OpenCV

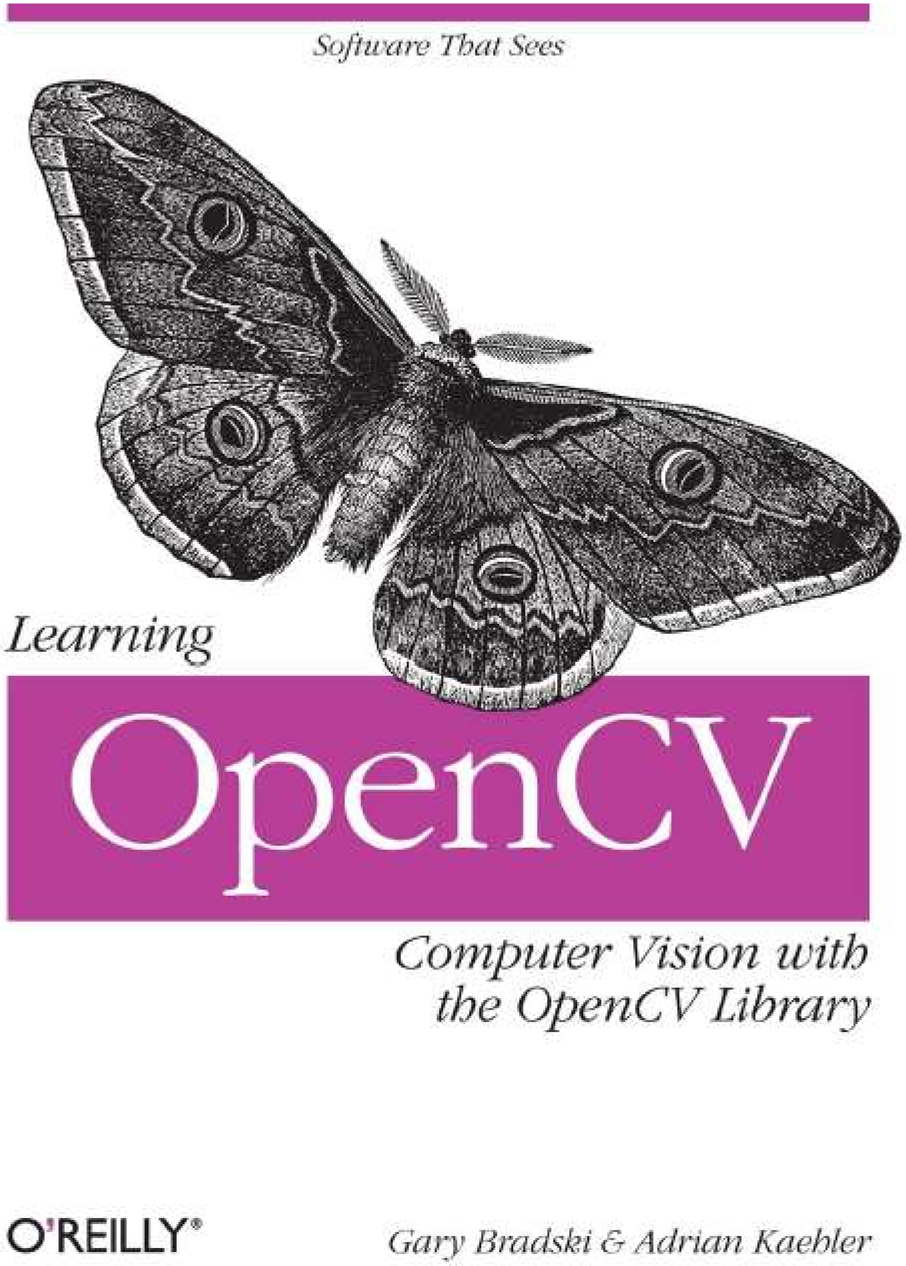
G. Bradski and A. Kaebler, *Learning OpenCV, Computer Vision with the OpenCV Library*, O’Reilly, 2008.

ISBN-10: 0596516134 or ISBN-13: 978-0596516130

Jenn-Jier James Lien (連 震 杰)
Professor

Computer Science and Information Engineering
National Cheng Kung University

(O) (06) 2757575 ext. 62540
jjlien@csie.ncku.edu.tw
http://robotics.csie.ncku.edu.tw



Content (1/3)

Image Processing

Content (2/3)

Computer Vision

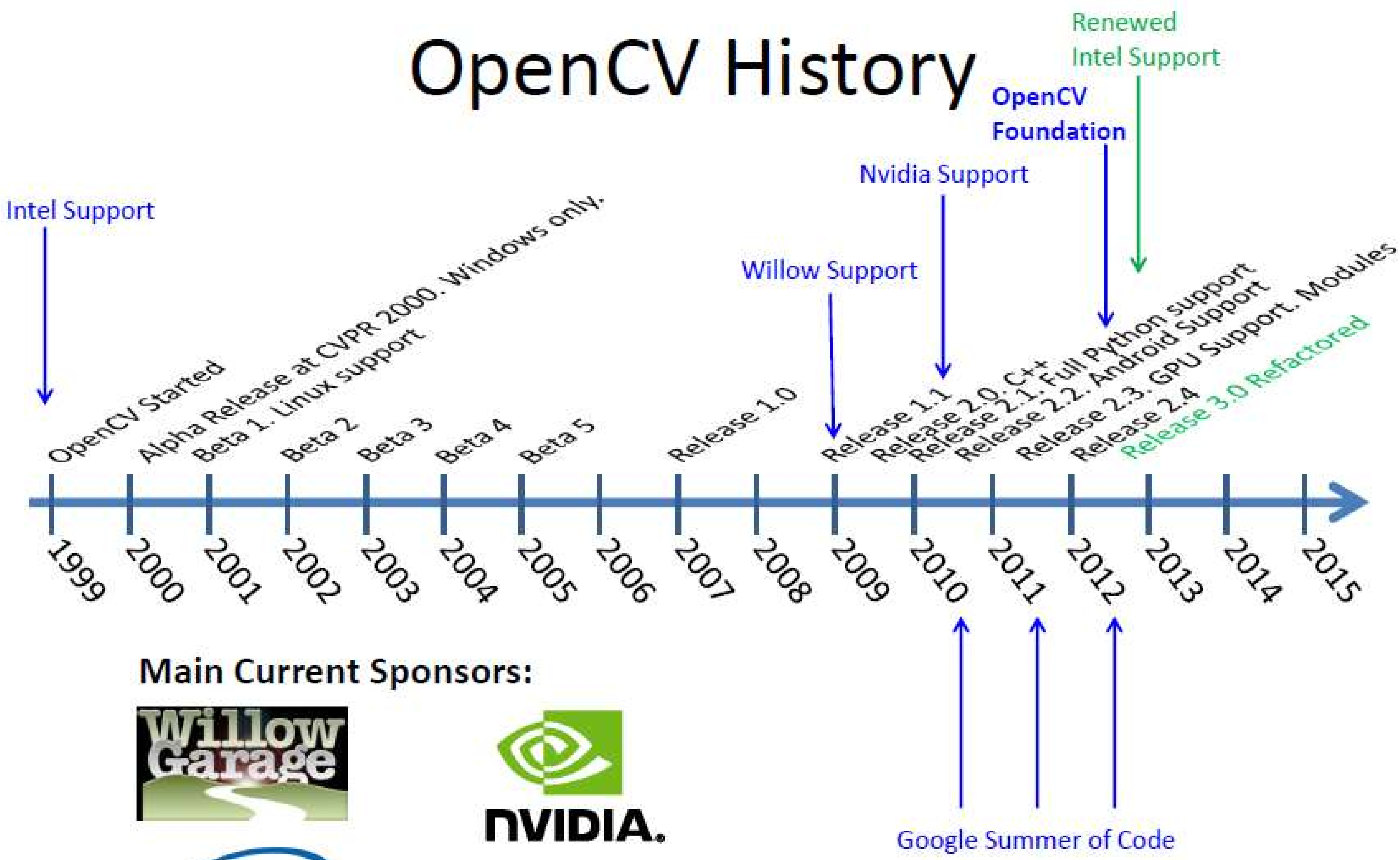
Machine Learning

Content (3/3)

What is OpenCV

- **Open** Source **C**omputer **V**ision Library
- Routines focused on real time image processing and 2D + 3D computer vision.
  - **On Linux, Windows, Mac, Android and iOS**
  - **C++, C, Java, Matlab and Python interfaces**
- <u>**Free**</u> for <u>commercial</u> or <u>research</u> use in whole or in part.

OpenCV History



Main Current Sponsors:

Environments, Platforms

- Languages:
  - C++, C#, Python, C, Java
- Platforms:

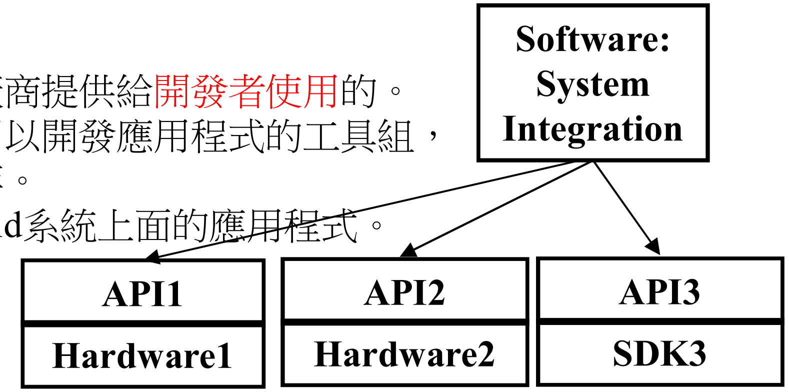
OpenCV Android Module

- **OpenCV 2.4 for Android:**
  - **Native Android Camera Support**
  - **Multithreading**
  - **Java API (soon)**
  - **Tegra HW Optimizations (soon)**

Wiki with the latest information:

http://blog.jyny.tw/2013/01/api-sdk.html

SDK Vs. API (1/2)



- **SDK (Software Development Kit)**
  翻譯成中文就是"軟體開發工具組"
  是用來幫一個 產品 或 平台 開發應用程式的工具組，由產品的廠商提供給開發者使用的。
  通常是 某一家廠商 針對某一 平台 或 系統 或 硬體 所發佈出來用以開發應用程式的工具組，
  在這個工具包裡面，可能包含了各式各樣的開發工具，模擬器等。
  例如：給 Android平台 使用的 Android SDK 就是用來開發 Android系統上面的應用程式。

- **API (Application Programming Interface)**
  翻譯成中文就是"應用程式介面"，其實這樣翻譯不太直觀。
  翻譯為介面，顧名思義就要溝通兩隻不同的東西用的，通常由一組函式所組成。
  在同一個平台下的兩個不同東西(程式 or 系統)，為了能取用對方的功能等等，
  所以一個 X程式 寫了一組函式，讓同一平台的其他程式取用 X程式 的功能，
  那組函式就可以說是那個 X程式 對外開放的 API。
  甚至是系統呼叫，
  因為作業系統的任務就是管理好電腦的各種資源，所以程式需要資源時必須跟作業系統溝通，申請使用某某功能等等，稱為系統呼叫(調用)。
  系統呼叫的時候也是取用OS作業系統提供的API。
  例如：我要在 自己的網頁 加入google map提供的功能，就使用"google map API“

