Computer Vision on GPU with OpenCV

Anatoly Baksheev, Itseez (anatoly.baksheev@itseez.com)

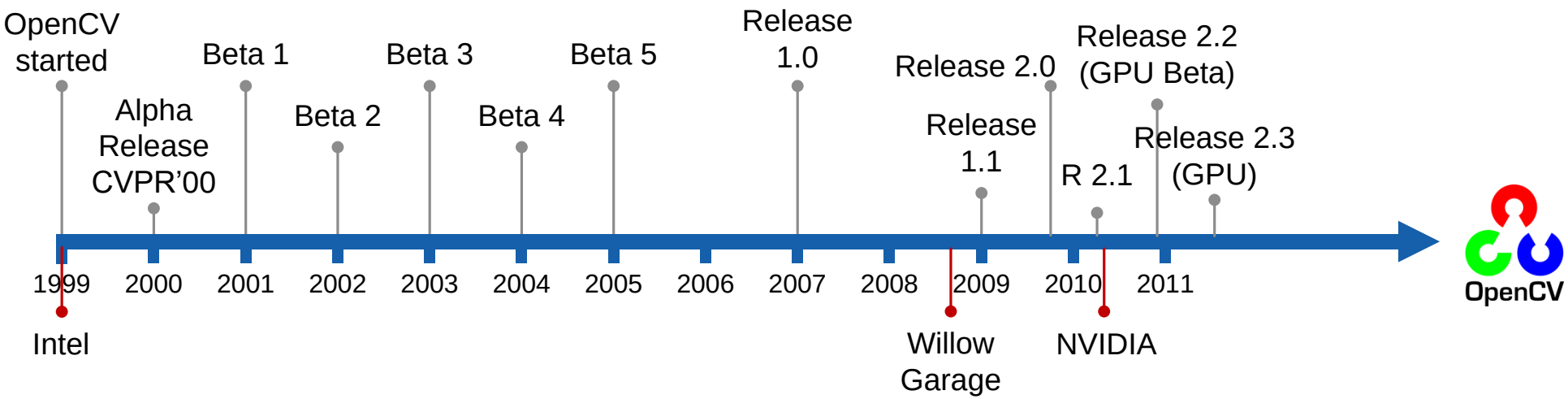
Outline

- Introduction into OpenCV
- Overview of OpenCV GPU module
- Pedestrian detection on GPU

itseez

Lure into

OpenCV History

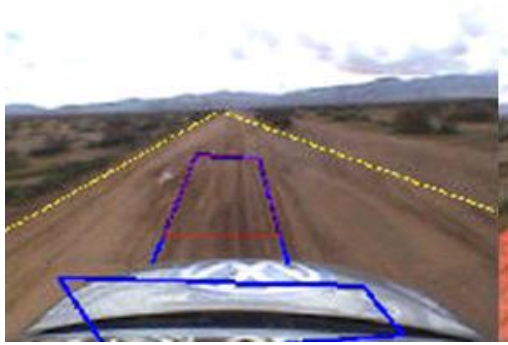


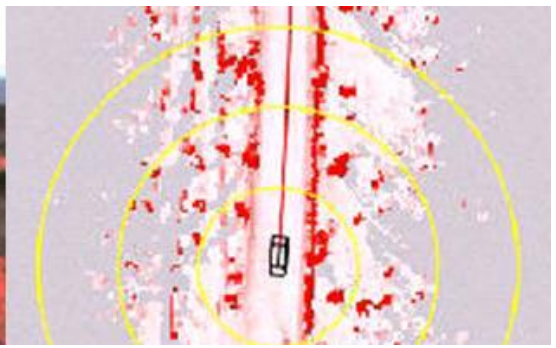
- Original goal:
  - Accelerate the field by lowering the bar to computer vision
  - Find compelling uses for the increasing MIPS out in the market
- Staffing:
  - Climbed in 1999 to average 7 first couple of years
  - Little development from 2002 – 2008
  - Willow entered in 2008 to accelerate development, NVIDIA joined in 2010
  - 8 full time professional developers, 3 of them dedicated to GPU

Projects Using OpenCV

- Academic and Research
  - Large community
- Industry
  - Google street view
  - Structure from motion in movies
  - Robotics
- Over 3 000 000 downloads





OpenCV Functionality Overview

Image processing

General Image Processing

Segmentation

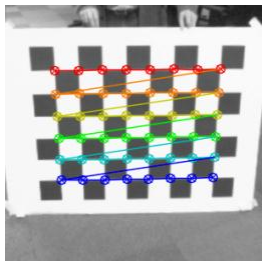
Machine Learning, Detection

Image Pyramids

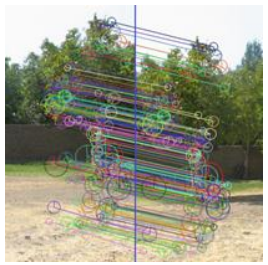
Transforms

Fitting

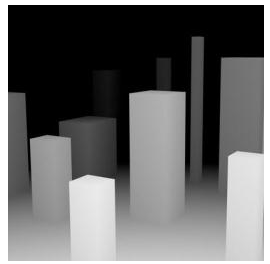
Video, Stereo, and 3D



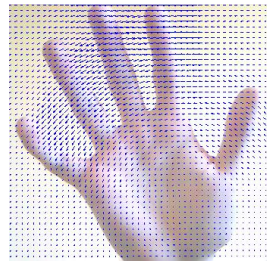
Camera Calibration



Features



Depth Maps



Optical Flow

Inpainting

Tracking

OpenCV License

Based on BSD license

- Free for commercial and research use
- Does not force your code to be open
- You need not contribute back

itseez

Outline

- Introduction into OpenCV
- OpenCV GPU module
- Pedestrian detection on GPU

itseez

OpenCV GPU Module

Goals:

- Provide developers with a convenient computer vision framework on the GPU
- Maintain conceptual consistency with the current CPU functionality
- Achieve the best performance with GPUs
  - Efficient kernels tuned for modern architectures
  - Optimized dataflows (asynchronous execution, copy overlaps, zero-copy)

itseez

OpenCV GPU Module Example

```
Mat frame;
VideoCapture capture(camera);
cv::HOGDescriptor hog;

hog.setSVMDetector(cv::HOGDescriptor::
        getDefaultPeopleDetectorector());

capture >> frame;



vector<Rect> found;
hog.detectMultiScale(frame, found,
    1.4,  Size(8, 8), Size(0, 0), 1.05, 8);
```

```
Mat frame;
VideoCapture capture(camera);
cv::gpu::HOGDescriptor hog;

hog.setSVMDetector(cv::HOGDescriptor::
        getDefaultPeopleDetectorector());

capture >> frame;

GpuMat gpu_frame;
gpu_frame.upload(frame);

vector<Rect> found;
hog.detectMultiScale(gpu_frame, found,
    1.4, Size(8, 8), Size(0, 0), 1.05, 8);
```

- Designed very similar!

itseez

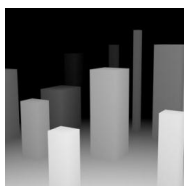
OpenCV GPU Module Contents

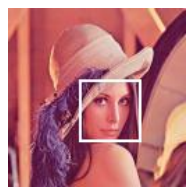
- Image processing building blocks:

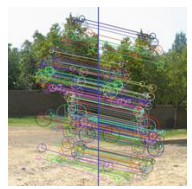
Color conversions | Geometrical transforms | Per-element operations | Integrals, reductions

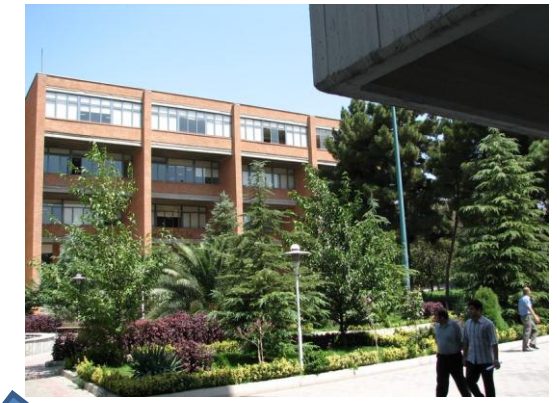
Template matching | Filtering engine | Feature detectors

- High-level algorithms:

Stereo matching 

Face detection 

SURF 



GPU Demo Pack:

http://sourceforge.net/projects/opencvlibrary/

Using in your application

- We recommend CMake for Quick Start.
  - Allows to generate project for your favorite IDE

CmakeLists.txt:

```
cmake_minimum_required(VERSION 2.8)

set(the_target YourProjName}
set(src "main.cpp" "other.cpp")

project( ${the_target} )
find_package( OpenCV REQUIRED )

include_directories( ${OpenCV_INCLUDE_DIR} )
add_executable( ${the_target} ${src} )
target_link_libraries( ${the_target} ${OpenCV_LIBS} )
```

itseez

OpenCV GPU Data Structures

- Class GpuMat
  - For storing 2D image in GPU memory, just like class cv::Mat
  - Reference counting
  - Can point to data allocated by user

```
cv::GpuMat edges;
cv::Mat edges_host;

cv::GpuMat image(Size(1024, 768), CV_8UC1); // allocates GPU memory

cv::Mat image_host = cv::imread("file.png"); // loads image from file
image.upload(image_host);


cv::gpu::Canny(image, edges, 1, 0, 3); // run GPU

edges.download(edges_host);
cv::imshow("derivate", edges_host); // displays result

// cv::gpu::imshow("edges OpenGL", edges); // coming soon
```

GpuMat: Types explained

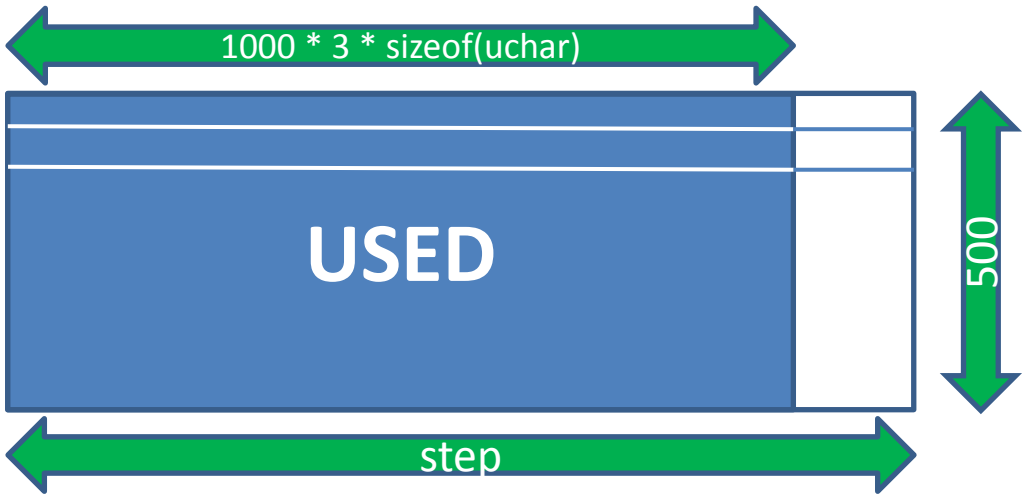
```
cv::GpuMat image(Size(1024, 768), CV_8UC3);
```

