Centro de Investigación en Matemáticas, A.C.

OpenCV & CUDA Exercises

By:
Ángel R. Aranda Campos
Francisco J. Hernández López.
Jorge F. Madrigal Díaz
{arac, fcoj23, pacomd}@cimat.mx

Guanajuato, Gto. October 2011

Yesterday

OpenCV

- Basic structures (Mat)
- Image processing
- Data access

CUDA introduction

- Device Management
  - cudaGetDeviceCount
  - cudaSetDevice
  - cudaGetDevice
  - cudaGetDeviceProperties
  - cudaChooseDevice
- Thread Management
  - cudaThreadSynchronize
- Qualifiers for a function
- Kernel functions calls
- Automatically Defined Variables

Device Management

- **cudaGetDeviceCount** - Returns the number of compute-capable devices
- **cudaSetDevice** - Sets device to be used for GPU executions
- **cudaGetDevice** - Returns which device is currently being used
- **cudaGetDeviceProperties** - Returns information on the compute-device
- **cudaChooseDevice** - Select compute-device which best matches criteria

Device Management - cudaGetDeviceProperties

- **Name** - is an ASCII string identifying the device;
- **totalGlobalMem** – total amount of global memory available on the device in bytes;
- **sharedMemPerBlock** – maximum amount of shared memory available to a thread block in bytes;
- **regsPerBlock** – maximum number of 32-bit registers available to a thread block;
- **warpSize** - is the warp size in threads;

Device Management - cudaGetDeviceProperties

- **maxThreadsPerBlock** - maximum number of threads per block;
- **maxThreadsDim[3]** - maximum sizes of each dimension of a block;
- **maxGridSize[3]** - maximum sizes of each dimension of a grid;
- **totalConstMem** - total amount of constant memory available on the device in bytes;
- **major, minor** - major and minor revision numbers defining the device's compute capability;
- **multiProcessorCount** - is the number of multiprocessors on the the device.

Device Management - cudaGetDeviceProperties

```
There is 1 device supporting CUDA

Device 0: "GeForce 9400M"
  CUDA Driver Version:                           4.0
  CUDA Runtime Version:                          4.0
  CUDA Capability Major/Minor version number:    1.1
  Total amount of global memory:                 265945088 bytes
  Multiprocessors x Cores/MP = Cores:            2 (MP) x 8 (Cores/MP) = 16 (Cores)
  Total amount of constant memory:               65536 bytes
  Total amount of shared memory per block:       16384 bytes
  Total number of registers available per block: 8192
  Warp size:                                     32
  Maximum number of threads per block:           512
  Maximum sizes of each dimension of a block:    512 x 512 x 64
  Maximum sizes of each dimension of a grid:     65535 x 65535 x 1
  Maximum memory pitch:                          2147483647 bytes
  Texture alignment:                             256 bytes
  Clock rate:                                    1.10 GHz
  Concurrent copy and execution:                 No
  Run time limit on kernels:                     Yes
  Integrated:                                    Yes
  Support host page-locked memory mapping:       Yes
  Compute mode:                                  Default (multiple host threads can use
this device simultaneously)
```

ThreadManagement

- **cudaThreadSynchronize** - Blocks until the device has completed all preceding requested tasks. cudaThreadSynchronize() returns an error if one of the preceding tasks failed.

Qualifiers for a function

- **__device__**
  - Runs on the device.
  - Called only from the device.

- **__global__**
  - Runs on the device
  - Called only from the host.

Kernel functions calls

- Example function

  __global__ void NameFunc(float *parameter, …);

  it must be called as follows:

  NameFunc <<< Dg, Db, Ns, St >>> (parameter1,…);

- **Dg**: Type *dim3*, dimension and size of the grid.
- **Db**: Type *dim3*, dimension and size of each block.
- **Ns**: Type *size_t*, number of bytes inshared memory.
- **St**: Type *cudaStream_t* that indicates which stream will use the kernel.

  (Ns and St are optional).

Automatically Defined Variables

- All __global__ and __device__ functions have access to the following variables:
  - **gridDim** (dim3), indicates the dimension of the grid.
  - **blockIdx** (uint3), indicates the index of the bloque within the grid.
  - **blockDim** (dim3), indicates the dimension of the block.
  - **threadIdx** (uint3), indicates the index of the thread within the block.

Exercises Outline

- CUDA
  - Example1: Add one (kernel parameters)
  - Exercise1: Add Vectors
  - Exercise2: Add Matrix
- OpenCV&CUDA
  - Example3: Memory management (OpenCV→CUDA→OpenCV)
  - Example4: Modify image
  - Exercise2: Compose images (Gray or RGB) ($\alpha I_1+(1- \alpha)I_2$)
  - Exercise3: Gradient Magnitude.
  - Example4: Mean filter.
  - Exercise4: Gaussian and Laplacian filters.
  - Exercise5: Diffusion image.