- **通常SDK(開發者工具包) 裡也會帶有很多 API** ，用來調用一些系統平台程式提供的功能
  例如說：視窗顯示，圖形特效等等。
  以下舉一個實際例子來說明，調用系統程式功能的API 是怎麼一回事
  開發Windows應用程式的SDK(開發者工具包) 裡就包含 Win32 API
  說明： Win32 API 是一個函式庫，可以給 Windows應用程式 調用 Windows系統的功能

- 在**PTT**看到有人問了差異性，我的看法是
  SDK是用來開發某一個平台的程式的工具包 (J: Toolkit)，API 是讓同一平台下的程式取用它的功能的函式庫 (J: Library)。

SDK Vs. API (2/2)

1. API 通常大家都不會弄錯，的確就是以功能為導向的"方法"或"函式"清單，看程式語言或平台而定(Methods, Functions...)，而每個 API 主要都是為了達成某特定功能所設計的。
   開發商可以為了不同平台，設計相同的 API 讓開發者使用，也可能會因應不同平台，製作不同的 API 讓開發者使用。

2. 當 API 數量夠多功能夠繁複並且可交互為用的時候，
   (例如為了達成某些功能，常需要同時引用某些 APIs 來完成)
   開發商就會為了開發便利，而預先撰寫好一些組合好 APIs 的 API供開發者使用，來統一有特定需求的開發者能有一致的開發與使用體驗，(例如讓使用"網路連線"的開發者不需自己處理網路的基礎溝通信息，與錯誤處理方式，使 API 在應用的時候有一定程度的便利性等)

   然後，也陸續發展出，甚至是設計不同平台開發環境所需的套件，測試、除錯工具，尤其針對不同平台，更是設計了對應的工具來協助開發、除錯；
   SDK 名詞之所以出現，儼然是為了匯整上述這些資源而誕生的，我想也可以說成是 API 的包含者(直接使用)與應用者(以便加速開發)，也因此可以說這兩個是屬於不同層級的東西...

以 Android 來說：
a. 我們要擁有 Android SDK 才能開發 Android 應用程式
(針對不同開發系統而不同 Linux, Windows)
b. Android SDK 裡的 APIs 統統都可以單獨使用，只不過你會發現他們都還有許多其他的應用，而且可能還比自己寫來得更有效率
c. Android SDK 跟開發環境整合後，除了提供程式碼語法錯誤檢查外，還提供模擬器平台讓我們不需要硬體就可以模擬測試
d. Android SDK 內有測試用的 APIs，來協助我們檢查記憶體用量、程式效能以及狀態顯現等功能 (當然它建議僅在測試除錯時才使用)

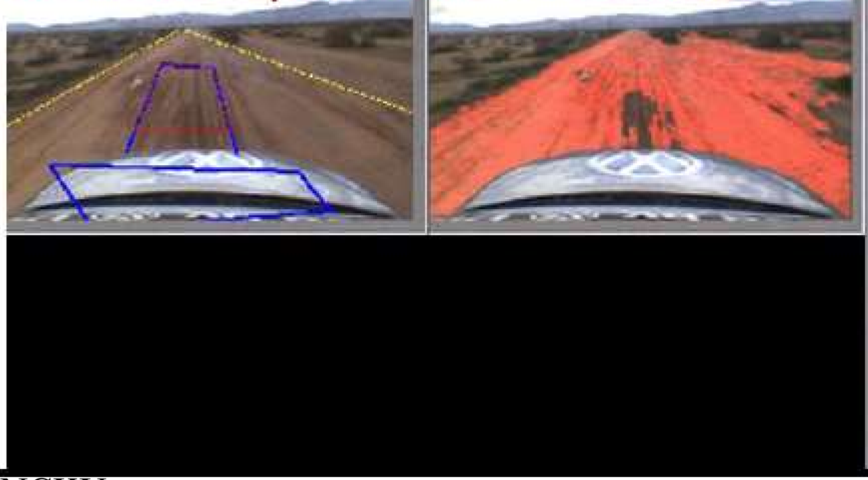
以 Facebook 來說：
a. 我們要下載 Facebook SDK 才能開發應用程式
(針對不同開發語言或平台而不同，PHP, JavaScript, Android, iOS)
b. Facebook 官網提供 SDK 詳細的 APIs 解說與使用方法、範例說明等
c. Facebook 官網提供 線上測試工具，測試某些API的指令與語法

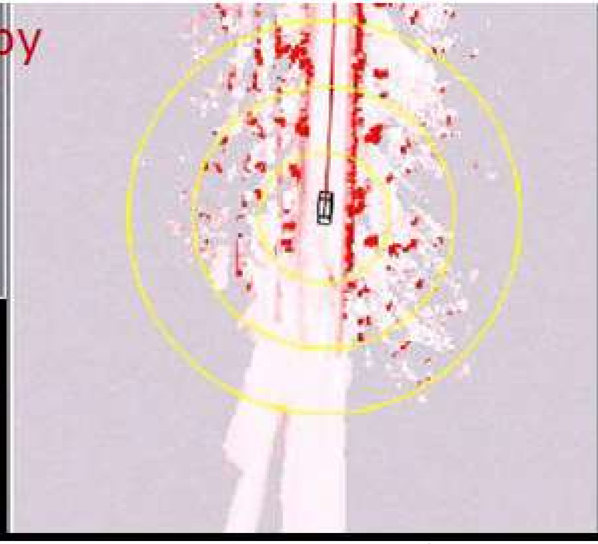
以 Google Map API 來說：
網頁開發，只需使用 Google Map API 即可在網頁上開發、使用其功能(但是在 Android, iOS 上開發則另外需要 Google Map SDK 才行)

由此可知，我們可以清楚知道 API 與 SDK 的定義差別了！

Where is OpenCV Used?

- Academic and Industry Research
- Security systems
- Google Maps, Streetview
- Image/video search and retrieval
- Structure from motion in movies
- Machine vision factory production inspection systems
- Automatic Driver Assistance Systems
- Safety monitoring (Dam sites, mines, swimming pools)
- Robotics – personal, industrial, hobby





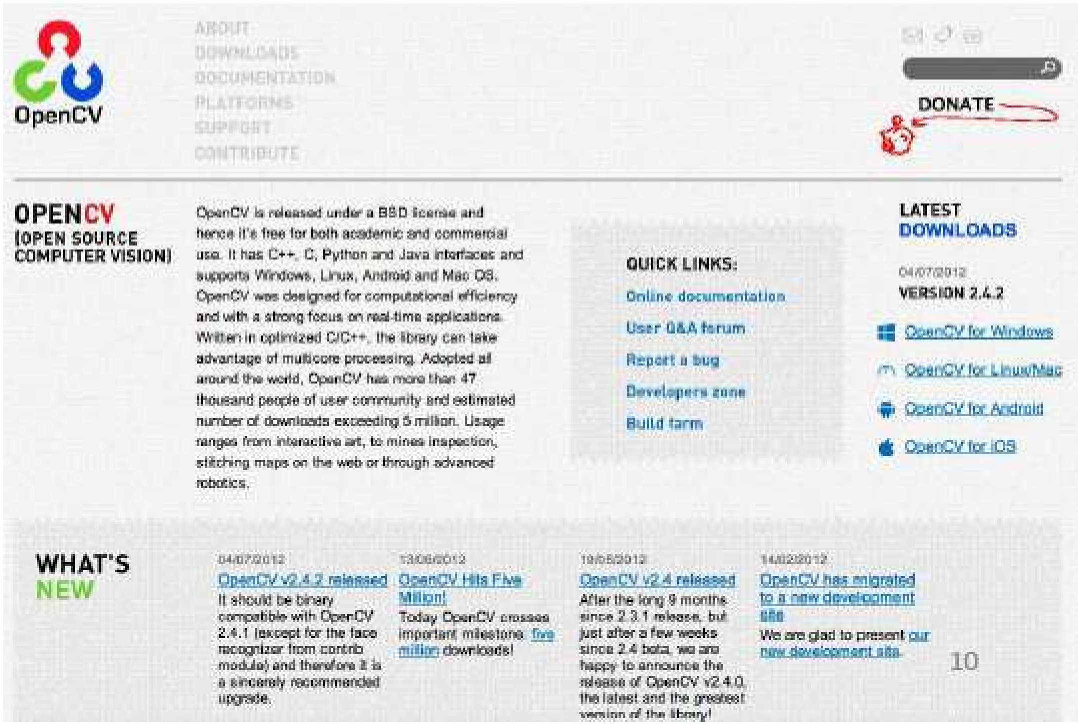
OpenCV Foundation

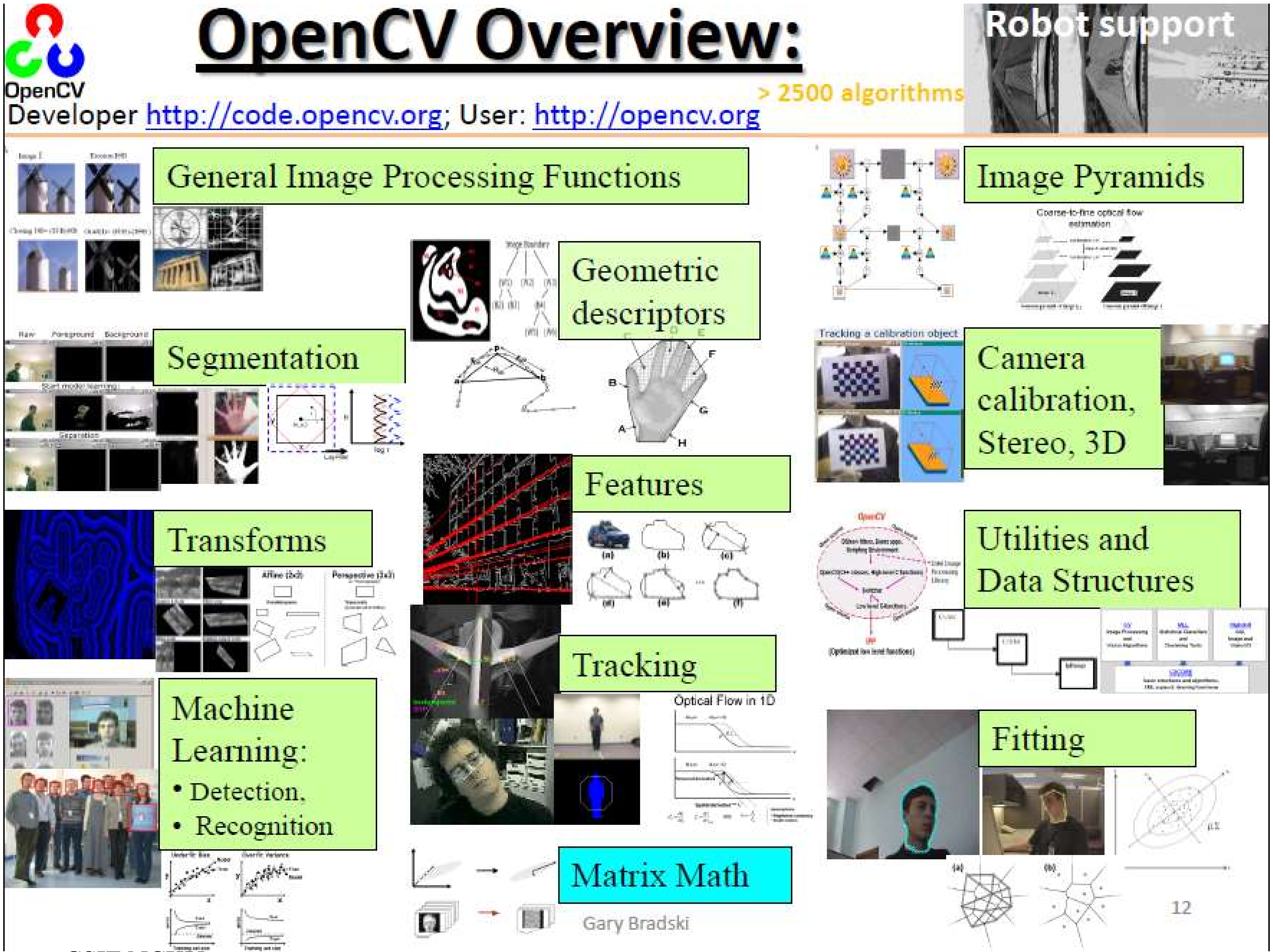
- Founded this July, 2012
- http://opencv.org (user site) Documentation: http://docs.opencv.org
- http://code.opencv.org (developer site)
- Contribute (via Credit, debit or paypal):
  - http://tinyurl.com/7eujyo2

