GpuCV: A GPU-accelerated framework for image processing and Computer Vision

Y. Allusse, P. Horain

Outline

- GpuCV in a few words
- Why accelerating Computer Vision and Image Processing?
- How can GPUs help?
- GpuCV description
- Results
- Future works
- Conclusion

GpuCV in a few words:

- **Initiated in 2005**
- **Aim: Accelerate computer vision with GPUs**
- **3 publications in major international conferences:**
  - ACM MM08 – Open Source competition.
  - ISCV08 – International Symposium on Visual Computing.
  - IEEE ICME06 – International Conference on Multimedia and Expo
- **Up to 100 daily visitors worldwide**
- **About 2.5 person·years effort**

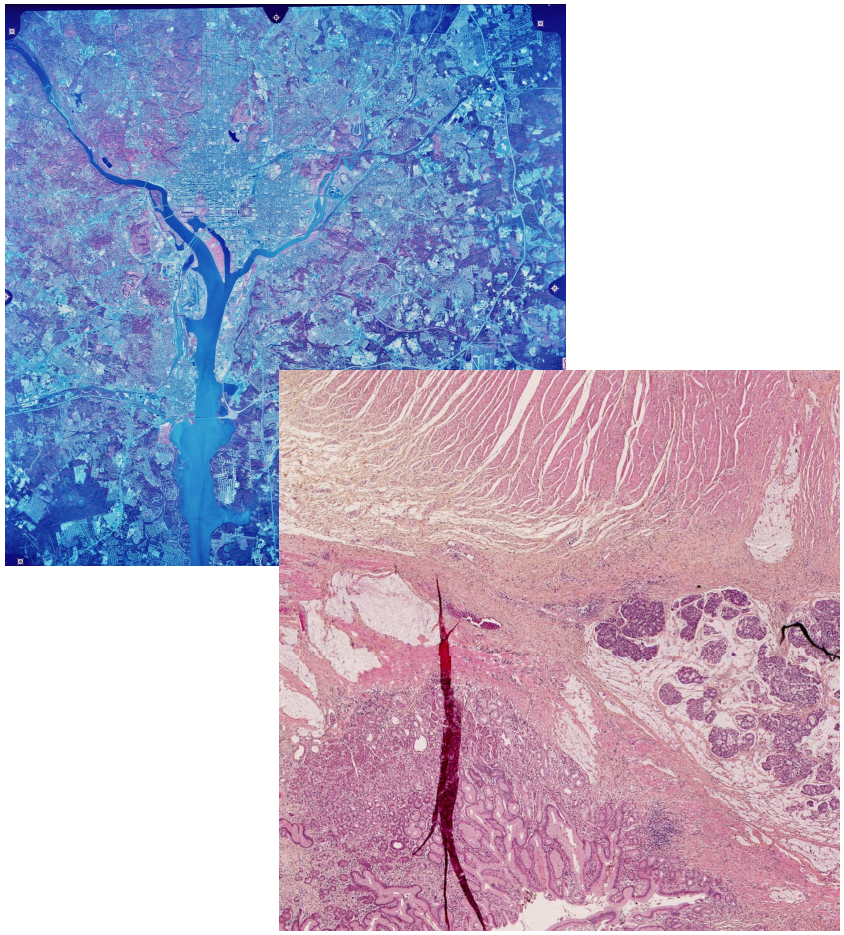
GpuCV

Why accelerating Computer Vision and Image Processing?

Processing large images & HD videos

Increasing data weight

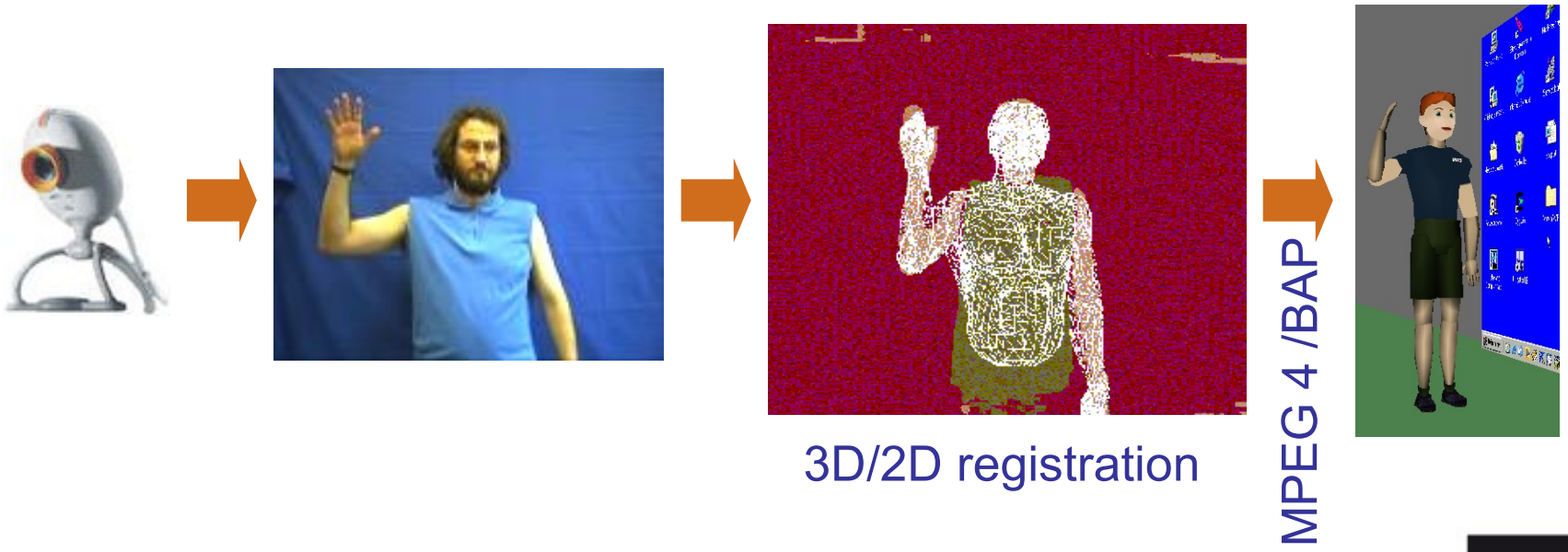
- Microscopy
- Satellite images
- Fine arts, printing
- Video databases



