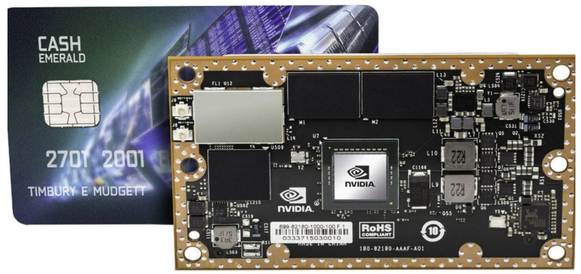
Embedded real-time stereo estimation via Semi-Global Matching on the GPU

Daniel Hernández Juárez, *Alejandro Chacón, Antonio Espinosa, David Vázquez,*

Juan Carlos Moure and Antonio M. López



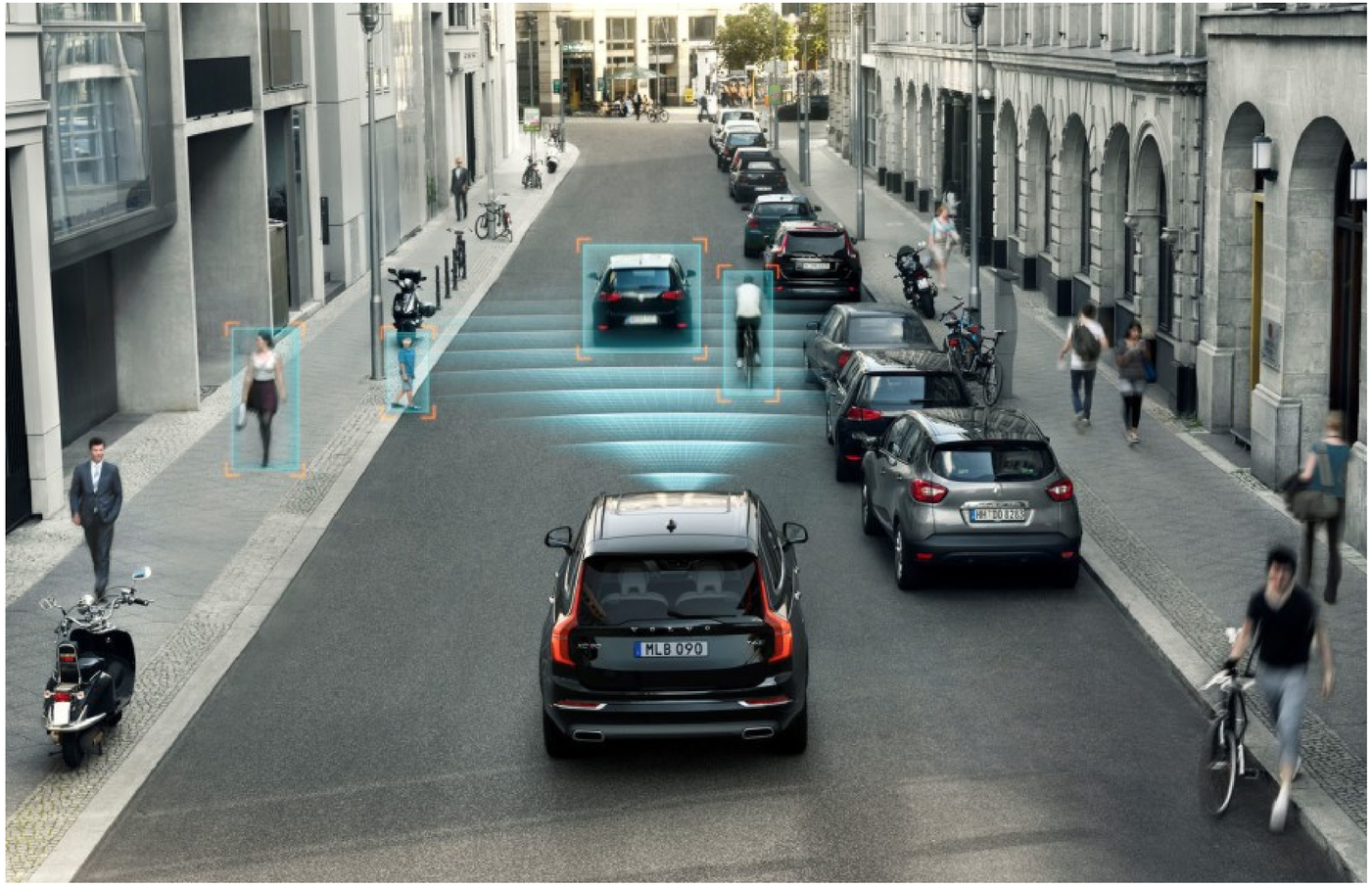
Computer Architecture & Operating Systems Department (CAOS)

Autonomous University of Barcelona

Universitat Autònoma de Barcelona

Centre de Visió per Computador

Motivation

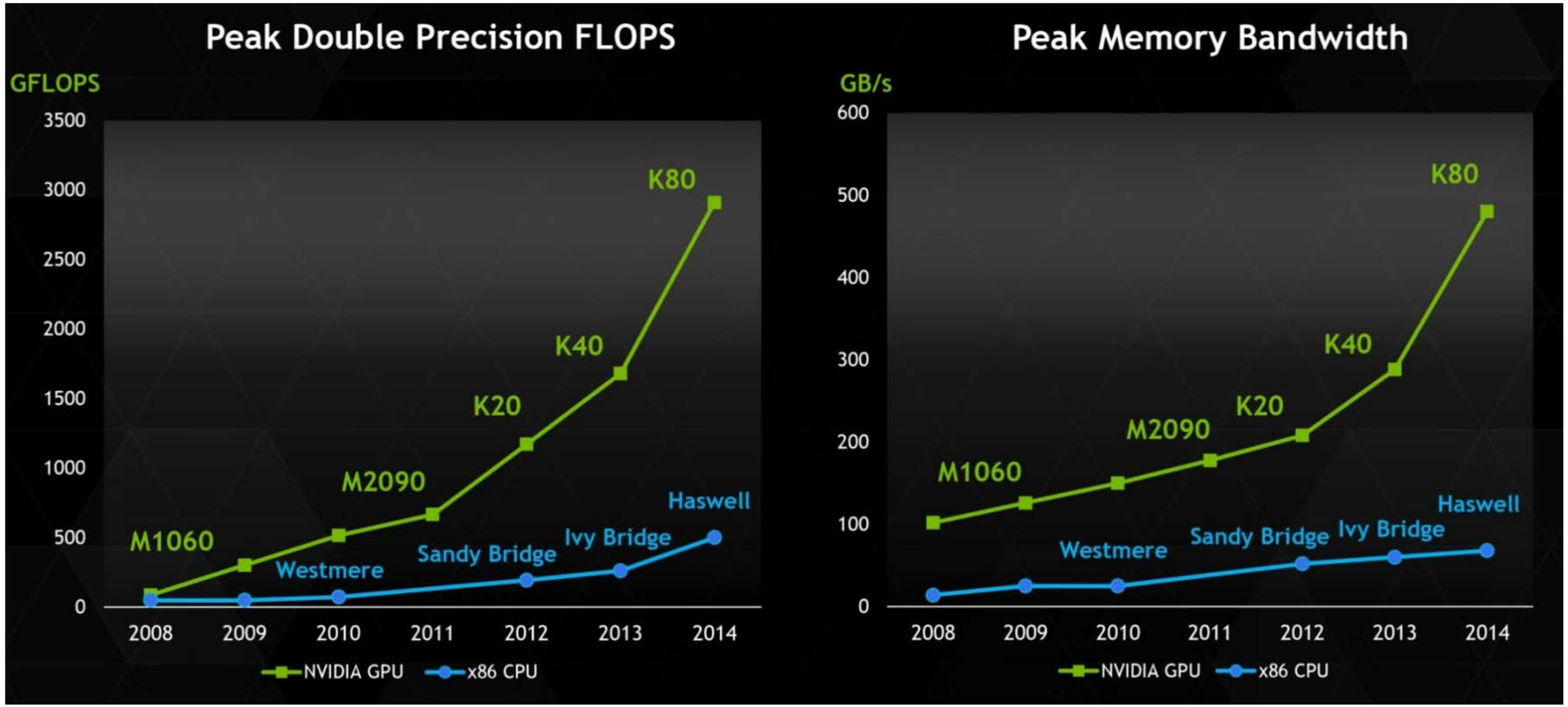


- **Driver Assistance**: Needs distance of each object (to brake before collision)
- **Embedded system:** Real-time (>= 20 Frames Per Second) & low energy consumption

Stereo Vision: cameras

- Infer depth of each pixel using two cameras (stereo pair)
- Reconstruct 3D world

Why GPU?



- More GFLOPS, more Memory Bandwidth & low Energy consumption
- Needs Massive Parallelism: found in many Computer Vision algorithms

Research Goals

- Identify and develop appropriate algorithms to reconstruct 3D world of objects from 2 images taken from cameras

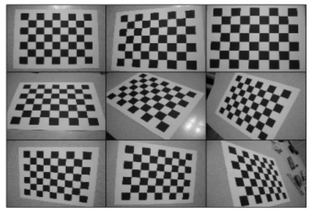
- Design and implement massively parallel schemes and data layouts for execution in accelerators (GPUs) to achieve real-time and low energy consumption

- Propose and apply methodology for developing massively parallel algorithms; identify most common parallel execution patterns and generalize adequate strategies for efficient implementation

Index

Stereo Vision Pipeline

Calibration → **Rectification** → **Stereo Matching** → **Triangulation**



Camera Calibration
Focal length, image center, Rotation & translation, …

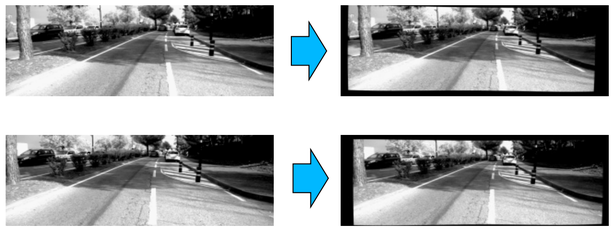
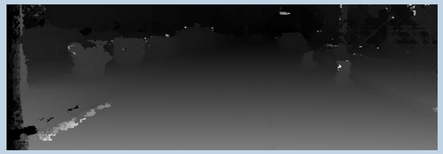
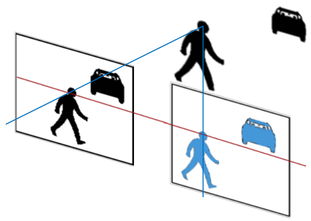


