OpenCL Imaging on The GPU: Optical Flow

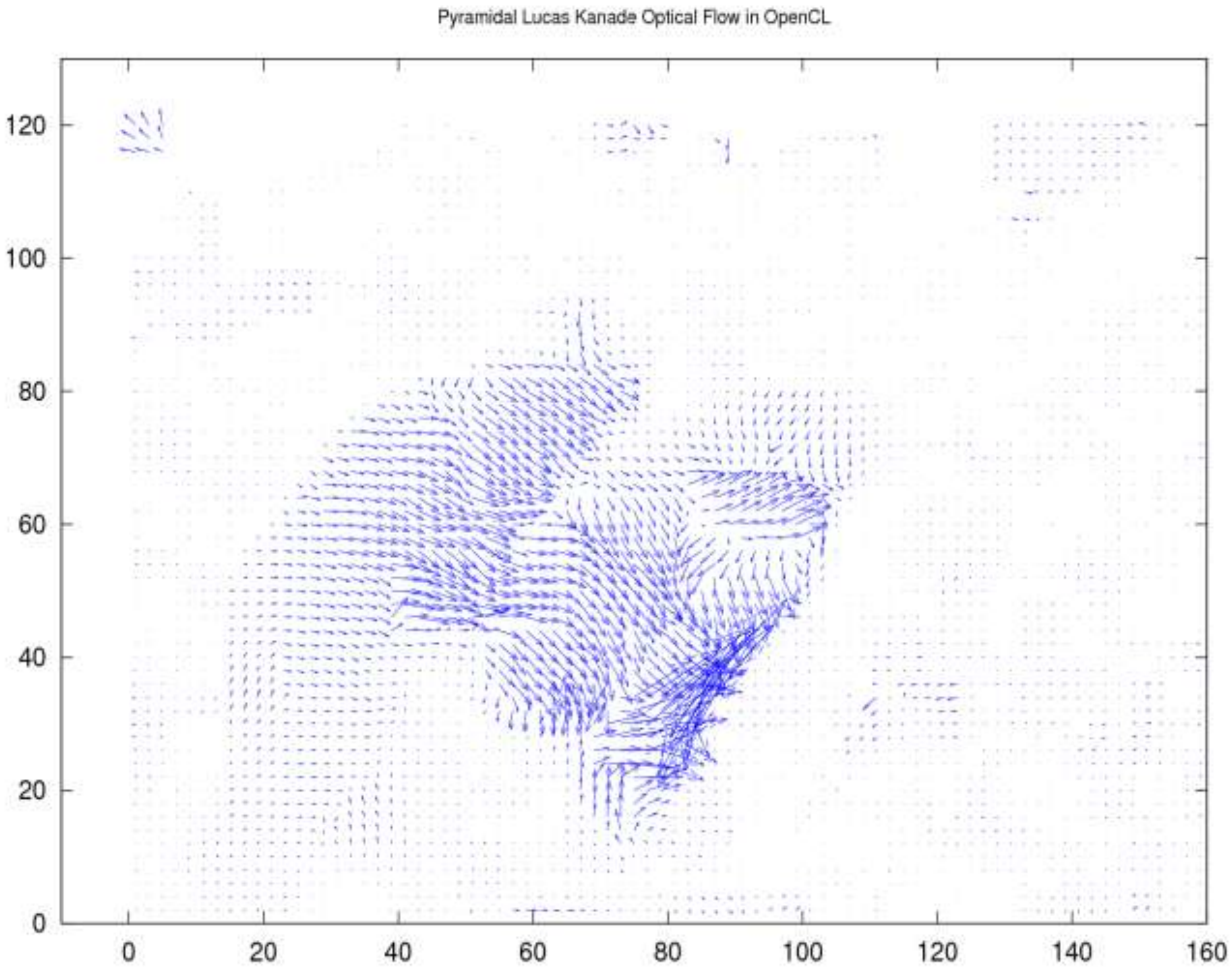
James Fung
Developer Technology, NVIDIA

Pyramidal Lucas Kanade Optical Flow in OpenCL

Algorithm by Jean-Yves Bouguet (Intel)
Implementation by James Fung (NVIDIA)

Optical Flow

- **Calculate movement of selected points in pairs of images**
- **Applications:**
  - Image stabilization
  - Feature tracking
  - Video encoding
- **May be used to track**
  - a few select points: sparse optical flow
  - All image points: dense optical flow
    - Computationally intensive!