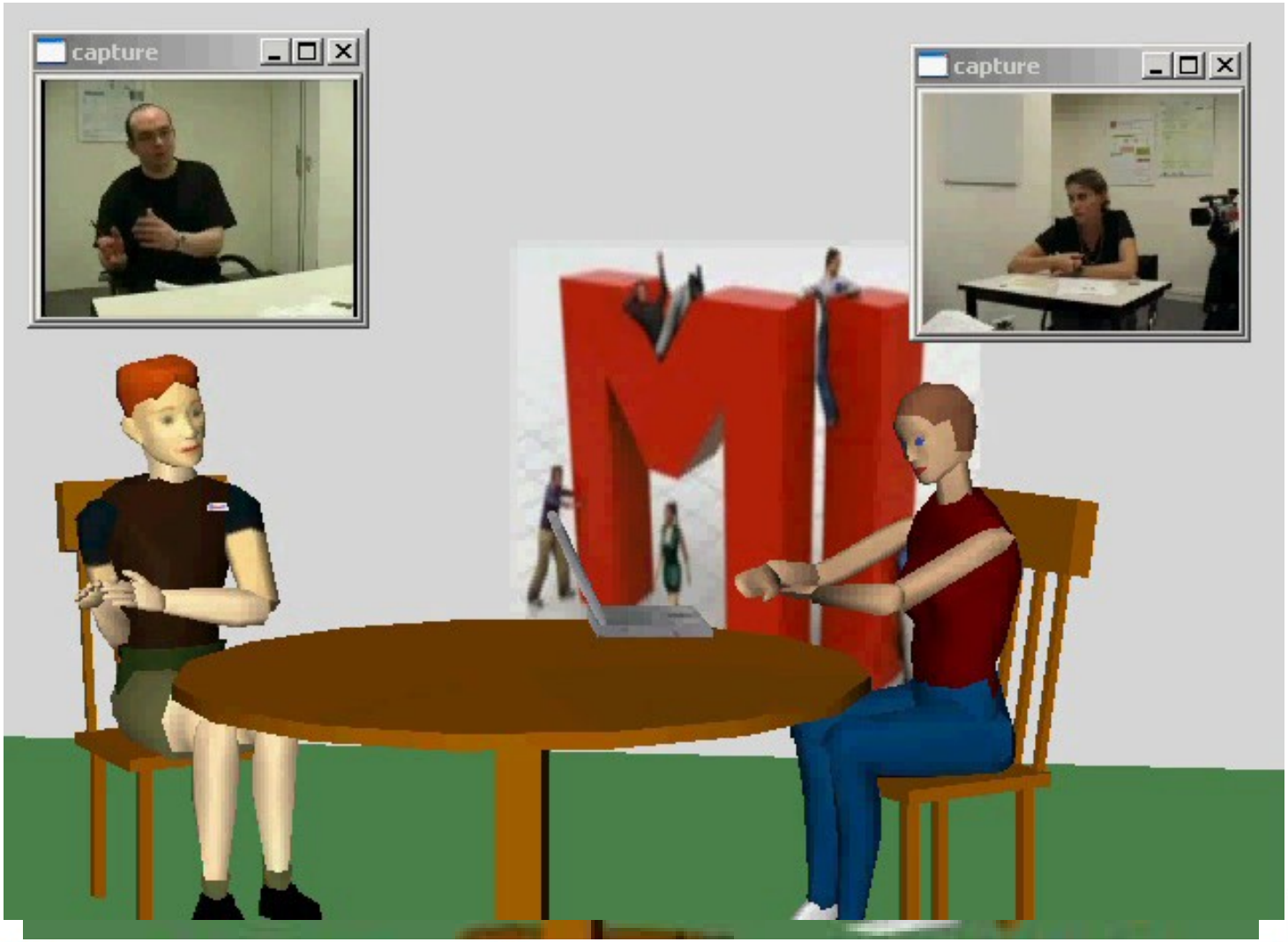
Up to 100 GBytes !

Real time computer vision

- **Interactive applications, security…**
  - Biometry
  - Multimodal applications
  - 3D motion tracking



Example application: Virtual conference



http://MyBlog3D.com demos

How to accelerate image processing?

Available technologies for acceleration:

- Extended instruction sets (ex.: Intel OpenCV / IPP)
- Multiple CPU cores (ex.: IBM Cell)
- Co-processor:
  - FPGA (field-programmable gate array)
  - GPU (graphical processing unit)

☺ GPUs available “for free” in consumer PCs

Graphics Processing Units

GPU: what?

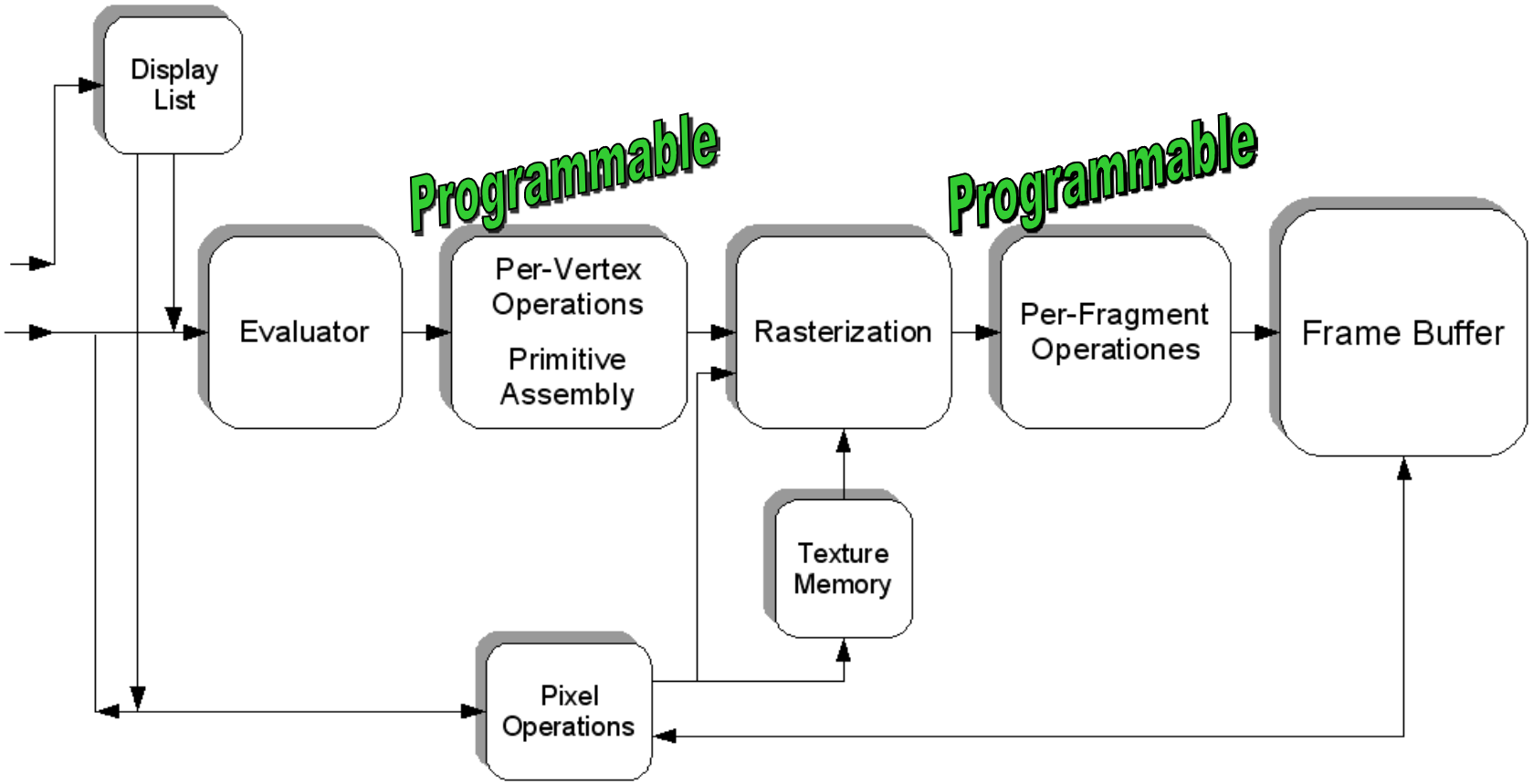
- **Originally in consumer PCs for gaming**
- **Designed for advanced rendering**
  - Multi-texturing effects.
  - Realistic lights and shadows effects.
  - Post processing visual effects.
- **Image rendering device**
  - ☺ **Highly parallel processor**
  - ☺ **High bandwidth memory**