Image Rectification
Make images parallel & remove lens distortion

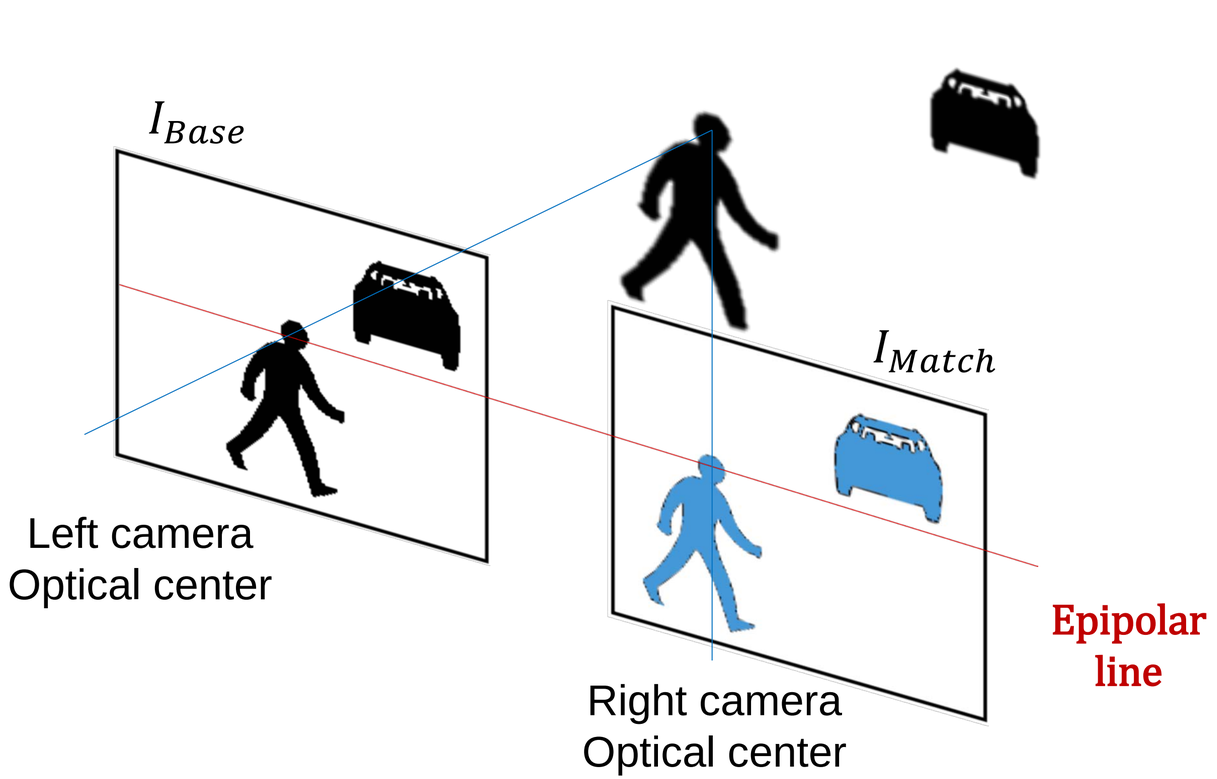


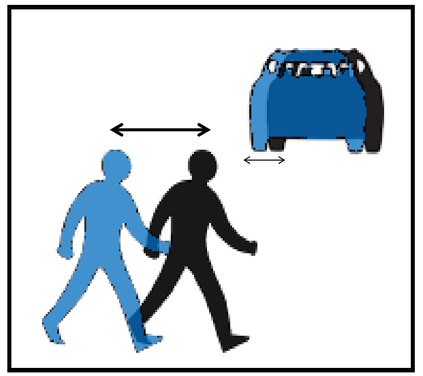
Stereo Matching
Compare pixels in both images to infer ***disparity***

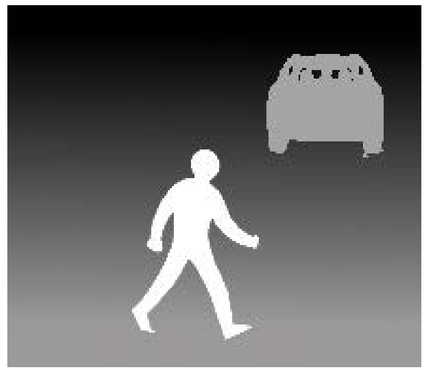


3D reconstruction
Extract 3D points from disparity

Stereo Vision: Overview







- **Disparity:** distance between the x position in the left and right image

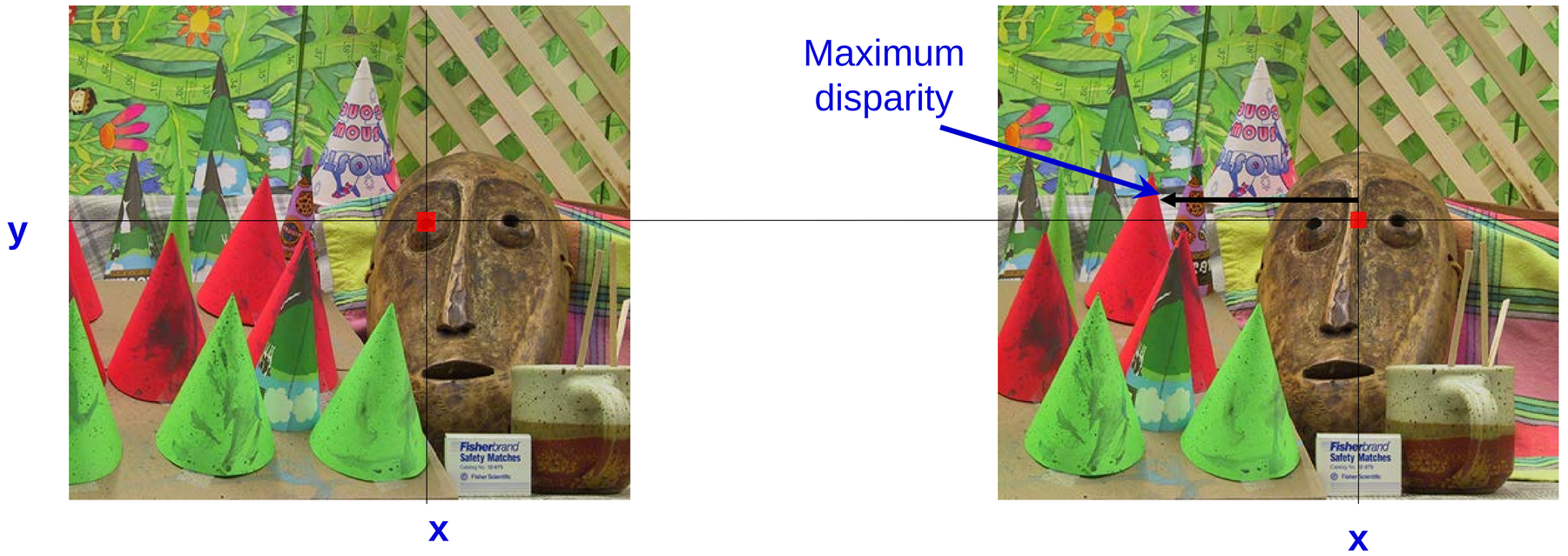
  Near objects = higher disparity Far objects = lower disparity

- **Epipolar** geometry limits the search to 1 dimension

Stereo Matching: Find pixel along epipolar line

Left Image = Base Image

Right Image = Matching Image



Problem:

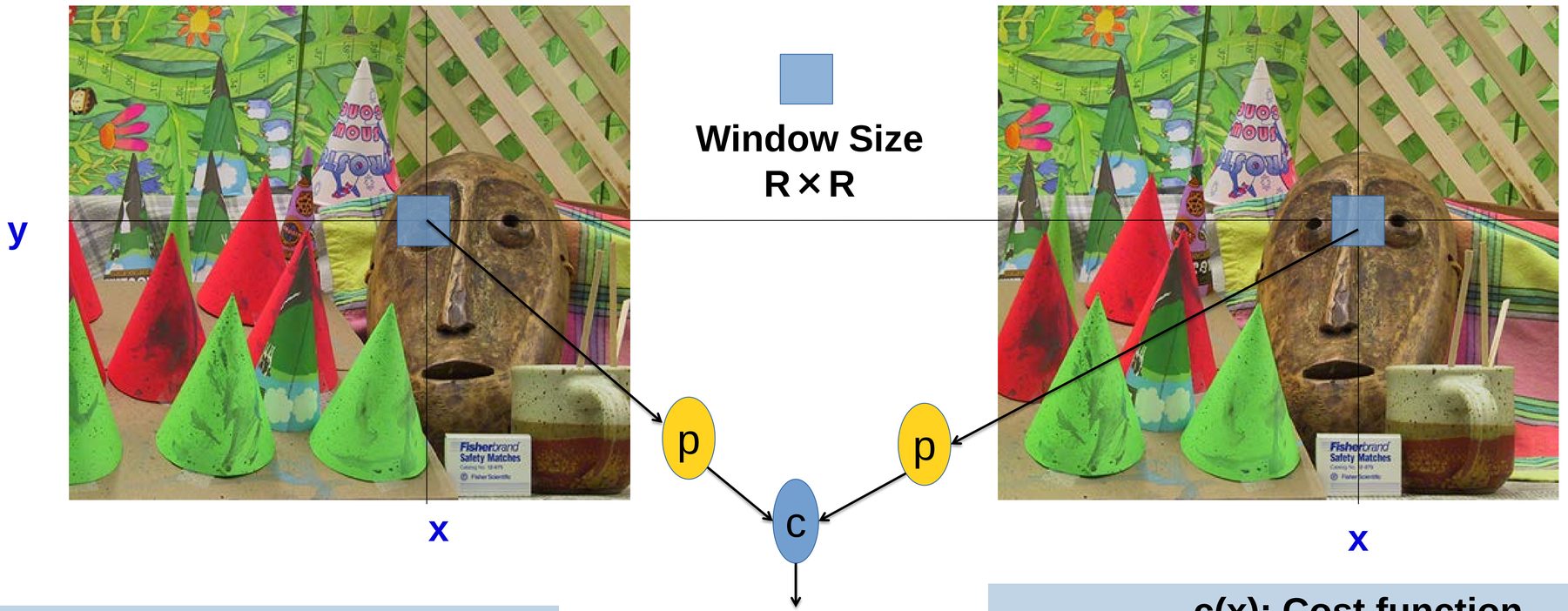
Find pixel $\boldsymbol{p}_{x,y}$ from Base Image (Left) into Matching Image (Right)