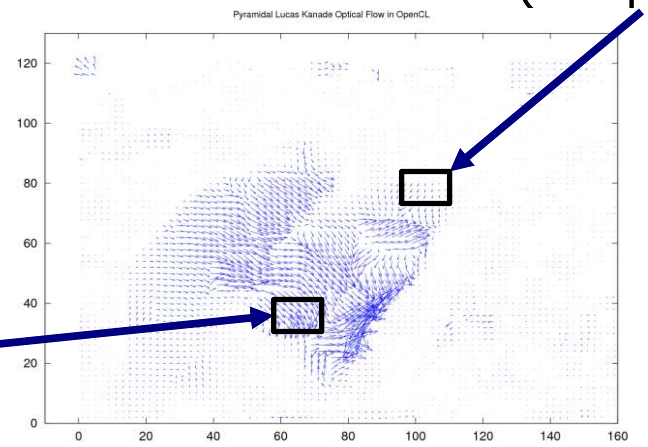

Pyramidal Lucas Kanade Optical Flow in OpenCL
0
20
40
60
80
100
120
0
20
40
60
80
100
120
140
160

Algorithm

- **For each point *u* on image *I*, find the corresponding point *v* on image *J***
- **Calculate spatial gradient *G* in vicinity of each point *u***
- **Iteratively solve for flow:**

Calculate image mismatch vector $\mathbf{b}$
Estimate motion $\quad \mathbf{v} = \mathbf{G}^{-1}\,\mathbf{b}(x',y')$
Update position $\quad (x',y') = (\,x'+v_x,\ y'+v_y\,)$
Repeat for N iterations or until convergence

Small motion
(< 3 pixels)



Large Motion
(3-7 pixels)

Algorithm

Downsample

Scharr Edge filter

Solve for flow
$v = G^{-1}b$

Source image pair

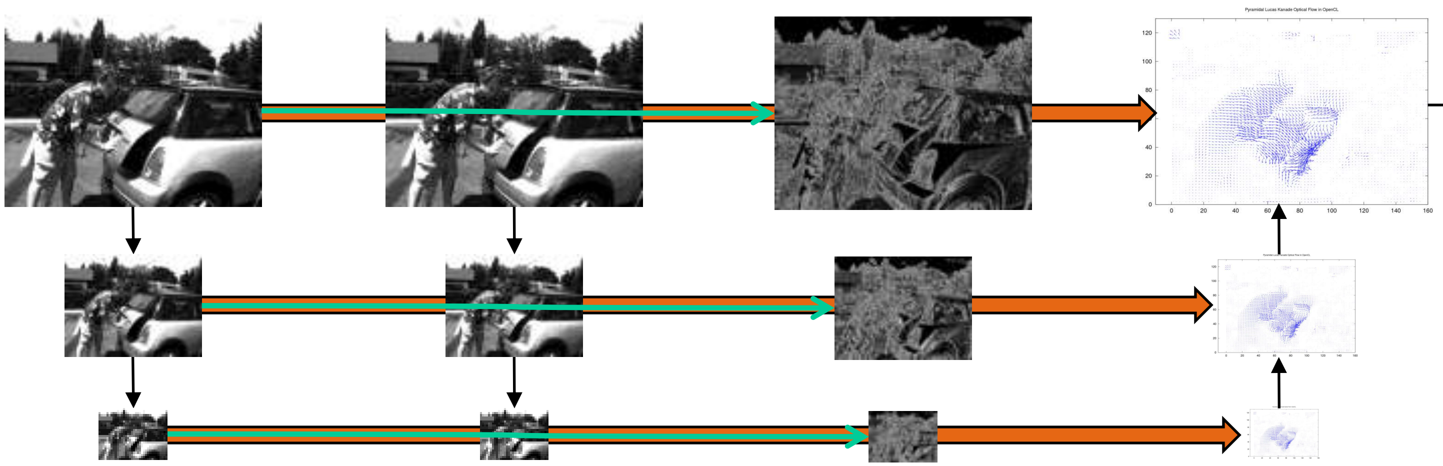
Precompute G from I

Image I

Image J

G_{2x2}

Flow (v)



