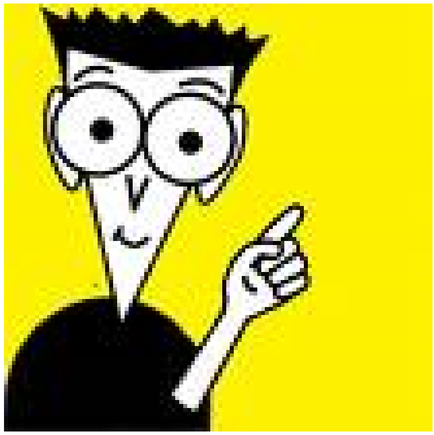
GPUs for Smarties

From Gaming to General Purpose Computing



Graphics Card for Computing?

NVIDIA GTX280 Graphics Card

- Is it only for hardcore gamers?
- Why we want to use it for general purpose computing?
- How easy to program on it?

Copyright © 2009 by Yong Cao

Graphics Card for Computing?

NVIDIA GTX280 Graphics Card

- Is it only for hardcore gamers?

No. More than 200 GPU-based research projects, and more than 100 university courses recorded by NIVIDA only.

Link for projects
Link for courses

- How easy to program on it?

Copyright © 2009 by Yong Cao

Graphics Card for Computing?

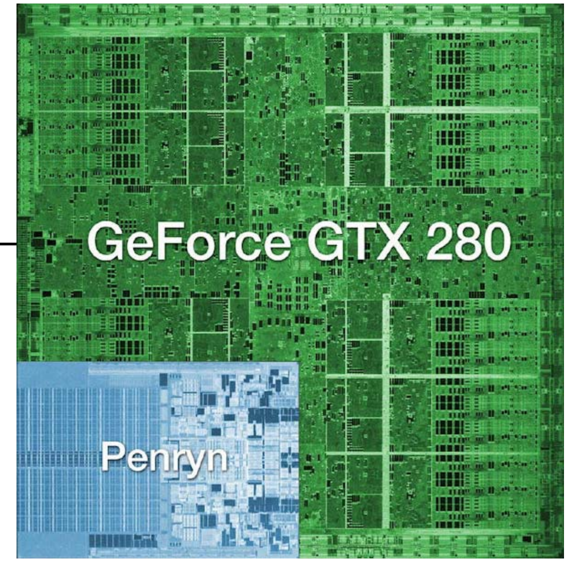
NVIDIA GTX280 Graphics Card

- Is it only for hardcore gamers?
- **Why we want to use it for general purpose computing?**

  It's powerful!
  And, it's commoditized and everywhere!
- How easy to program on it?

Copyright © 2009 by Yong Cao

Graphics Card for Computing?



- **Powerful, Cheap and Everywhere:**
  - **1.4 Billion Transistors. (GTX280)**
  - **Nearly 1 Tera FLOPs. (GTX280)**
  - **$350 for each card.**
  - **Over one million GPGPUs sold. (NVIDIA**
  - **Personal Supercomputer!**

3 NVIDIA GTX295 Graphics Cards
6 GT200 GPUs
1440 Processing Codes
5.367 Tera FLOPs
About 1,500 Watts
Cost: $4,000

IBM ASCI White at 2000
512 Nodes
8192 Processors
7.266 Tera FLOPs
106 tons
3 Million Watts
Cost: $110 Millions

Copyright © 2009 by Yong Cao

Graphics Card for Computing?

NVIDIA GTX280 Graphics Card

- Is it only for hardcore gamers?
- Why we want to use it for general purpose computing?
- **How easy to program on it?**

  Improved a lot recently, with NVIDIA's CUDA (Compute Unified Device Architecture) programming framework.

Copyright © 2009 by Yong Cao

CUDA "Compute Unified Device Architecture"

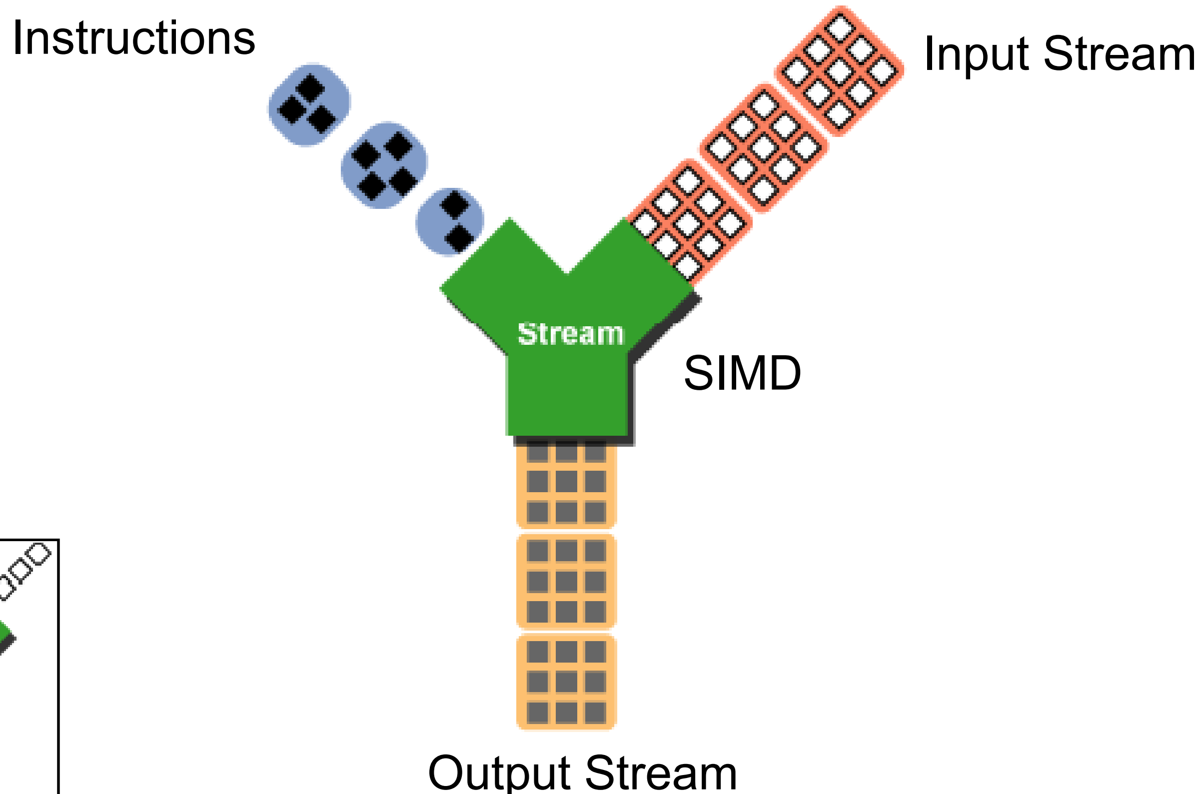
- General purpose parallel programming model
- Much easier to use
  - C language, no Graphics APIs
  - Shallow learning curve: tutorials, sample projects, forum
- Key features
  - Simple management of threads
  - Simple execution model
  - Simple synchronization
  - Simple communication

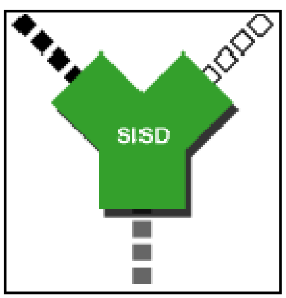
Nice! Let's do it!

- **Wait! Limitations:**
  - **Tricky for non-data-parallel applications. Worst performance on GPU than CPU!**
    - **Data and control dependencies.**
    - **Research question: How to parallelize these algorithms?**
  - **Performance heavily impacted by hardware limitations.**
    - **Register file size, cache size, threads scheduling parameters.**
    - **Research question: How to redesign these algorithms to meet these hardware limitations?**

Computing on GPUs

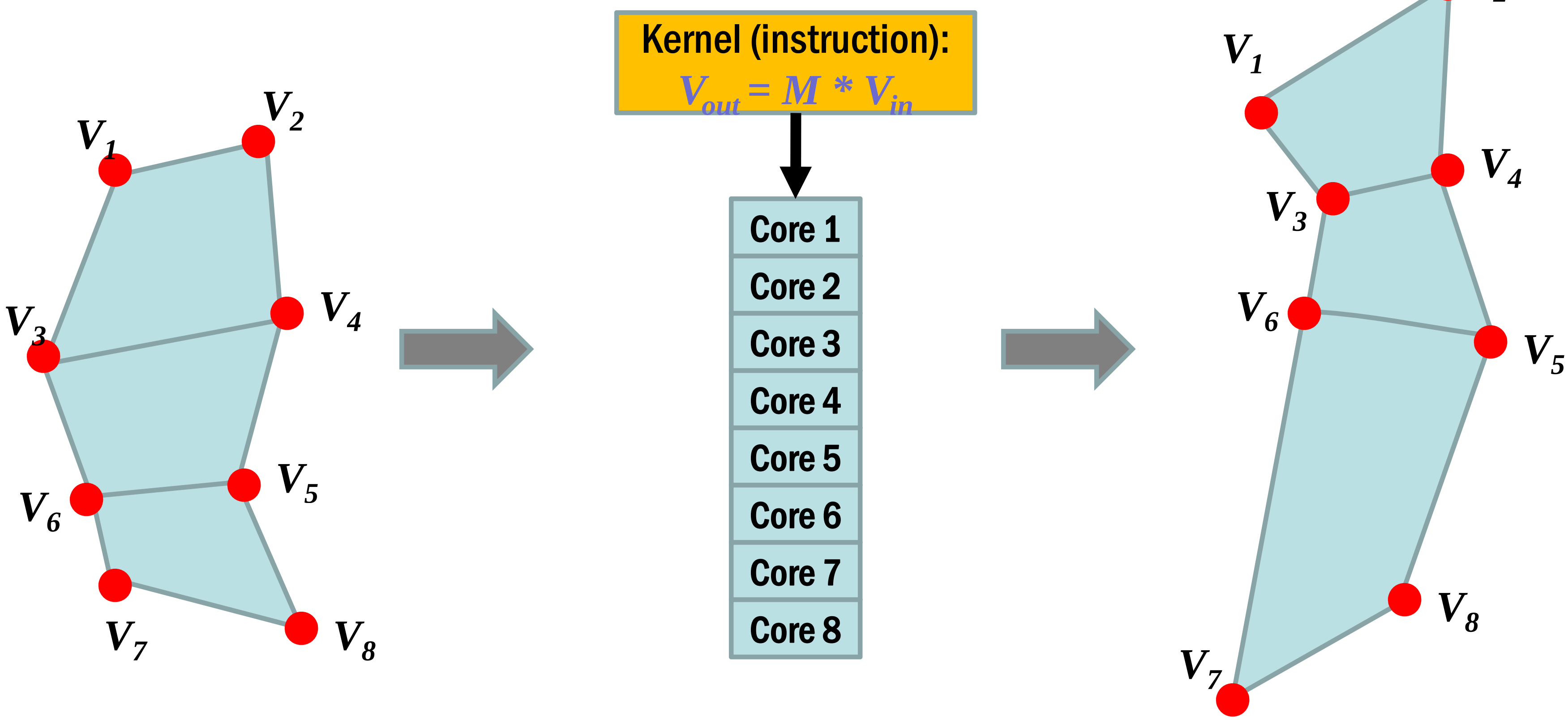
- **Steam processing and Vectorization (SIMD)**