Only at the left along epipolar line

Stereo Matching: General Scheme

Left Image = Base Image

Right Image = Matching Image



p(x): Preprocessing function
Laplacian of Gaussian, Bilateral filtering, Rank transform, **Census transform**

matching cost

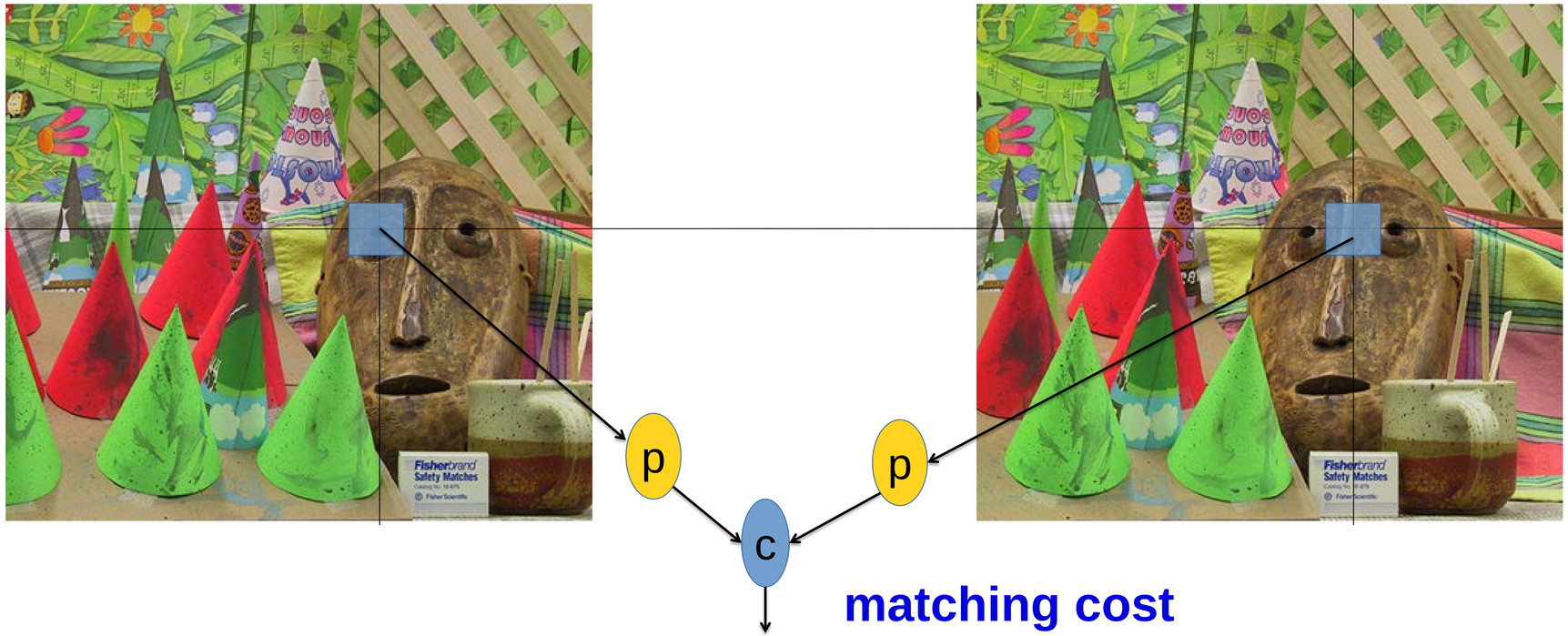
c(x): Cost function
Sum of Absolute or Squared or Truncated Absolute Differences, Mutual Information, **Hamming Distance**

- **Support window**: reduces ambiguity
- **Preprocessing + Cost function:** compensates for photometric distortions

Stereo Matching: Computing Disparity

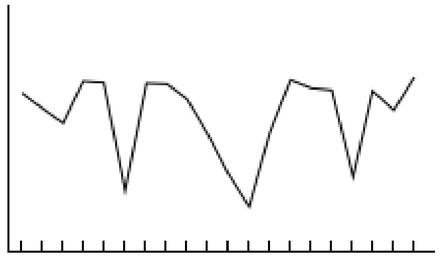
Left Image = Base Image

Right Image = Matching Image



100	85	95	12	69	46	32	112	95	99

d=0 … Dmax



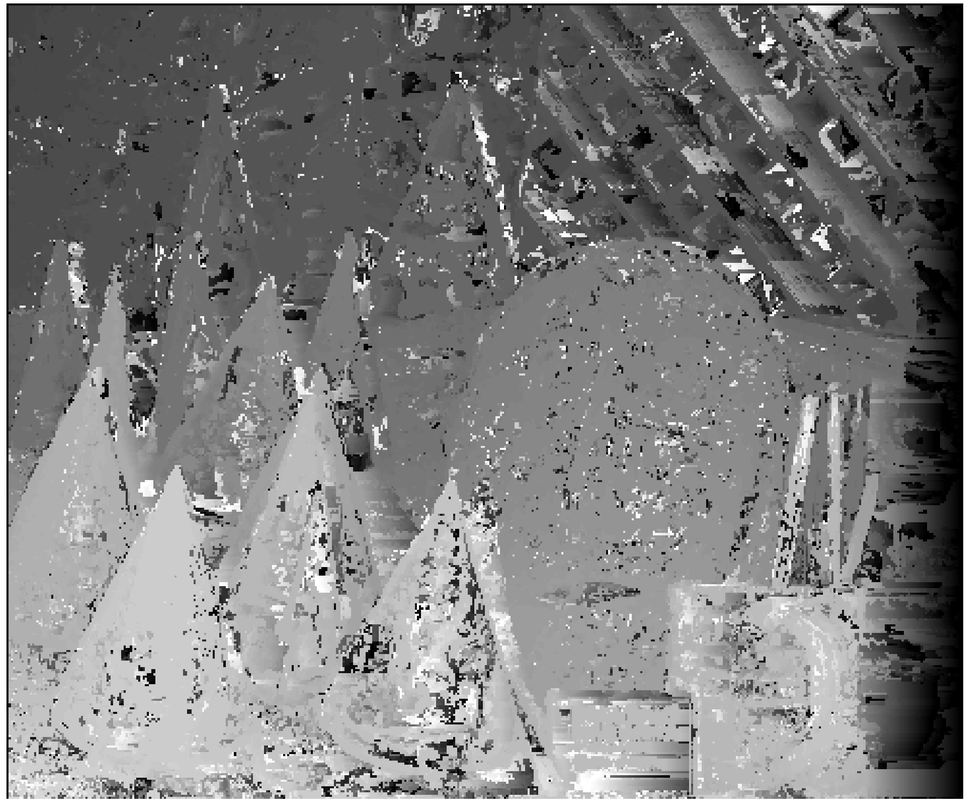
d: Disparity

?

Stereo Matching: Local algorithm

Local method

disparity d is index with minimum cost



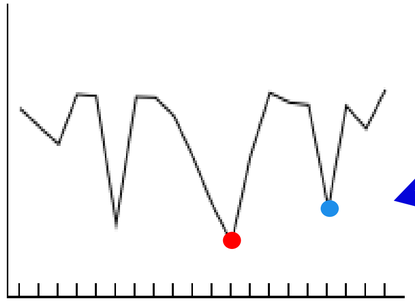
?

Consider **global information**

Decision depends on neighbors

100	85	95	**12**	69	46	**32**	112	95	99

d



A better candidate?