- CV_<depth>C<number of channels>
  - depth: 8U, 8S, 16U, 16S, 32S, 32F, 64F
  - channels: 1, 2, 3, 4

- Ex:
  - CV_8UC1, CV_8UC3, CV_8UC4
  - CV_32F, CV32FC3

GpuMat: step explained

```
cv::GpuMat image(Size(1000, 500), CV_8UC3);
```



Bad idea: ~~Size(1, n)~~

(x,y):

- `image.data + image.step * y + 3 * sizeof(unsigned char) * x`
- `image.ptr<unsigned char>(y) + image.channels() * x;`

Memory Allocation Policy

- GPU memory allocations are very expensive

OpenCV GPU functions allocate memory inside if necessary

```
cv::GpuMat dx;
cv::gpu::Sobel(image, dx, 1, 0, 3);
```

Pre-allocation on initialization stage

```
cv::GpuMat dx(size, CV_8U);

//main functionality
...
cv::gpu::Sobel(image, dx, 1, 0, 3);
...
```

Smart code organization

```
cv::GpuMat dx;
for(;;) // video loop
{
    GpuMat image = getNextVideoFrame();
    cv::gpu::Sobel(image, dx, 1, 0, 3);
}
```

itseez

Using OpenCV GPU with your code

- Constructing for memory allocated by you
  - GpuMat(rows, cols, type, pointer, step)
- Passing to any other GPU libraries (CUFFT, CUBLAS, MAGMA)
  - GpuMat::cols, rows, data, step
- Passing to your CUDA code
  - Convertible to PtrStep<T>/PtrStepSz<T>

```
__global__ void func(cv::gpu::PtrStepSz<uchar3> mat)
{
    int x = blockIdx.x * blockDim.x + threadIdx.x;
    int y = blockIdx.y * blockDim.y + threadIdx.y;

    if (x < mat.cols && y < mat.rows)
    {
        int val = min(255, x + y);
        matr.ptr(y)[x] = make_char3(val, val, val);
    }
}
```

itseez

OpenCV GPU Module Performance

Tesla C2050 (Fermi) vs. Core i5-760 2.8GHz (4 cores, TBB, SSE)

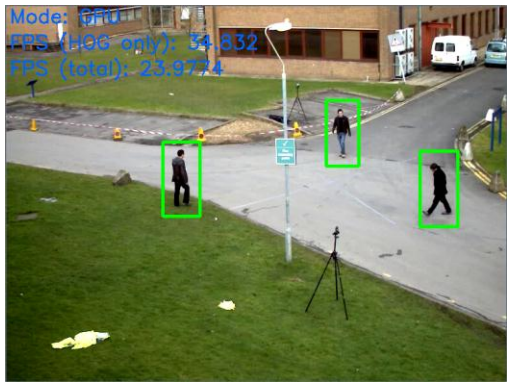
- Average speedup for primitives: 33×
  - For “good” data (large images are better)
  - Without copying to GPU

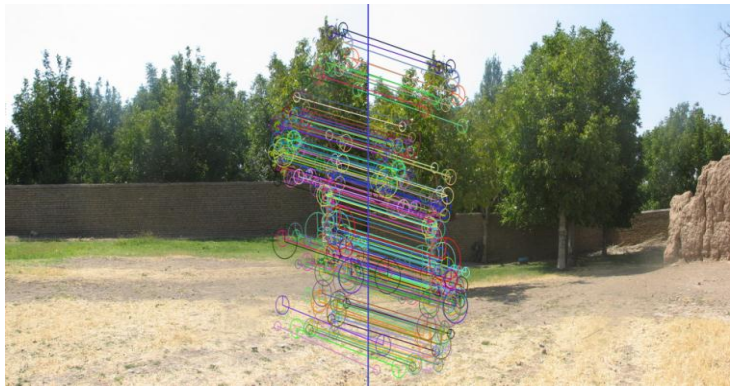
What can you get from your computer?