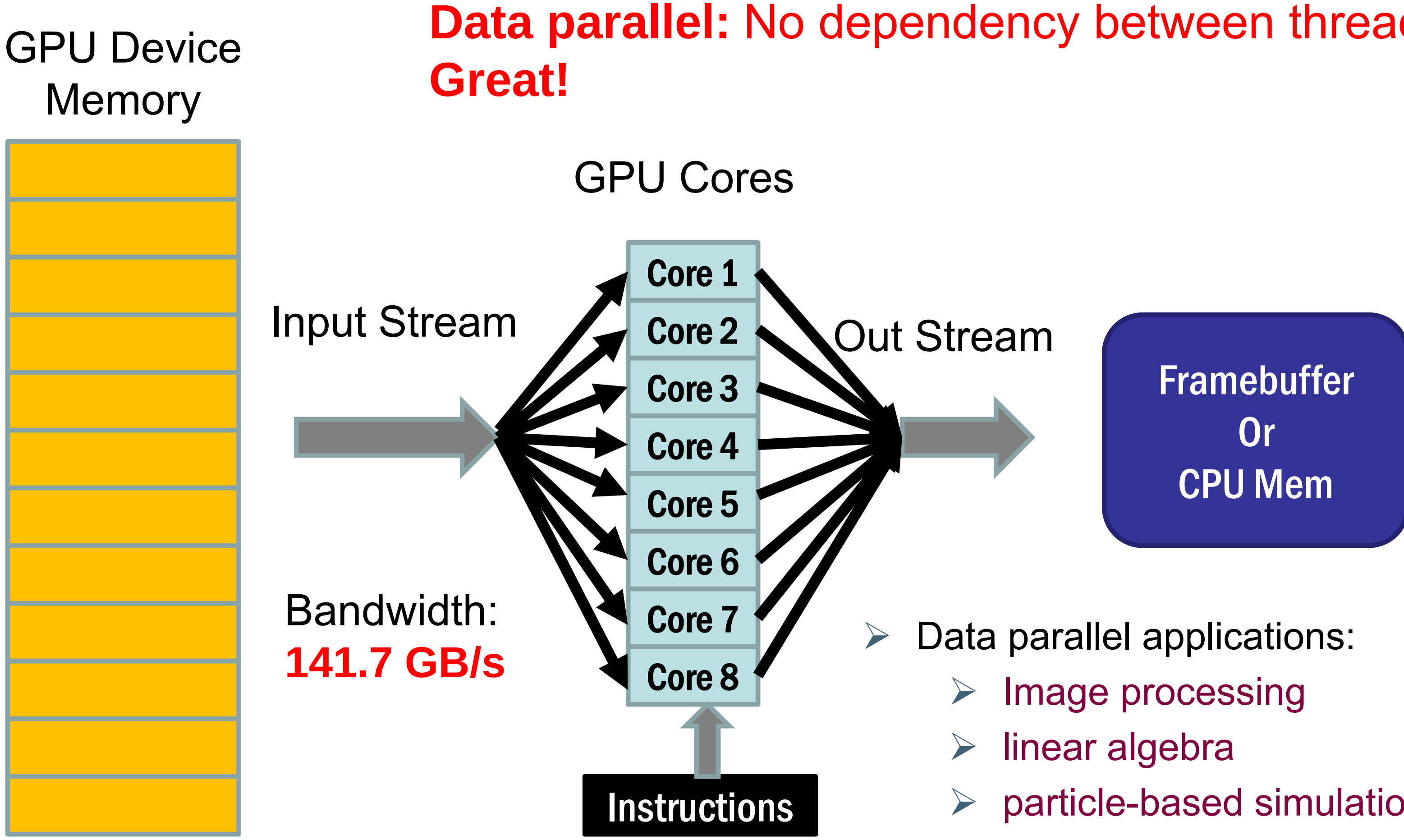
Computing on GPUs

- A graphics example and a demo



GPU Architecture (Simplified)

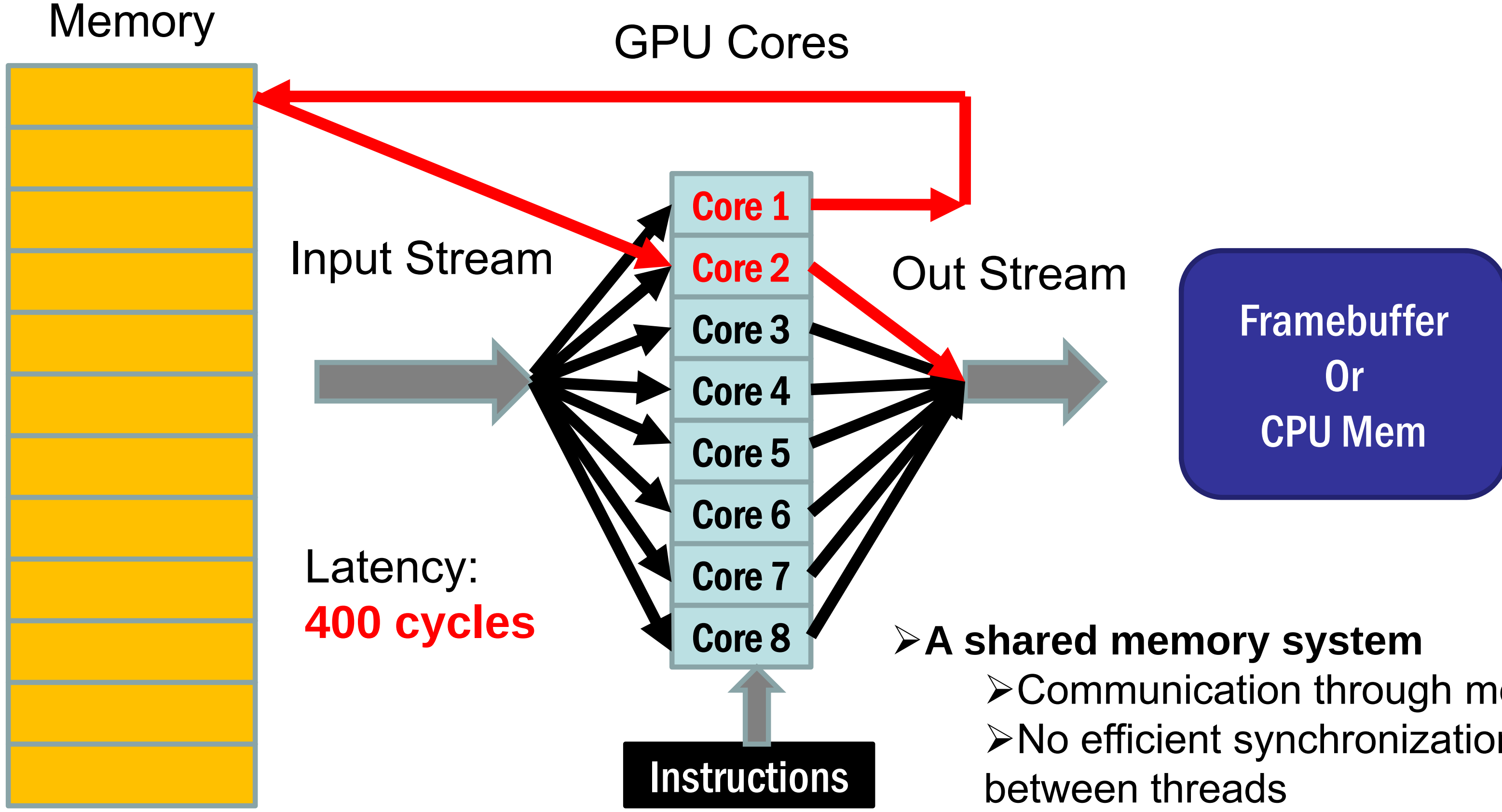
Data parallel: No dependency between threads
Great!



- Data parallel applications:
  - Image processing
  - linear algebra
  - particle-based simulation
  - graphics …

GPU Architecture (Simplified)

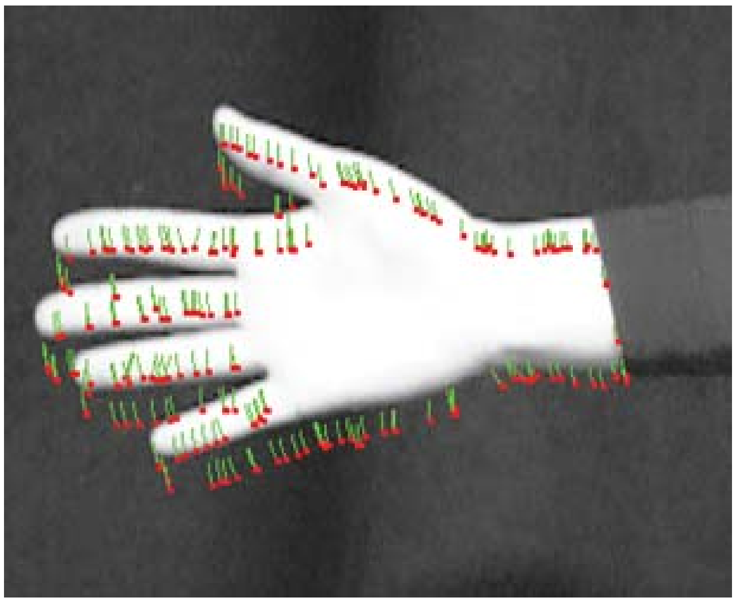
Non-Data parallel: Dependencies between threads