GPU history: towards programming flexibility

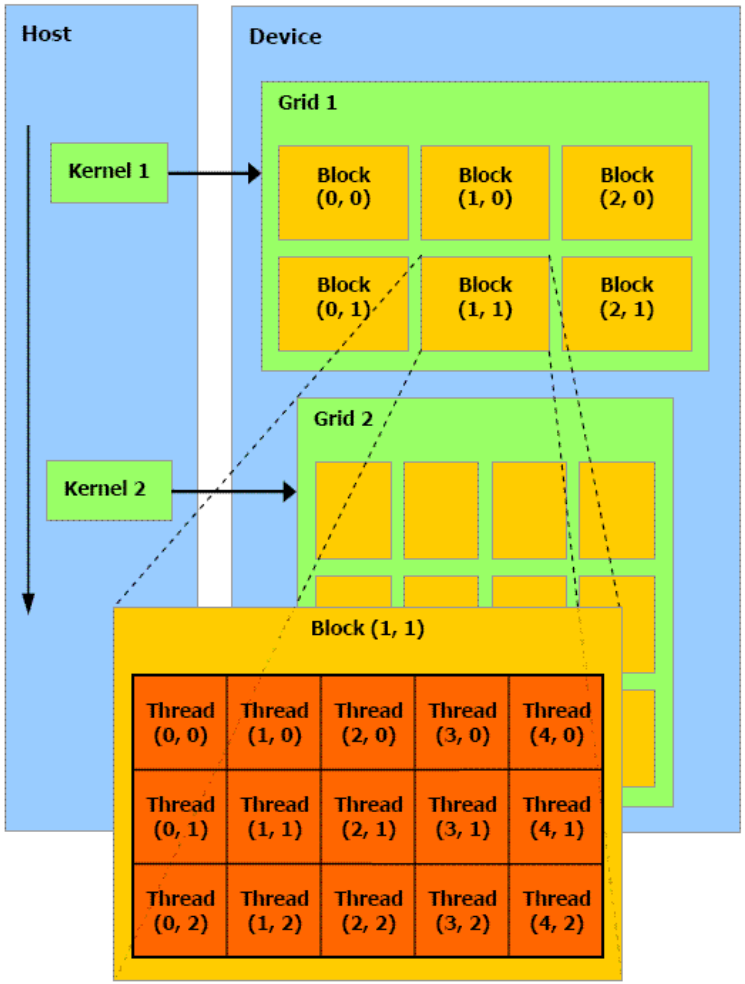
- **Until 2000: fixed architecture** (*not* programmable)
- **2000-01: Pixel and Vertex shaders**
  GeForce 3 and ATI R200.
- **2005: Geometry shaders**
  GeForce 6800 and ATI Radeon X800.
- **2006: ATI CTM™ (for "Close To Metal").**
  ATI Radeon based GPUs.
- **2007: NVIDIA CUDA, ATI Stream SDK**
  NVIDIA GeForce 8, AMD FireStream.
- **2009** (Q3)**: OpenCL drivers & SDK available**
- **2010** (S1)**: Intel Larrabee coming**

