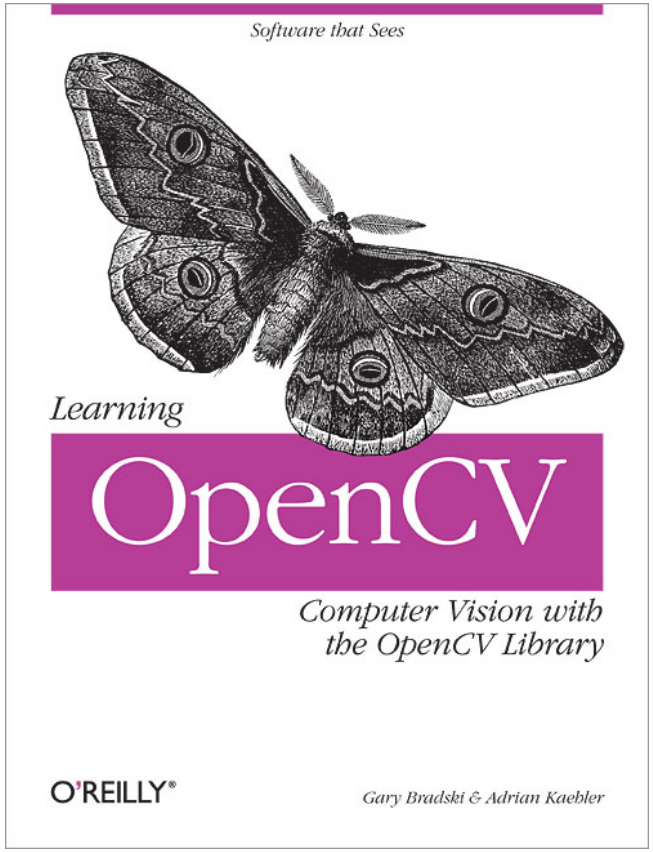
Visão Computacional
Aula 02

Ferramenta Computacional - OpenCV (Noções Básicas)

OpenCV


Software that Sees
Learning
OpenCV
Computer Vision with the OpenCV Library
O'REILLY®
Gary Bradski & Adrian Kaebler


Open Source
Computer Vision Library
intel

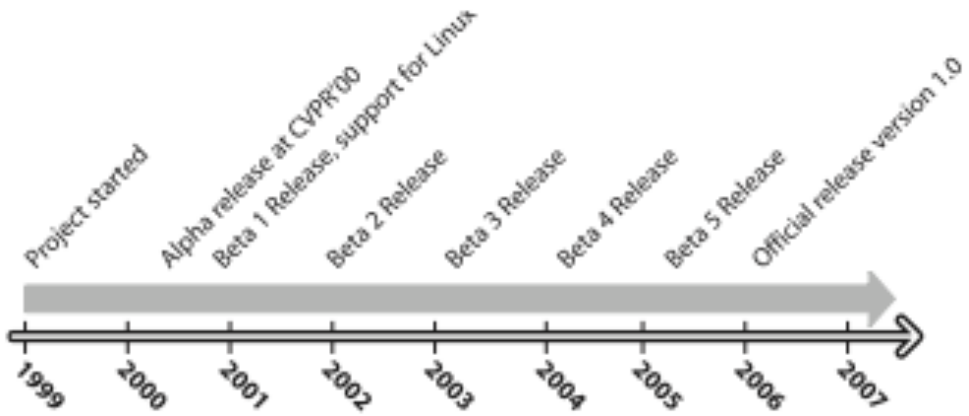
OpenCV

- Open source computer vision library
  - Infra-estrutura de visão computacional de fácil uso
  - Aplicações sofisticadas de CV rapidamente construídas
  - Otimizada para tempo real
  - Multi-plataforma, C/C++
  - APIs low-level e high-level