Stereo Matching: Global algorithm

Global algorithm

Key assumption: Changes in distance are smooth

Define energy function over images and minimize it

Local Cost

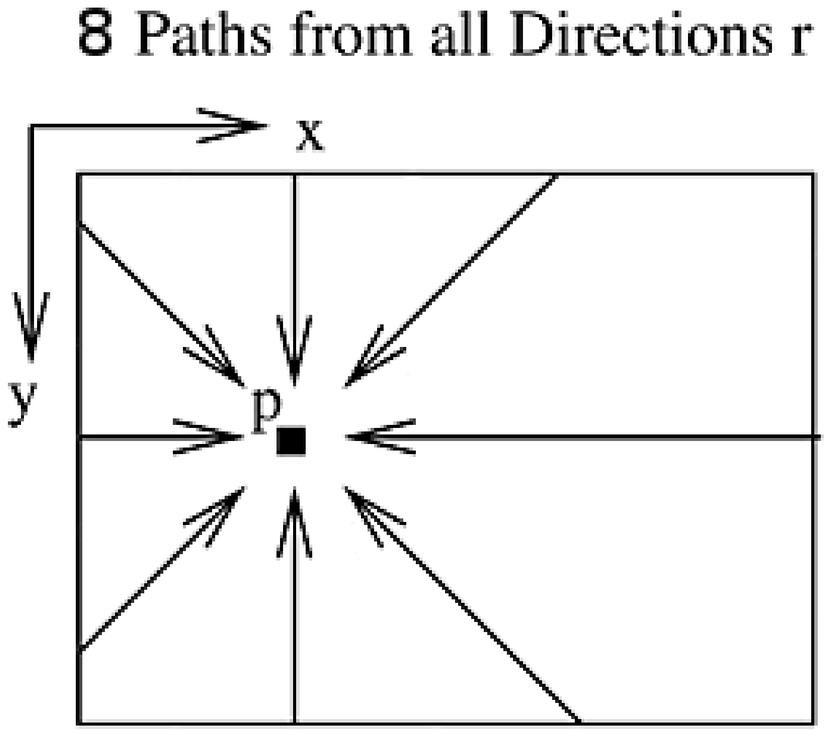
Considers neighbor pixels

$$E(D) = E_{data}(D) + E_{smooth}(D)$$

Problem: Global minimization in 2D is NP-Complete

But, there is a faster option …

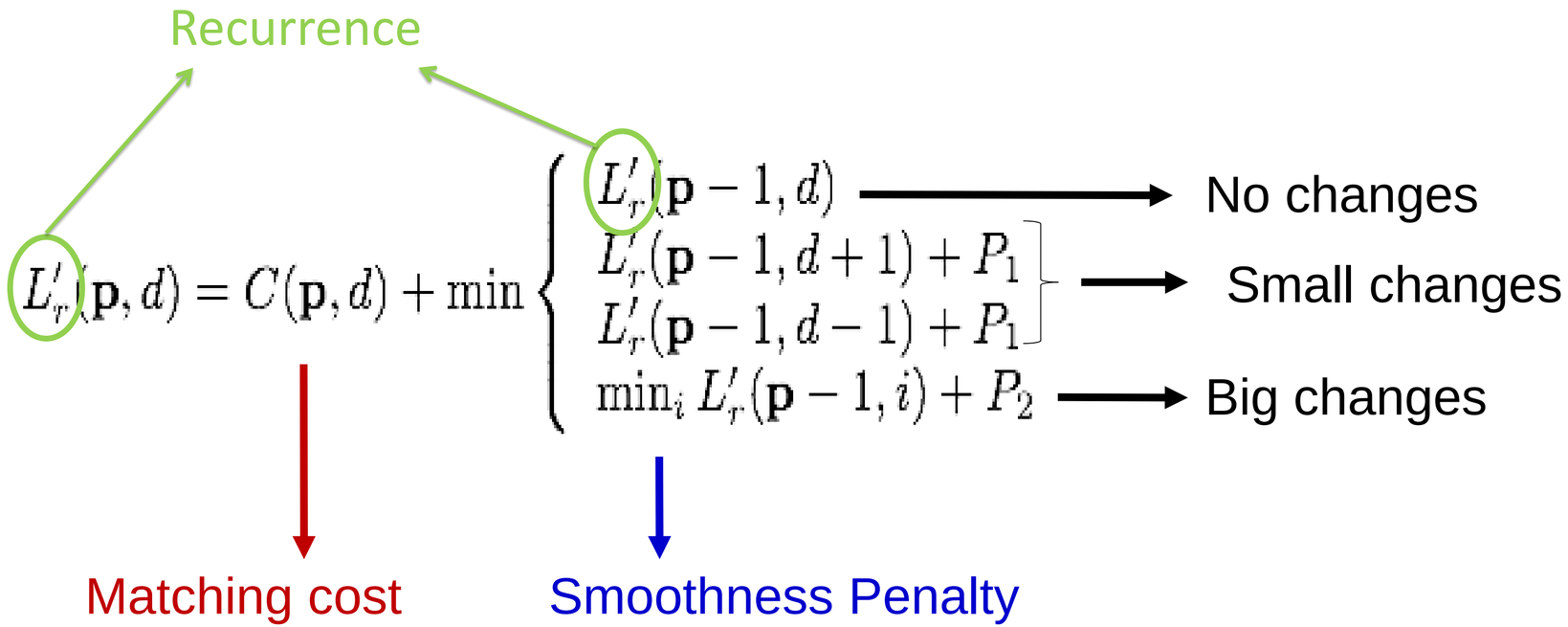
Stereo Matching: Semi-Global Matching (SGM)



SGM:

- add smoothing penalties along several directions in 1D (dynamic programming method)
- aggregate penalties from all directions and minimize
- similar accuracy of global algorithms, but lower computational complexity

SGM: Smoothness function



- **Key idea:** Penalize abrupt changes
- Penalty for small changes: considers slanted or curved surfaces
- Penalty for bigger changes: considers boundaries of real objects