Iterate $v = G^{-1}b$

Flow Result

Iterate $v = G^{-1}b$

Iterate $v = G^{-1}b$

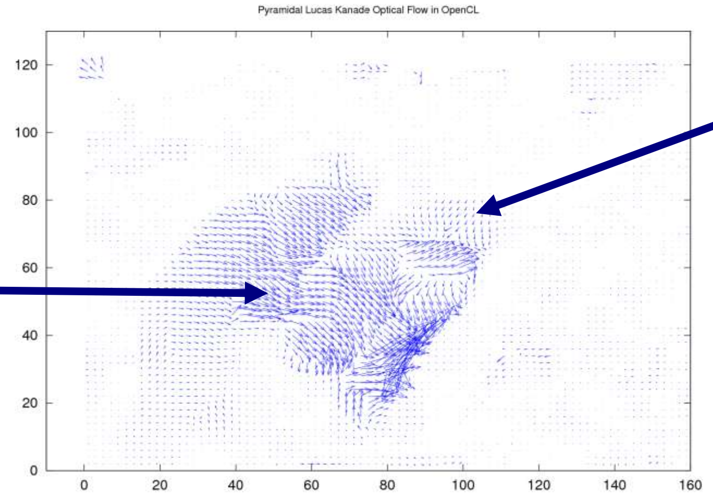
Optical Flow in OpenCL

- **Embarrassingly parallel**
- **Computationally intensive**
- **Can take advantage of special-purpose GPU hardware:**
  - Texture cache for 2D spatial locality
  - Hardware bilinear interpolation for sub-pixel lookups
  - Local Memory
- **Can use OpenCL-OpenGL interoperability for visualization**

Texture Cache

- **Different areas of image have different amounts of motion, but motion is spatially coherent**
  - Lookup window offset varies inside the image
  - *Texture cache* captures 2D locality

Large Motion
(3-7 pixels)



Small motion
(< 3 pixels)

Hardware Interpolation

- **Sub-pixel accuracy & sampling is crucial**
- **Between iterations, x,y is non-integer**

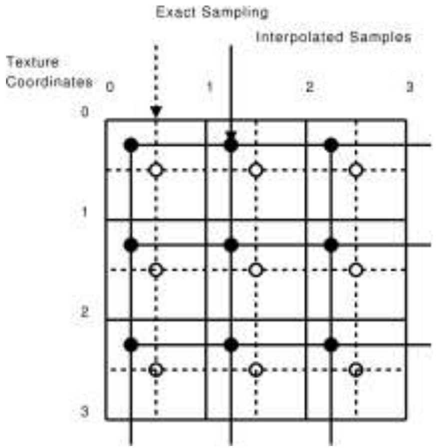
$$\partial I_k(x,y) = I^L(x,y) - J^L(x + g_x^L + v_x^L, y + g_y^L + v_y^L)$$

$$b_k = \sum_{x=p_x-w_x}^{p_x+w_x} \sum_{y=p_y-w_y}^{p_y+w_y} \begin{bmatrix} \partial I_k(x,y) I_x(x,y) \\ \partial I_k(x,y) I_y(x,y) \end{bmatrix}$$

Image level ***L***
Pixel center ***p***
Window ***w***
Guess ***g*** *from previous level* ***L+1***