Example1: Add one

- Create a host vector ("vector_h").
- Initialize "vector_h".
- Create a device vector ("vector_d").
- Copy memory from "vector_h" to "vector_d".
- Add 1 to "vector_d"
- Copy memory from "vector_d" to "vector_h".
- Finally, show the result: "vector_h".

Exercise1: Add vectors

- Add vectors (c = a + b).
  - Create host memory: “a_h”, “b_h” and “c_h”
  - Initialize the vectors “a_h” and “b_h”.
  - Create device memory: “a_d”, “b_d” and “c_d”.
  - Copy memory from host to device of vectors a and b.
  - Add vectors a_d and b_d; the result is saved in vector c_d.
  - Copy memory from device to host of vector c.
  - Finally, show the result.

Exercise2: Add Matrix

- Create host memory: “a_h”, “b_h” and “c_h”.
- Initialize “a_h” and “b_h”.
- Crete device memory: “a_d”, “b_d” y “c_d”.
- Copy memory from host to device.
- Add matrix in the device.
- Copy memory from device to host.
- Finally, show the result.

Example3: Memory management

Example4: Modify image

Exercise 3: Image Composition

- Load two images and reserve memory to the output image.
- Create memory on Device (for the 3 images).
- Copy memory of the Host to Device.
- Loop:
  - Kernel (CUDA_Compose_Images)
  - Return the result on the Host
  - Show the result
- Free the memory

Exercise3: Gradient Magnitude

- Load the original image in host memory.
- Create device memory: Imag_dev, ImagDx_dev, ImagDy_dev, ImagMG_dev.
- Copy the original image from host to device memory.
- Calculate Dx, Dy and GM in the device.
- Copy the result from device to host memory.
- Show the result.

$$D_x(x,y) = I(x,y) - I(x-1,y)$$

$$D_y(x,y) = I(x,y) - I(x,y-1)$$

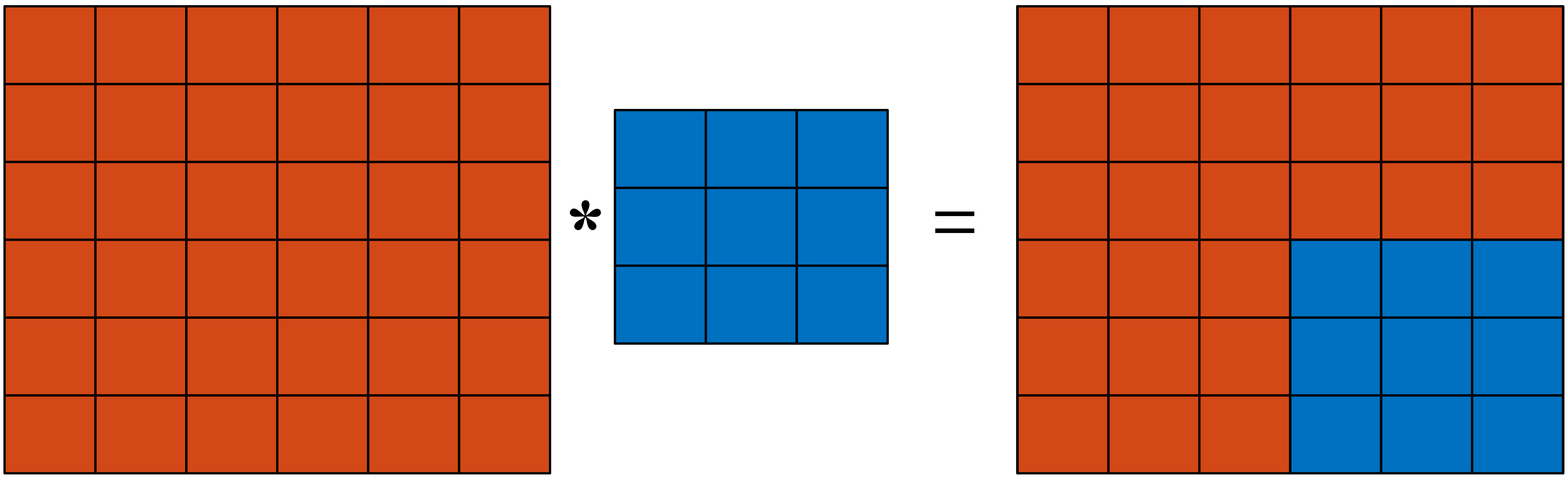
$$GM(x,y) = \sqrt{D_x^2(x,y) + D_y^2(x,y)}$$

Example4: Mean filter

- Load the original image in host memory.
- Create device memory.
- Copy the original image from host to device memory.
- Calculate the mean filter.
- Copy the result from device to host memory.
- Show the result.

Example4: Mean filter

- With window size of 3x3:



Image

Convolution Kernel

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

Exercise4: Gaussian and Laplacian filters

- Load the original image in host memory.
- Create device memory.
- Copy the original image from host to device memory.
- Calculate the Gaussian or Laplacian filter.
- Copy the result from device to host memory.
- Show the result.