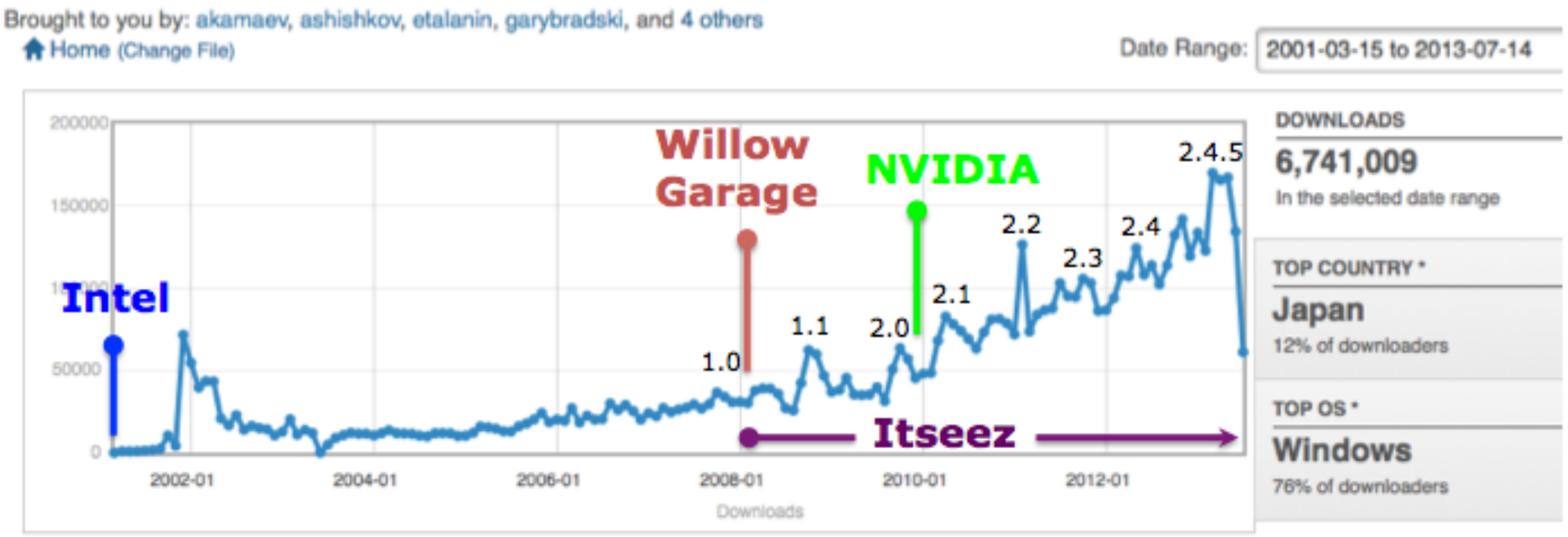
OpenCV - Contexto

- 99/2000 (Primeira versão disponível)
- Intel Research Initiative
- Aplicações de uso intensivo da CPU
  - Real-time ray tracing
- MIT Media Lab
  - Código passado de mão em mão

Linha Cronológica



Linha Cronológica (# Downloads)



OpenCV

- Infra-estrutura de visão computacional largamente disponível
  - Core de código implementado
  - Especificações de algoritmos
    - Intel Russia: gerenciou, codificou, otimizou

OpenCV - Objetivos

- Avançar a pesquisa em CV
  - Prover código aberto e otimizado
  - Não reiventar a roda
- Disseminar o conhecimento de CV
  - Infra-estrutura comum
  - Código prontamente legível e transferível
- Avançar aplicações comerciais em CV
  - Código portável, otimizado e gratuito
  - Licença BSD Intel

OpenCV - Objetivos

- Crescimento da área de CV
  - O crescimento das aplicações exigiria processadores mais rápidos
  - Mais lucro para a Intel

OpenCV

- C/C++, Python, Visual Basic, Ruby, Matlab
- Linux (POSIX, ROS), Windows, Mac OS X e principais plataformas móveis.