GPU pipeline

Shaders allow application to run their own code in the graphic pipeline

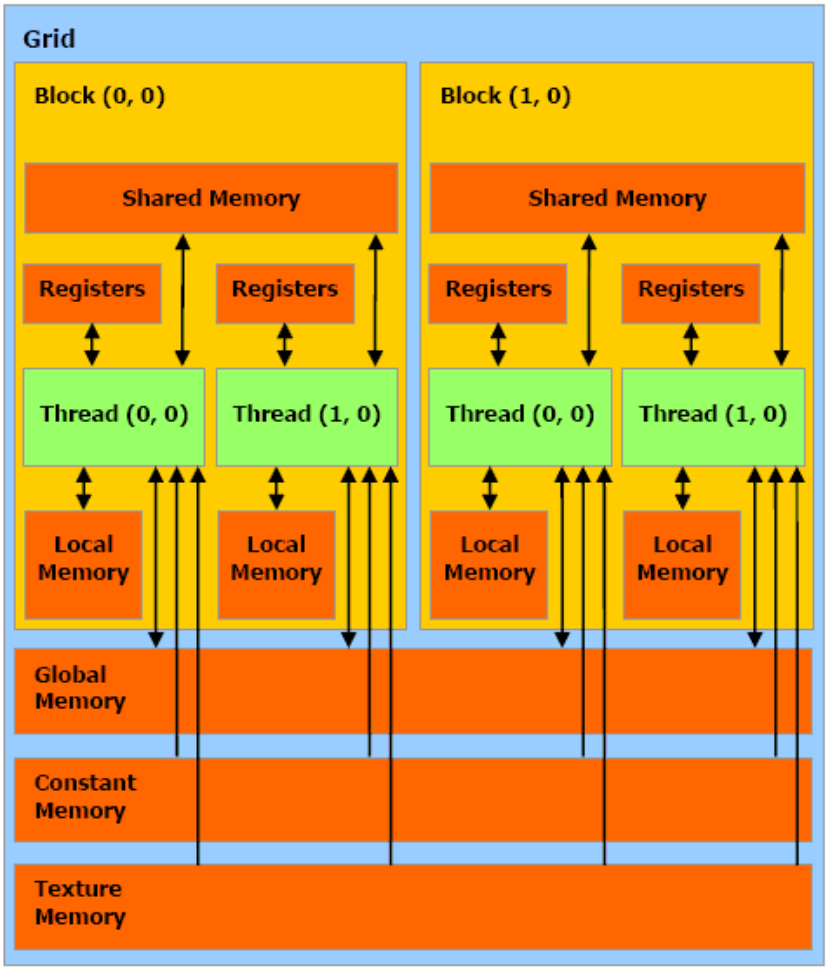


CUDA thread batching



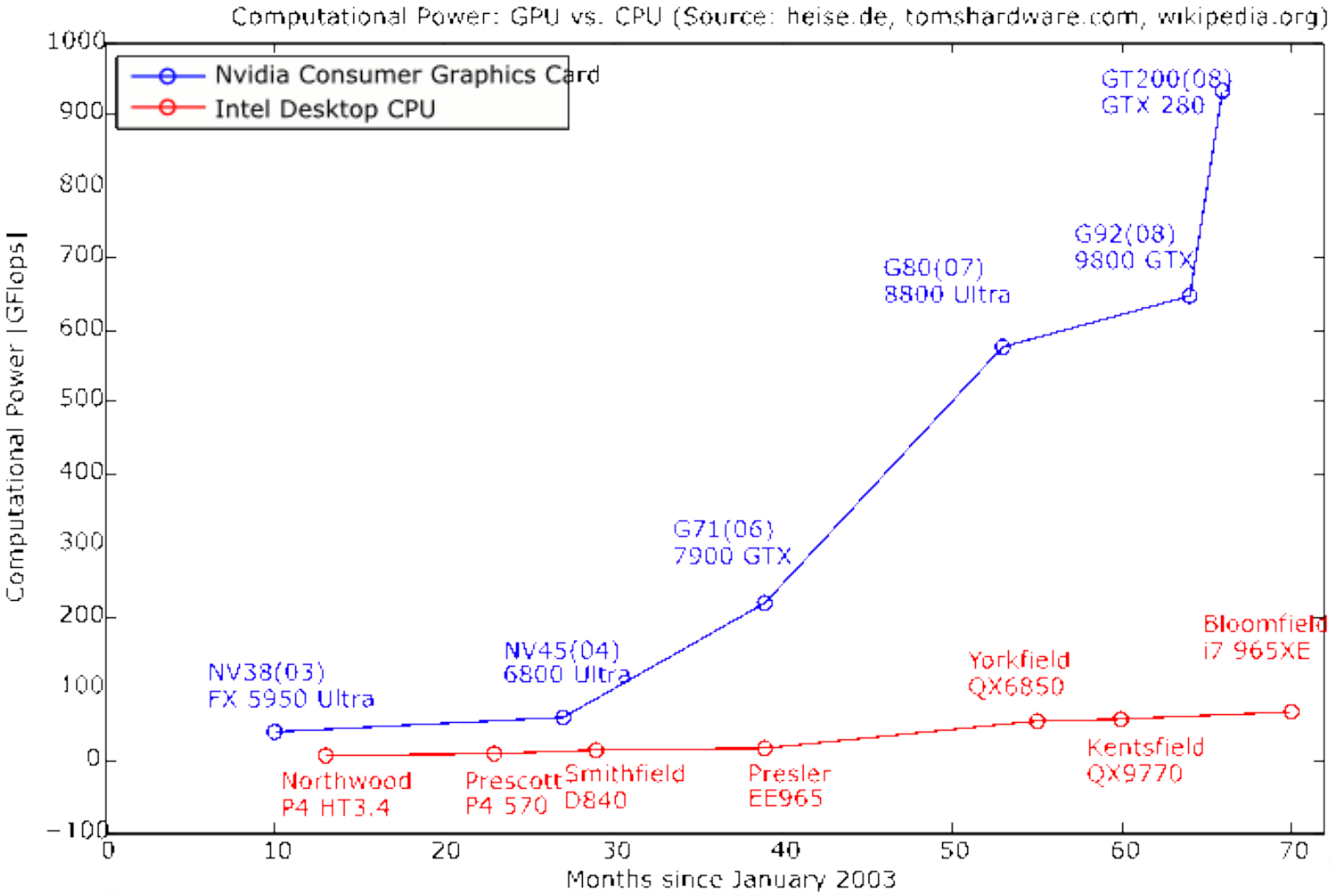
- GPU processing library from NVIDIA.
- Subdivide processing tasks in thousands of threads using blocks.
- C Style programming
- Full memory access

Cuda memory model



- Threads share memory => Synchronization mechanism
- Cache memory really fast (Shared Memory, Registers, Local Memory)
- Texture and constant Memory are fast but *READ ONLY*.
- Global Memory is slower but *WRITABLE*.

GPU history: The power race



Source: GPU4Vision, http://gpu4vision.icg.tugraz.at

TELECOM SudParis

GPU vs CPU: Specifications

Model	Avg Price in € (12/01/2008)	Effective processing power in Gflops	Power in Watts	Millions of transistors	Nbr. of processing units
NVIDIA GeForce 8800 GT	300	336	110	754	112
ATI Radeon HD 2900 XT	300	475	215	700	320
Intel Core 2 Duo E6700	169	17	65	291	2
AMD 64 x2 6000+	168	19	125	227,4	2

[Source : Naga Govindaraju]

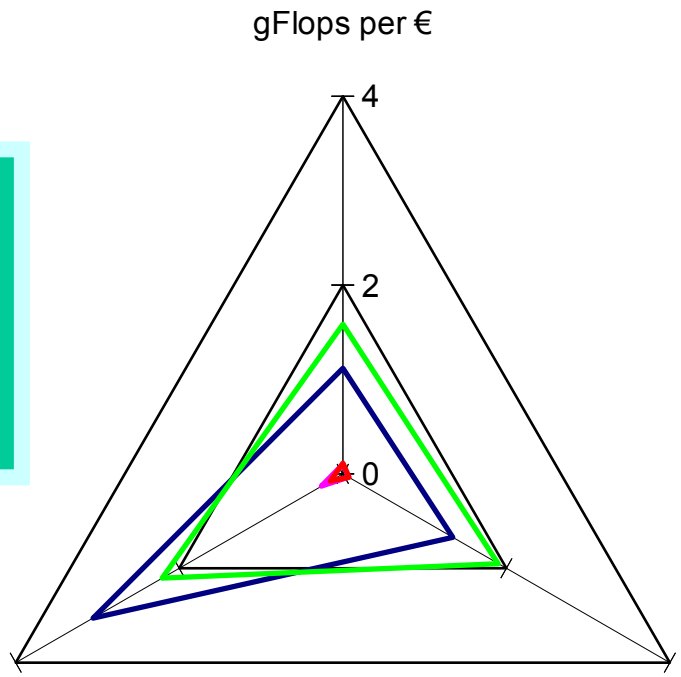
GPU *vs.* CPU

Processing power ratio on price and power consumption

Different purpose

Different architecture

Different efficiency



NVIDIA GeForce 8800 GT — ATI Radeon HD 2900 XT — Intel Core 2 Duo E6700 — AMD 64 x2 6000+

GPU *vs.* CPU(2)

- **Benefits of GPU:**
  - Processing power:
    - increasing faster than CPU,
    - cheaper than CPU,
    - highly parallel,
  - Easily upgradable.
- **Benefits of CPU:**
  - Flexible and general processing unit,
  - Stable programming languages.

GPU programming challenges

- **Algorithms**
  - Highly parallel
  - Coding limitations
- **Dedicated APIs**
  - OpenGL, shading languages,
  - Brook, CUDA, OpenCL…
- **Development tools**
  - Rapidly evolving APIs
  - Heterogeneous and scattered documentation

GPU complexity to be hidden for wide acceptance

GpuCV

Framework description

GpuCV: main features

Transparently manages:

- Hardware capabilities.
- Data synchronization.
- Activation of low level GPU code (GLSL & CUDA).
- On-the-fly benchmarking and switching to the most efficient implementation.