Gaussian Filter:

$$\begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$$

Laplacian Filter:

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

Exercise5: Diffusion image

- Given an image g(x) with noise.
- Smooth the image g(x) with the following functional:

$$U(f(x)) = \frac{1}{2}\sum_{x}(f(x)-g(x))^2 + \frac{\lambda}{2}\sum_{<x,y>}(f(x)-f(y))^2$$

- Differentiating and equating to zero, we obtain:

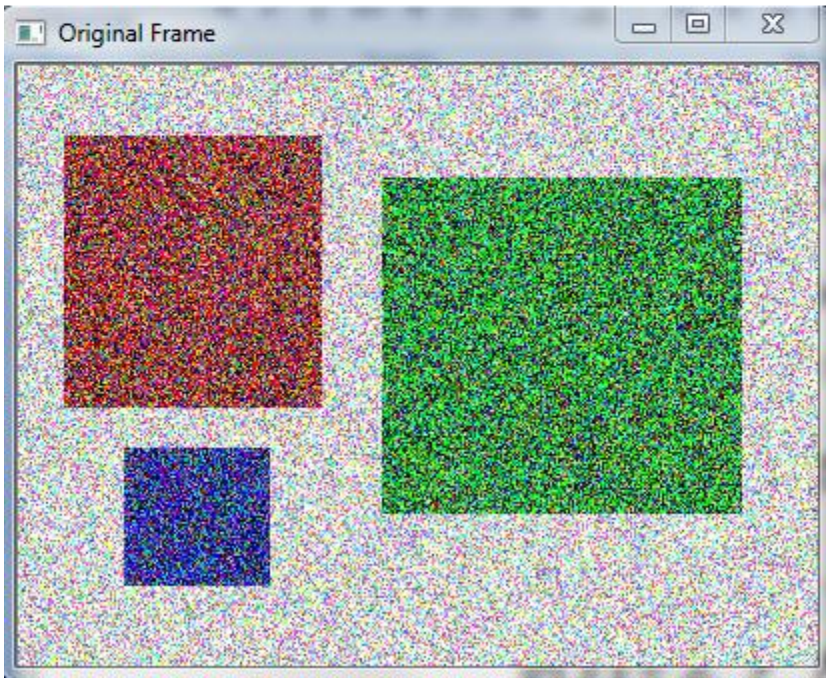
$$f^{k+1}(x) = \frac{g(x)+\lambda\sum_{<x,y>} f^{k}(y)}{1+\lambda|N_x|}$$

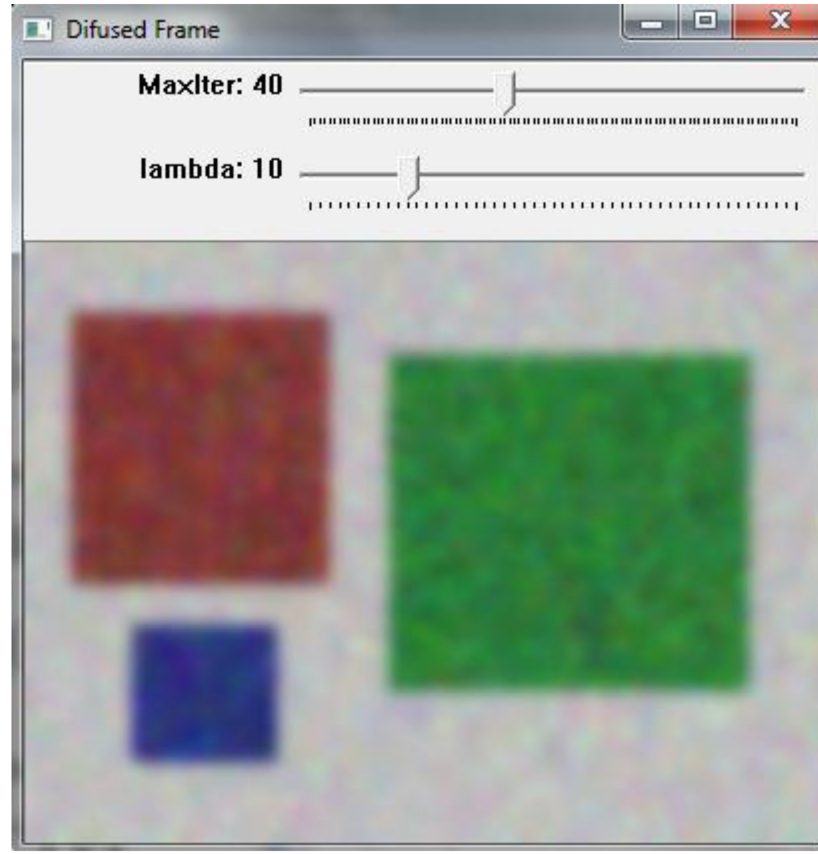
$|N_x|$ # neighborhoods of pixel x

$$f^{0}(x) = g(x)$$

- We can solve by:
  - Jacobi
  - Gauss-Seidel

Exercise5: Diffusion image





Questions

?

Thank you