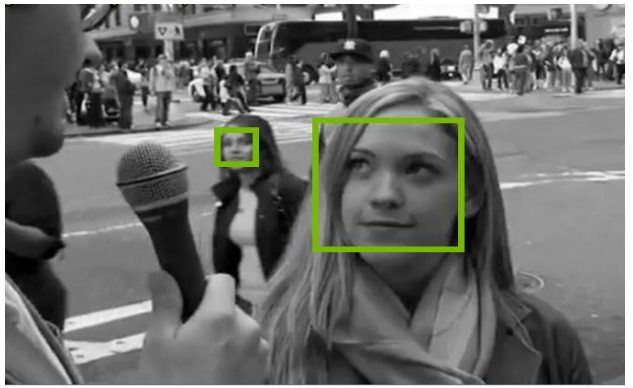
- opencv\samples\gpu\performance

itseez

OpenCV GPU: Performance of High level algorithms

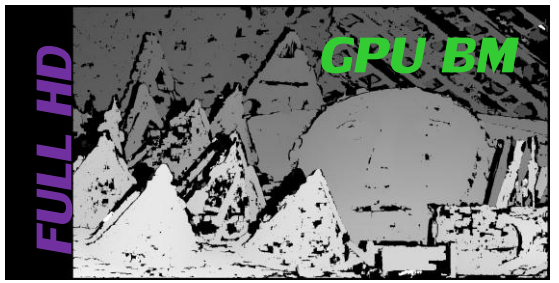


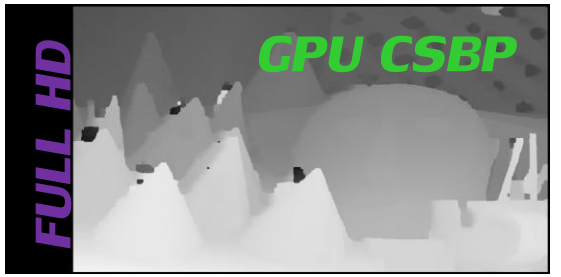




- HOG (**8×**)
- SURF (**12×**)
- Viola-Jones (**6×**)

- Stereo matching
  - BM (**12×**)
  - BP (**20×**)





Documentation and tutorials

- http://opencv.itseez.com/
- http://opencv.itseez.com/doc/tutorials/tutorials.html

gpu. GPU-accelerated Computer Vision

- GPU Module Introduction
- Initalization and Information
- Data Structures
- Operations on Matrices
- Per-element Operations
- Image Processing
- Matrix Reductions
- Object Detection
- Feature Detection and Description
- Image Filtering
- Camera Calibration and 3D Reconstruction

itseez

Outline

- Introduction into OpenCV
- OpenCV GPU module
- Pedestrian detection on GPU

itseez

Pedestrian Detection

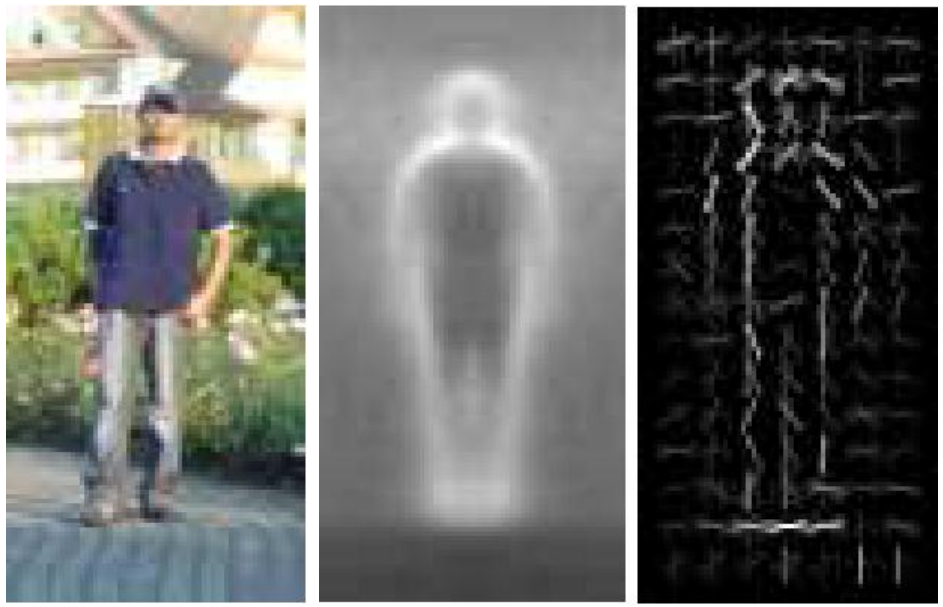
- HOG descriptor
  - Introduced by Navneet Dalal and Bill Triggs (2005)
  - Feature vectors are compatible with the INRIA Object Detection and Localization Toolkit http://pascal.inrialpes.fr/soft/olt/

itseez

Pedestrian Detection: HOG Descriptor

- Object shape is characterized by distributions of:
  - Gradient magnitude
  - Edge/Gradient orientation