GpuCV: Integration with OpenCV

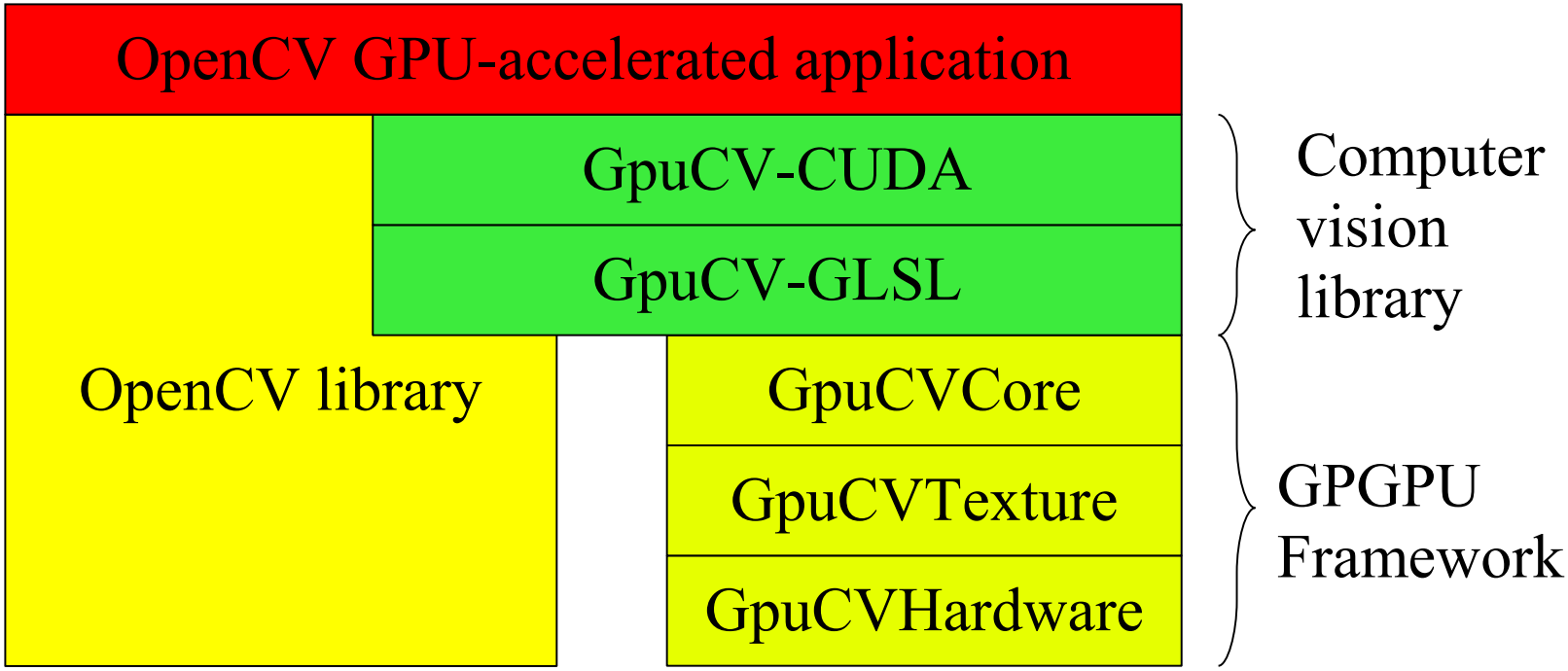
- **Compatible on multiple OS such as MS Windows XP and LINUX.**
- **Designed to be fully compliant with existing OpenCV applications:**
  - OpenCV function:

  void cvAdd(CvArr*src1, CvArr*src2, CvArr*dst)
  - GpuCV function:

  void cvgAdd(CvArr*src1, CvArr*src2, CvArr*dst)
- **Change header and lib files to GpuCV and call init function:**
  - *cvgInit()*

GpuCV: layered framework

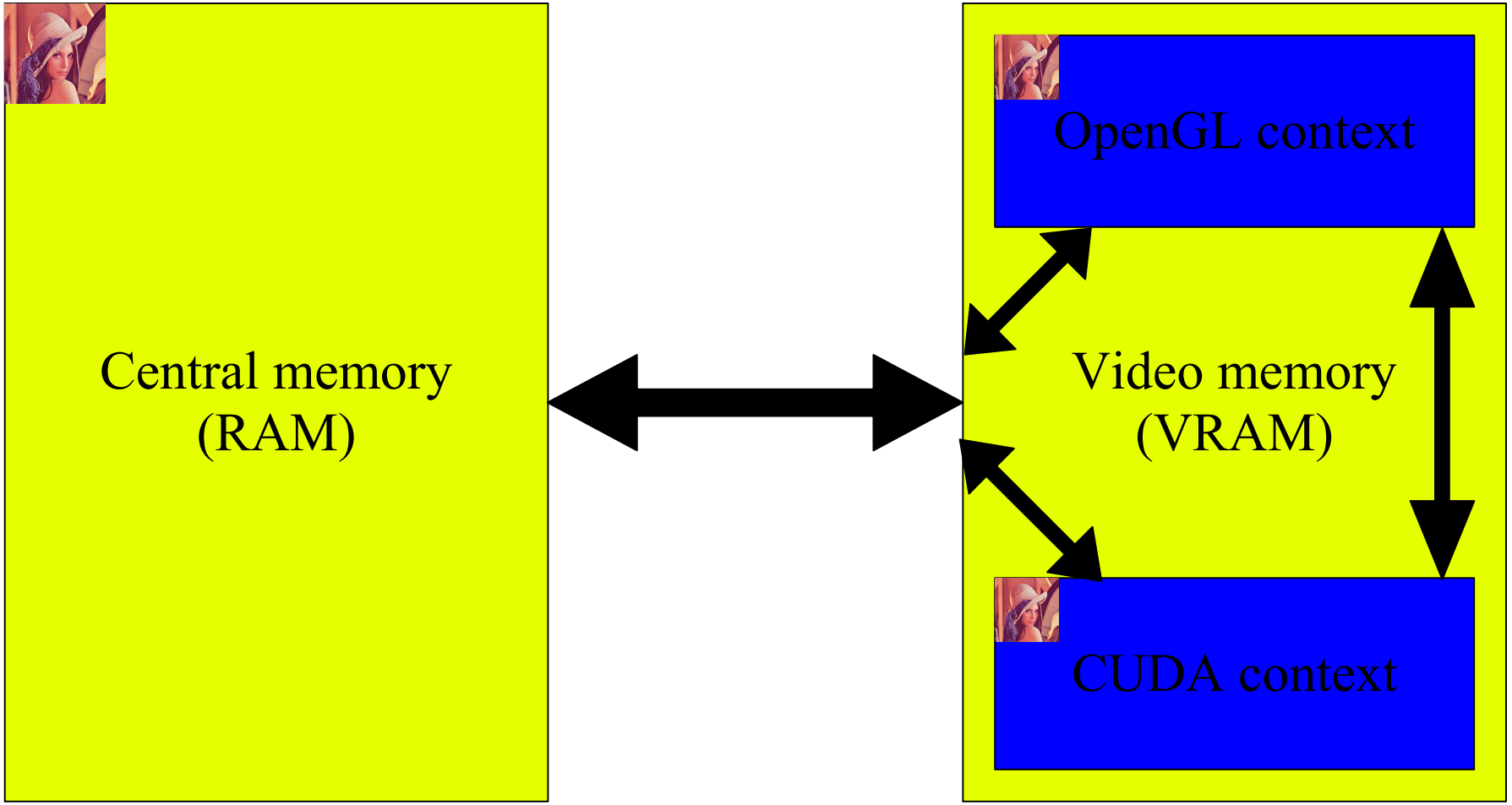
GpuCV = GPGPU framework +
GPU-accelerated Computer Vision library



GpuCV

Data management

GpuCV: Memory locations



GpuCV: Data management

- Processing data with either CPU or GPU requires storing data in central memory and/or in graphics memory.
- Data are automatically transferred to required locations.
- 'Smart transfer' option can estimate all possible transfer time costs and select the fastest one.
- GpuCV operators know about input and output images, so writing to an output image discards all the other existing instances for data consistency sake.