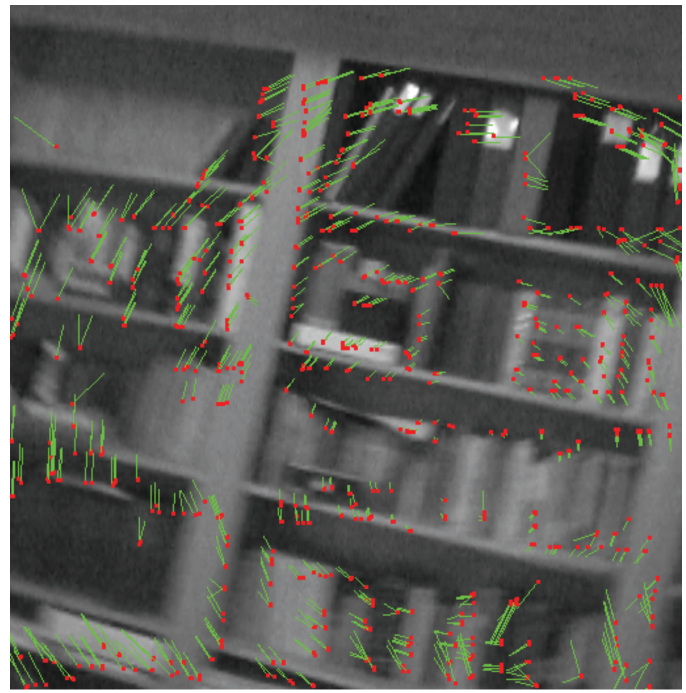
- **A shared memory system**
  - Communication through memory
  - No efficient synchronization between threads

First application: Video Processing

- **Goal: Real-time motion tracking in video stream using vectors.**



Hand movement



Camera rotating on its optical axis

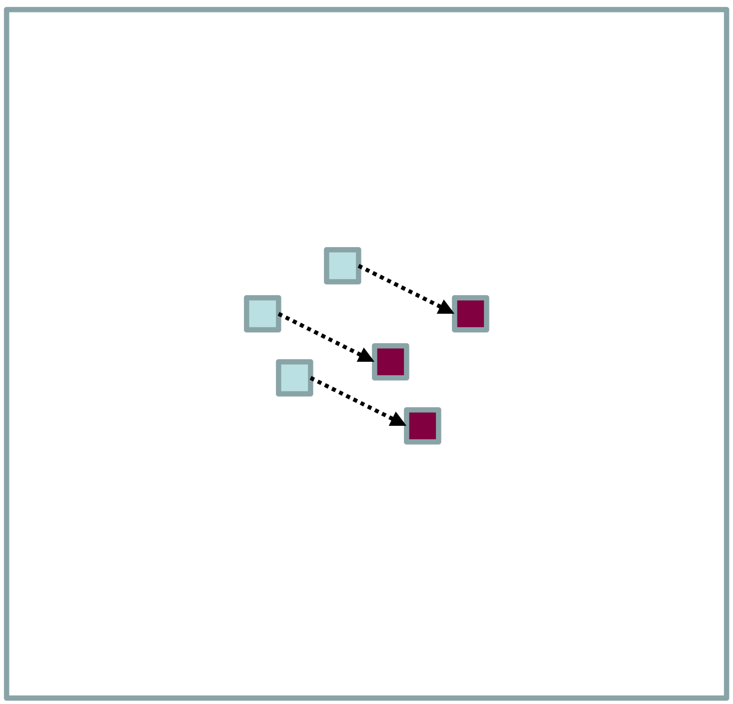
Participants: Francis Quek, Jing Huang, Sean Ponce, Seung-In Park, Yong Cao

Vector Coherence Mapping[1]

- Compute the movement vectors between frames by applying
  - Spatial coherence constraints
  - Temporal coherence constraints

[1]. Francis Quek and Robert K. Bryll, “Vector coherence mapping: A parallelizable approach to image flow computation”

Vector Coherence Mapping[1]

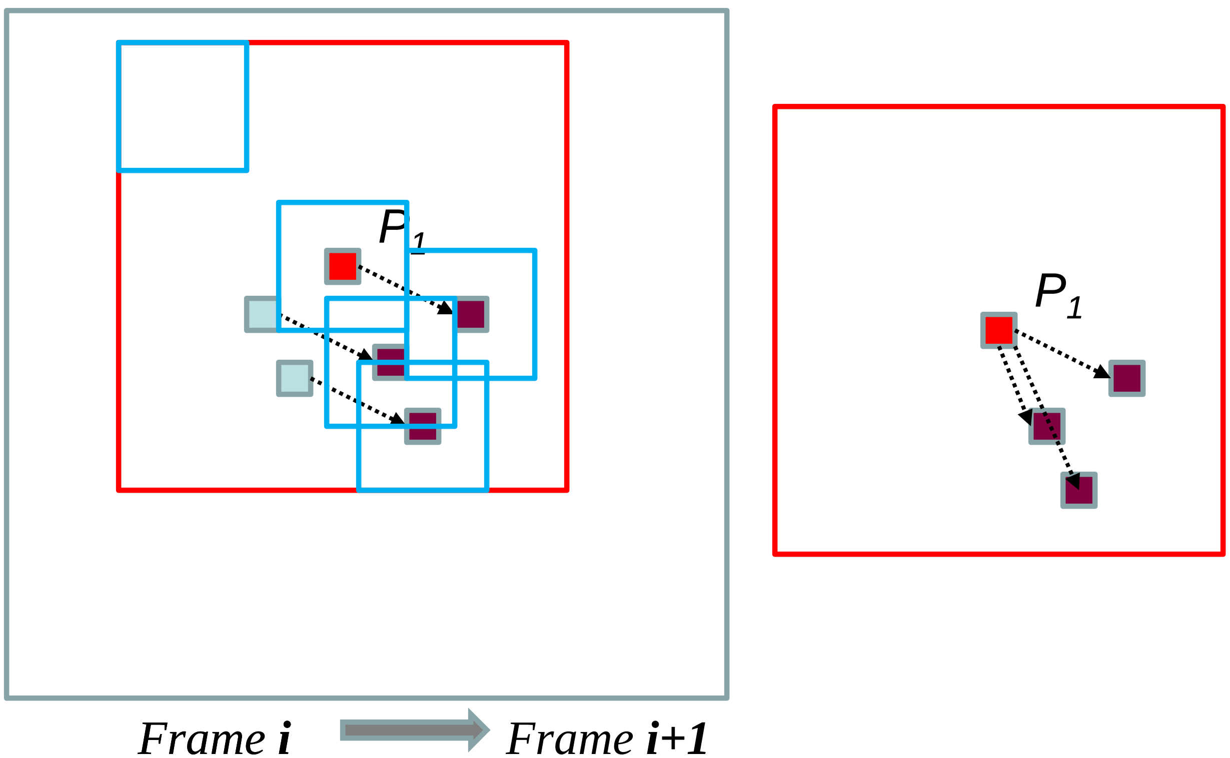


Frame ***i*** → *Frame* ***i+1***

[1]. Francis Quek and Robert K. Bryll, “Vector coherence mapping: A parallelizable approach to image flow computation”

Vector Coherence Mapping[1]

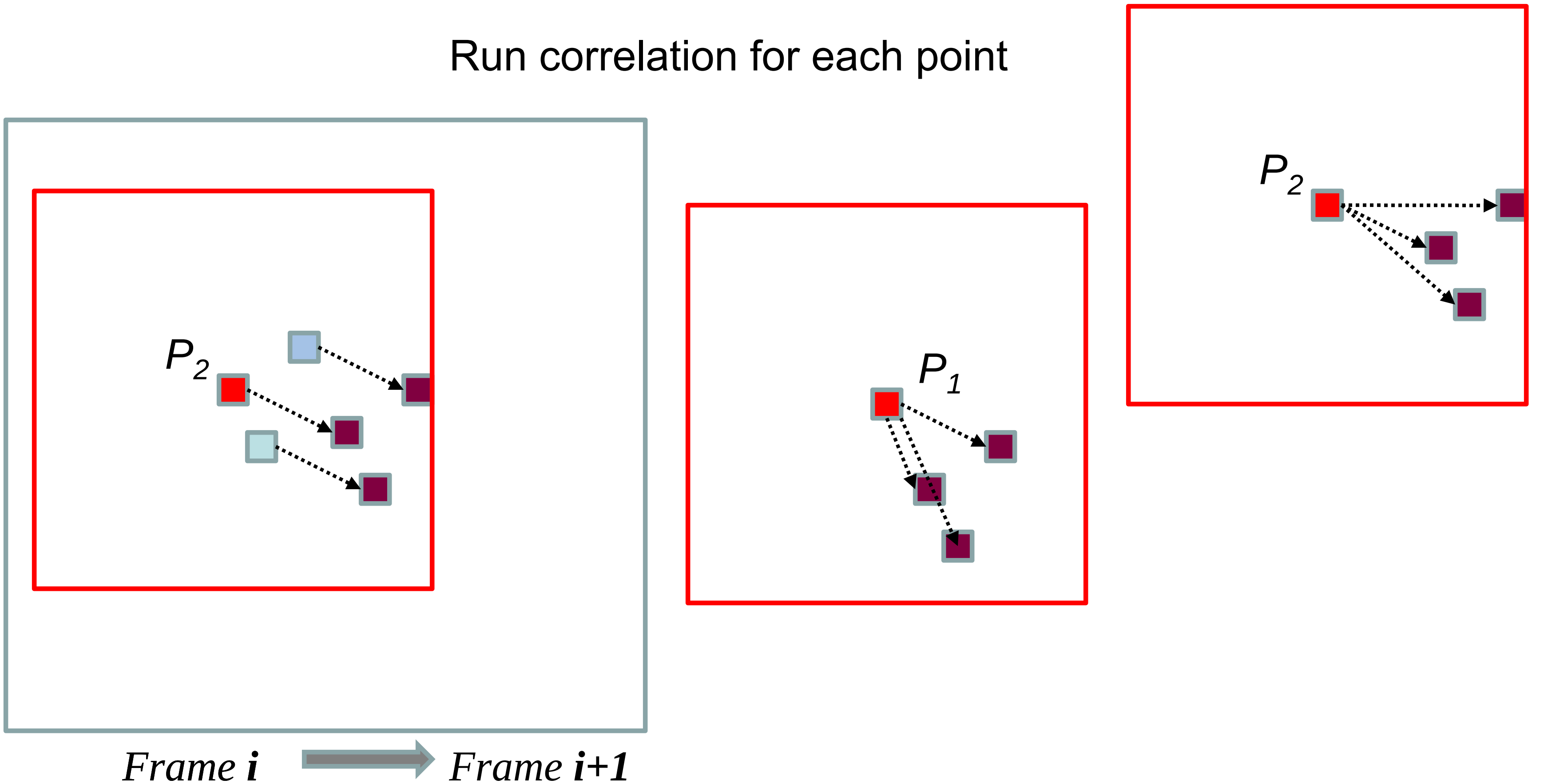
Run correlation for each point



[1]. Francis Quek and Robert K. Bryll, "Vector coherence mapping: A parallelizable approach to image flow computation"