For larger corporate support And/or partnership, contact
Garybradski@gmail.com

Support levels:
- Diamond
- Platinum
- Gold
- Silver
- Bronze

Includes support, brainstorming and development sprints. Higher levels include strategic control.




OpenCV Overview:
OpenCV
Developer http://code.opencv.org; User: http://opencv.org
> 2500 algorithms
Robot support
General Image Processing Functions
Image Pyramids
Coarse-to-fine optical flow estimation
Geometric descriptors
Segmentation
Tracking a calibration object
Camera calibration, Stereo, 3D
Features
Transforms
Affine (2x2)
Perspective (3x3)
Utilities and Data Structures
OpenCV
Tracking
Optical Flow in 1D
Machine Learning:
• Detection,
• Recognition
Fitting
Matrix Math
Gary Bradski
12

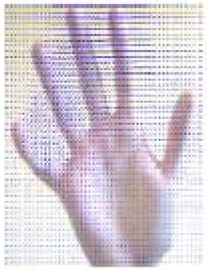
OpenCV Algorithm Modules Overview

HighGUI:
I/O, Interface

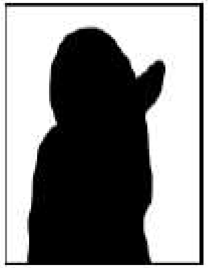
Image Processing

Transforms

Fitting



Optical Flow
Tracking



Segmentation

Calibration

Features
VSLAM

Depth, Pose
Normals, Planes,
3D Features

Object recognition
Machine learning

Computational
Photography

CORE:
Data structures, Matrix math, Exceptions etc

13

Machine Learning Library (MLL)

CLASSIFICATION / REGRESSION

(new) Fast Approximate NN (FLANN)

(new) Extremely Random Trees

CART

Naïve Bayes

MLP (Back propagation)

Statistical Boosting, 4 flavors

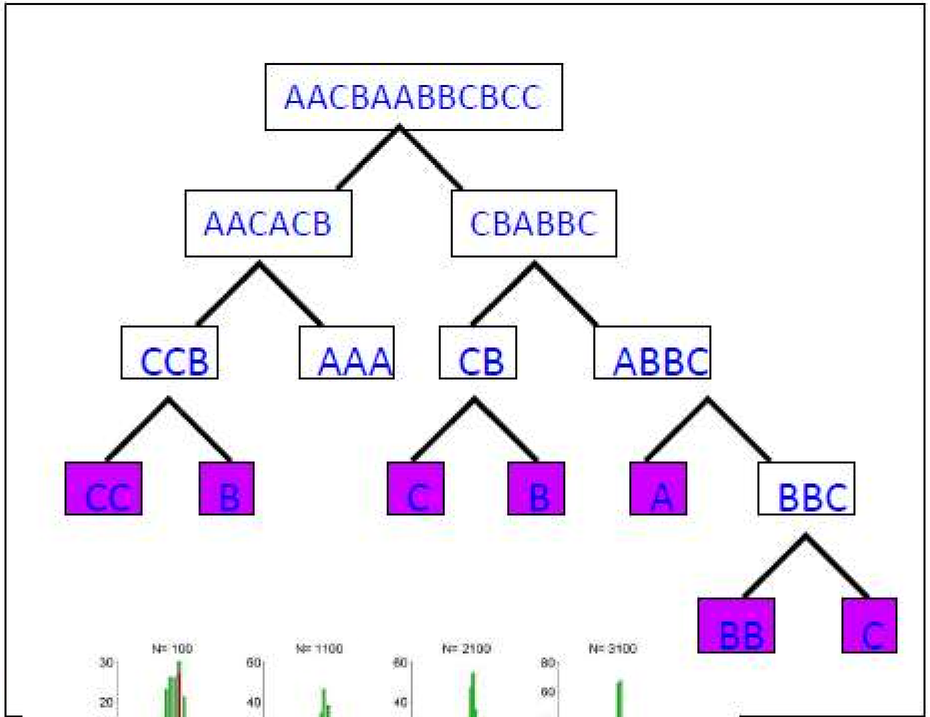
Random Forests

SVM

Face Detector

(Histogram matching)

(Correlation)

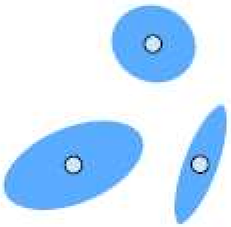


CLUSTERING

K-Means

EM

(Mahalanobis distance)



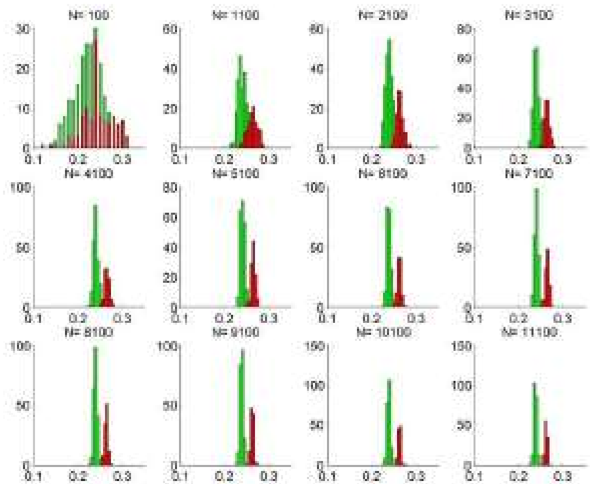
TUNING/VALIDATION

Cross validation

Bootstrapping

Variable importance

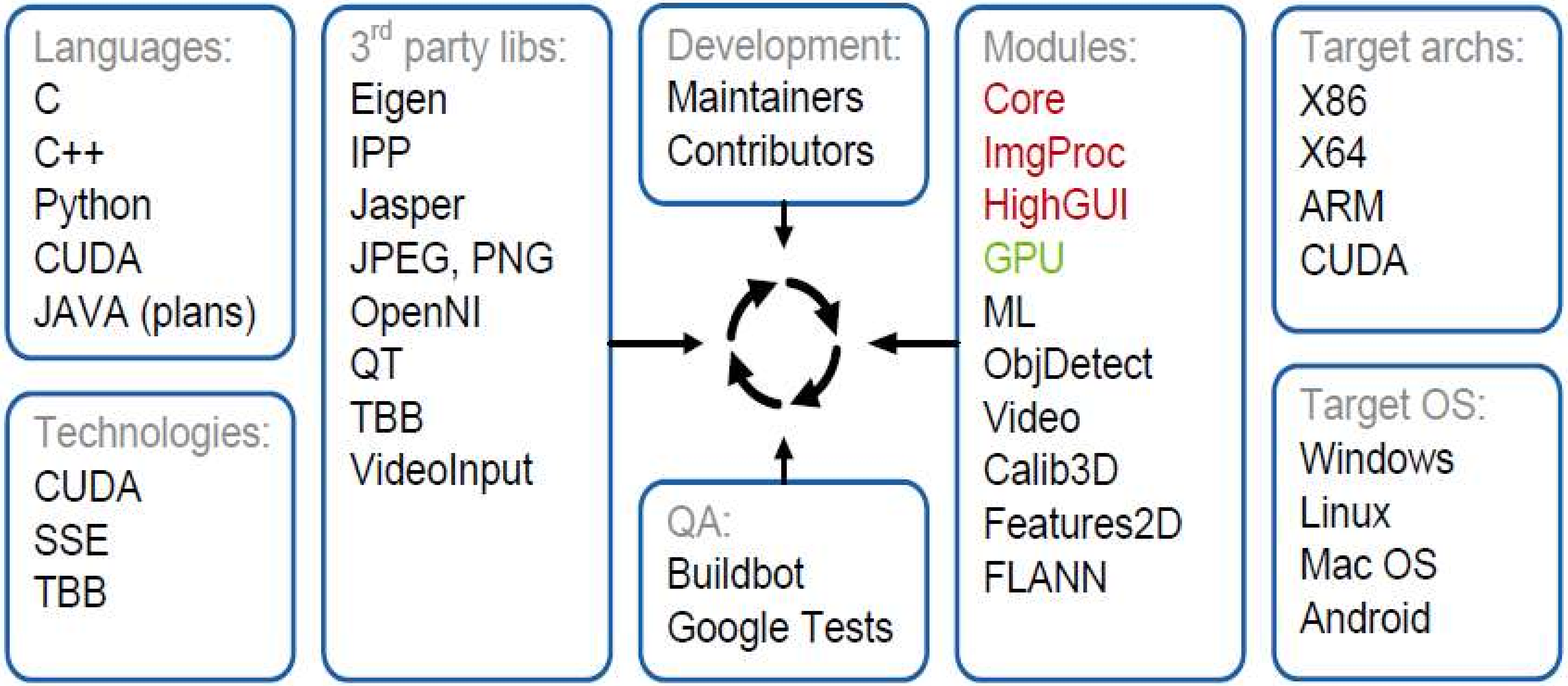
Sampling methods



14

http://opencv.org

OpenCV Architecture and Development



CORE:
Data structures, Matrix math, Exceptions etc

OpenCV Modules: Core

OpenCV Core (C++)

The OpenCV C++ reference manual is here: *http://opencv.willowgarage.com/documentation/cpp/.* *Use **Quick Search** to find descriptions of the particular functions and classes*