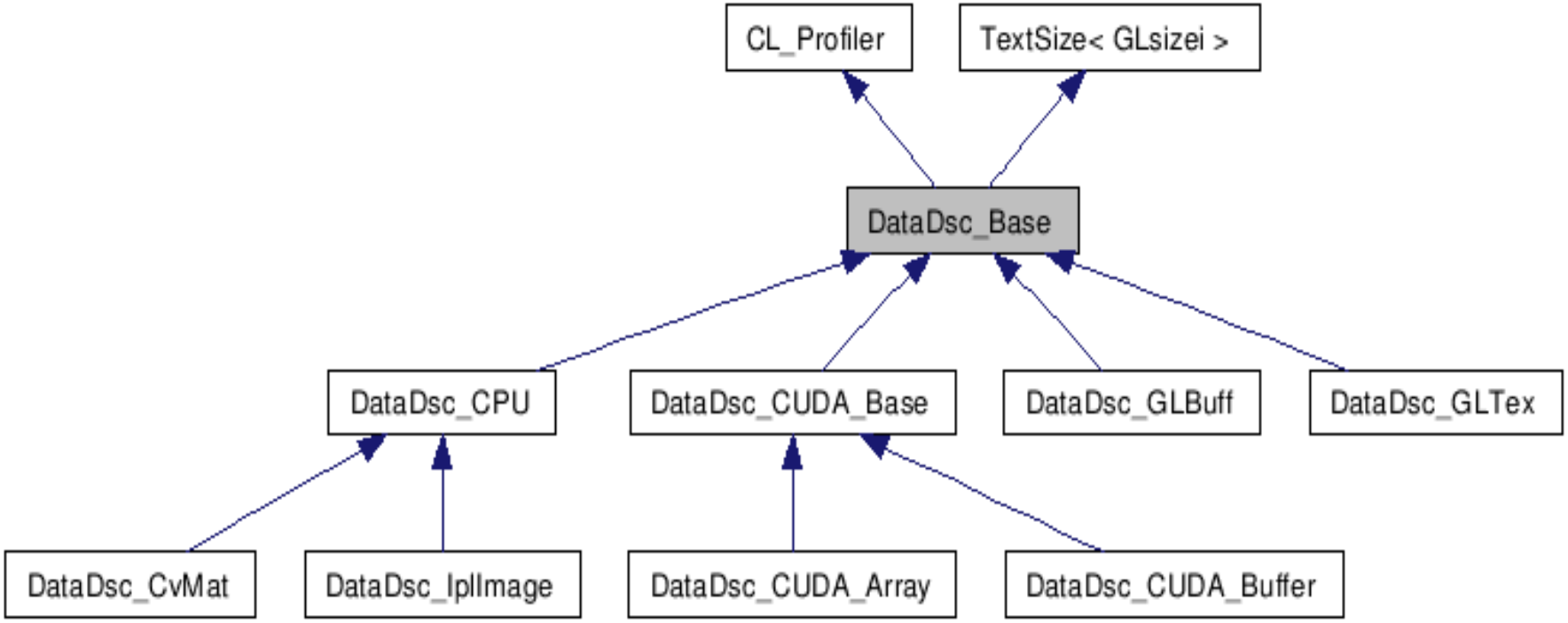
GpuCV: Data descriptors

- **Holds image properties:**
  - Data size / format (number of channels, element type).
  - Pointer to allocated data memory.
  - Flag raised if data present.

- **Holds methods:**
  - To copy/convert properties with other data descriptors.
  - To copy data to other data descriptors.

GpuCV: Data container

Container that describes and stores data



GpuCV supports transparent data synchronization

GpuCV: selecting the memory type

Choosing a data location is easy:

cvgSetLocation<DestinationType>(OpenCV_Image , DataTransferFlag);

With:

- DestinationType: destination data descriptor class.
- OpenCV_Image: pointer to OpenCV image/matrix.
- DataTransferFlag: specify if we transfer data or only allocate memory.

GpuCV

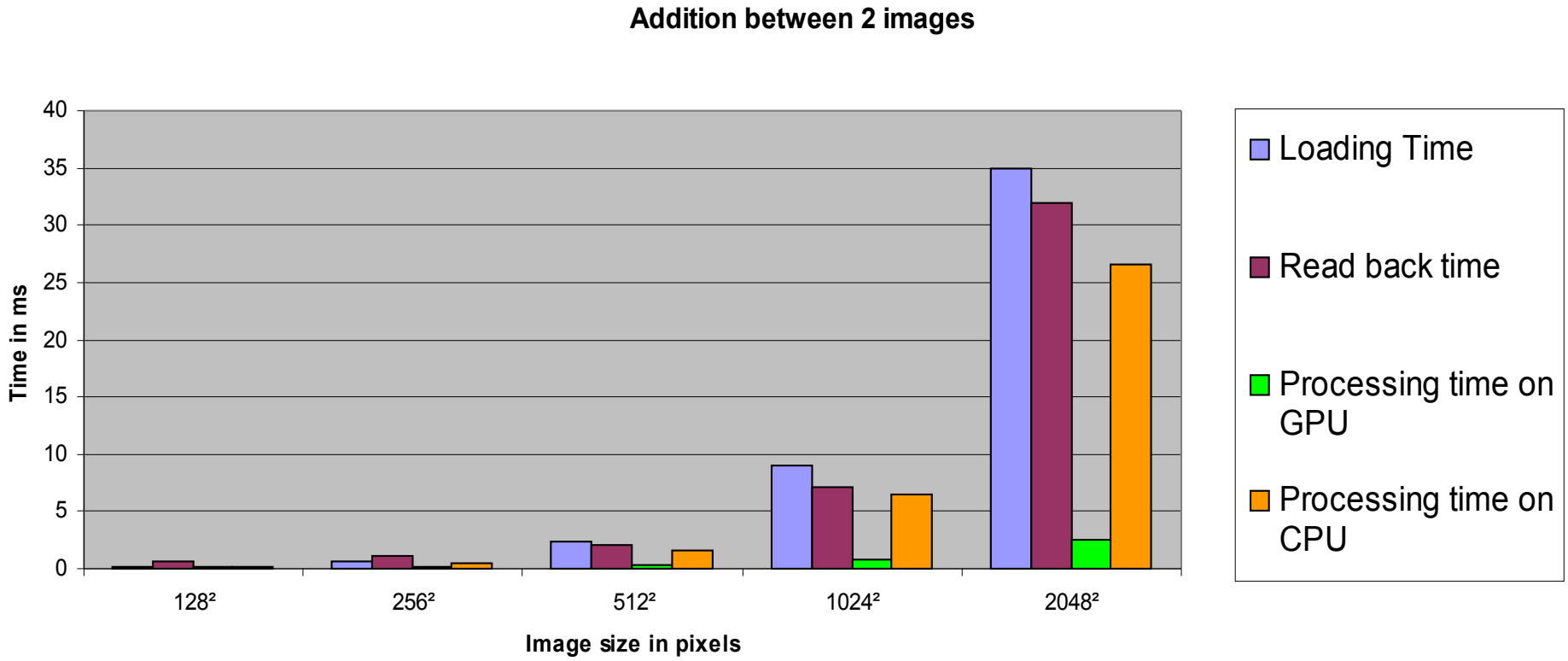
Implementation switching

GpuCV: Performance issues

- **Operator performance depends on:**
  - Implementation used (CPU, GLSL, CUDA).
  - Current data location(s) and eventual transfer.
  - Operator parameters (image size, format, options)
  - Host computer hardware.