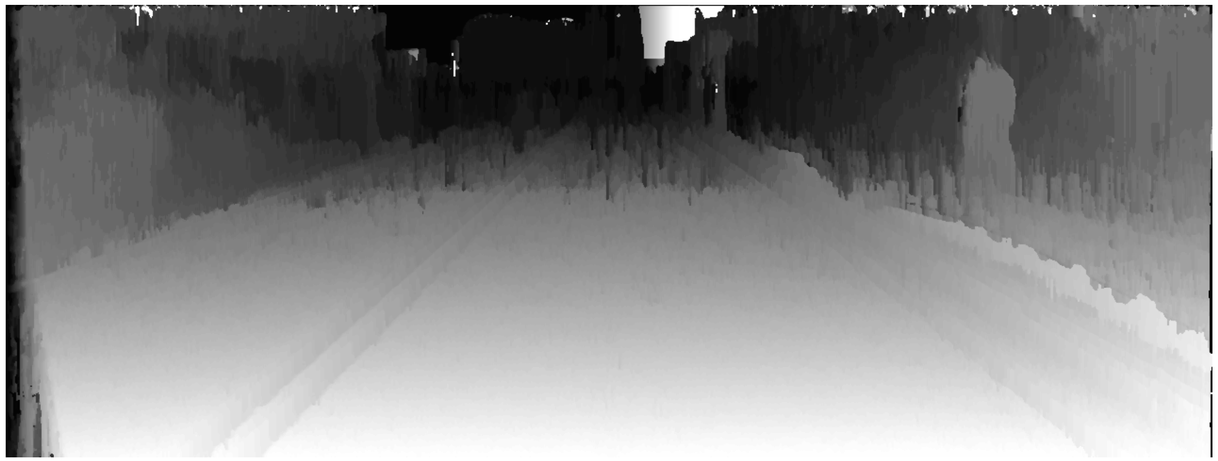
Semi-global matching: Visual Example

Visual Example: effect of using different path directions

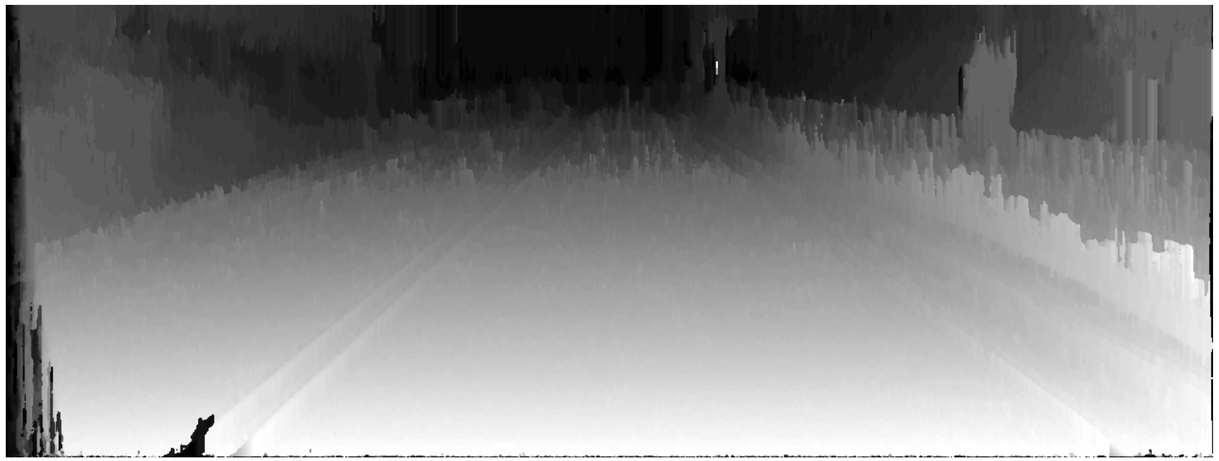
Direction →

Direction ←

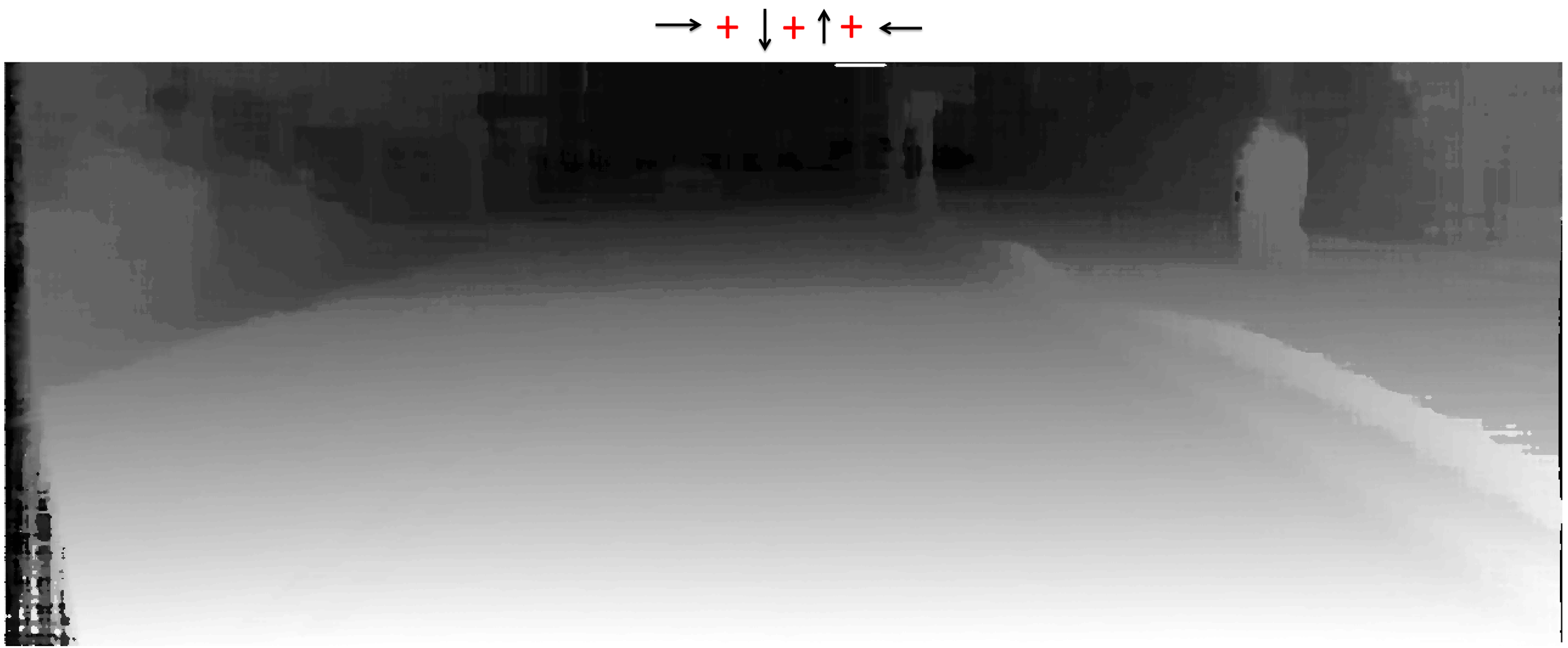
Direction ↓



Direction ↑



Visual Example: Effect of Aggregation



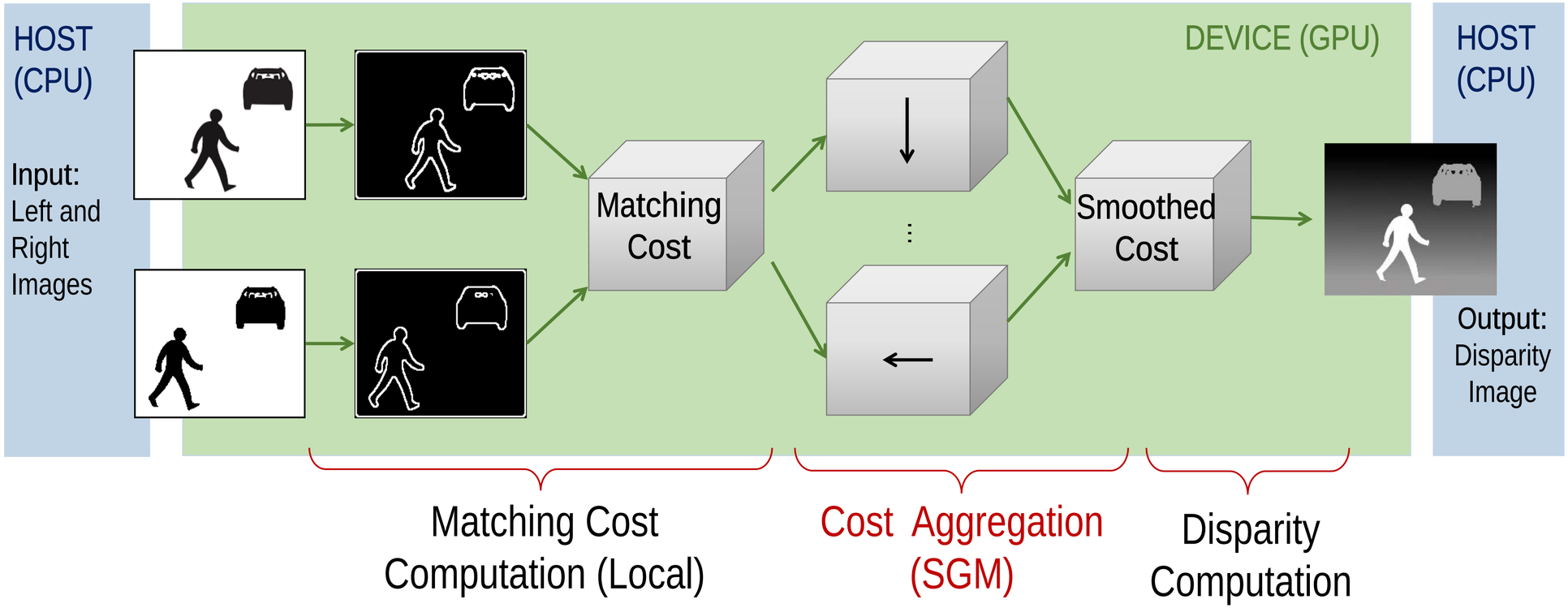
- Aggregation on all directions provides more **robust** results

Index

Massively Parallel Programming: Methodology

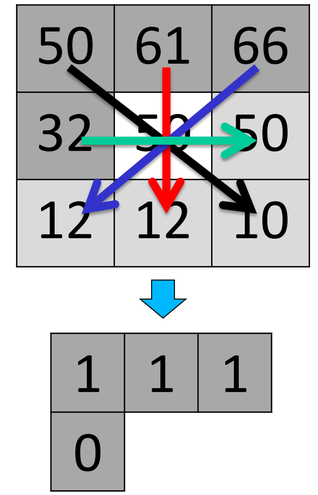
1. Identify massive **parallelism** & parallel **patterns** (Map, Reduce, Recurrence ...)
2. Propose **work distribution** into blocks of threads
3. Analyze **global** memory access
    - estimate **arithmetic intensity**
    - consider **data reuse** strategies
4. Design language-independent **pseudocode**
5. Translate to CUDA (or OpenCL, OpenACC …)

Pipeline: Host and Device mapping



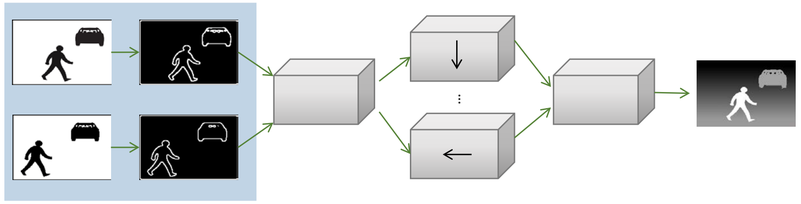
- Complete **stereo matching pipeline** implemented on GPU (accelerator)

CSCT Transform

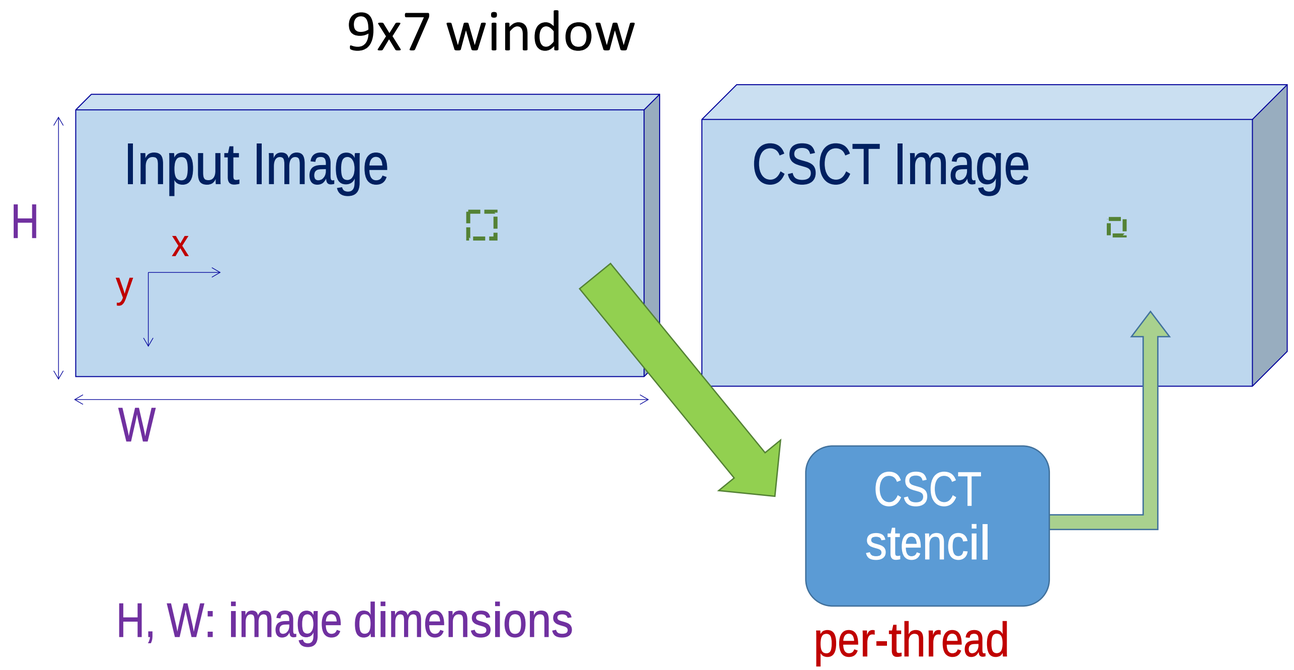


- **Rationale:** invariant to local intensity changes & tolerant to outliers
- Encode local neighborhood around each pixel into bit-vector

  9x7 window (31-bit result)



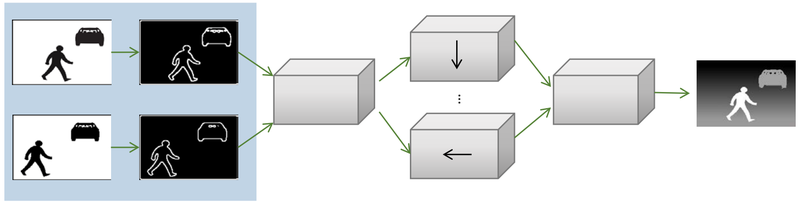
CSCT Transform: task distribution & performance analysis



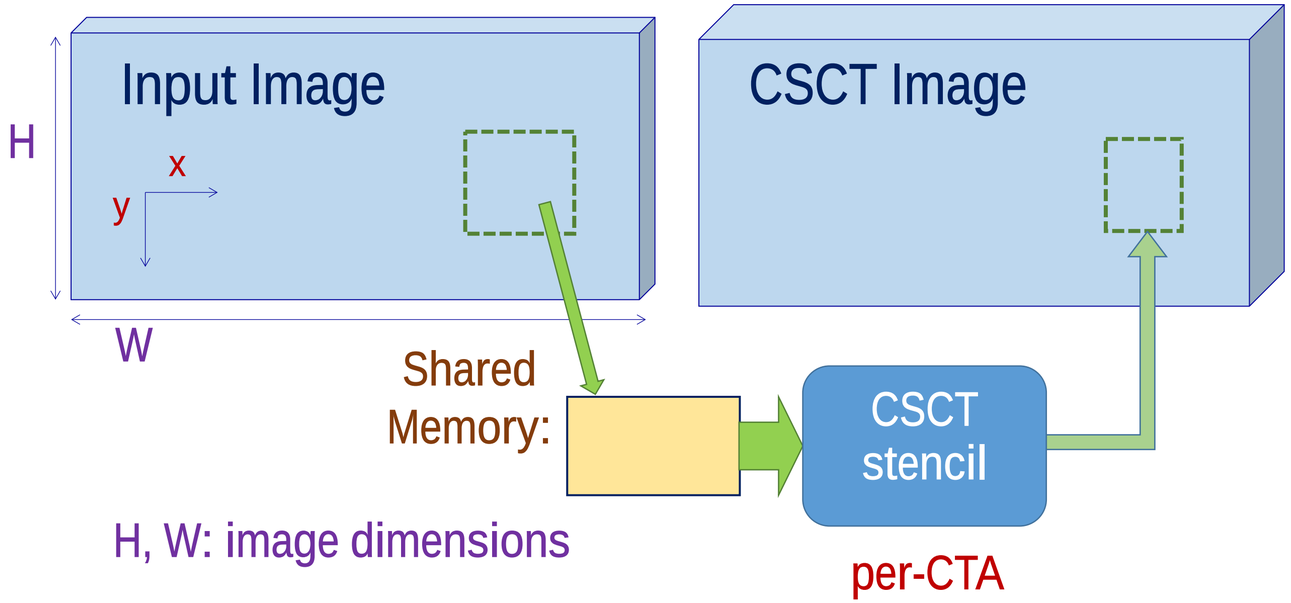
Parallelization Scheme	Naïve
Thread Parallelism (per image)	W×H
Compute Work per thread	62 ops
Total Global Data Read (Bytes)	62×W×H
Total Global Data Written (Bytes)	4×W×H

Arithmetic Intensity:
62 ops / (62+4) Bytes

1. 2D Stencil-pattern: parallelism in x & y axis
2. Naïve scheme: embarrasingly parallel (per-thread)



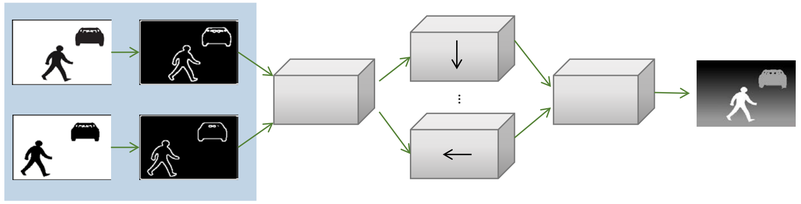
CSCT Transform: task distribution & performance analysis (2)