- Grid of orientation histograms

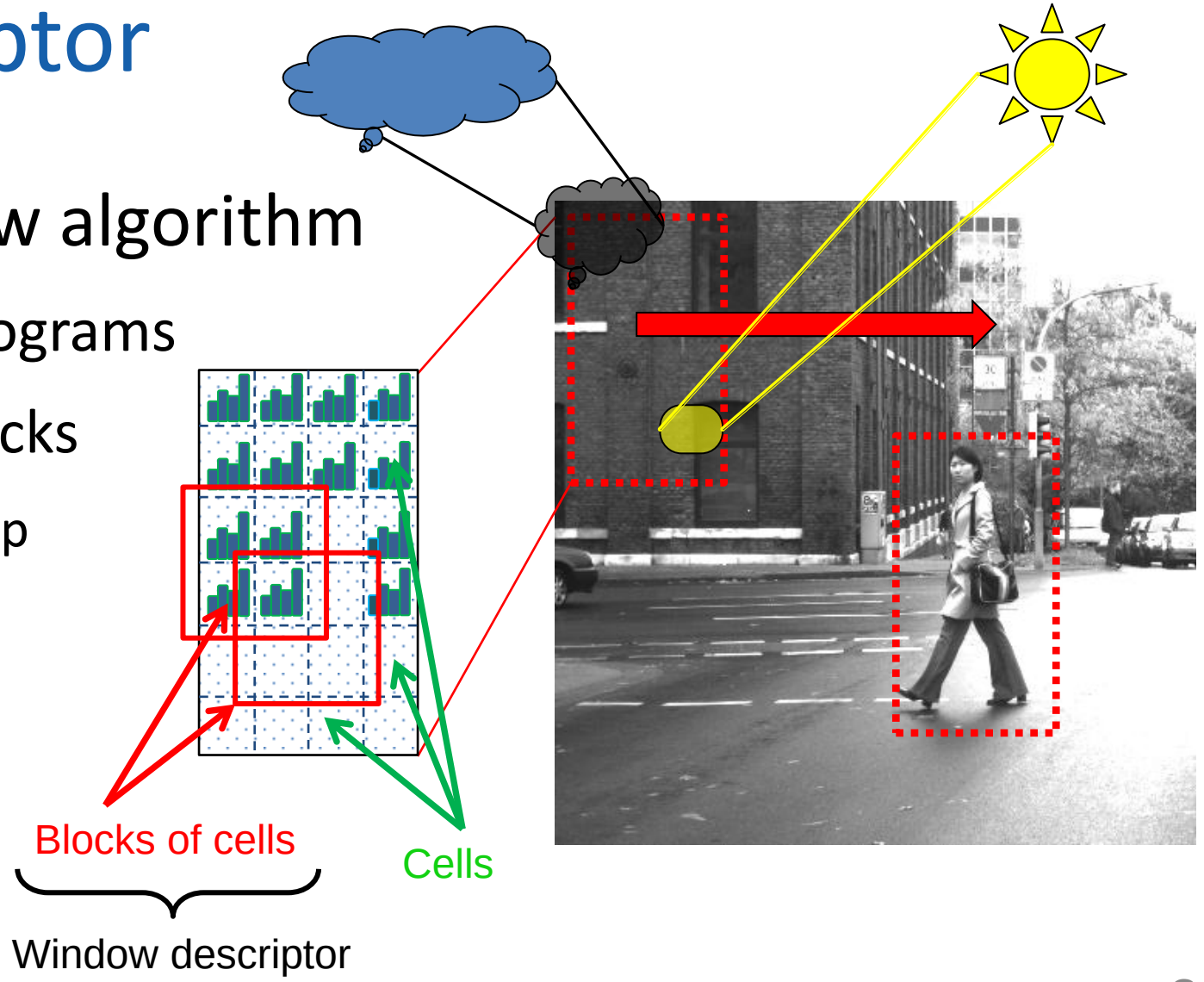


Magnitude

Edge Orientation

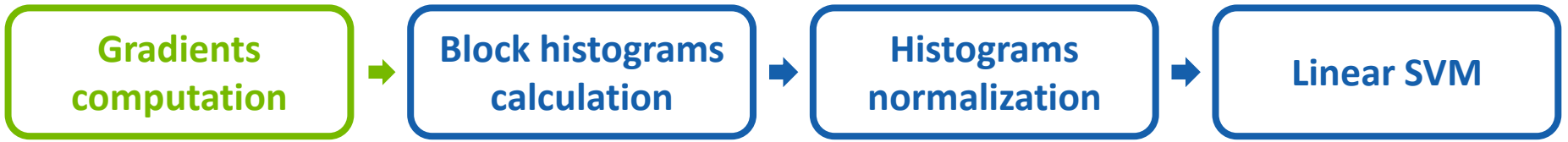
HOG Descriptor

- Sliding window algorithm
  - Compute histograms
  - Normalize blocks
    - Blocks overlap
  - Linear SVM
- Multi-scale



Multi-scale

Pedestrian Detection: Step 1



- Gamma correction improves quality
- Sobel filter 3x3 by columns and rows
- Output: magnitude and angle

$$\mathbf{G}_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * \text{Image}$$

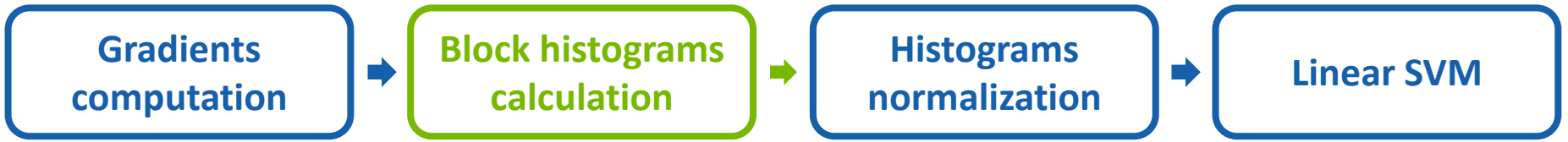
$$\mathbf{G}_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * \text{Image}$$