OpenCV

- Eficiência Computacional
  - Foco em tempo real, C otimizado, processadores multicore
  - Mais otimização: Intel's IPP
    - Integrated Performance Primitives
    - Algoritmos de baixo nível otimizados

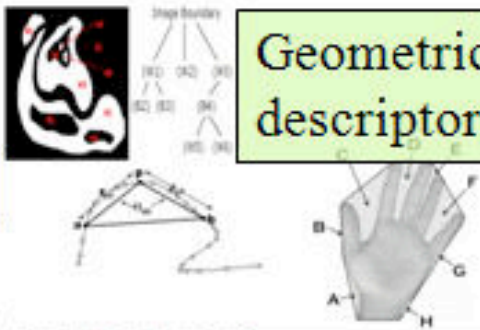
OpenCV Overview: > 500 functions

opencv.willowgarage.com

General Image Processing Functions

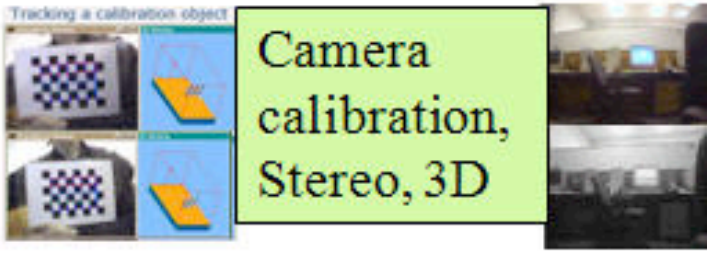
Image Pyramids

Geometric descriptors

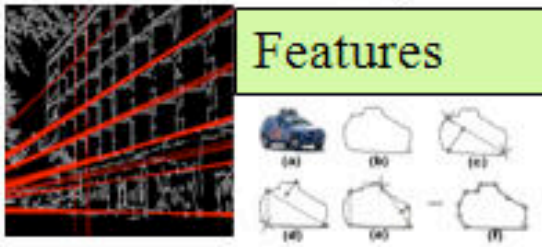


Segmentation

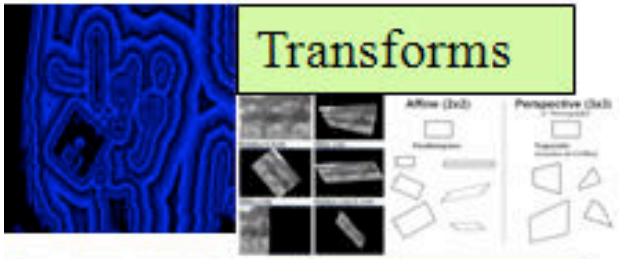
Camera calibration, Stereo, 3D



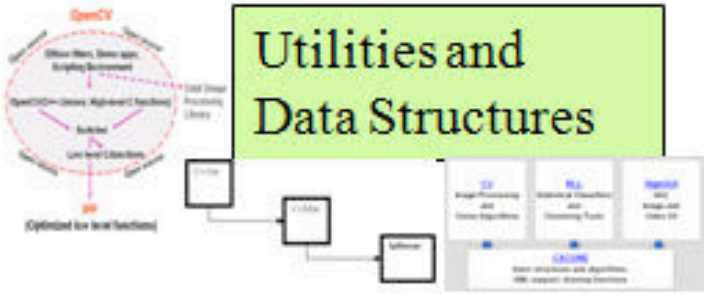
Features



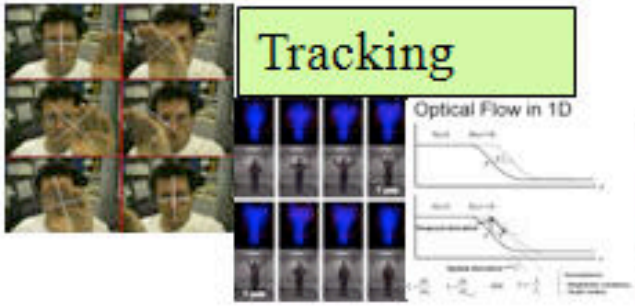
Transforms



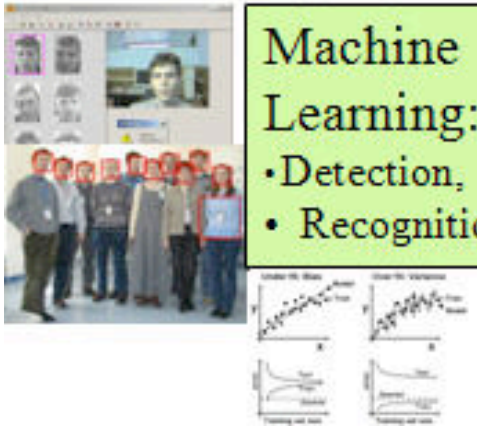
Utilities and Data Structures



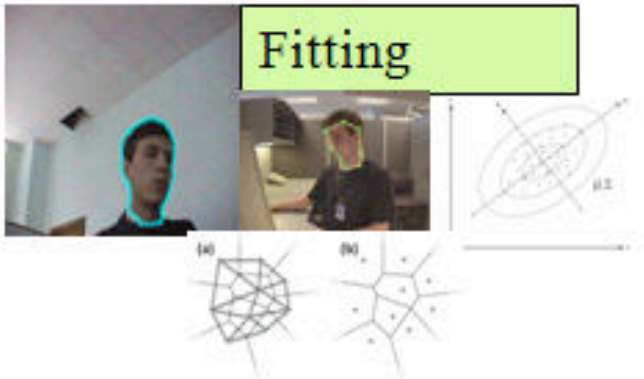
Tracking



Machine Learning:
- Detection,
- Recognition



Fitting



Matrix Math

OpenCV - Funcionalidades

- Manipulação de dados de imagens
- E/S de imagem e vídeo
- Manipulação de matrizes e vetores
- Rotinas de álgebra linear
- Estruturas de dados dinâmicas
- Processamento de imagem básico

OpenCV - Funcionalidades

- Análise estrutural
- Calibragem de câmera
- Análise de movimento (**tracking**)
- Reconhecimento de objetos
- GUI Básica
- Rotulagem de imagem

OpenCV - Módulos

- cxcore
  - Estruturas de dados e álgebra linear
  - Transformadas de dados, persistência de objetos, gerenciamento de memória, manipulação de erros, carregamento dinâmico de código
  - Desenho, texto, matemática básica
- cv
  - Processamento de imagem, análise de estrutura, movimento e tracking, reconhecimento de padrões, calibragem de câmera (em C)

OpenCV - Módulos

- cvcam
  - Interface de câmera