Parallelization Scheme	Naïve	2D-tiled
Thread Parallelism (per image)	W×H	W×H
Compute Work per thread	62 ops	62 ops
Total Global Data Read (Bytes)	62×W×H	≈ 1.5×W×H
Total Global Data Written (Bytes)	4×W×H	4×W×H

Arithmetic Intensity:
62 ops / 5,5 Bytes

1. 2D Stencil-pattern: parallelism in x & y axis
2. 2D input data tile: cooperative read (per CTA)
   Data reuse from shared memory (per thread)



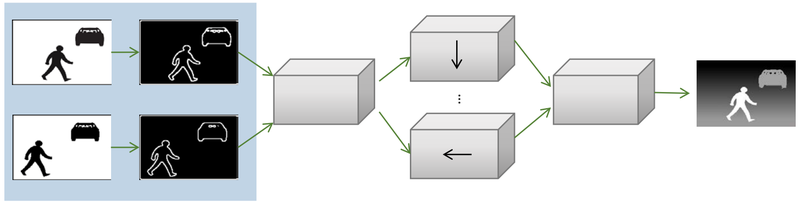
CSCT Transform: pseudocode

Algorithm 1: CSCT: 2D-tiled, read-cooperative parallel scheme

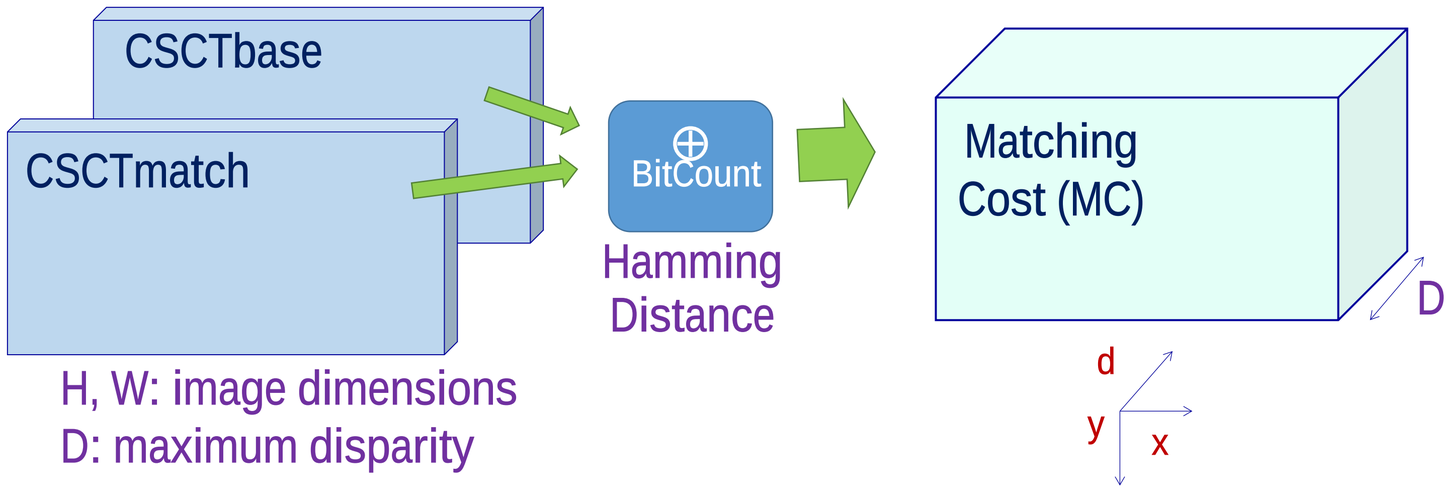
```
   input  : I[H][W], H, W
   output: CSCT[H][W]
1  parallel for y=0 to H step WarpSize do
2  |  parallel for x=0 to W step WarpSize do
3  |  |  CTA parallel for yCTA,xCTA=(0,0) to (WarpSize,WarpSize) do
4  |  |  |  copy (WarpSize+8)×(WarpSize+6) tile of I[][] into SharedI[][];
5  |  |  |  CTA Barrier Synchronization;
6  |  |  |  CSCT[y+yCTA][x+xCTA] = CSCT_{9,7}(SharedI, xCTA, yCTA);
```

→ per-CTA cooperative read

→ per-thread computation, reuse data from shared memory



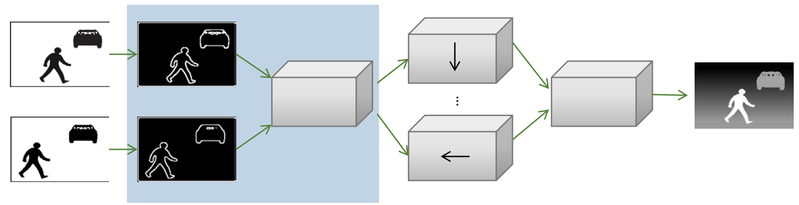
Matching Cost: task distribution & performance analysis



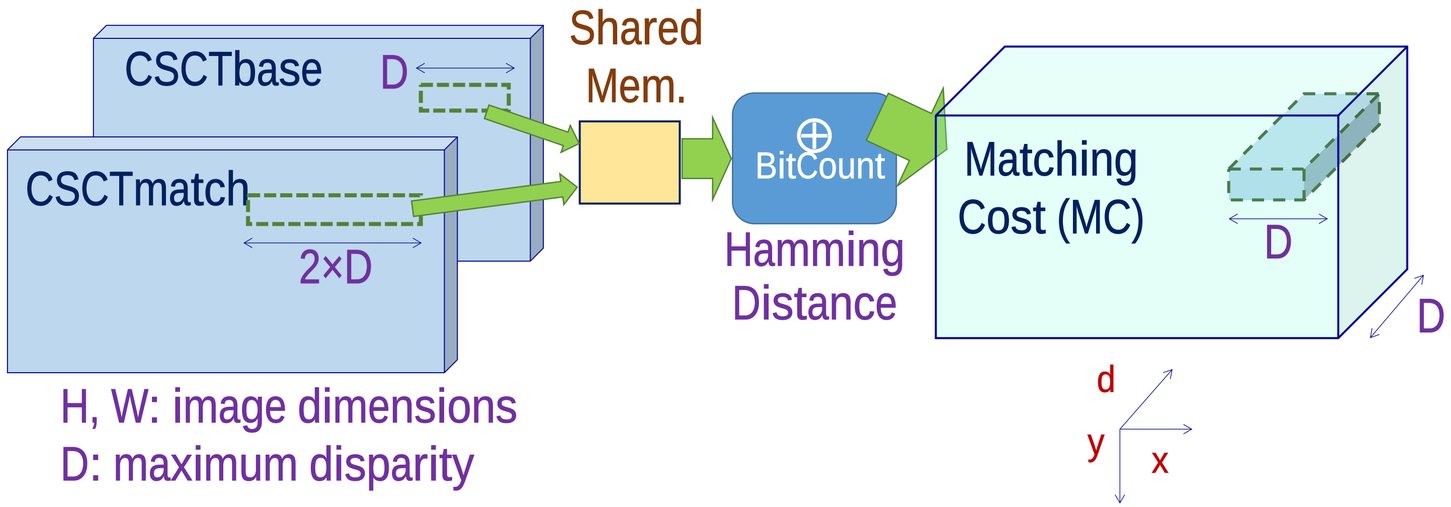
Parallelization Scheme	Naïve
Thread Parallelism	W×H×D
Compute Work per thread	2 ops
Total Global Data Read (Bytes)	8×W×H×D
Total Global Data Written (Bytes)	W×H×D

Arithmetic Intensity:
1 op / (8+1) Bytes

1. 2D to 3D pattern: parallelism in x, y & d axis
2. Naïve scheme: embarrasingly parallel (per-thread)



Matching Cost: task distribution & performance analysis (2)



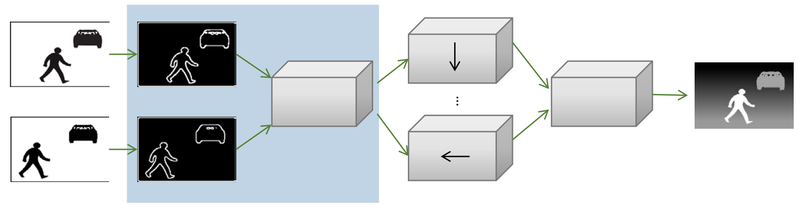
Parallelization Scheme	Naïve	1D-tiled
Thread Parallelism	W×H×D	W×H
Compute Work per thread	2 ops	2D ops
Total Global Data Read (Bytes)	8×W×H×D	12×W×H
Total Global Data Written (Bytes)	W×H×D	W×H×D

Arithmetic Intensity:
2D ops / (12+D) Bytes

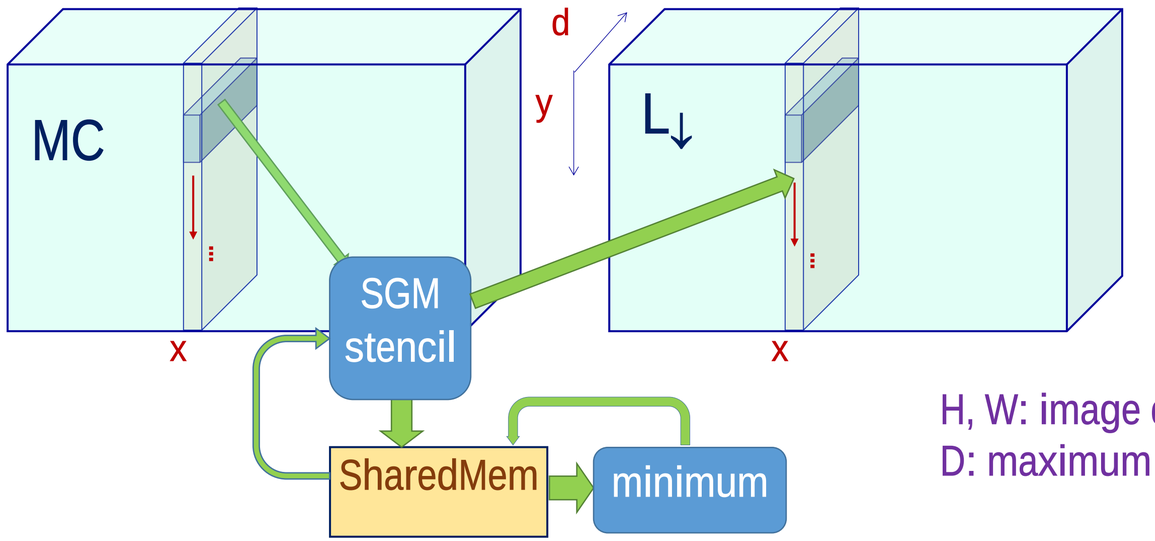
1. 2D to 3D pattern: parallelism in x, y & d axis
2. Increase work per thread to D output results (reduce parallelism)