- **Use texture hardware linear interpolation**

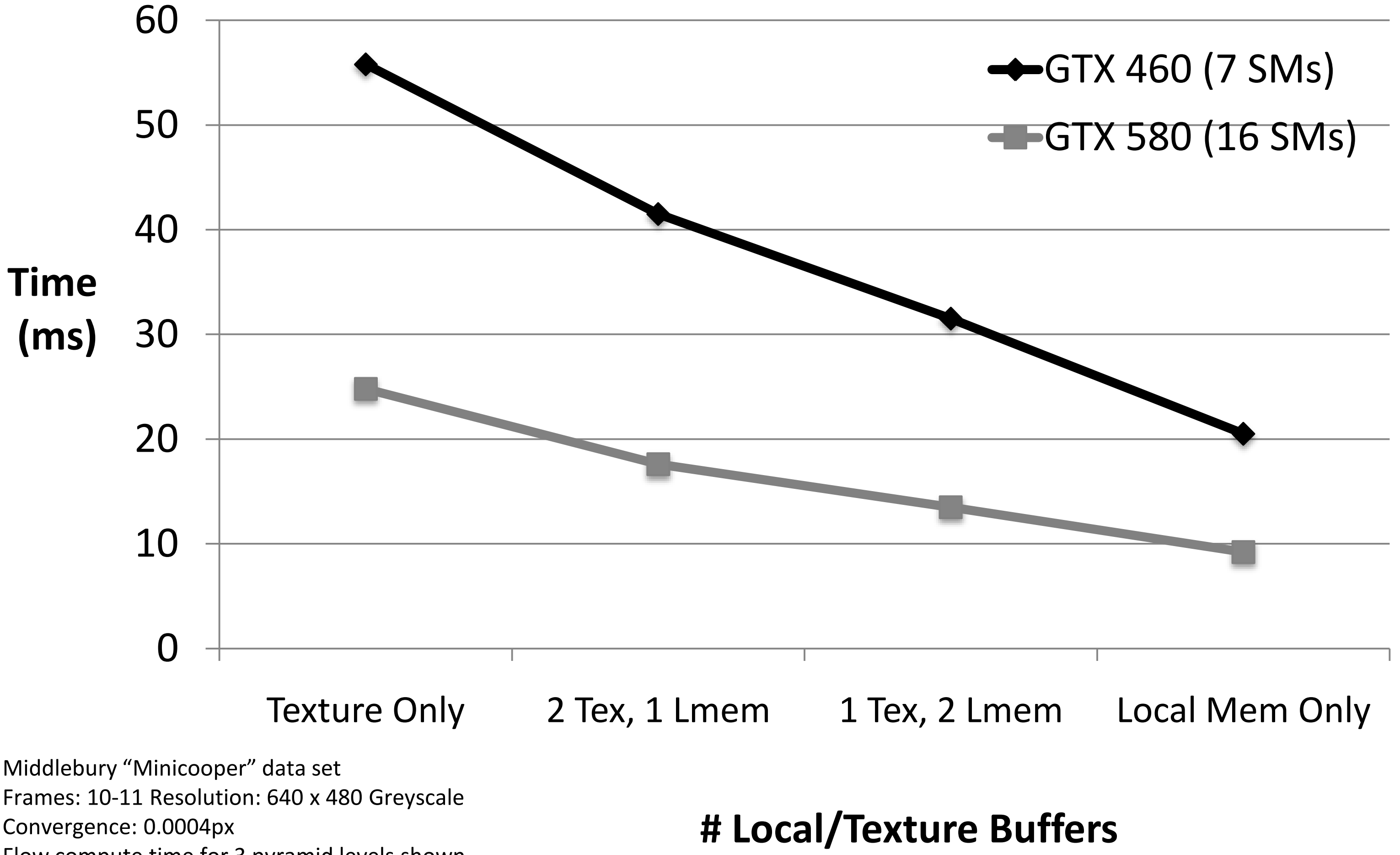
```
sampler_t bilinSample =
        ... | CLK_FILTER_LINEAR;
...
float Jsample = read_imagef( J_float,
        bilinSampler, Jidx+(float2)(i,j) ).x;
...
```



Using OpenCL Local Memory

- **Lookup locations for I, Ix and Iy are static per iteration**
- **Can be explicitly cached in Local Memory**
  - Local memory is on-chip
  - Local memory is faster to access, useful for repeated access

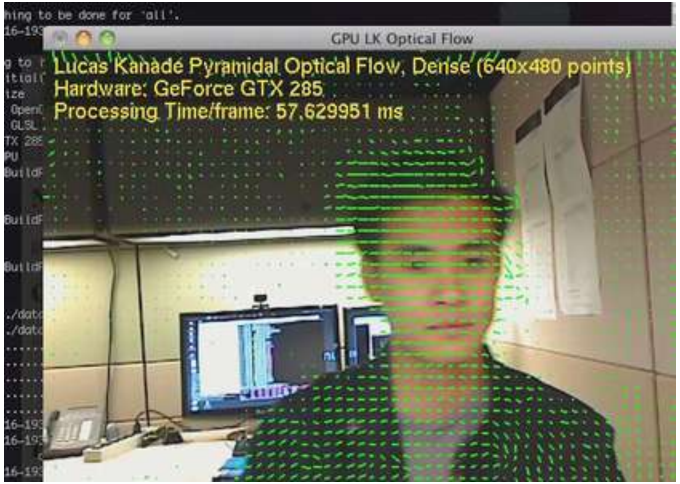
Effect of moving from Texture to Local Memory Usage