- cvaux
  - Eigen objects (técnica de reconhecimento), gestures, contorno de regiões, matching de formas, descritores de texturas, tracking de olhos e bocas, descoberta de esqueletos, segmentação de background-foreground, calibragem de câmera (em C++)
  - Alguns migrarão para cv, outros, não

OpenCV - Módulos

- HighGUI
  - Interface de usuário
  - Armazenamento e chamada de imagem/vídeo
- ml
  - Aprendizagem de máquina
  - Clustering, classificação e análise de dados
  - Suficientemente genérica

OpenCV – Quem usa?

- IBM, Microsoft, Intel, Sony, Siemens, Google
- Stanford, MIT, CMU, Cambridge, INRIA
- Yahoo Groups: 20,000 membros
  - China, Japão, Rússia, Europa, Israel

- Estabilidade (?)
- Informações: http://opencv.willowgarage.com/wiki/Welcome

OpenCV

- Câmeras de vigilância, imagens e vídeo na web, interfaces de jogos, imagens aéreas, monitoramento de segurança, veículos não-tripulados, análises biomédicas, inspeção automática de produção, robótica

OpenCV – Hands on!!

- Diversos Exemplos podem ser encontrados na Web
- Criando simples aplicações...

Como é uma imagem digital?

But the camera sees this:

194	210	201	212	199	213	215	195	178	158	182	209
180	189	190	221	209	205	191	167	147	115	129	163
114	126	140	188	176	165	152	140	170	106	78	88
87	103	115	154	143	142	149	153	173	101	57	57
102	112	106	131	122	138	152	147	128	84	58	66
94	95	79	104	105	124	129	113	107	87	69	67
68	71	69	98	89	92	98	95	89	88	76	67
41	56	68	99	63	45	60	82	58	76	74	65
20	41	69	75	56	41	51	73	55	70	63	44
50	50	57	69	75	75	73	74	53	68	59	37
72	59	53	66	84	92	84	74	57	72	63	42
67	61	58	65	75	78	76	73	59	75	69	50

Como o OpenCV trata uma imagem?