   1D input data tile: cooperative read (per CTA)

   Data reuse from shared memory (per thread)



SGM Cost Aggregation: path direction ↓

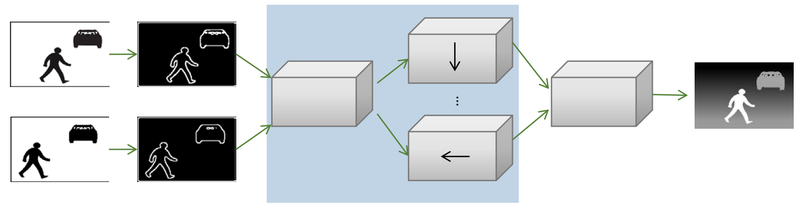


H, W: image dimensions
D: maximum disparity

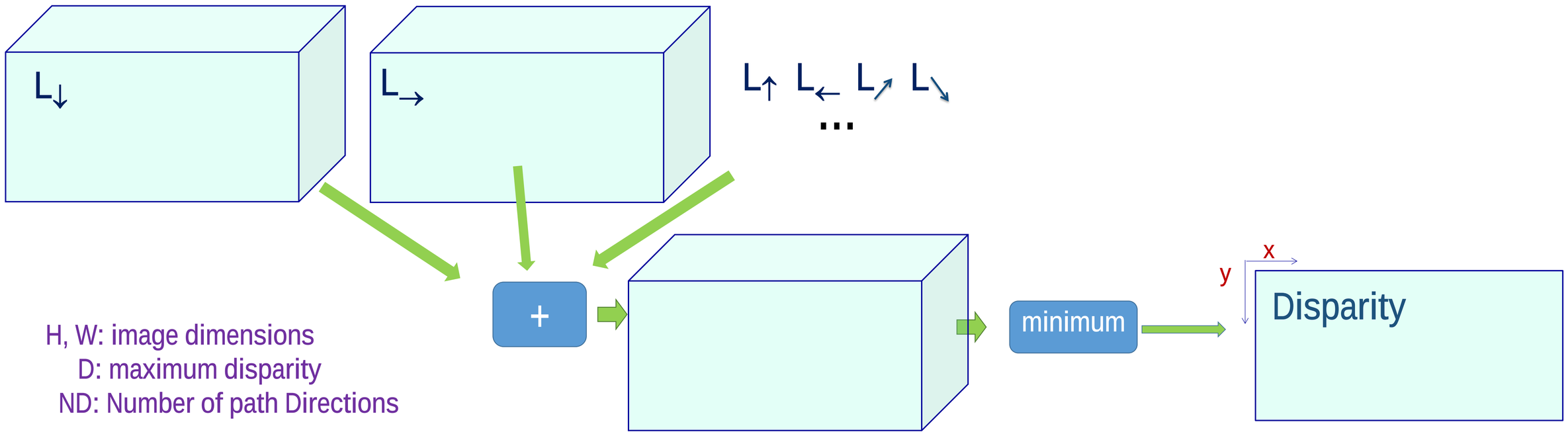
Parallelization Scheme	Recurrent
Thread Parallelism	W×D
Compute Work per thread	k×H ops
Total Global Data Read (Bytes)	H×W×D
Total Global Data Written (Bytes)	H×W×D

Arithmetic Intensity:
kH ops / 2H Bytes

1. Parallelism in x axis + Recurrent pattern in y axis +
   3-1 stencil and minimum reduction in d axis
2. 1D CTA arrangement:
   stencil communication (SharedM) & cooperative minimum reduction

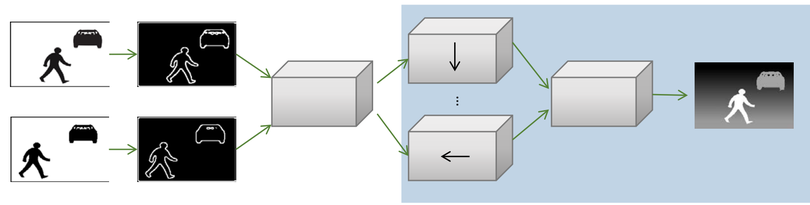


SGM aggregation and Disparity Computation

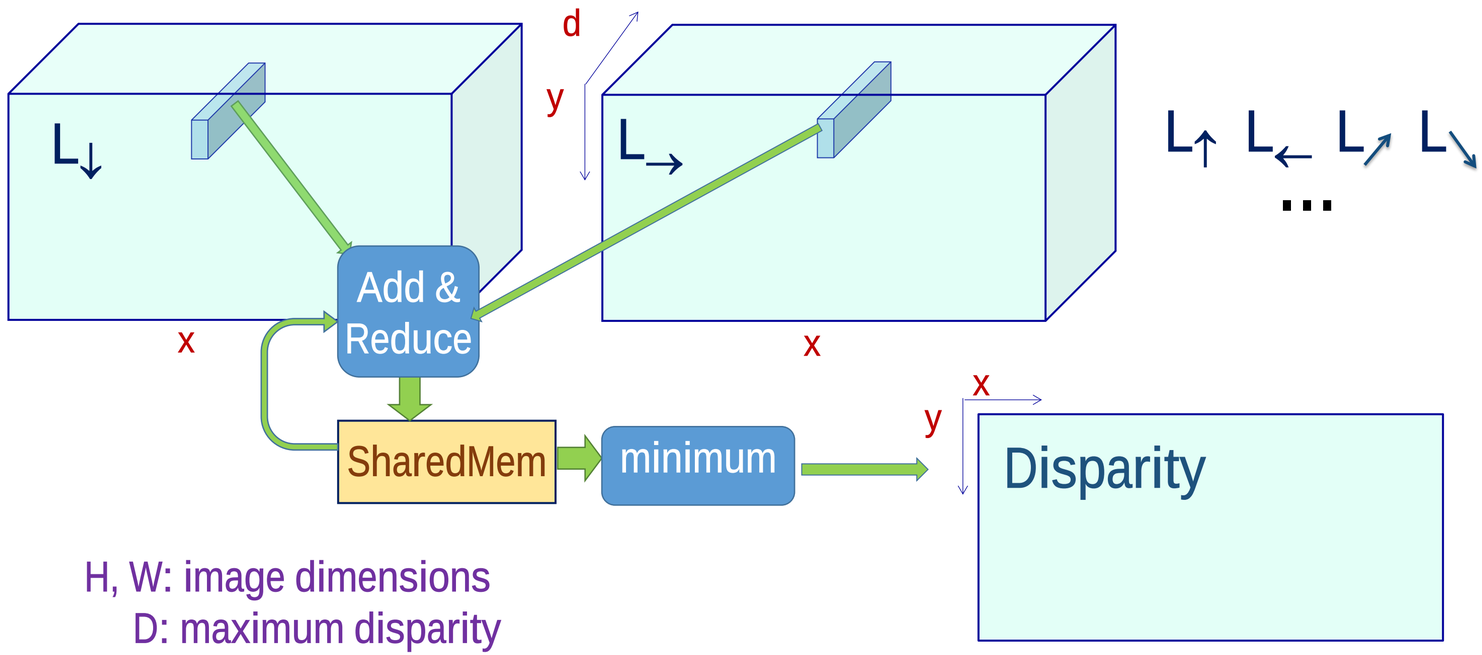


1. 3D addition to 3D: parallelism in x, y and d axis

   3D to 2D (d axis) reduction pattern (minimum)



SGM aggregation and Disparity Computation (2)



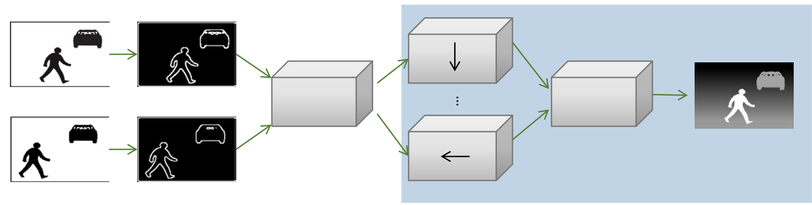
$L_{\uparrow}$ $L_{\leftarrow}$ $L_{\nearrow}$ $L_{\searrow}$ ...

H, W: image dimensions
D: maximum disparity
ND: Number of path Directions

Parallelization Scheme	reduction
Thread Parallelism	W×H×D
Compute Work per thread	ND ops
Total Global Data Read (Bytes)	ND×H×W×D
Total Global Data Written (Bytes)	H×W

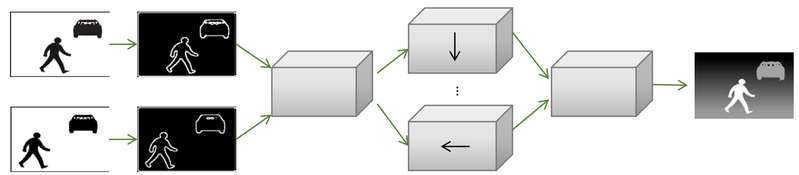
Arithmetic Intensity:
ND ops / ND Bytes

1. 3D to 2D (d axis) reduction pattern: parallelism in x & y axis
2. 1D CTA arrangement: cooperative minimum reduction



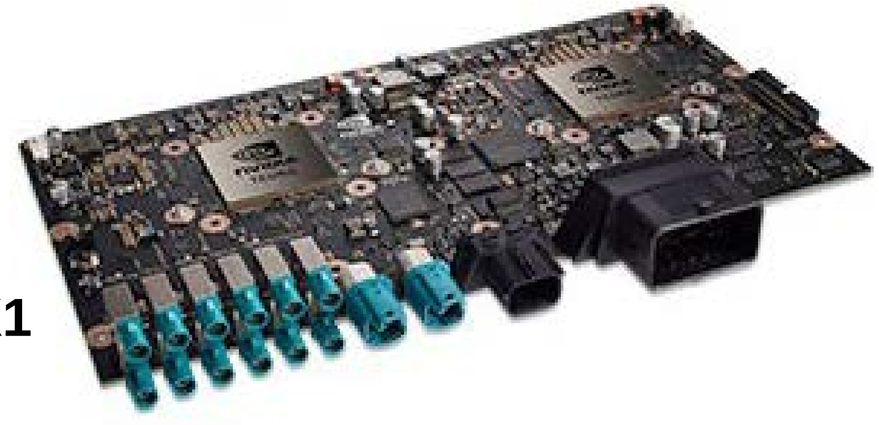
Additional Optimizations

- **Fused**: Last path + Disparity Computation (1.35× speed up)
- **Fused**: Matching Cost + horizontal path directions (1.13× speed up)
- **CTA-to-Warp** conversion (Kepler architecture and newer):
  - avoids shared memory by using fast register-to-register communication (*shuffle*)
  - reduces instruction count and increases instruction parallelism
- **Vectorize** Cost Aggregation inner loop: compute 4 costs at the same time (3× speed up)