Vector Coherence Mapping[1]

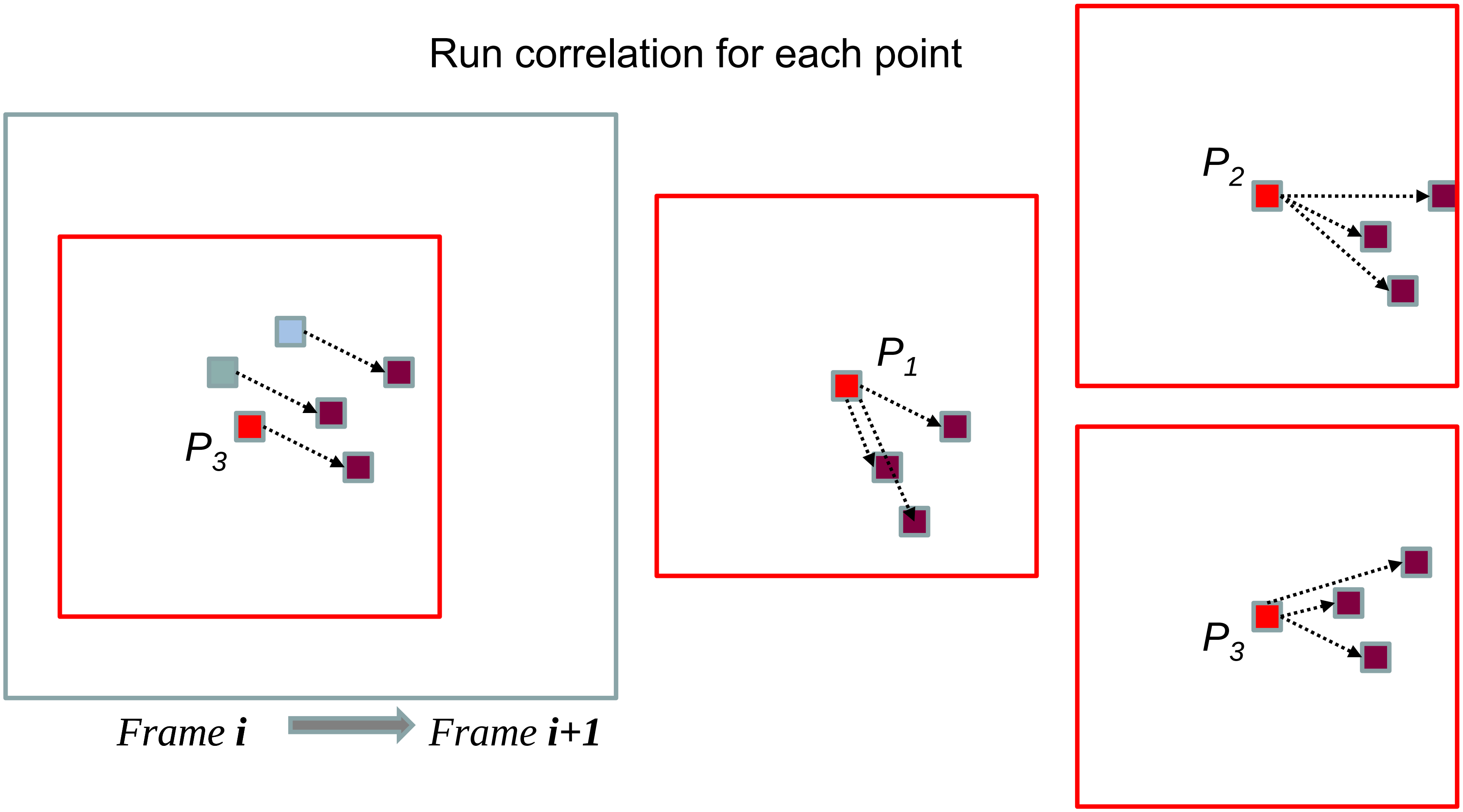
Run correlation for each point



[1]. Francis Quek and Robert K. Bryll, "Vector coherence mapping: A parallelizable approach to image flow computation"

Vector Coherence Mapping[1]

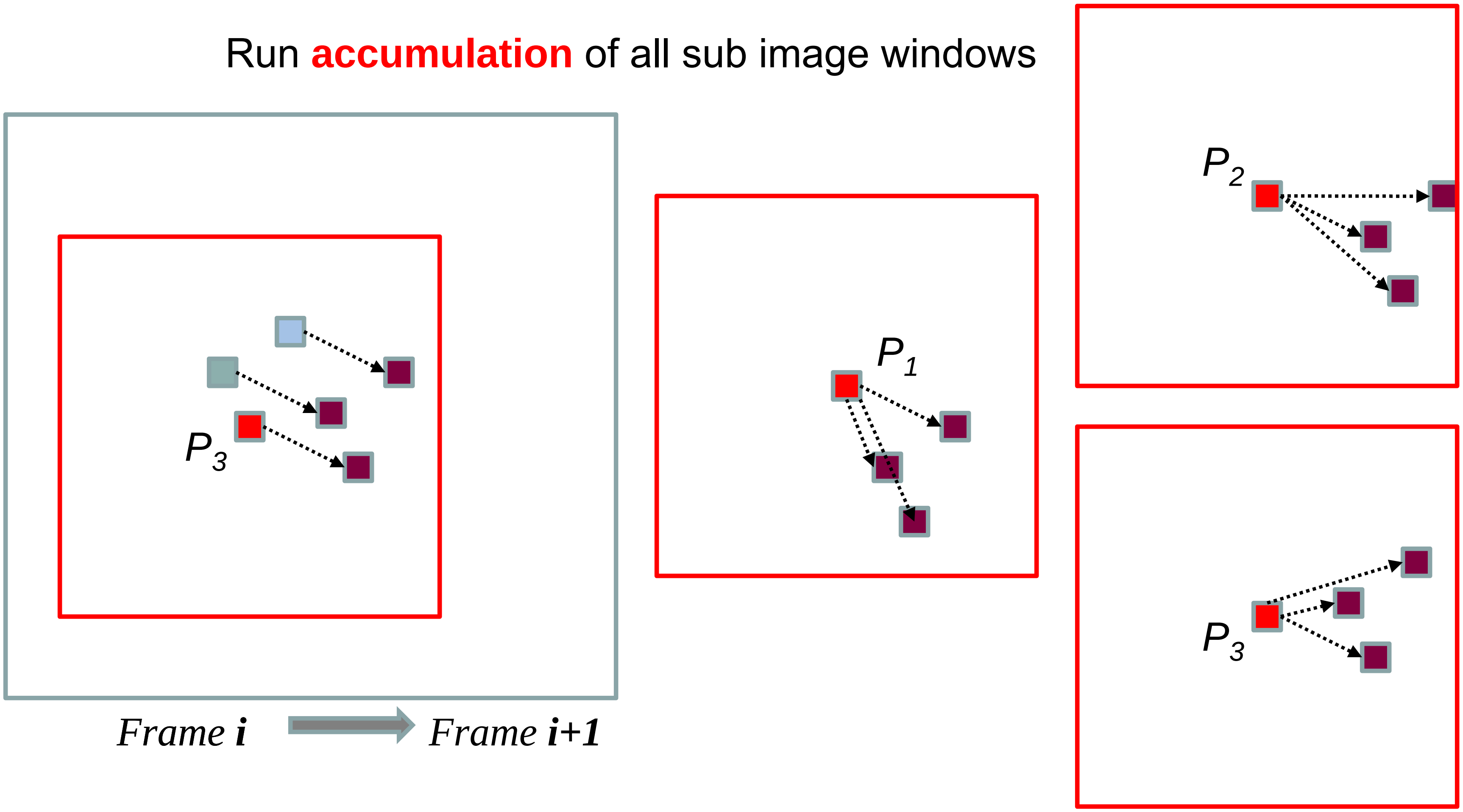
Run correlation for each point



[1]. Francis Quek and Robert K. Bryll, “Vector coherence mapping: A parallelizable approach to image flow computation”

Vector Coherence Mapping[1]

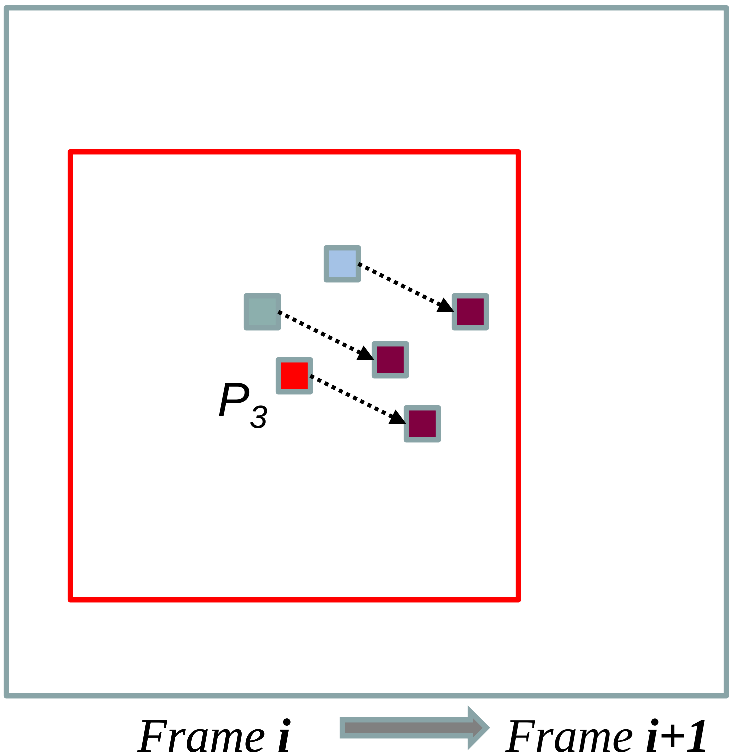
Run **accumulation** of all sub image windows