$$\mathbf{G} = \sqrt{\mathbf{G}_x^{\,2} + \mathbf{G}_y^{\,2}}$$

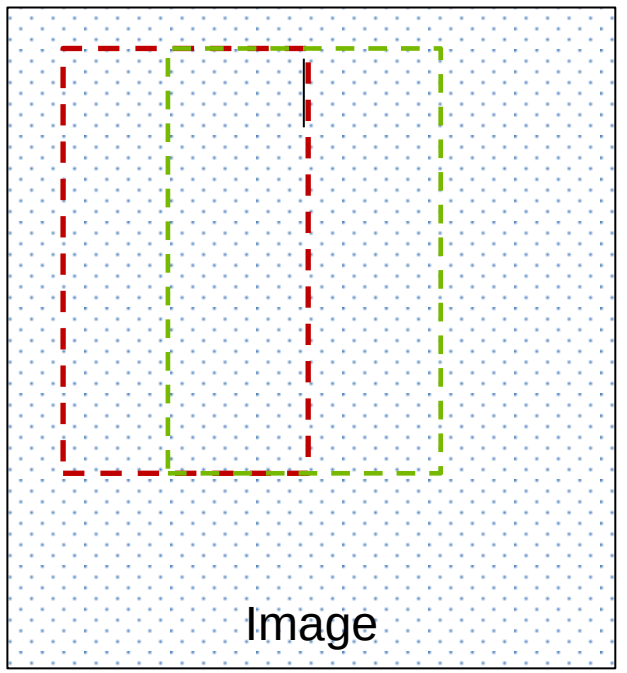
$$\mathbf{\Theta} = \arctan\left(\frac{\mathbf{G}_y}{\mathbf{G}_x}\right)$$

itseez

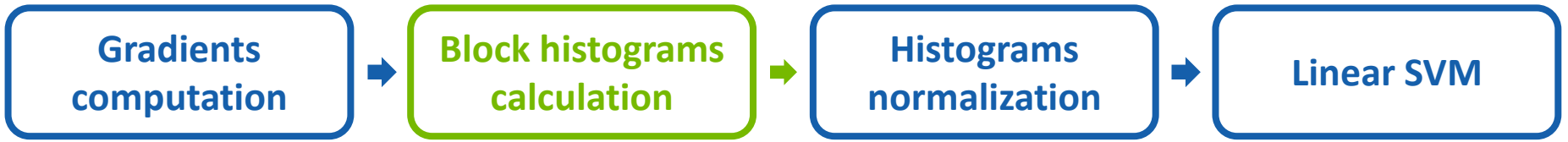
Pedestrian Detection: Step 2



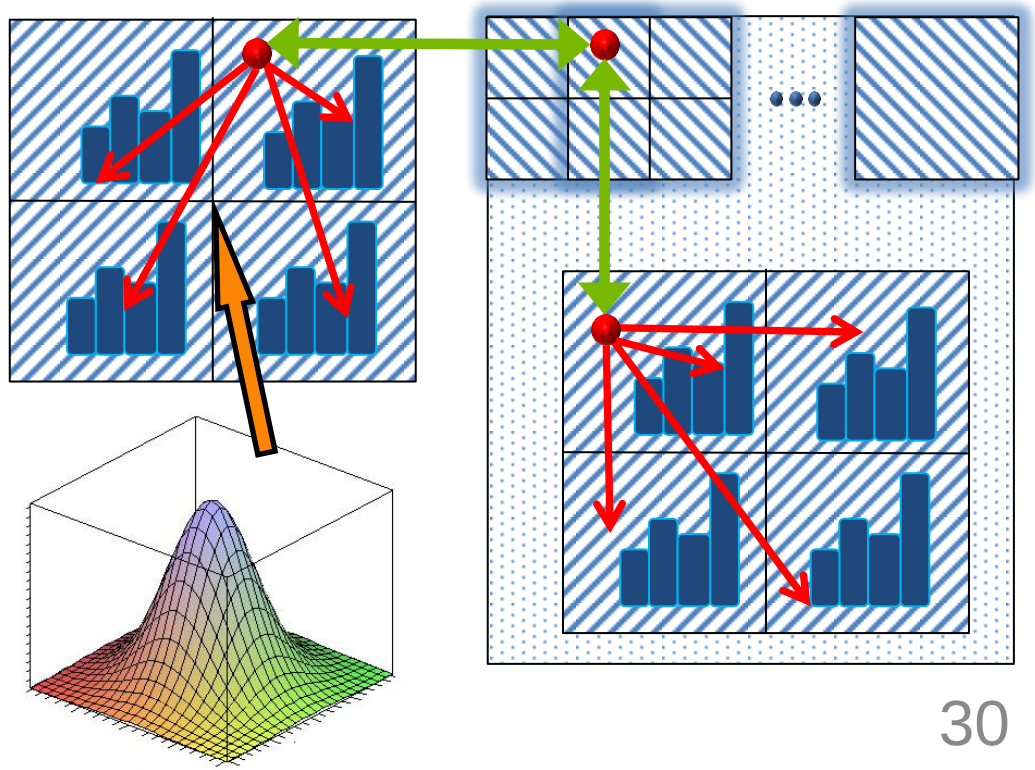
- Big intersection in close positions
- Require window stride to be multiple of cell size
- Histograms of blocks are computed independently