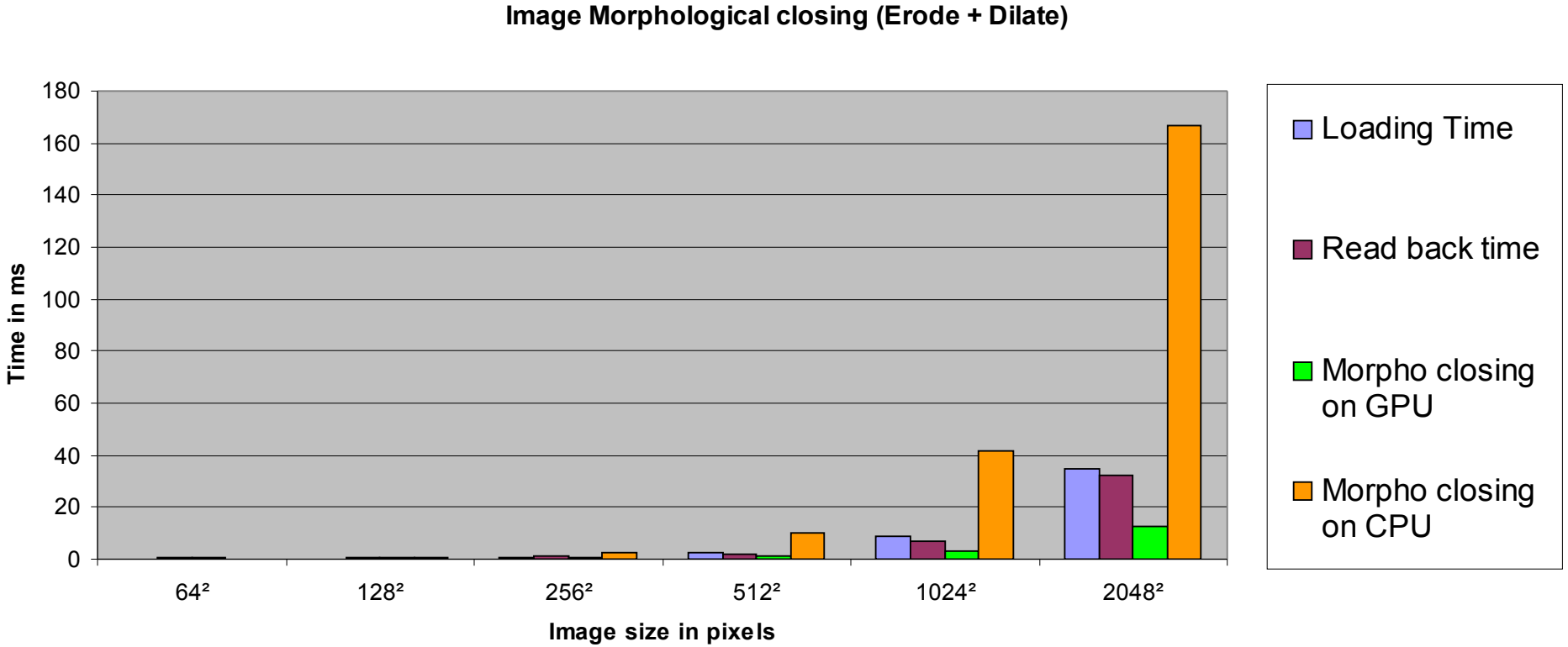
- **Too many parameters to optimize manually an application for many target platforms**

GpuCV: the transfer bottleneck



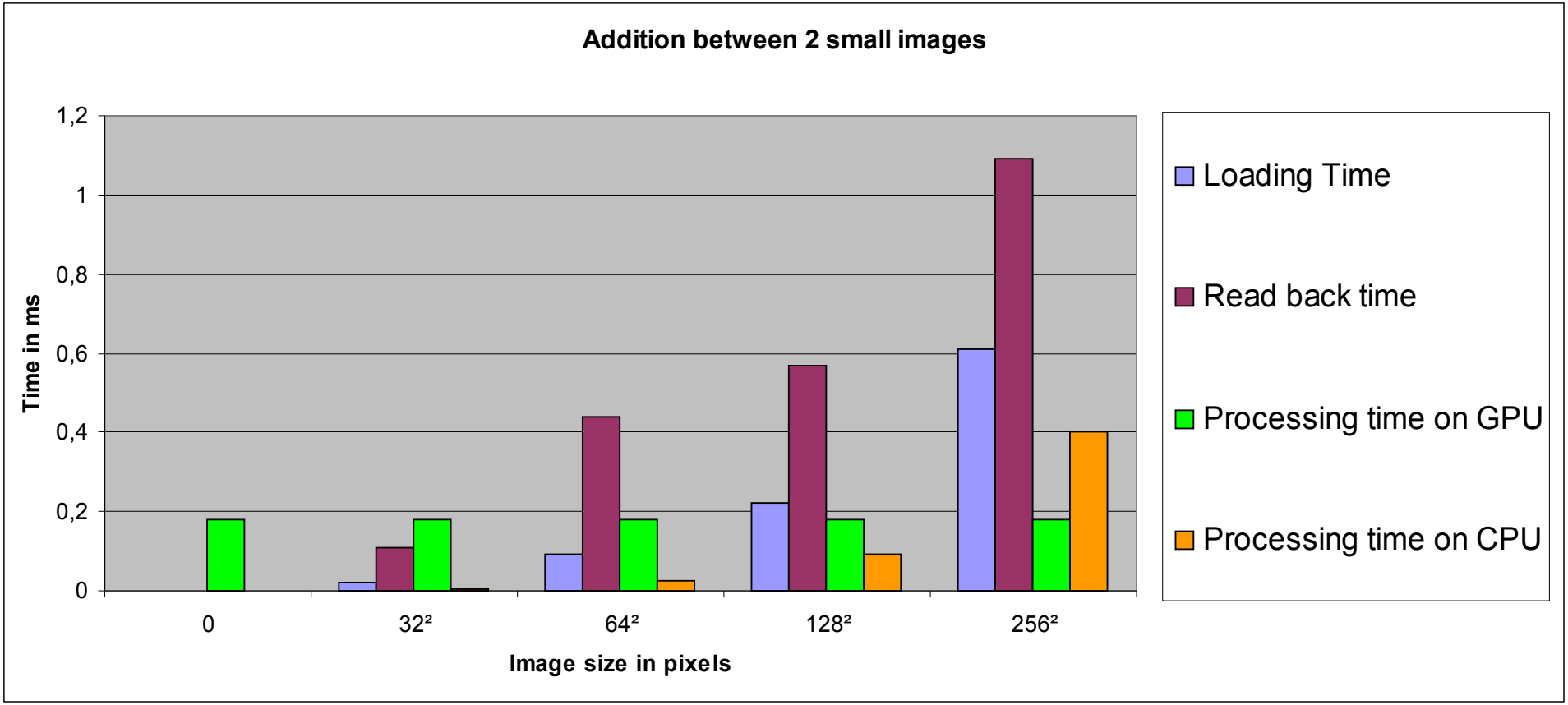
GPU much slower *with* transfer,
much faster *without* transfer!!

GpuCV:
fast for compute intensive operators



GPU can be faster even *with* transfer!

GpuCV: the activation issue



GPU implies a *constant activation delay*

not efficient on small images!

GpuCV:
Dynamic implementation switching

- **GpuCV operators switch between implementations: CPU, GLSL or CUDA.**
  - Dynamic switching based on previous on-the-fly benchmarks.
  - Selects the most efficient implementation, including transfer delay and processing time.
  - Can be turned off *e.g.* for manual benchmarks.

- **Has an additional cost of about 300 µs**

  Usually acceptable for image larger than 256×256.