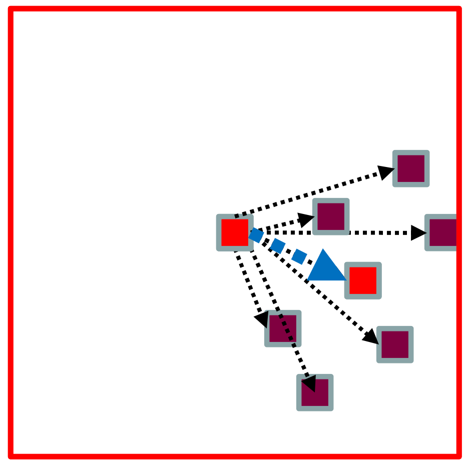


[1]. Francis Quek and Robert K. Bryll, "Vector coherence mapping: A parallelizable approach to image flow computation"

Vector Coherence Mapping[1]

Run **accumulation** of all sub image windows





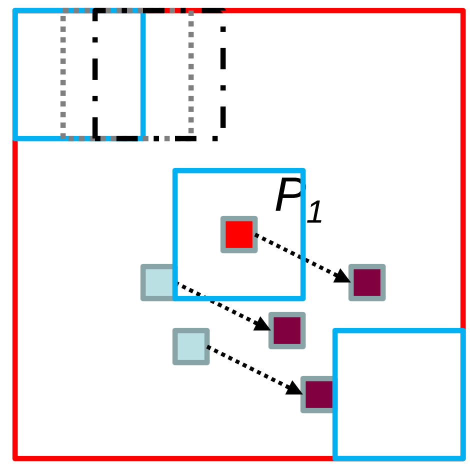
Accumulated 3 times

That's our vector!

[1]. Francis Quek and Robert K. Bryll, "Vector coherence mapping: A parallelizable approach to image flow computation"

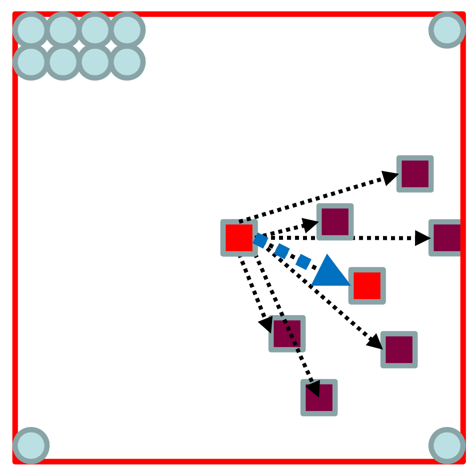
Parallelizing VCM on GPU Demo

- **Correlation: (data parallel)**
  - **Each thread handle on correlation**



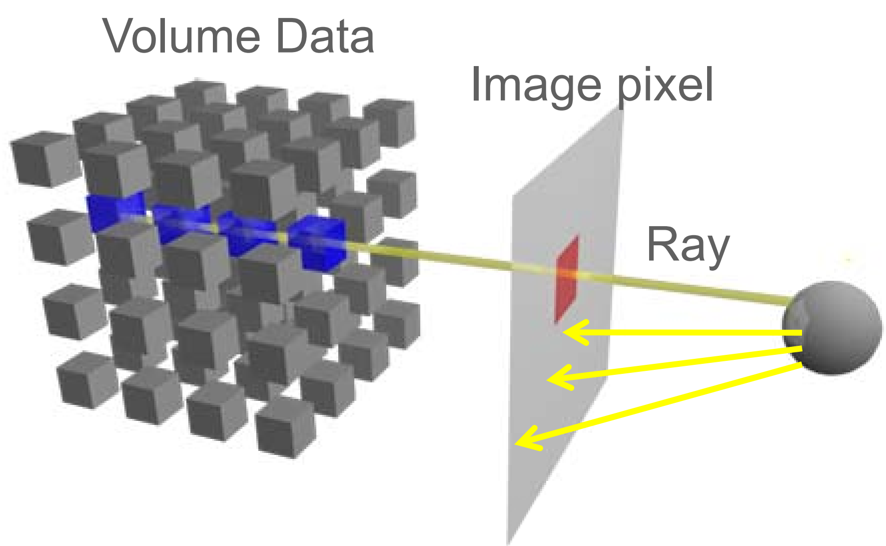
More than 40X speed up

- **Accumulation: (data parallel)**
  - **Each thread handle accumulation of one pixel**



Volume rendering project: (data parallel) Demo

- **Each thread processing one *Ray***



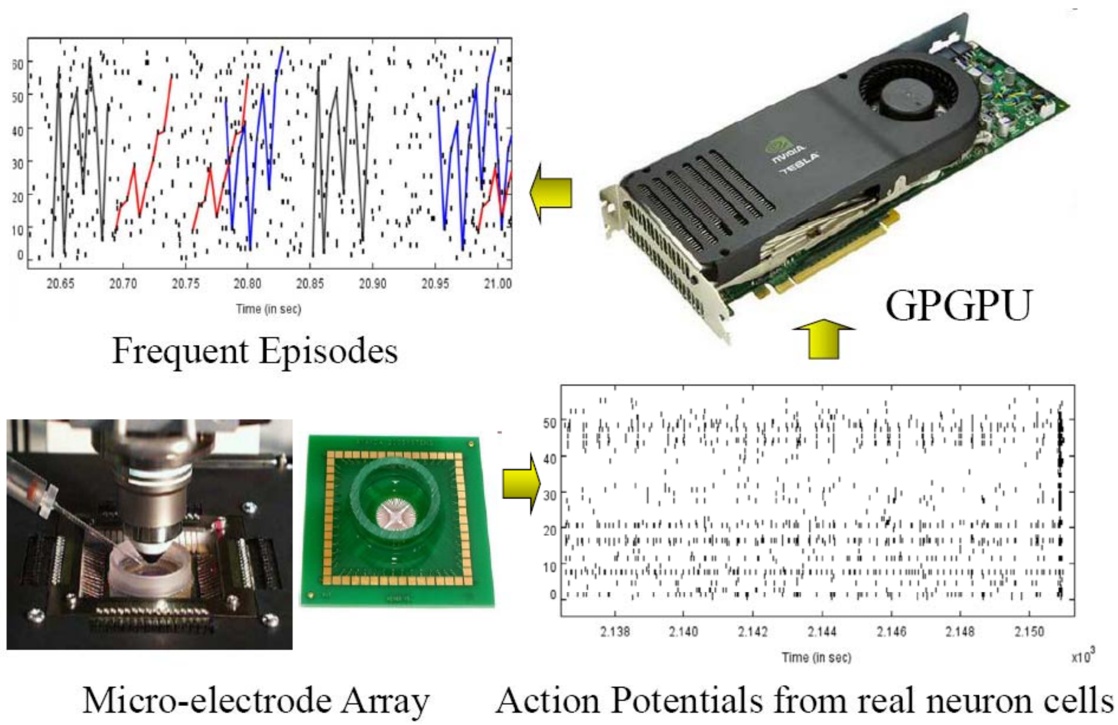
Volume Ray Casting

More than 100X speed up

Participants: Colin Braley, Robert Hagan, Yong Cao

Temporal Data Mining: (non-data-parallel)

- **Application: Computational Neuroscience**



Participants: **Naren Ramakrishnan, Wu-chun Feng, Debprakash Patnaik, Sean Ponce, Jeremy Archuleta, Tom Scogland, Patrick Butler, Yong Cao**

Temporal Data Mining: (non-data-parallel)

Input stream:

MDAFBDNCABFDBCAABMADOAOEYBCD … … FQQEGTOKMAKMBCFOTUMCAHEBPC

Episode: $A \rightarrow B \rightarrow C$

Temporal Data Mining: (non-data-parallel)

Input stream:

MDAFBDNCABFDBCAABMADOAOEYBCD … … FQQEGTOKMAKMBCFOTUMCAHEBPC

Episode: A →B →C

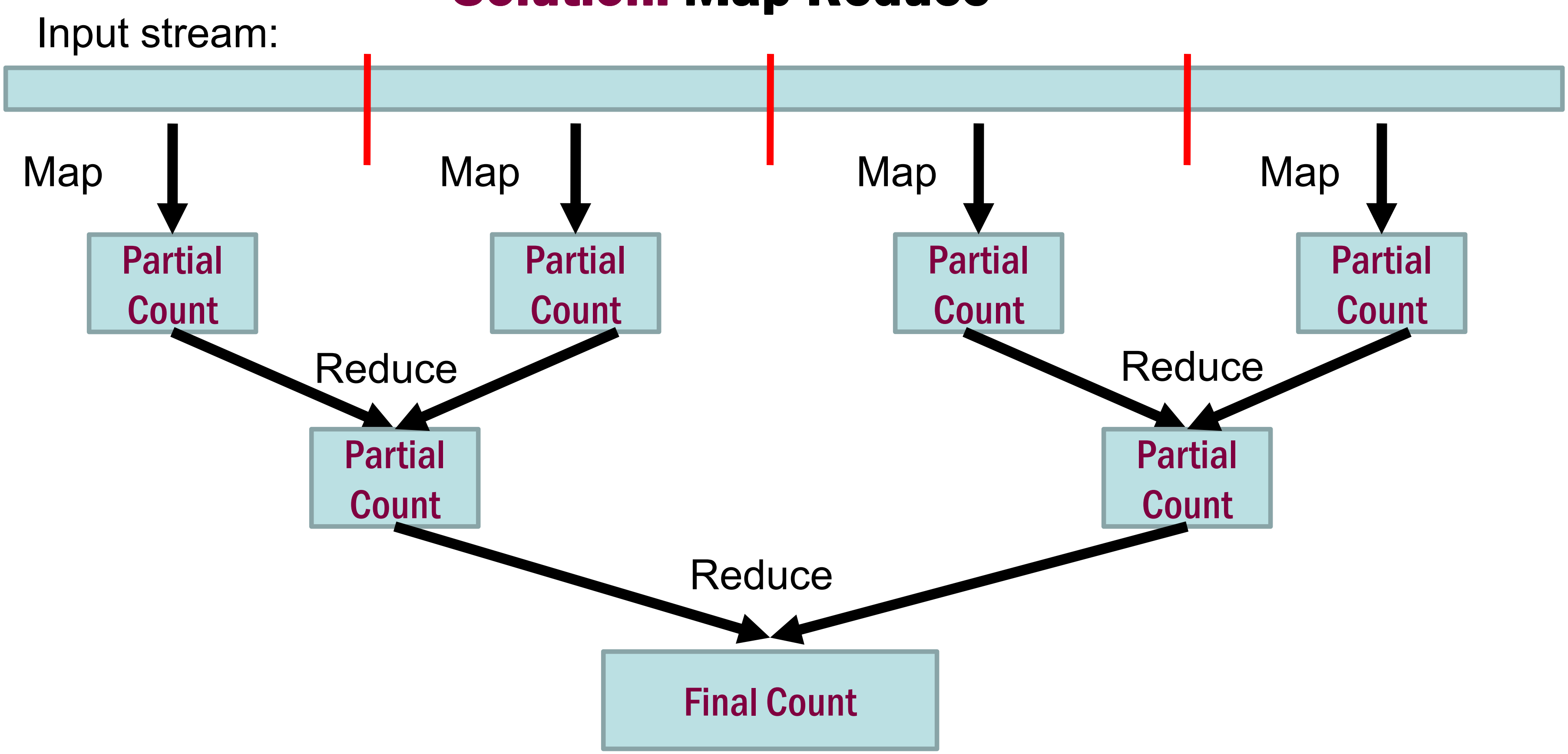
Algorithm: **scan the input stream using a FSM to count.**

FSM based scan has many data and control dependencies. Non-data-parallel.