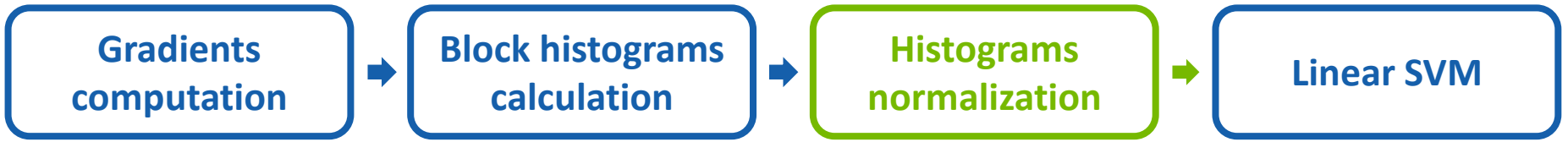
Pedestrian Detection: Step 2



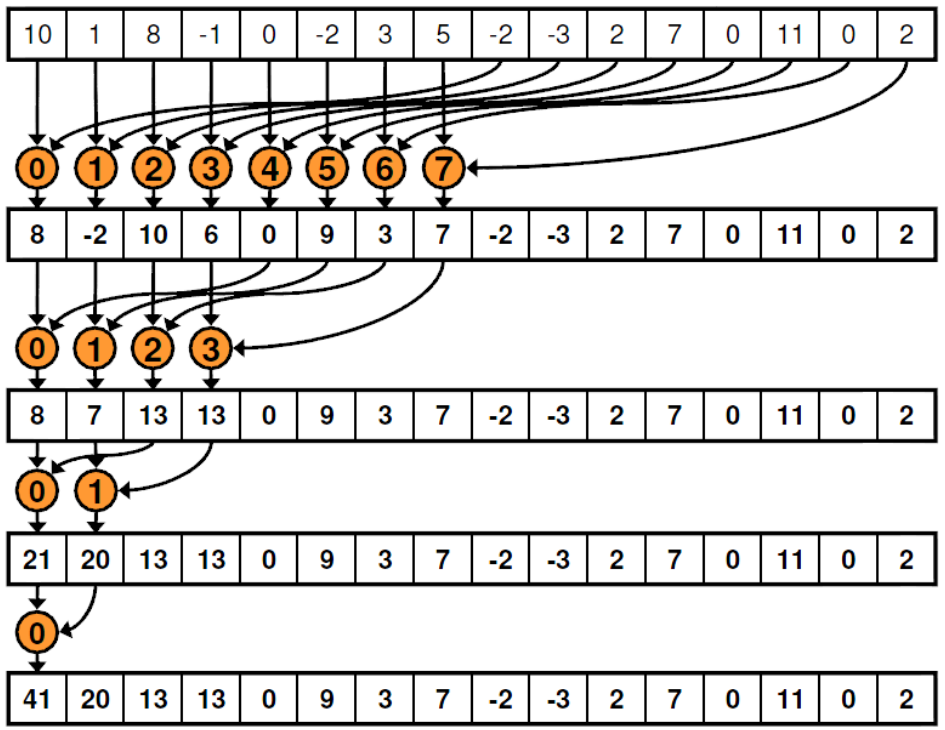
- Pixels vote in proportion to gradient magnitude
- Tri-linear interpolation
  - 2 orientation bins
  - 4 cells
- Gaussian
  - Decreases weight of pixels near block boundary



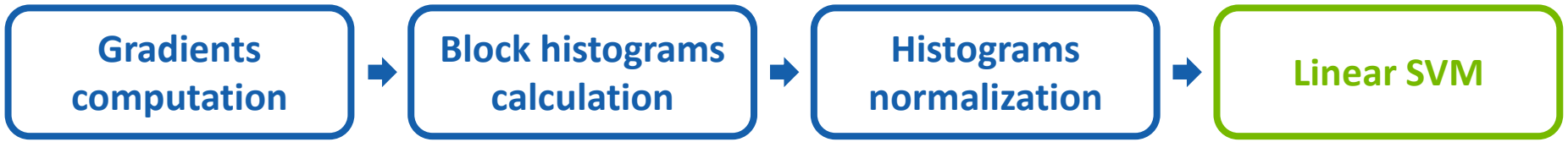
Pedestrian Detection: Step 3



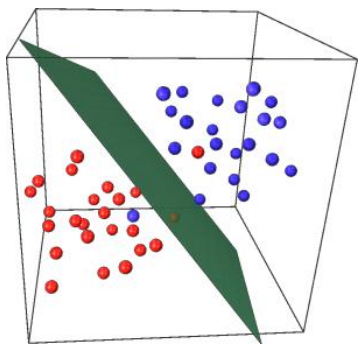
- Normalization
  - L2-Hys norm
    - L2 norm, clipping, normalization
  - 2 parallel reductions in shared memory