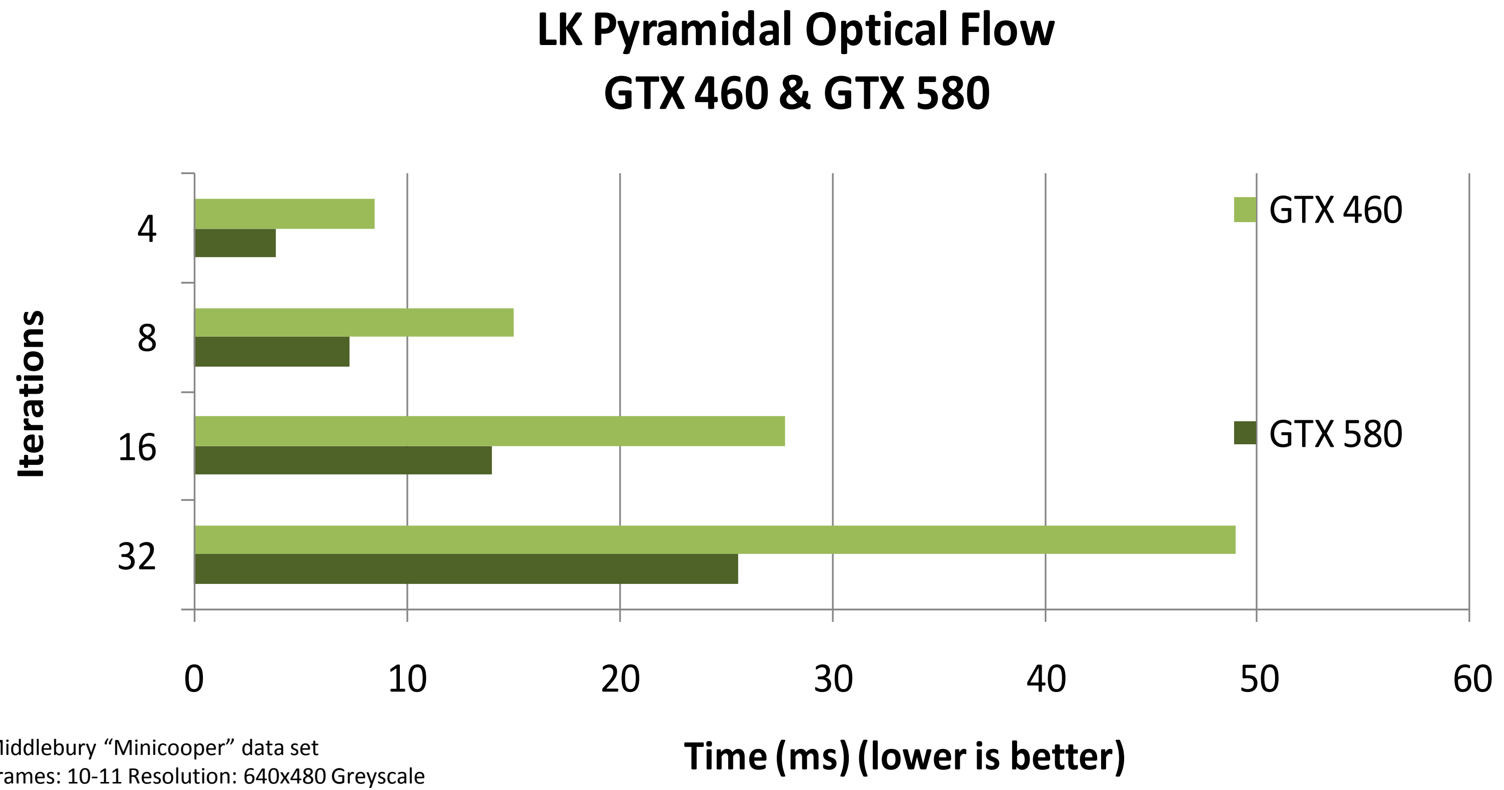
Middlebury "Minicooper" data set
Frames: 10-11 Resolution: 640 x 480 Greyscale
Convergence: 0.0004px
Flow compute time for 3 pyramid levels shown