How to increase the level of parallelism?

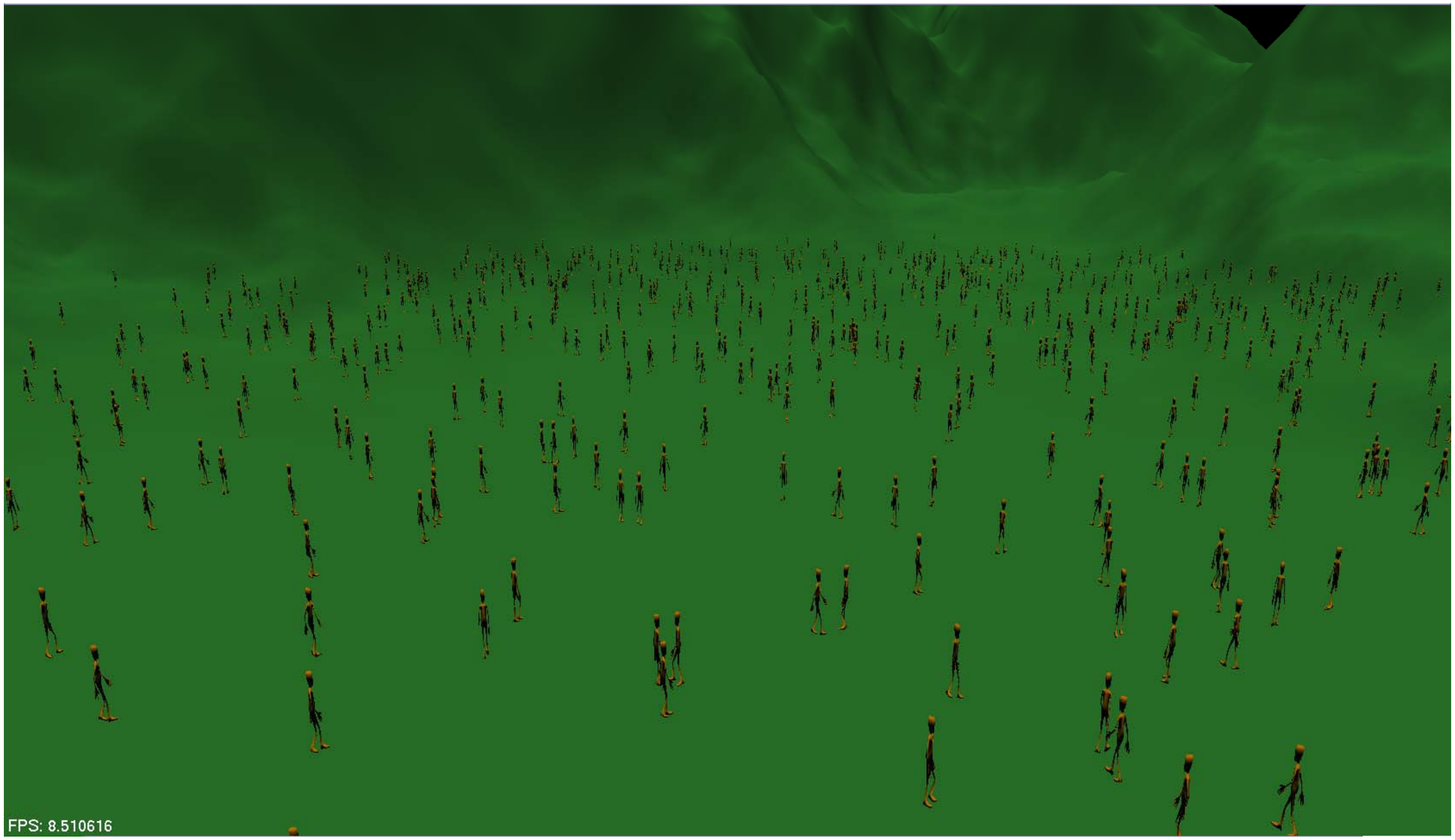
Temporal Data Mining: (non-data-parallel)

Solution: Map-Reduce



Get 15X folds of speedup.
Support real-time mining of frequent episode.

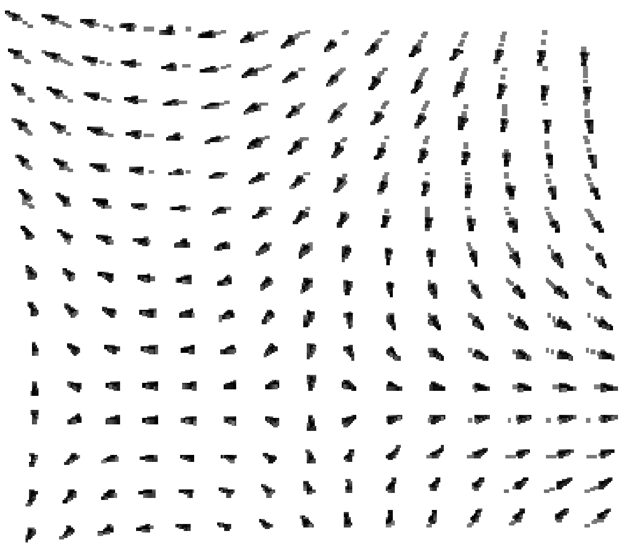
Crowd Simulation: (non-data-parallel)



Participants: Francis Quek, Wu-chun Feng, Yang Cao, Steve Harrison, Denis Gracanin, Sean Ponce, Richard Battle, Lee Calgett, Yong Cao

Crowd Simulation:

- Global path planning for massive agents in real-time
  - Can not solve for each agent (does not scale)
- Solution:
  - Models a crowd as a flow
  - Uses physics-based equations (PDEs)



- Divides the area into a discrete grid
- Creates a vector field for each grid cell
- Each agent simply follows the vectors of its nearest cells

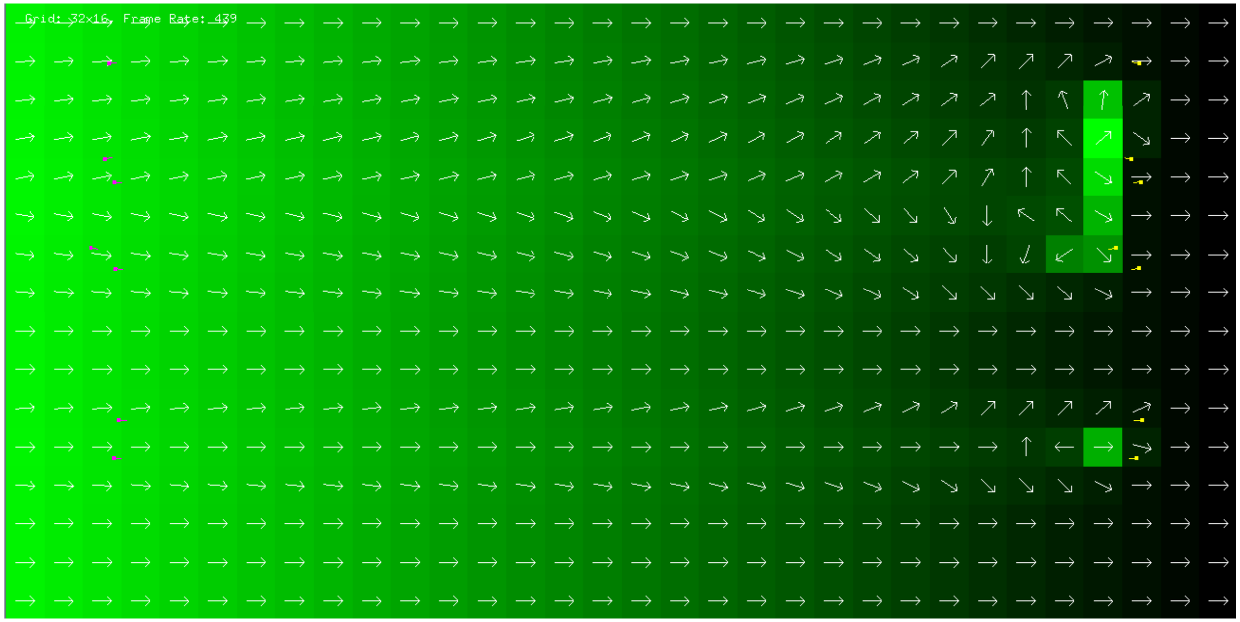
Eikonal Equation

- Physics-based wave propagation equation
- Guarantees minimum cost path with no local minima