Index

Experimental Methodology



NVIDIA Tegra X1

Cores: 256

Peak Bandwidth: 25.6 GB/s

TDP 10 W

NVIDIA Titan X

Cores: 3072

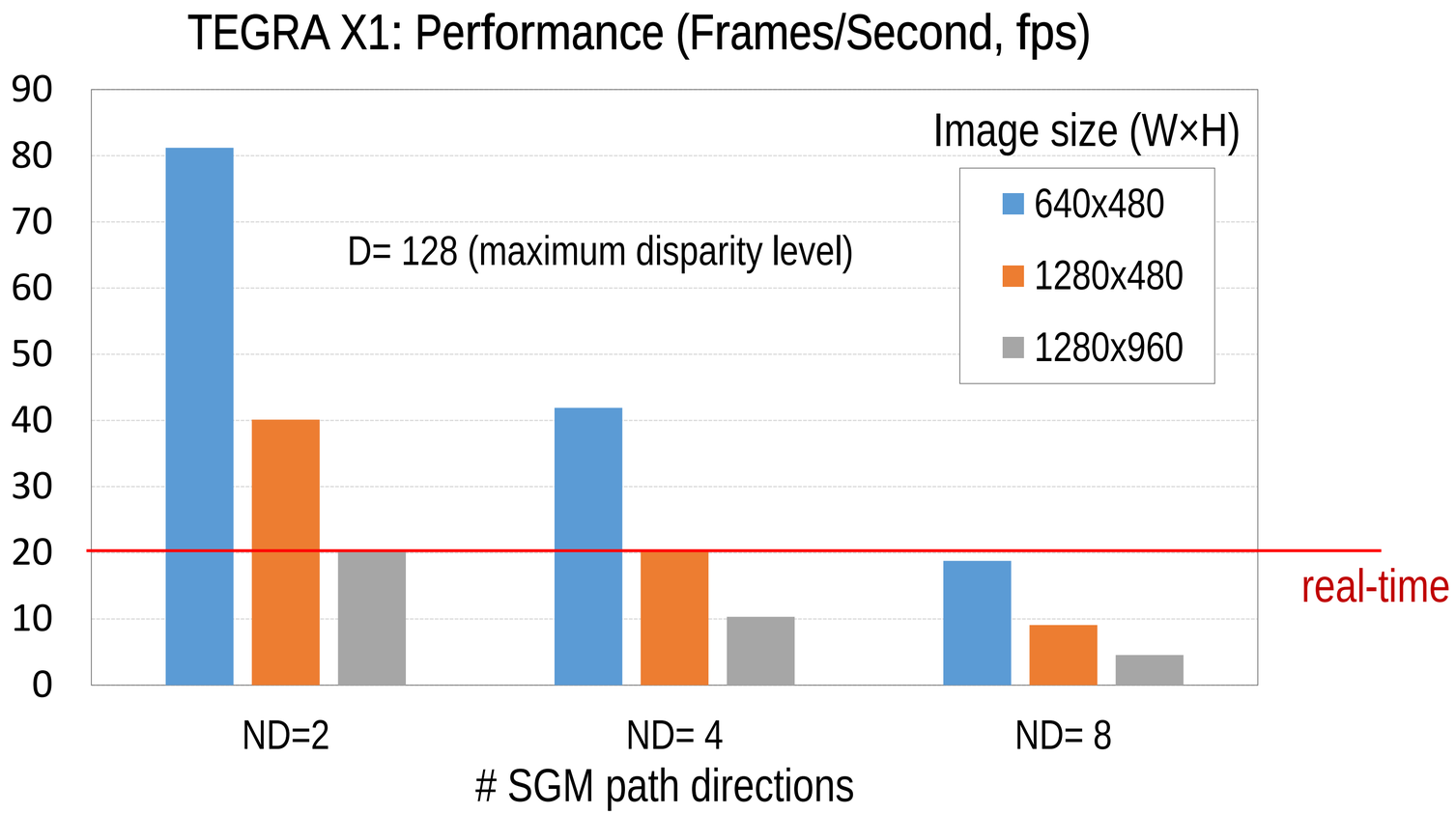
Peak Bandwidth: 336.5 GB/s

TDP 250 W

- **Metrics**:
  - Performance= Frames Per Second (fps)
  - Energy Efficiency= fps / Watt
  - Detailed performance per kernel: Instruction Count, Memory Bandwidth & IPC
- Images stored in memory

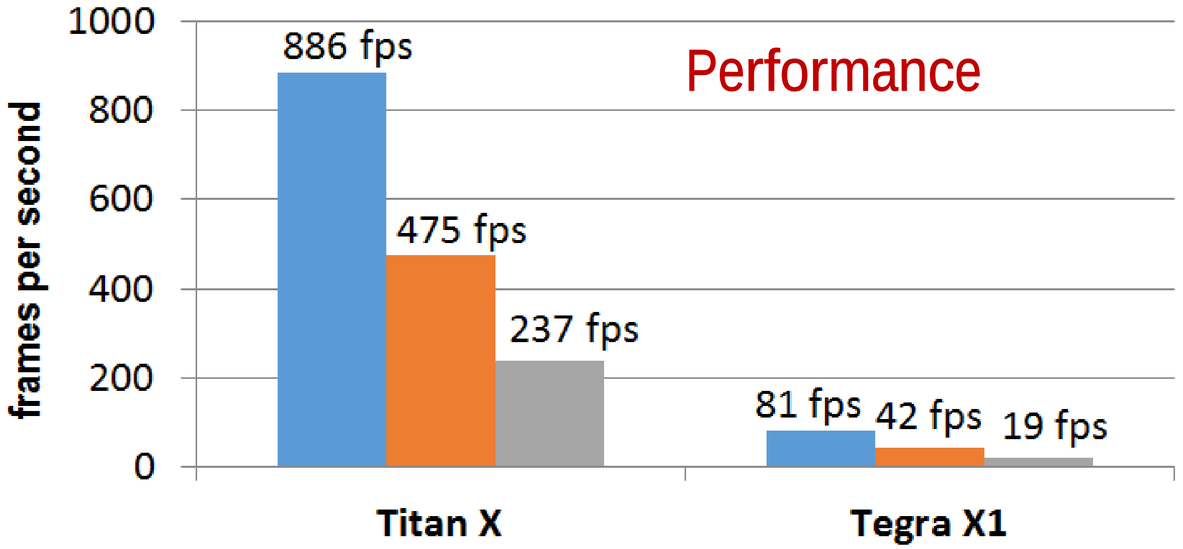
Results: Accuracy & Performance

Accuracy (ND: # of SGM Path Directions)



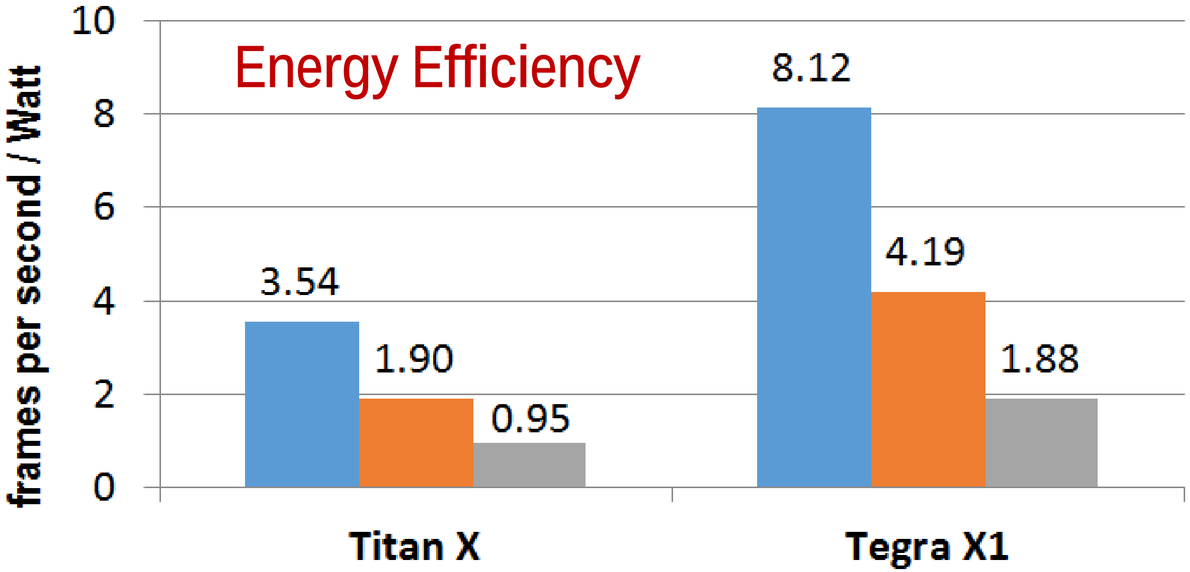
- Accuracy reduced very slightly with 4 SGM path directions with respect to 8 SGM path directions
- Real-time on Tegra X1 is achieved for 4 SGM path directions on images of 1280x480 pixels

Results: comparison with Titan X



W×H = 640×480
D= 128

# path directions
ND= 2
ND= 4
ND= 8



- Titan X GPU provides more than 10× the performance of Tegra X1
- Tegra X1 offers more than 2× performance per Watt

Index

Conclusions & Future work

- Low-consumption embedded GPU systems (Tegra X1) are capable of attaining real-time stereo

- We have proposed baseline parallel schemes and data layouts that follow general optimization rules

---

- Evaluate performance on the new embedded Pascal GPUs with larger images and a higher number of disparity levels

- Include and evaluate well-known post-filtering algorithms:
  - Left-Right Consistency Check
  - Subpixel Estimation
  - Adaptive P2

Thank you

Daniel Hernández Juárez, *Alejandro Chacón, Antonio Espinosa, David Vázquez, Juan Carlos Moure and Antonio M. López*

More information:
www.cvc.uab.es/people/dhernandez

Autonomous University of Barcelona

caos

UAB
Universitat Autònoma de Barcelona

CVC
Centre de Visió per Computador