Key OpenCV Classes

`Point_`	Template 2D point class
`Point3_`	Template 3D point class
`Size_`	Template size (width, height) class
`Vec`	Template short vector class
`Matx`	Template small matrix class
`Scalar`	4-element vector
`Rect`	Rectangle
`Range`	Integer value range
`Mat`	2D or multi-dimensional dense array (can be used to store matrices, images, histograms, feature descriptors, voxel volumes etc.)
`SparseMat`	Multi-dimensional sparse array
`Ptr`	Template smart pointer class

Matrix Basics

Create a matrix

```
Mat image(240, 320, CV_8UC3);
```

[Re]allocate a pre-declared matrix

```
image.create(480, 640, CV_8UC3);
```

Create a matrix initialized with a constant

```
Mat A33(3, 3, CV_32F, Scalar(5));
Mat B33(3, 3, CV_32F); B33 = Scalar(5);
Mat C33 = Mat::ones(3, 3, CV_32F)*5.;
Mat D33 = Mat::zeros(3, 3, CV_32F) + 5.;
```

Create a matrix initialized with specified values

```
double a = CV_PI/3;
Mat A22 = (Mat_<float>(2, 2) <<
  cos(a), -sin(a), sin(a), cos(a));
float B22data[] = {cos(a), -sin(a), sin(a), cos(a)};
Mat B22 = Mat(2, 2, CV_32F, B22data).clone();
```

Initialize a random matrix

```
randu(image, Scalar(0), Scalar(256)); // uniform dist
randn(image, Scalar(128), Scalar(10)); // Gaussian dist
```

Convert matrix to/from other structures
(without copying the data)

```
Mat image_alias = image;
float* Idata=new float[480*640*3];
Mat I(480, 640, CV_32FC3, Idata);
vector<Point> iptvec(10);
Mat iP(iptvec); // iP – 10x1 CV_32SC2 matrix
IplImage* oldC0 = cvCreateImage(cvSize(320,240),16,1);
Mat newC = cvarrToMat(oldC0);
IplImage oldC1 = newC; CvMat oldC2 = newC;
```

... (with copying the data)

```
Mat newC2 = cvarrToMat(oldC0).clone();
vector<Point2f> ptvec = Mat_<Point2f>(iP);
```

Access matrix elements

```
A33.at<float>(i,j) = A33.at<float>(j,i)+1;
Mat dyImage(image.size(), image.type());
for(int y = 1; y < image.rows-1; y++) {
  Vec3b* prevRow = image.ptr<Vec3b>(y-1);
  Vec3b* nextRow = image.ptr<Vec3b>(y+1);
  for(int x = 0; y < image.cols; x++)
    for(int c = 0; c < 3; c++)
      dyImage.at<Vec3b>(y,x)[c] =
      saturate_cast<uchar>(
      nextRow[x][c] - prevRow[x][c]);
}
Mat_<Vec3b>::iterator it = image.begin<Vec3b>(),
  itEnd = image.end<Vec3b>();
for(; it != itEnd; ++it)
  (*it)[1] ^= 255;
```

Matrix Manipulations: Copying, Shuffling, Part Access

`src.copyTo(dst)`	Copy matrix to another one
`src.convertTo(dst,type,scale,shift)`	Scale and convert to another datatype
`m.clone()`	Make deep copy of a matrix
`m.reshape(nch,nrows)`	Change matrix dimensions and/or number of channels without copying data
`m.row(i)`, `m.col(i)`	Take a matrix row/column
`m.rowRange(Range(i1,i2))` `m.colRange(Range(j1,j2))`	Take a matrix row/column span
`m.diag(i)`	Take a matrix diagonal
`m(Range(i1,i2),Range(j1,j2))`, `m(roi)`	Take a submatrix
`m.repeat(ny,nx)`	Make a bigger matrix from a smaller one
`flip(src,dst,dir)`	Reverse the order of matrix rows and/or columns
`split(...)`	Split multi-channel matrix into separate channels
`merge(...)`	Make a multi-channel matrix out of the separate channels
`mixChannels(...)`	Generalized form of split() and merge()
`randShuffle(...)`	Randomly shuffle matrix elements

Example 1. Smooth image ROI in-place

```
Mat imgroi = image(Rect(10, 20, 100, 100));
GaussianBlur(imgroi, imgroi, Size(5, 5), 1.2, 1.2);
```

Example 2. Somewhere in a linear algebra algorithm

```
m.row(i) += m.row(j)*alpha;
```

Example 3. Copy image ROI to another image with conversion

```
Rect r(1, 1, 10, 20);
Mat dstroi = dst(Rect(0,10,r.width,r.height));
src(r).convertTo(dstroi, dstroi.type(), 1, 0);
```

Simple Matrix Operations

OpenCV implements most common arithmetical, logical and other matrix operations, such as

- `add()`, `subtract()`, `multiply()`, `divide()`, `absdiff()`, `bitwise_and()`, `bitwise_or()`, `bitwise_xor()`, `max()`, `min()`, `compare()`
  – correspondingly, addition, subtraction, element-wise multiplication ... comparison of two matrices or a matrix and a scalar.

  Example. Alpha compositing function:

```
void alphaCompose(const Mat& rgba1,
    const Mat& rgba2, Mat& rgba_dest)
{
    Mat a1(rgba1.size(), rgba1.type()), ra1;
    Mat a2(rgba2.size(), rgba2.type());
    int mixch[]={3, 0, 3, 1, 3, 2, 3, 3};
    mixChannels(&rgba1, 1, &a1, 1, mixch, 4);
    mixChannels(&rgba2, 1, &a2, 1, mixch, 4);
    subtract(Scalar::all(255), a1, ra1);
    bitwise_or(a1, Scalar(0,0,0,255), a1);
    bitwise_or(a2, Scalar(0,0,0,255), a2);
    multiply(a2, ra1, a2, 1./255);
    multiply(a1, rgba1, a1, 1./255);
    multiply(a2, rgba2, a2, 1./255);
    add(a1, a2, rgba_dest);
}
```

- `sum()`, `mean()`, `meanStdDev()`, `norm()`, `countNonZero()`, `minMaxLoc()`,
  – various statistics of matrix elements.
- `exp()`, `log()`, `pow()`, `sqrt()`, `cartToPolar()`, `polarToCart()`
  – the classical math functions.
- `scaleAdd()`, `transpose()`, `gemm()`, `invert()`, `solve()`, `determinant()`, `trace()` `eigen()`, `SVD`,
  – the algebraic functions + SVD class.
- `dft()`, `idft()`, `dct()`, `idct()`,
  – discrete Fourier and cosine transformations

For some operations a more convenient algebraic notation can be used, for example:

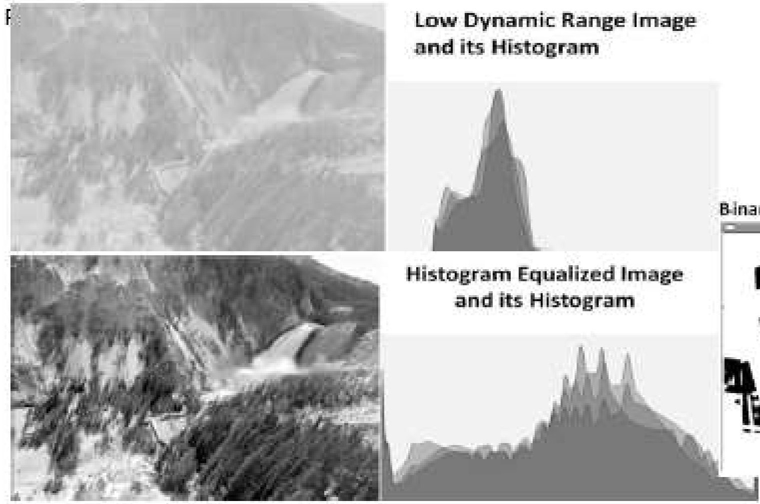
```
Mat delta = (J.t()*J + lambda*
   Mat::eye(J.cols, J.cols, J.type()))
   .inv(CV_SVD)*(J.t()*err);
```

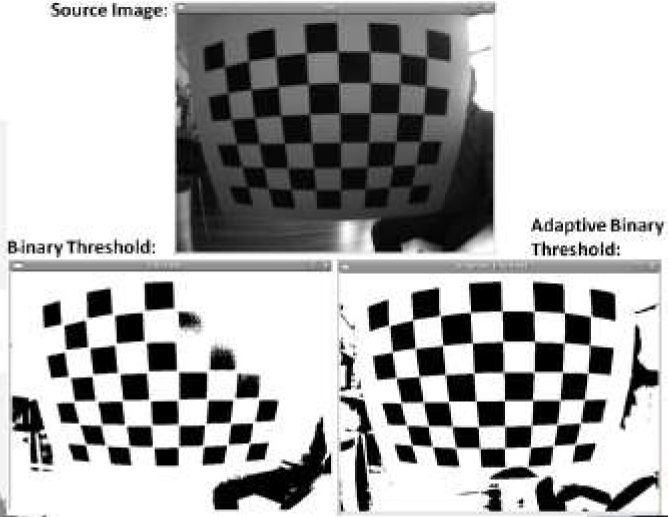
implements the core of Levenberg-Marquardt optimization algorithm.