$$\left\| \nabla \phi(x) \right\| = C(x)$$

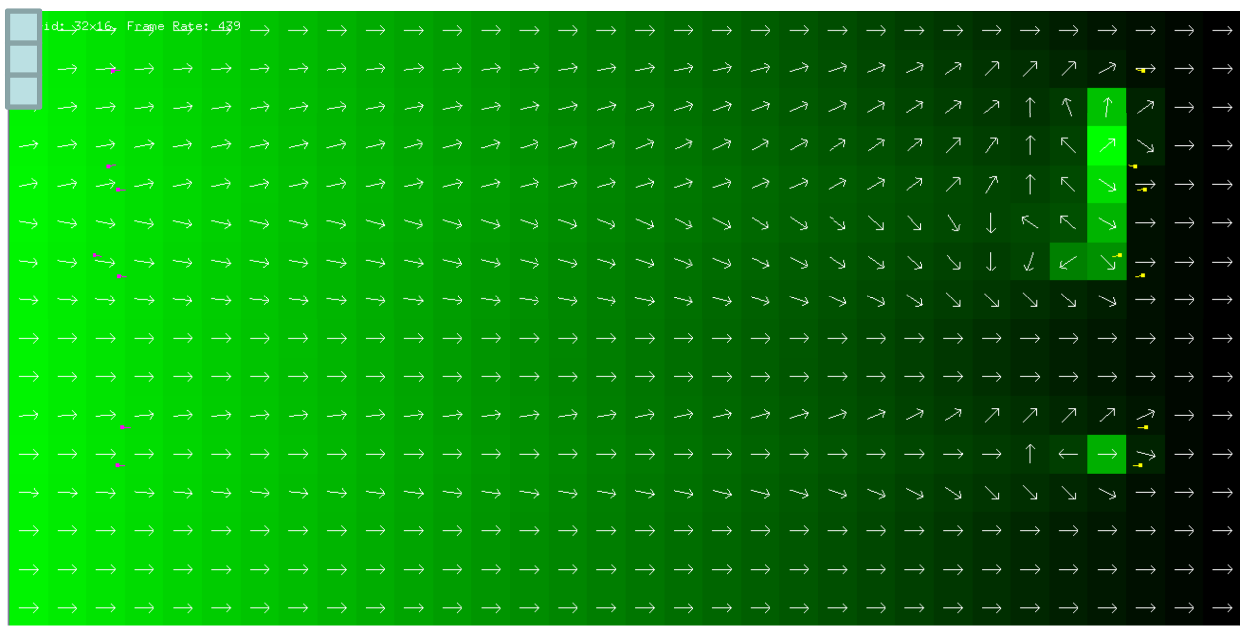
Cost function

Potential field



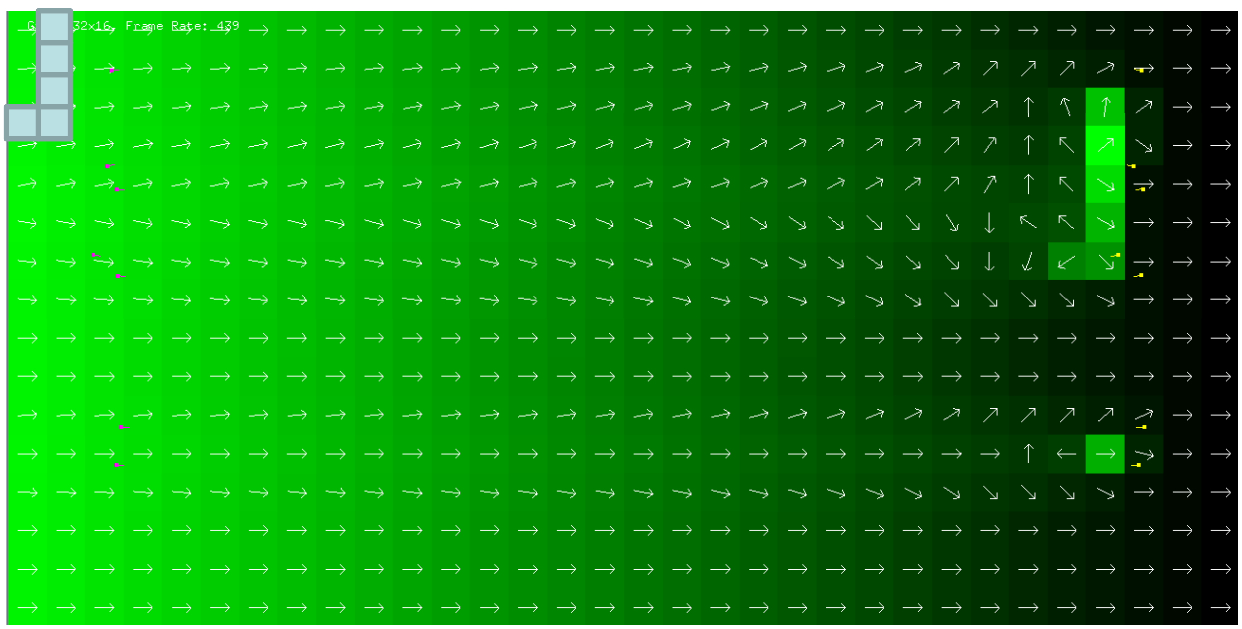
Eikonal Equation

- The numeric method for solving Eikonal equation is **non-data-parallel**
- Guarantees minimum cost path with no local minima



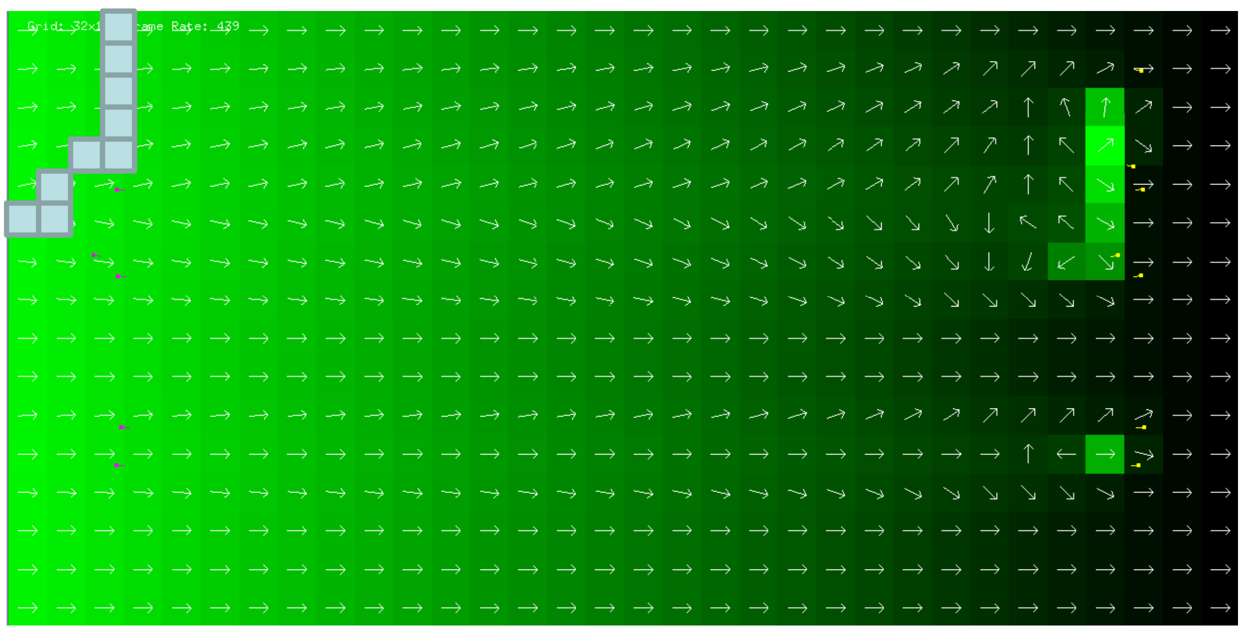
Eikonal Equation

- The numeric method for solving Eikonal equation is non-data-parallel
- Guarantees minimum cost path with no local minima



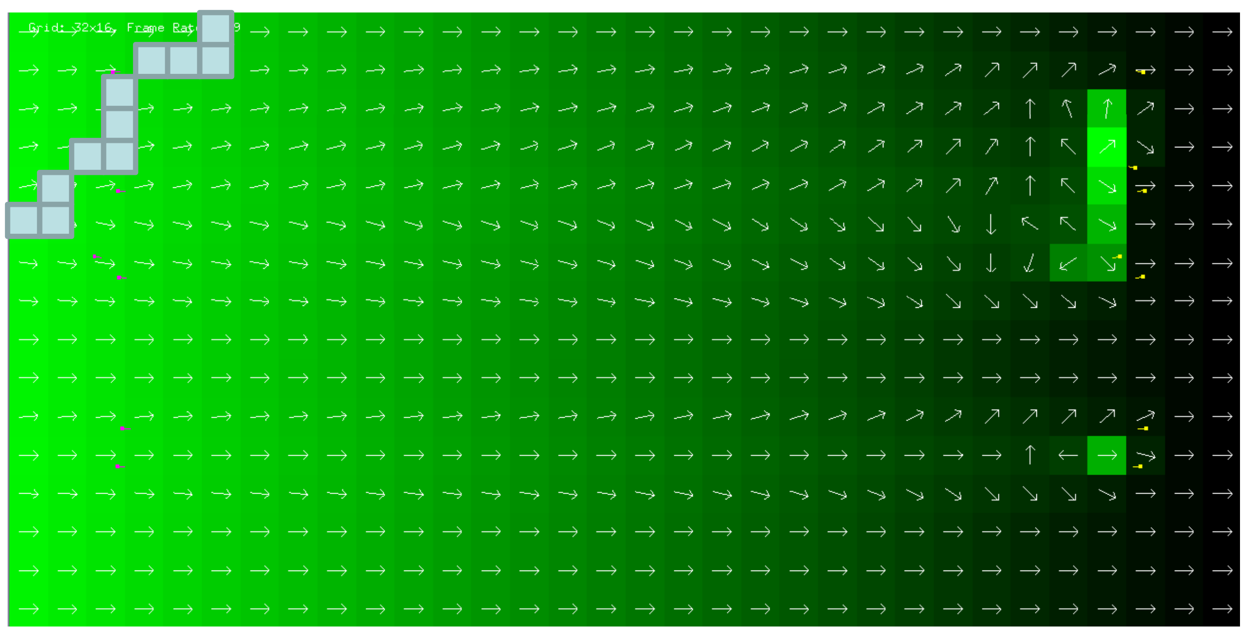
Eikonal Equation

- **The numeric method for solving Eikonal equation is non-data-parallel**
- **Guarantees minimum cost path with no local minima**