GpuCV: Internal benchmarking

- *SugoiTracer* for embedded benchmarking
- *Benchmarking results saved in XML:*

```
<function name="cvgSub">
<RecordList>
  <Records id="|type:GPUCV||size:=512*512|" TotalRecords="10">
  <TotalTime sec="0.000000" usec="3170.000000" />
  <MinTime sec="0.000000" usec="278.000000" />
  <MaxTime sec="0.000000" usec="358.000000" />
  </Records>
  <Records id="|type:OPENCV||size:=512*512|" TotalRecords="10">
  <TotalTime sec="0.000000" usec="18220.000000" />
  <MinTime sec="0.000000" usec="1605.000000" />
  <MaxTime sec="0.000000" usec="2046.000000" />
  </Records>

</RecordList>
</function>
```

GpuCV: Auto-switching operators

CXCORE library (Operation on array):

- Initialization: cvCreateImage, cvCreateMat, cvReleaseImage, cvReleaseMat, cvCloneImage, cvCloneMat, cvGetRawData, cvSetData.
- Copying and Filling: cvCopy, cvSetZero
- Transforms and Permutations: cvSplit, cvMerge
- Arithmetic, Logic and Comparison: cvgAdd, cvAddS, cvConvertScale, cvDiv, cvMax, cvMaxS, cvMin, cvMinS, cvMul, cvSub, cvSubRS, cvSubS
- Statistics: cvAvg, cvSum, cvMinMaxLoc.
- Linear Algebra: cvScaleAdd, cvGEMM
- Math Functions: cvPow
- And more…

GpuCV: Auto-switching operators

CV library

- Image Processing:
  - Sampling, Interpolation and Geometrical Transforms: cvResize
  - Morphological Operations: cvDilate, cvErode, cvMorphologyEx, cvSobel, cvLaplace, cvDeriche,...
  - Filters and Color Conversion: cvCvtColor, cvThreshold
  - Histograms: cvQueryHistValue_*D
  - And more...

GpuCV achievements

Benchmarks

Benchmarks

Ex. processing 2048 x 2048 images
with NVIDIA GeForce GTX 280
& Intel Core2 Duo 2.2 GHz (online benchmark)

(time in ms)	OpenCV	GpuCV-CUDA	Acceleration
Deriche	1997	19,35	103,2
Erode 3 x 3	85,1	1,2	70,92
Mul.	73,6	0,99	74,34
Mat. Mul.	11172	200	55,86
DFT	435,4	9,9	43,98

https://picoforge.int-evry.fr/projects/svn/gpucv/bBenchs
/0.4/0.4.1.rev.175/NV8800_Core2Duo-2.2

Conclusion

Summary

- **Benefits of GPUs:**
  - High processing power
  - Lower power/price ratio than CPU
- **Penalties:**
  - Requires additional data transfer
  - Activation delay → not efficient for small images
  - GPU operators implementations depend on hardware compatibilities.
- **GpuCV**

  A ready to use GPU-accelerated CV library.

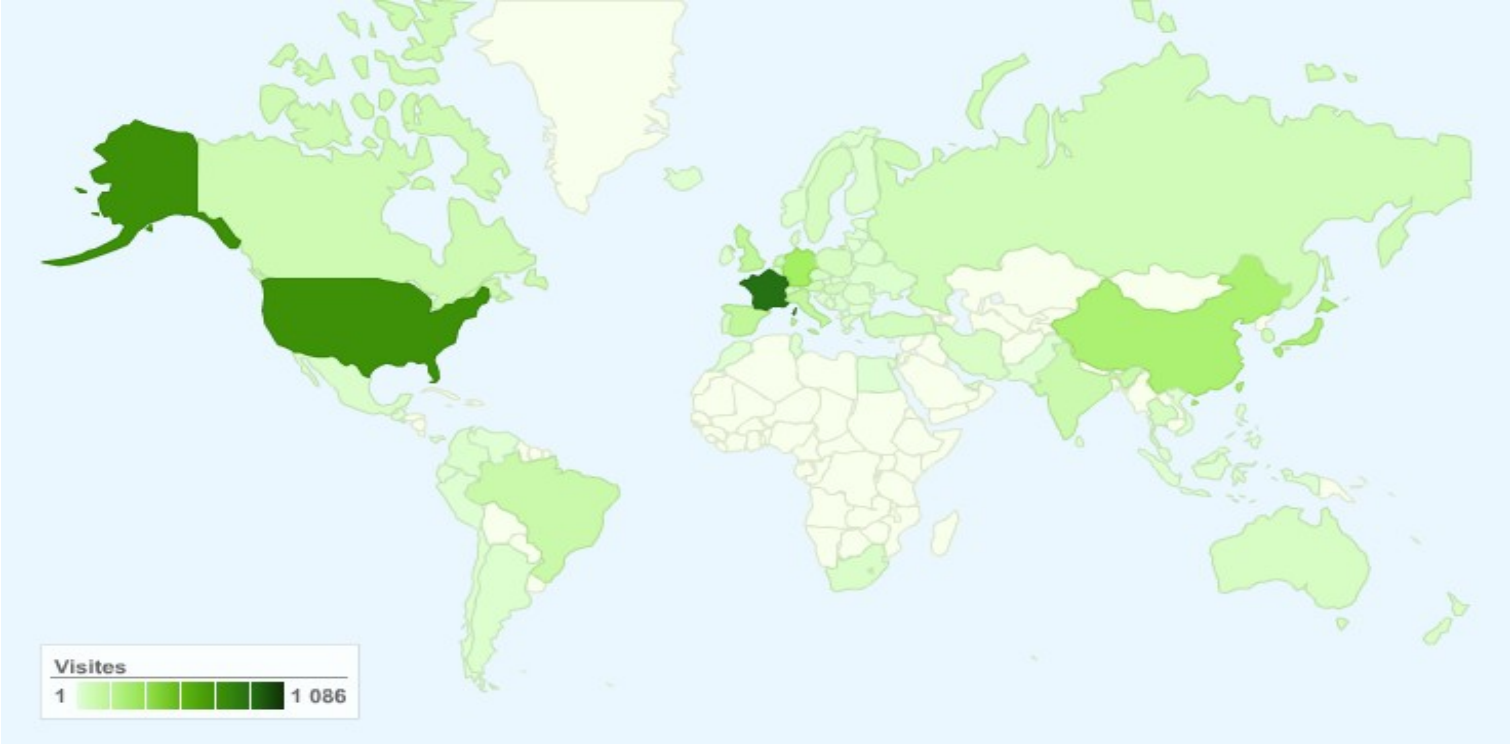
The GpuCV framework

- Meant for GPU acceleration
  - image processing and Computer Vision operators
- compatible with the popular OpenCV library
  - replacement to OpenCV routines
- hides the GPU programming complexity
  - data synchronization
  - codelets (kernels) management (GLSL,CUDA)
- adaptive to hardware platform
  - integrated benchmarking and implementation switching
- multi-platform library
  - MS Windows & Linux
- open source
  - CeCill-B license

GpuCV available as open source

Home: http://picoforge.int-evry.fr/projects/gpucv

Visitors from around the world:



Top 10 countries by connections number since January 2008:

- France: 1 086
- United States: 886
- Germany: 360
- China: 316
- Japan: 297
- Spain: 180
- United Kingdom: 179
- Italy: 167
- Brazil: 126
- India: 124

Thank you!

http://picoforge.int-evry.fr/projects/gpucv

http://www-public.it-sudparis.eu/~horain/OffreCDD.html

Any question?