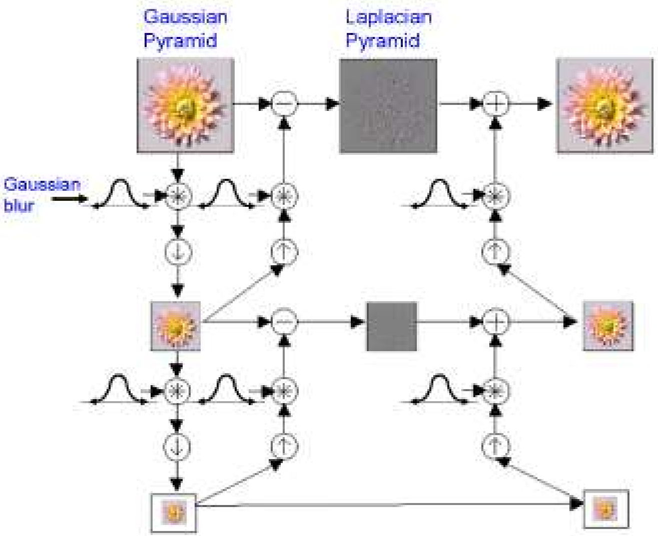
17

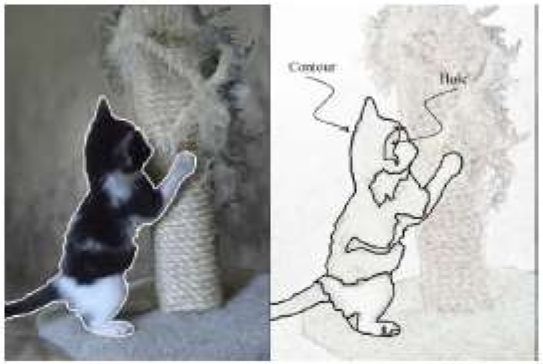
OpenCV Modules: Image Processing

Image



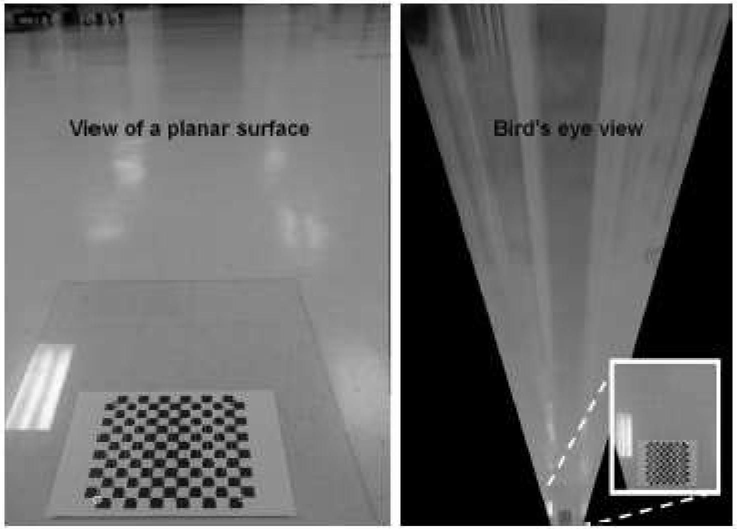


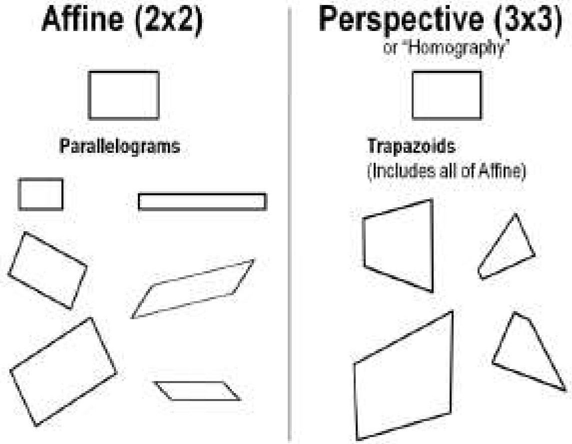


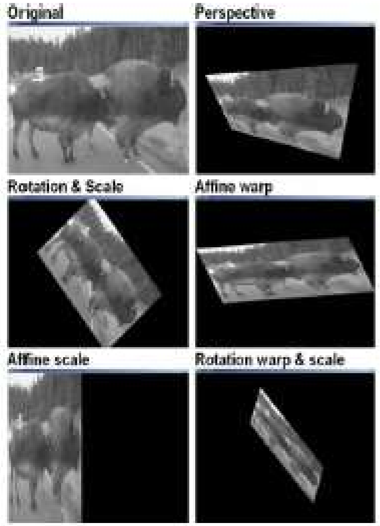


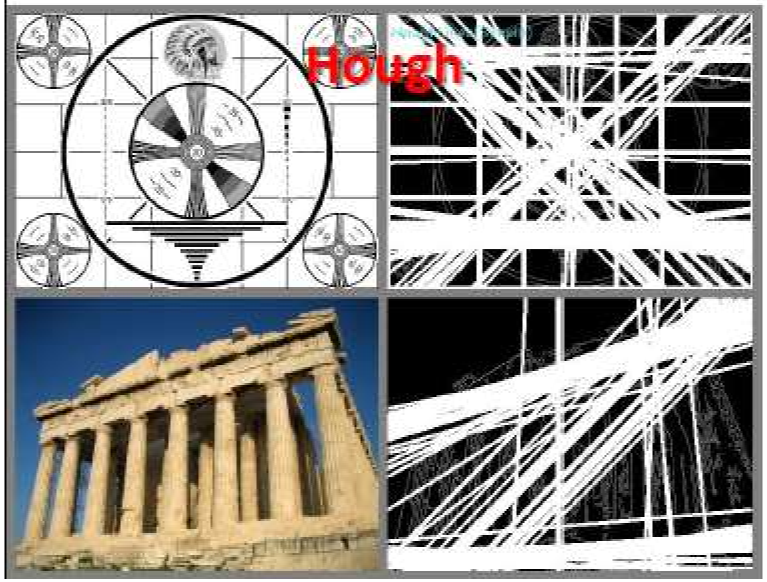
OpenCV Modules: Transforms

Transforms



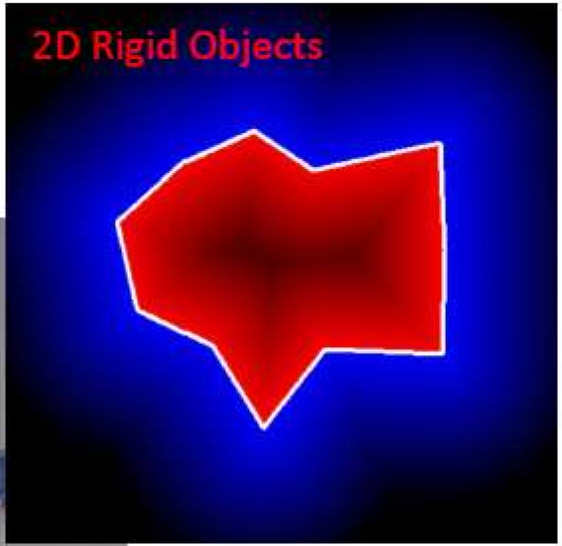


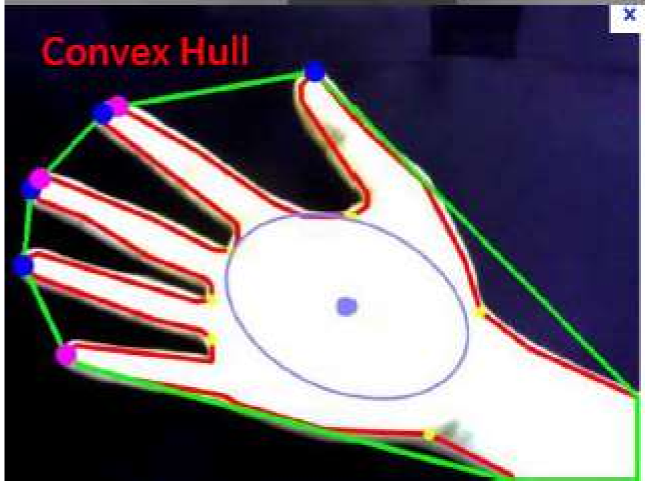


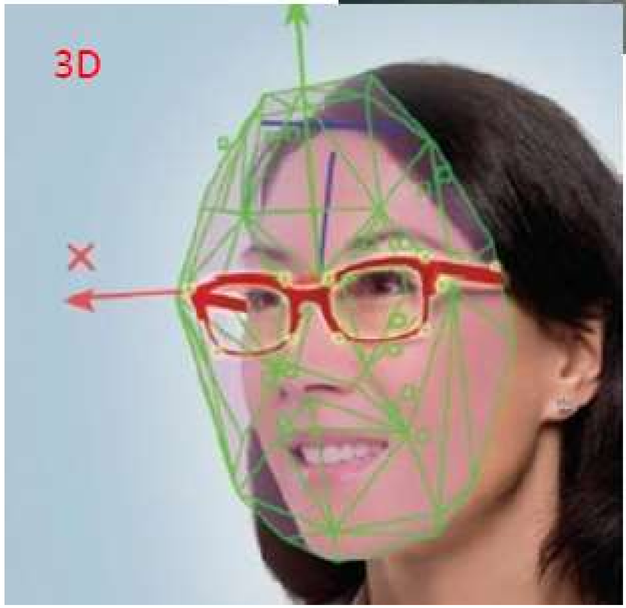


OpenCV Modules: Fitting

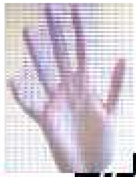

Fitting







21



Optical Flow Tracking

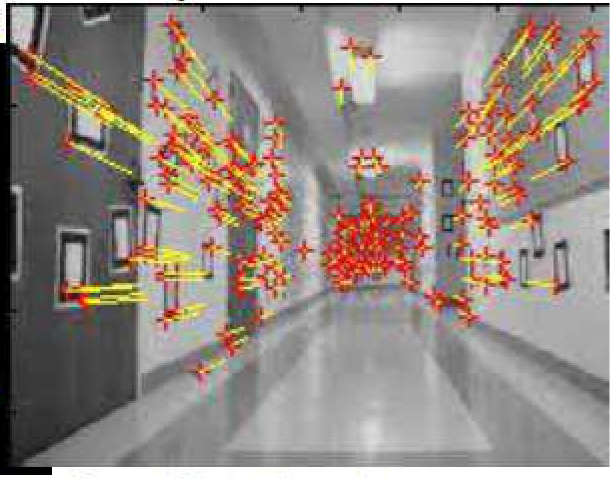
OpenCV Modules: Optic Flow, Track

http://www.youtube.com/watch?v=bWyBGmzfP-g

```
// opencv/samples/c/lkdemo.c
int main(…){
…
CvCapture* capture = <…> ?
    cvCaptureFromCAM(camera_id) :
    cvCaptureFromFile(path);
if( !capture ) return -1;
for(;;) {
  IplImage* frame=cvQueryFrame(capture);
  if(!frame) break;
  // … copy and process image
cvCalcOpticalFlowPyrLK( …)
  cvShowImage( "LkDemo", result );
  c=cvWaitKey(30); // run at ~20-30fps speed
  if(c >= 0) {
      // process key
 }}
cvReleaseCapture(&capture
```

lkdemo.c, 190 lines

(needs camera to run)

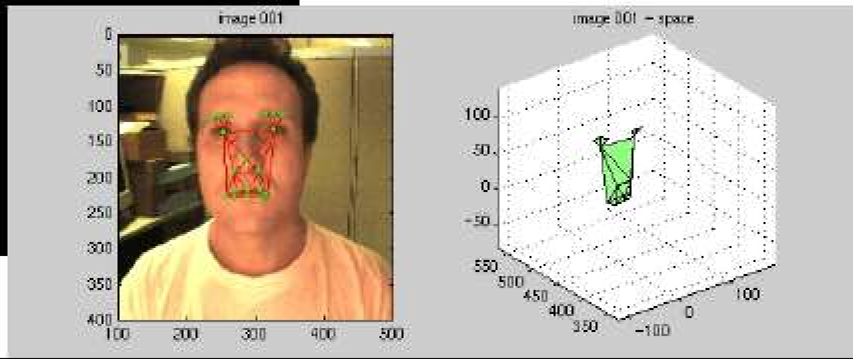


$$I(x+dx, y+dy, t+dt) = I(x, y, t);$$

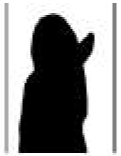
$$-\partial I/\partial t = \partial I/\partial x \cdot (dx/dt) + \partial I/\partial y \cdot (dy/dt);$$

$$G \cdot \partial X = b,$$

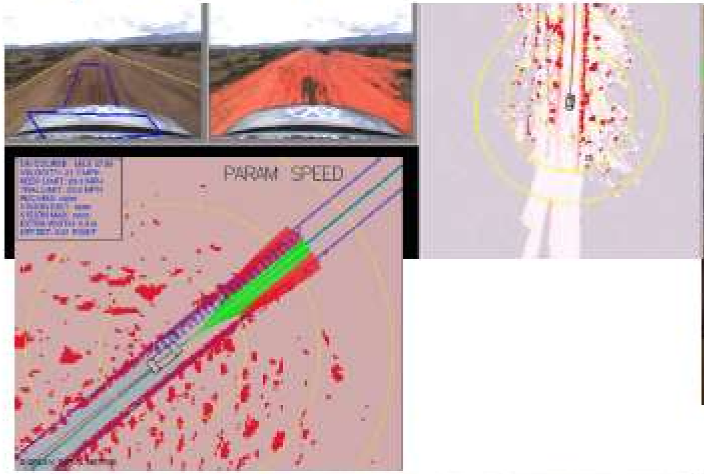
$$\partial X = (\partial x, \partial y), G = \sum \begin{bmatrix} I_x^2, & I_x I_y \\ I_x I_y, & I_y^2 \end{bmatrix}, b = \sum I_t \begin{bmatrix} I_x \\ I_y \end{bmatrix}$$