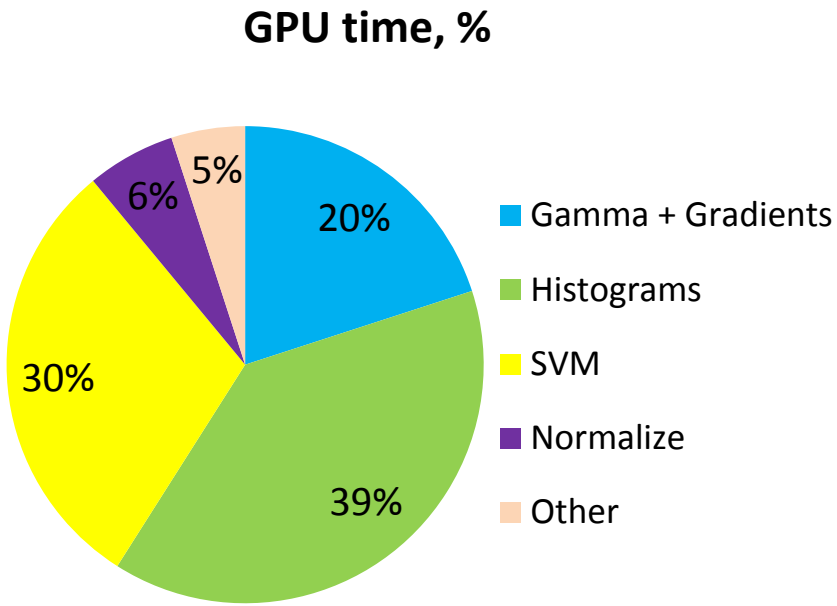


Pedestrian Detection: Step 4



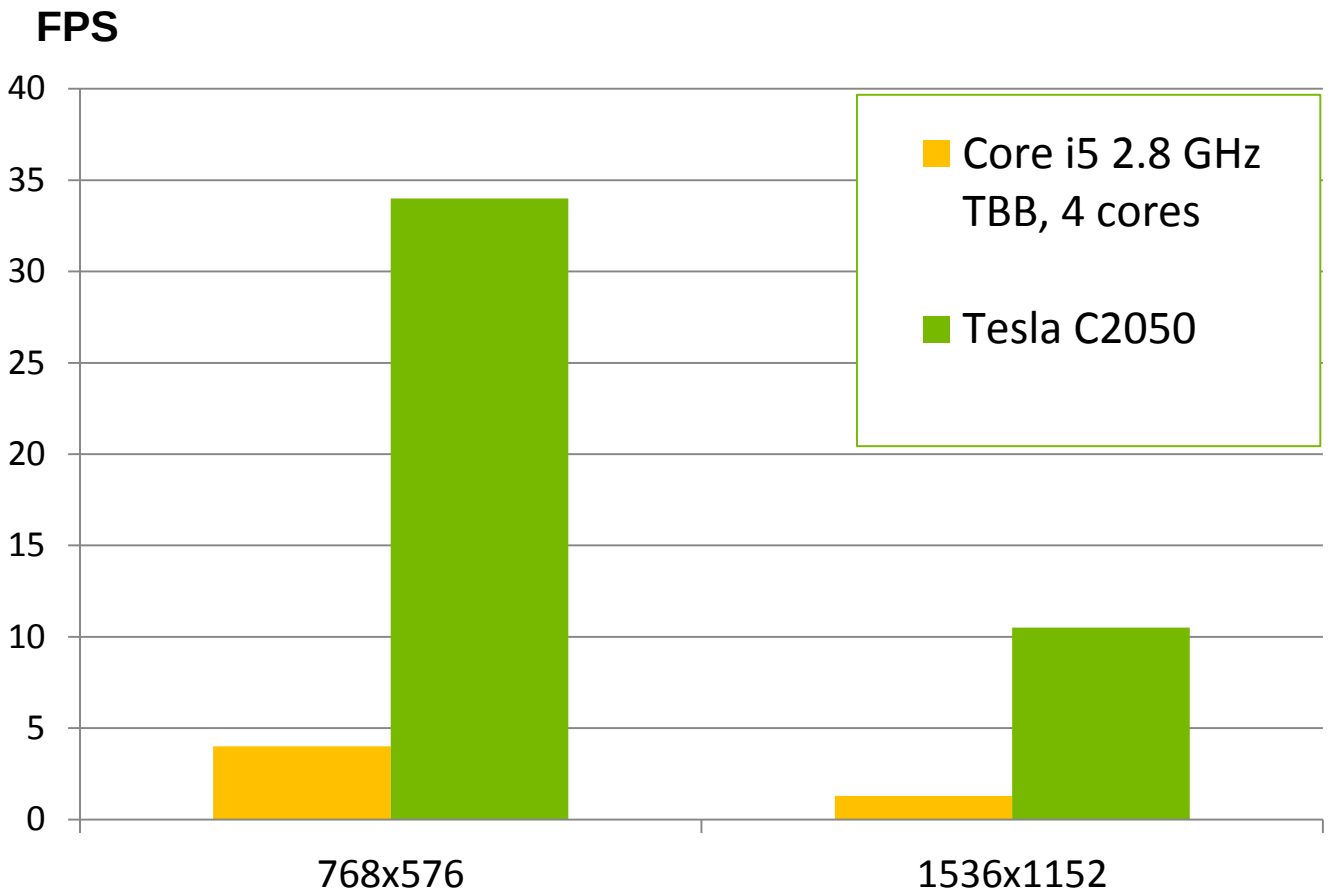
- Linear SVM
  - Classification is just a dot product
  - 1 thread block per window position





Pedestrian Detection Performance

- **8×** times faster!
- Detection rate
  - Same as CPU



Coming soon

- Implementation of Microsoft Kinect Fusion
  - License ???

itseez

Questions?

http://opencv.willowgarage.com/wiki

http://opencv.itseez.com

anatoly.bakshev@itseez.com

itseez