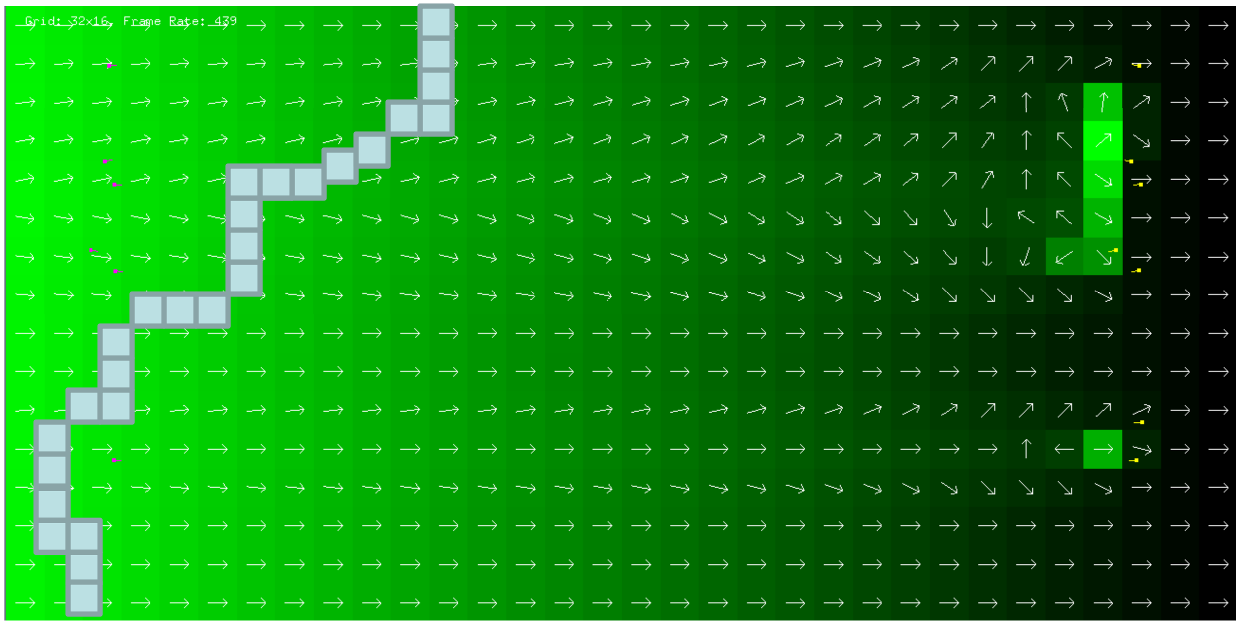
Eikonal Equation

- The numeric method for solving Eikonal equation is non-data-parallel
- Guarantees minimum cost path with no local minima

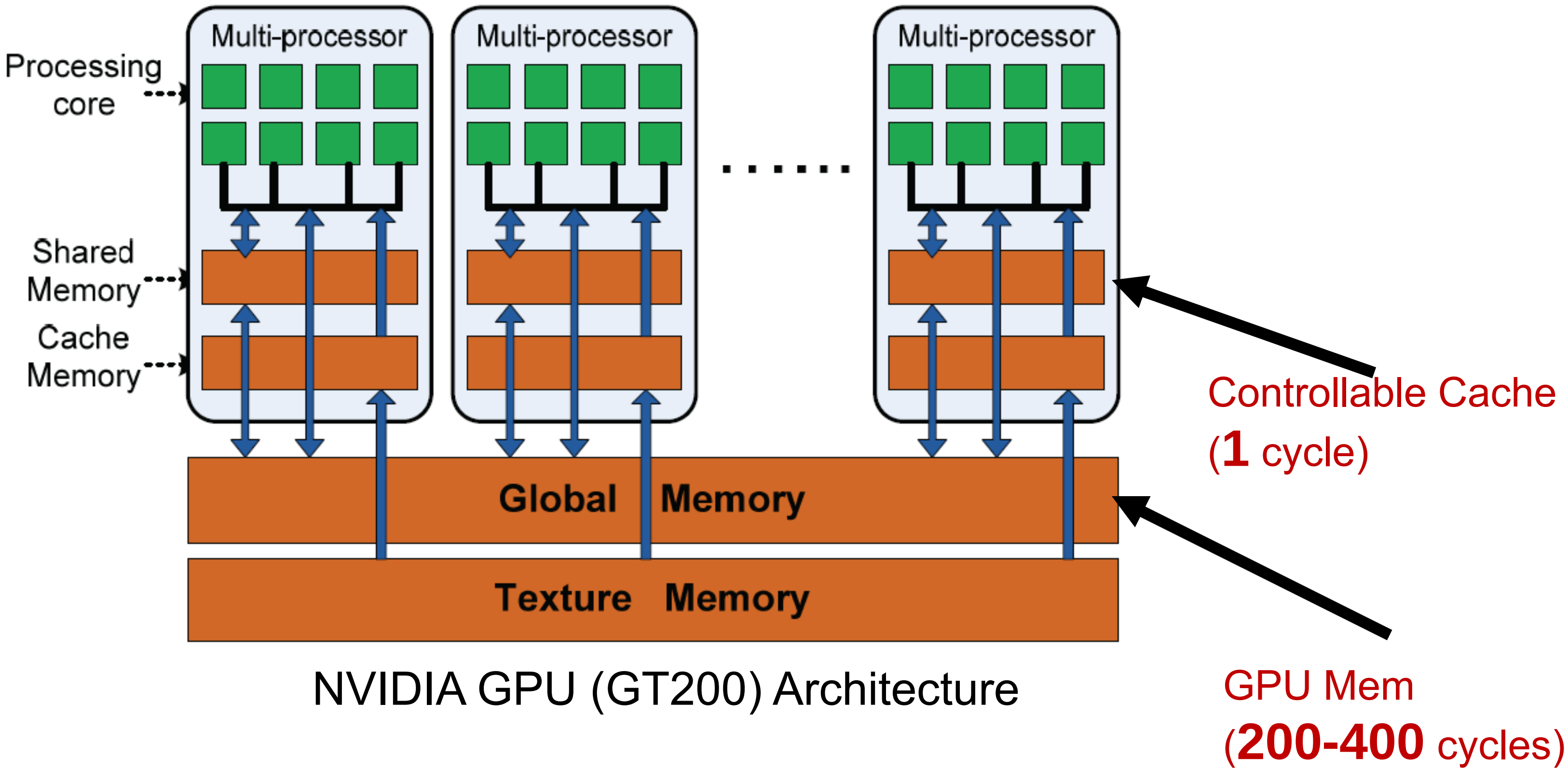


Eikonal Equation

- The numeric method for solving Eikonal equation is non-data-parallel
- Guarantees minimum cost path with no local minima

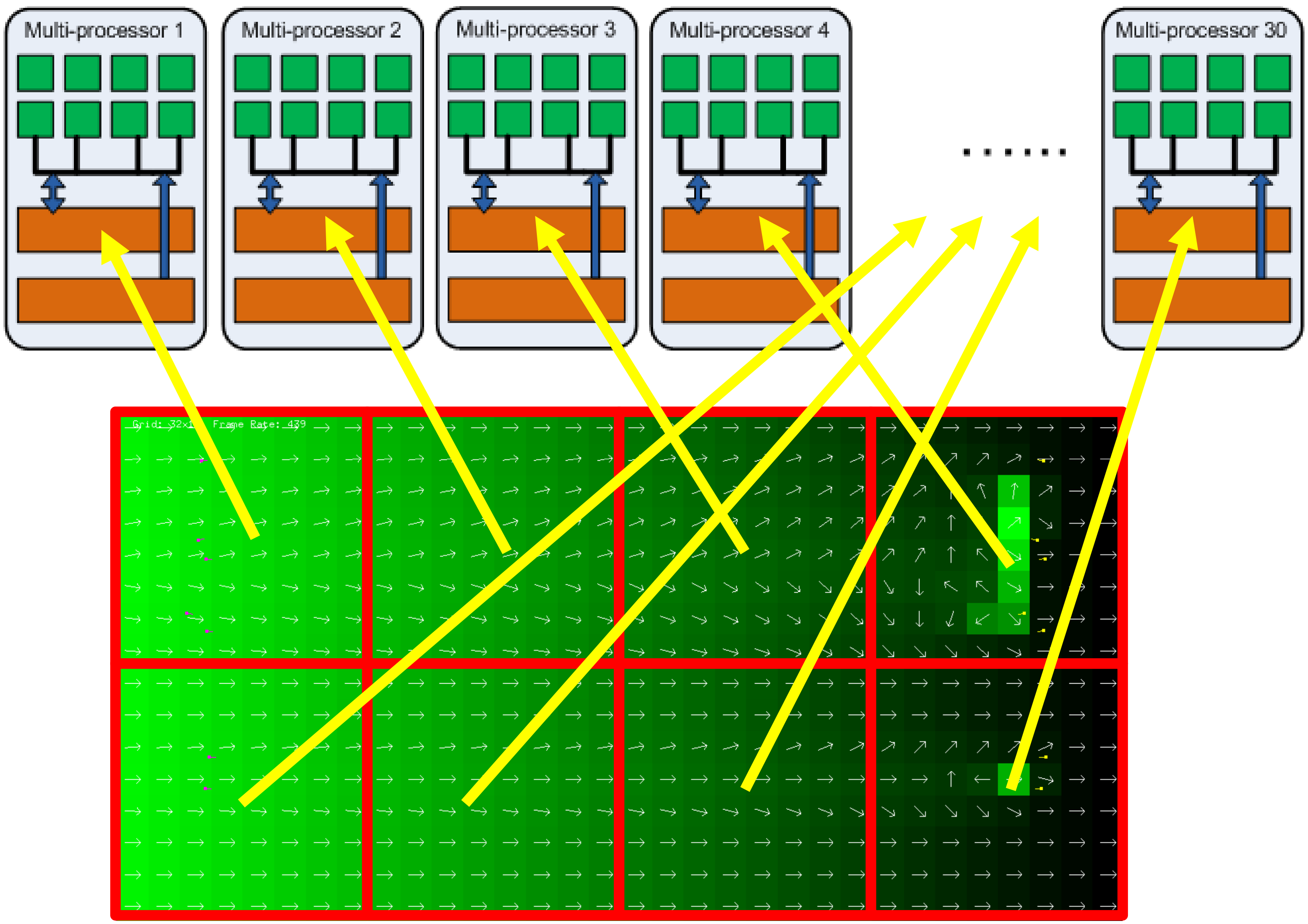


GPU acceleration with controllable cache



NVIDIA GPU (GT200) Architecture

GPU acceleration with controllable cache



More research questions

- High level parallel programming frameworks
  - Compiler, runtime scheduling, …
- Algorithm transformation

  ~~Algorithm → Hardware optimization~~

  - Hardware abstraction → Algorithm
    - Time complexity ≠ Performance
- Parallel computing on hybrid devices
- Applications: what if your program can execute in real-time?