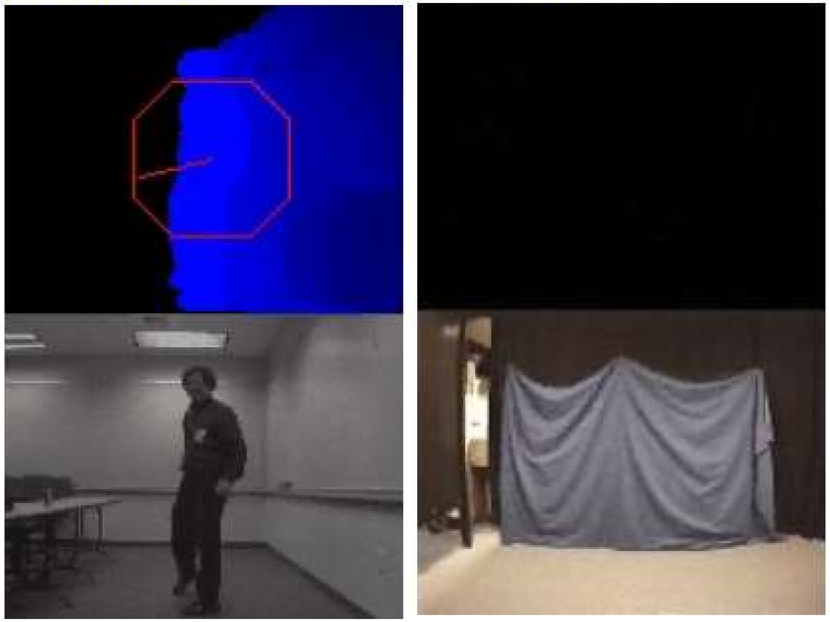
http://www.youtube.com/watch?v=1osj7kRgswk


Segmentation

OpenCV Modules: Segmentation



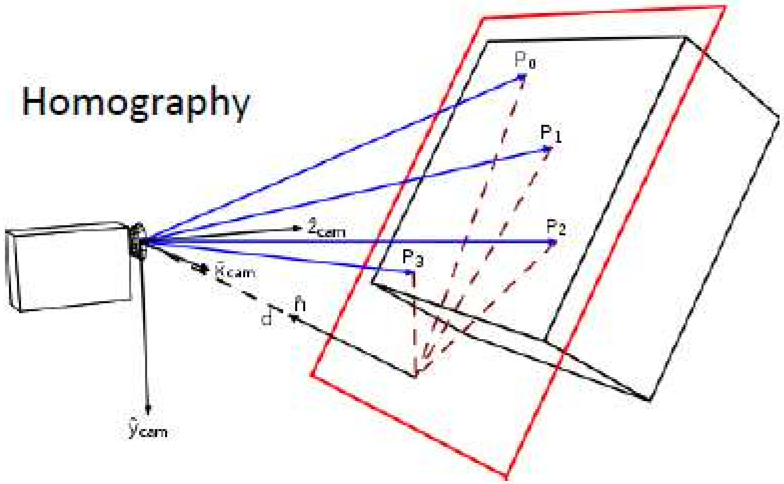
https://www.youtube.com/watch?v=OxmDonZja74
http://www.youtube.com/watch?v=Ktrjh5-KLKo

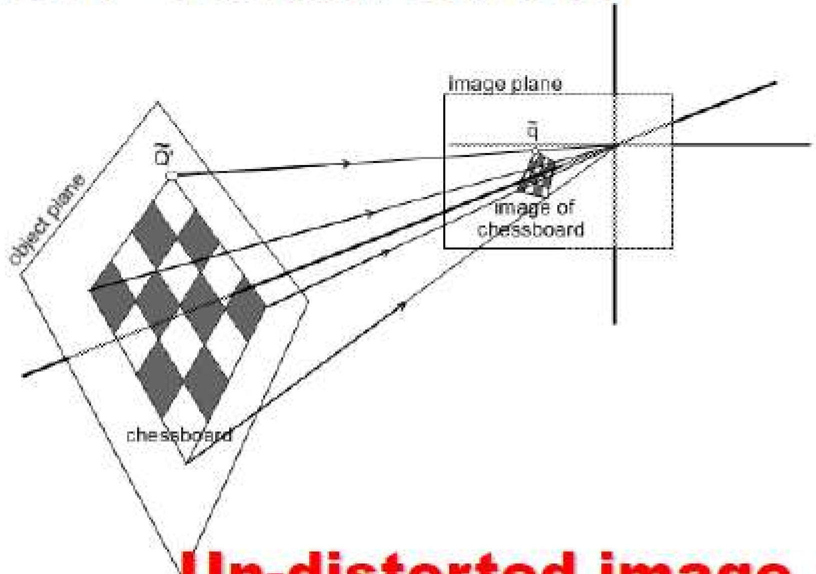


Calibration

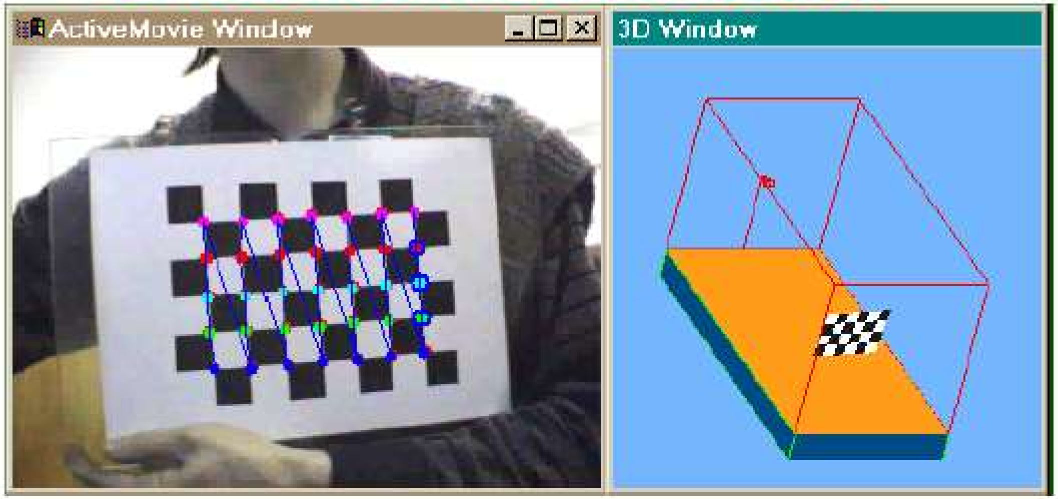
OpenCV Modules: Calibration

Homography





3D view of checkerboard



Un-distorted image

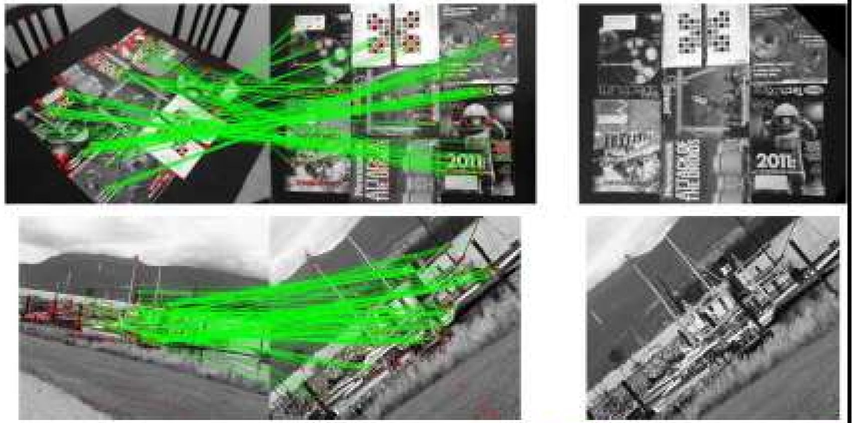
24

http://www.youtube.com/watch?v=DrXIQfQHFv0

http://www.youtube.com/watch?v=PuWQnCReIeE

OpenCV Modules: Features, VSLAM

Features
VSLAM



Read two input images:

```
Mat img1 = imread(argv[1], CV_LOAD_IMAGE_GRAYSCALE);
```

Detect keypoints in both images:

```
// detecting keypoints
FastFeatureDetector detector(15);
vector<KeyPoint> keypoints1;
detector.detect(img1, keypoints1);
```

Compute descriptors for each of the keypoints:

```
// computing descriptors
SurfDescriptorExtractor extractor;
Mat descriptors1;
extractor.compute(img1, keypoints1, descriptors1);
```

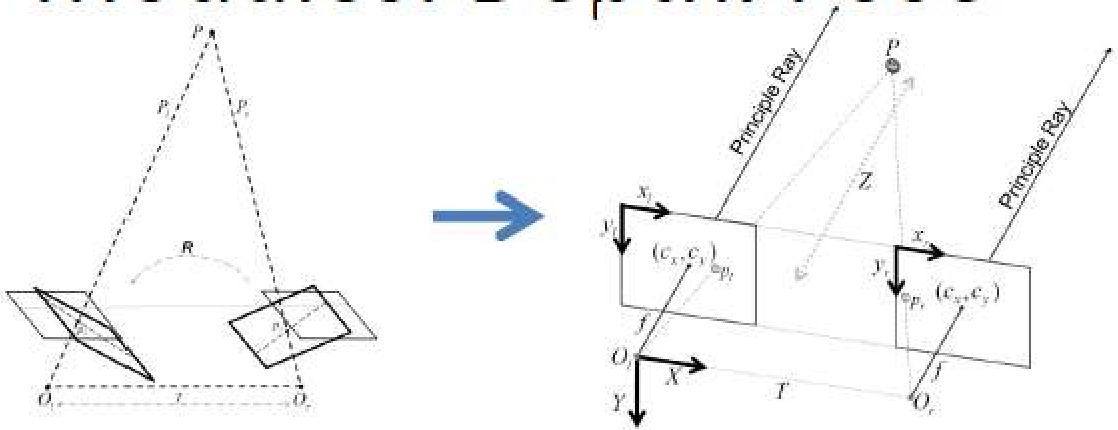
Now, find the closest matches between descriptors from the first image to the second:

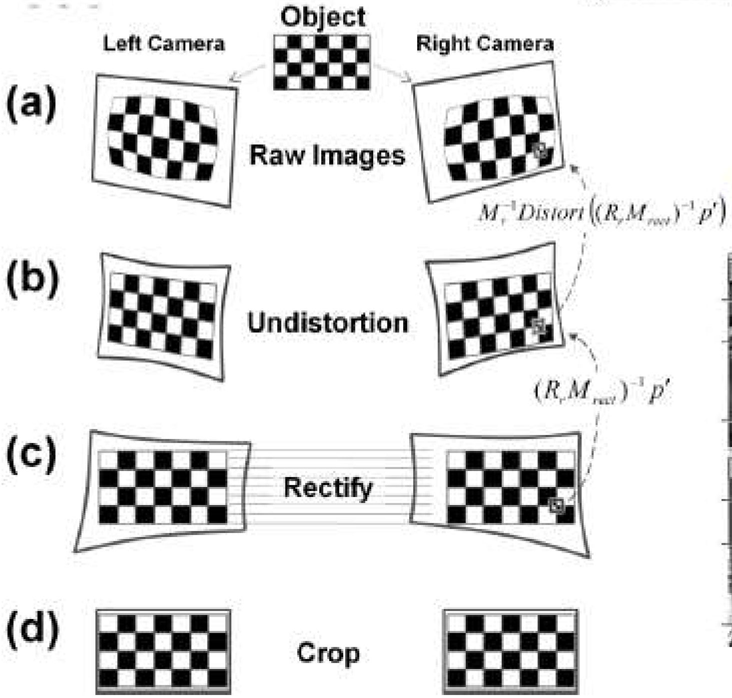

```
// matching descriptors
BruteForceMatcher<L2<float> > matcher;
vector<DMatch> matches;
matcher.match(descriptors1, descriptors2, matches);
```

Change one or both of these lines to switch detector and/or descriptor types

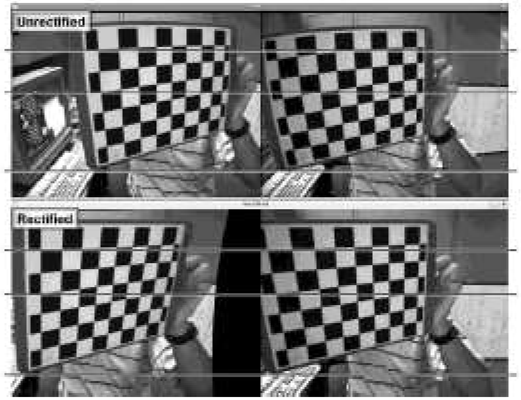
Depth, Pose
Normals, Planes,
3D Features

OpenCV Modules: Depth. Pose

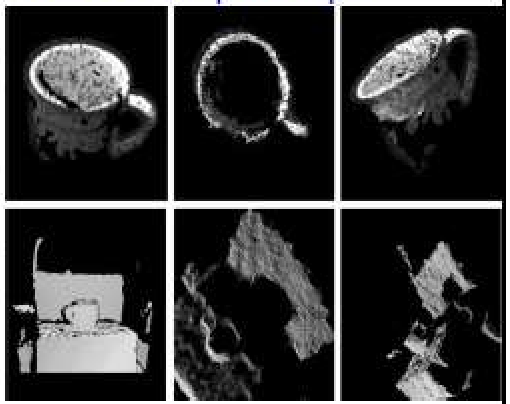




Left – right feature alignment:



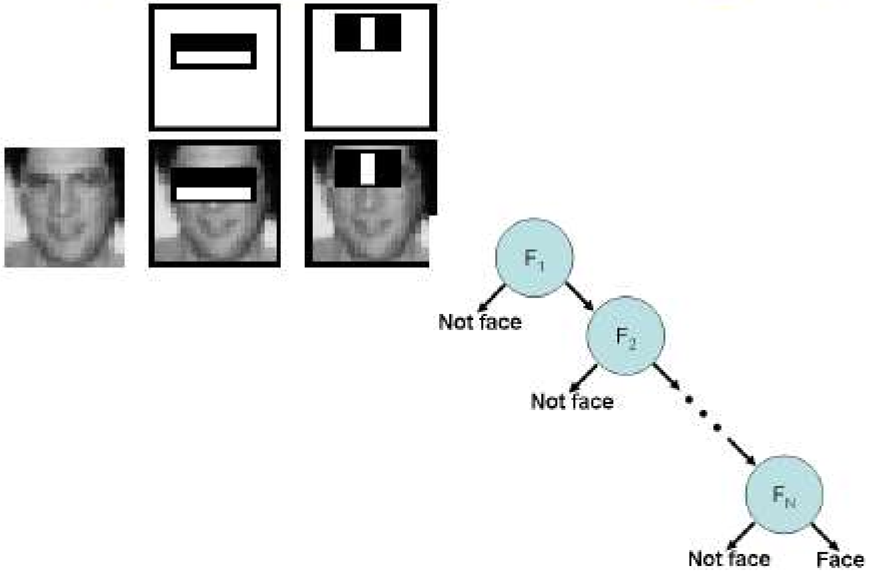
Some examples of
3D stereo depth maps:

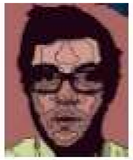


OpenCV Modules: Obj Rec/ML

https://www.youtube.com/watch?v=_RF0VpR4xog

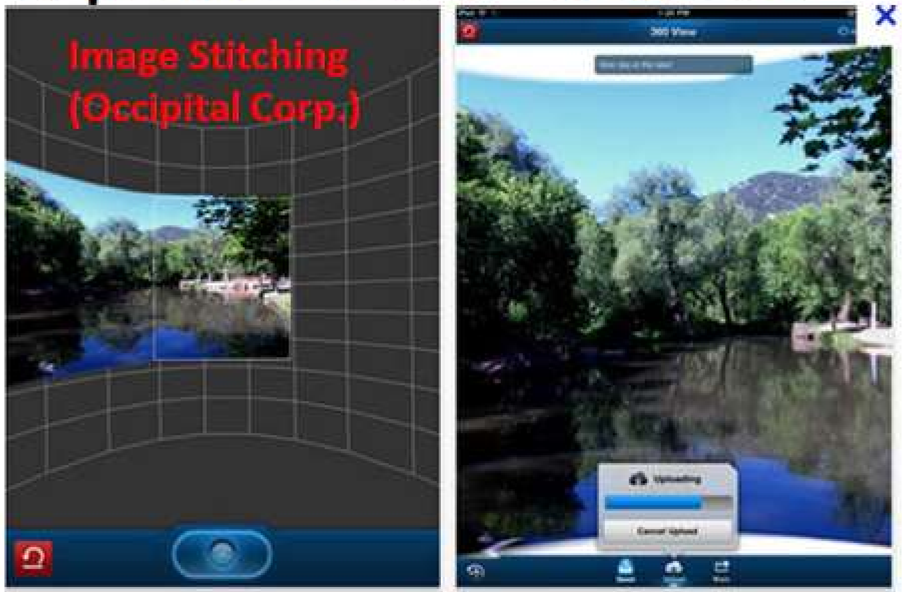
http://youtu.be/i1uUuWwbIcc





Computational Photography

OpenCV Modules: Comp Photog



OpenCV GPU Module:

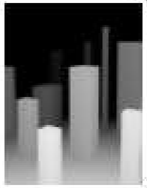
- Image processing building blocks:

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

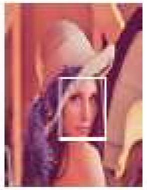
Template matching | Filtering engine | Feature detectors

- High-level algorithms:

Stereo matching



Face detection



Feature matching

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!

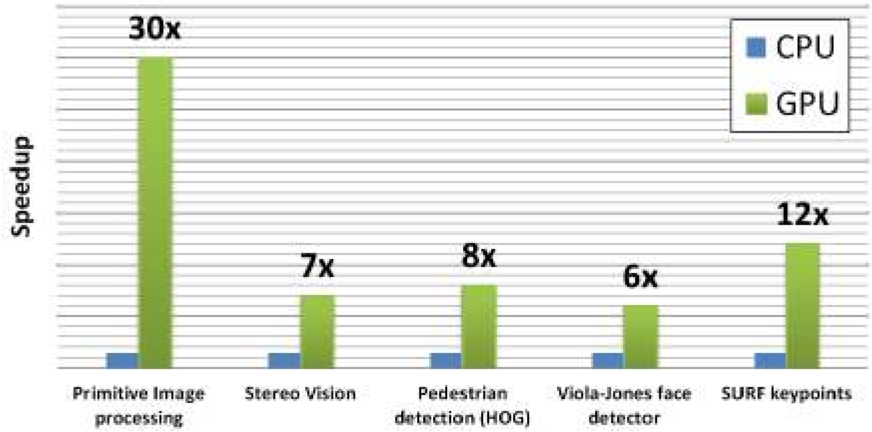
OpenCV GPU Module Performance

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\perfomance



7

OpenCV: Nvidia Drive PX – Tegra + CUDA + Deep Learning

Google Summer of Code 2013

- Google Summer of Code Page: http://www.google-melange.com/gsoc/org/google/gsoc2013/opencv
- Our ideas page: http://code.opencv.org/projects/gsoc2013/wiki

FUTURE

- Contribution based
- 3.0
- OpenVX (Khronos)
- Learning OpenCV V2.0
- Foundation

OpenCV Timeline

Version	Released	Reason	Lifetime
pre 1.0	2000 (first alpha)	-	6 years
1.0	2006 (ChangeLog)	maturity	3 years
2.0	2009 (ChangeLog)	C++ API	>3 years
3.0	**2013?**	several (next level maturity, ...)	

OpenCV 2.x is 3.5-year old already, time to bump the version number!

Dropping old skin

- OpenCV 1.x: C API
- OpenCV 2.x: new C++ API + fully supported C API. It's quite a burden!
- OpenCV 3.0:
  - refined C++ API + officially deprecated C API in a separate module(s)
  - no old-style Python bindings
  - cleaned documentation (just new-style API)
  - even a few wrong things from 2.x C++ API will be corrected or deprecated

  (*no way we could do that in 2.5!*)

Emphasis on binaries

- For a long time OpenCV principles were:
  - Source-level compatibility
  - "Build it yourself!"
- Binary compatibility in 2.4.x
- In OpenCV 3.0 we continue the trend:
  - provide high-quality binary packages for each major platform => easier to maintain, more convenient for users
  - maintain binary compatibility for years!