Optical Flow Demo

- **Pyramidal Lucas Kanade Optical Flow**
- **Visualization done on GPU by sharing data between OpenGL & OpenCL**

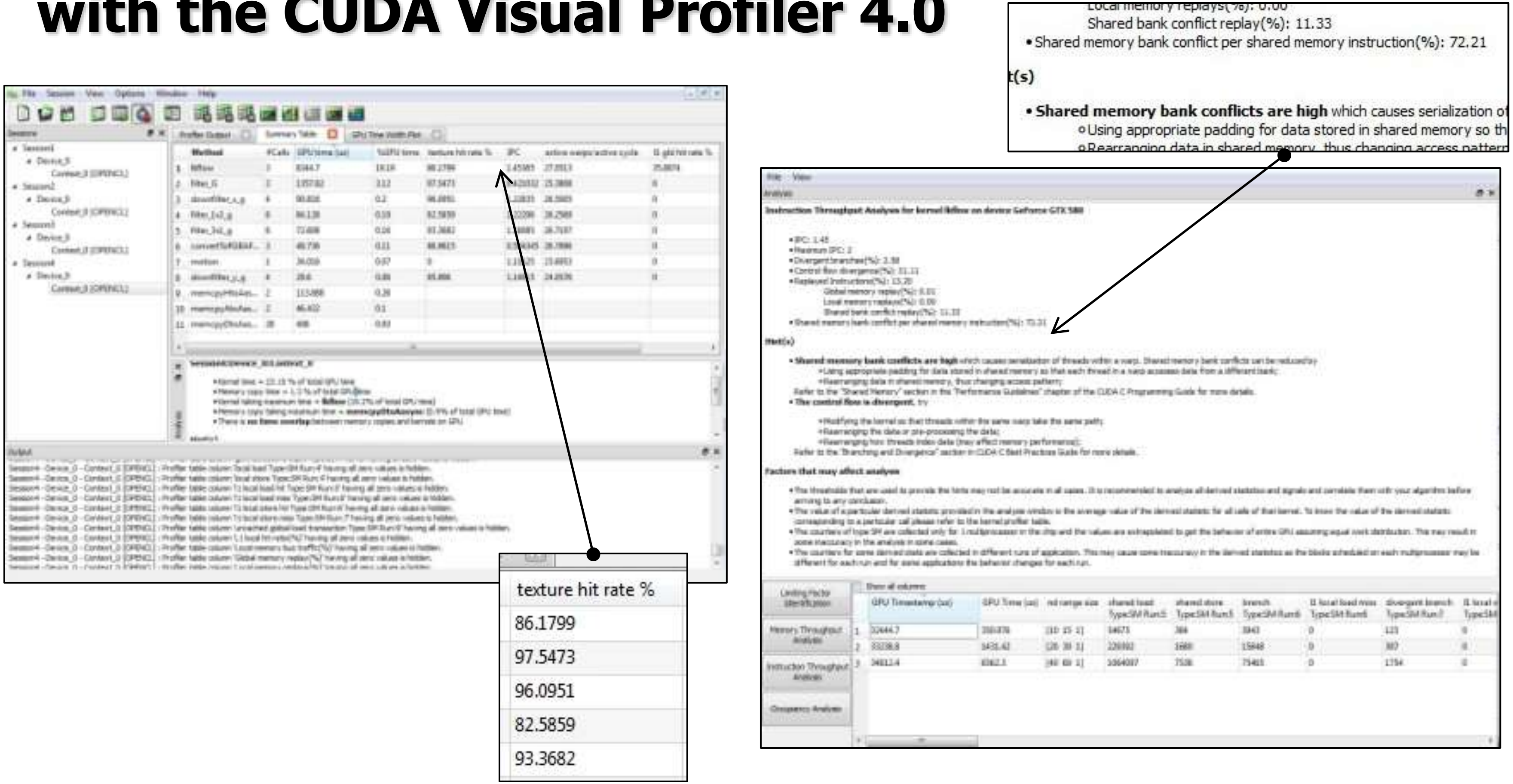


Optical Flow Performance



Middlebury “Minicooper” data set
Frames: 10-11 Resolution: 640x480 Greyscale
Convergence: 0.0004px
Pyramid 3-level flow compute time shown

OpenCL Performance Analysis with the CUDA Visual Profiler 4.0



Thank You!

Contact:

James Fung
jfung@nvidia.com