device buffer d_o [$k - 1)\%3$] $oper(d_i$ [$k - 1)\%3$] on *D2D* stream
9: **end if**
10: **if** $k > 2$ and $k - 2 <= N$ **then**
11: Write output d_o [$k - 2)\%3$] to the storage device $oImg$ [$k - 2$) on the *D2H* stream
12: **end if**
13: **Synchronize streams**
14: **end for**

Algorithm 5: Explicit pipelining MIMO operator