The HAL + Accelerators

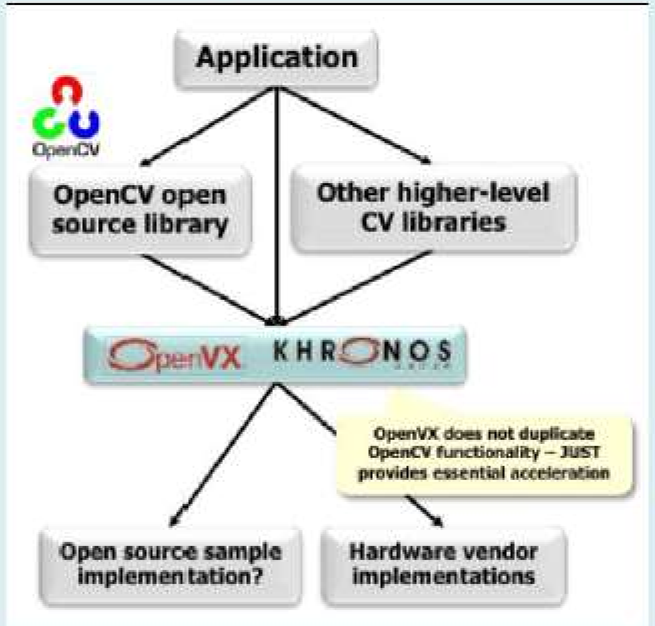
- `opencv_hal` - IPP-like, fastcv-like low-level API to accelerate OpenCV for different platforms.
- `opencv_ocl` module (OpenCL acceleration) will be universal (any SDK) and the binary will be shipped within official OpenCV packages.
- Possible universal `Mat` (`vMat`, `xMat` …?) structure instead of existing `cv::Mat`, `GpuMat`, `OclMat`.
- Preliminary OpenVX support?

khronos 發表電腦視覺API 標準：**OpenVX**. 在2011 年的時候，Khronos 曾經發表過一個名為「Vision」的API 標準，希望可以為電腦視覺（Computer Vision）的處理、定義一套標準的介面，作為硬體加速的抽象層；當時，基本上只是剛開始的階段，並沒有完整的介面出來。2014年10月21日

New Functionality

- RGBD – processing data from depth sensors
- Wrappers for bundle adjustment engines (libmv, ceres …)
- Viz – VTK-based visualization
- Numerical optimization
- New denoising algorithms
- Text detection, barcode readers
- Python 3.0 bindings
- Matlab bindings

OpenVX (Khronos HAL)



khronos 發表電腦視覺API 標準：**OpenVX**.
在2011 年的時候，Khronos 曾經發表過一個名為「Vision」的API 標準，希望可以為電腦視覺（Computer Vision）的處理、定義一套標準的介面，作為硬體加速的抽象層；當時，基本上只是剛開始的階段，並沒有完整的介面出來。2014年10月21日

OpenVX

- Khronos group - OpenVX:
  - Connecting software to silicon
  - OpenVX is an open, royalty-free standard for cross platform acceleration of computer vision applications.
  - It is designed by the Khronos Group to facilitate portable, optimized and power-efficient processing of methods for vision algorithms.
  - This is aimed for embedded and real-time programs within computer vision and related scenarios. It uses a connected graphics representation of operations.

Computer Vision: Algorithms and Applications

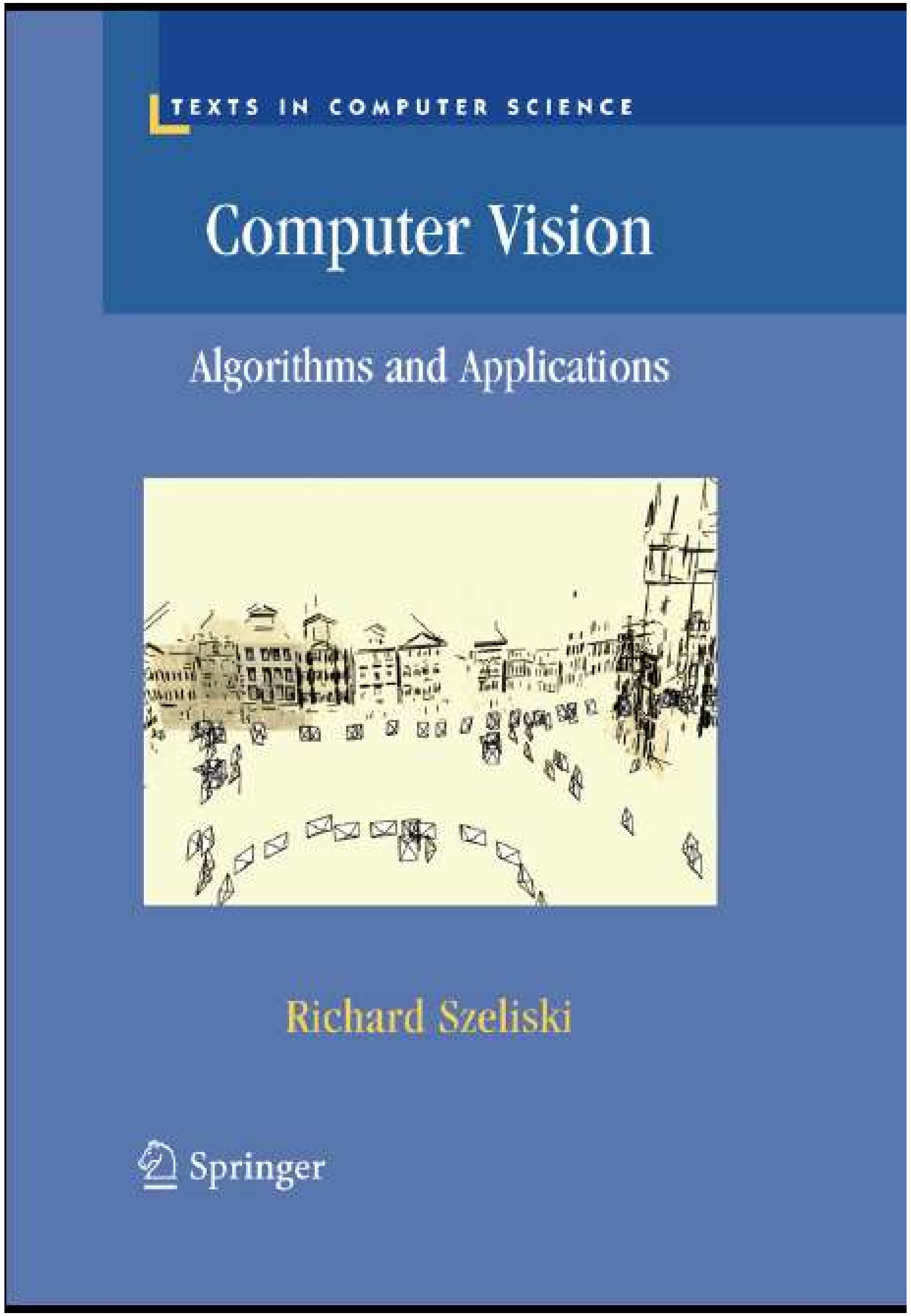
R. Szeliski, *Computer Vision: Algorithms and Applications*, Springer, 2010.

ISBN-10: 1848829345 or ISBN-13: 978-1848829343

Jenn-Jier James Lien (連震杰)
Professor

Computer Science and Information Engineering
National Cheng Kung University

(O) (06) 2757575 ext. 62540
jjlien@csie.ncku.edu.tw
http://robotics.csie.ncku.edu.tw



Content (1/2)

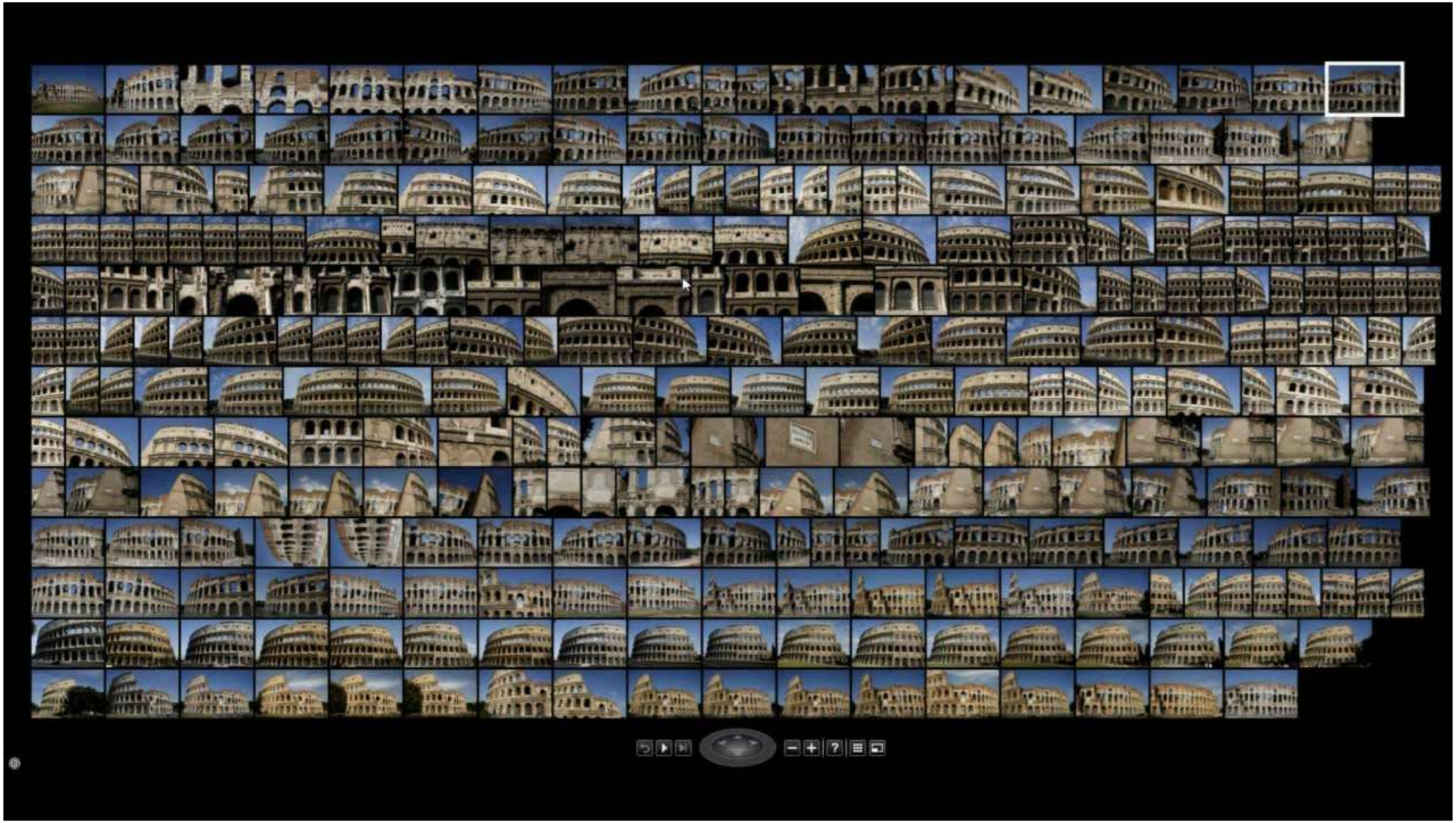
Content (2/2)

Slow Motion with Panorama (1/3)

[redacted]

Panoramic Image Stitching with Local and Global Registration (2/3)

Panoramic Image Stitching with Local and Global Registration (3/3)



References

1. G. Bradski and A. Kaebler, *Learning OpenCV, Computer Vision with the OpenCV Library*, O'Reilly, 2008. ISBN-10: 0596516134 or ISBN-13: 978-0596516130.
2. R. Szeliski, *Computer Vision: Algorithms and Applications*, Springer, 2010. ISBN-10: 1848829345 or ISBN-13: 978-1848829343.