IplImage

Comments from the Wiki

IplImage

IPL image header

```
typedef struct _IplImage
{
    int  nSize;
    int  ID;
    int  nChannels;
    int  alphaChannel;
    int  depth;
    char colorModel[4];
    char channelSeq[4];
    int  dataOrder;
    int  origin;
    int  align;
    int  width;
    int  height;
    struct _IplROI *roi;
    struct _IplImage *maskROI;
    void  *imageId;
    struct _IplTileInfo *tileInfo;
    int  imageSize;
    char *imageData;
    int  widthStep;
    int  BorderMode[4];
    int  BorderConst[4];
    char *imageDataOrigin;
}
IplImage;
```

OpenCV – Exemplo 01

- Abrir uma imagem:

```
#include "highgui.h"
#include "cv.h"
int main( int argc, char** argv ) {
  IplImage* img = cvLoadImage( "fruits.jpg",
  CV_LOAD_IMAGE_COLOR);
  cvNamedWindow( "Example1", CV_WINDOW_AUTOSIZE );
  cvShowImage( "Example1", img );
  cvWaitKey(0);
  cvReleaseImage( &img );
  cvDestroyWindow( "Example1" );
}
```

Obs.: Diretivas de Compilação

- Compilação automatizada build_all.sh (linux)
  - Estrutura Básica – “Arquivos .c”

```
#!/bin/sh
if [[ $# > 0 ]] ; then
  base=`basename $1 .c`
  echo "compiling $base"gcc -ggdb `pkg-config opencv --cflags
  --libs` $base.c -o $base
else
  for i in *.c; do
    echo "compiling $i"
    gcc -ggdb `pkg-config --cflags opencv` -o `basename
  $i .c` $i `pkg-config --libs opencv`;
  done
fi
```

OpenCV – Exemplo 02

- Capturando imagens (“frame”) da câmera:

```
#include "highgui.h"
#include "cv.h"
#include "stdio.h"
 int main( int argc, char **argv )
    {
        CvCapture *capture = 0;
        IplImage *frame = 0;
        int key = 0;

        cvNamedWindow("Minha Webcam", 1);
        capture = cvCaptureFromCAM(-1);
```

OpenCV – Exemplo 02

- Capturando imagens (“frame”) da câmera (Cont.):

```
        while(key != 'q') {
            frame = cvQueryFrame(capture);
            cvShowImage("Minha Webcam", frame);
            key = cvWaitKey(1);
        }
        cvReleaseCapture(&capture);
        cvDestroyWindow("Minha Webcam");

        return 0;
    }
```

OpenCV - Exemplo 03

- Exibindo um arquivo de vídeo:

```
#include "cv.h"
#include "cvcam.h"
#include "highgui.h"
int main(int argc, CHAR* argv[]){
  cvNamedWindow( "Example3", CV_WINDOW_AUTOSIZE );
  CvCapture* capture =
  cvCreateFileCapture("video.avi");
  IplImage* frame;
```

OpenCV – Exemplo 03

- Exibindo um arquivo de vídeo (Cont.):

```
while(1) {
  frame = cvQueryFrame( capture );
      if( !frame )
      break;
  cvShowImage( "Example2", frame );
  char c = cvWaitKey(33);
  if( c == 27 )
      break;
  }
  cvReleaseCapture( &capture );
  cvDestroyWindow( "Example2" );
  return 0;
}
```

OpenCV - Exemplo 04

- Fazendo as coisas “rapidamente”... Em C++

```
int main(int argc, char* argv[])    {
      Mat img, gray;
      img = imread("lena.jpg",1);
      imshow("Original", img);
      cvtColor(img,gray, COLOR_BGR2GRAY);
      GaussianBlur(gray, gray, Size(7,7), 1.5);
      Canny(gray, gray, 0, 50);
      imshow("Bordas", gray);
       waitKey();
      return 0;
}
```

C vs. C++ (um comparativo)

C

```
double calcGradients(const IplImage *src,
                     int aperture_size = 7)
{
  CvSize sz = cvGetSize(src);

  IplImage* img16_x = cvCreateImage(sz, IPL_DEPTH_16S, 1);
  IplImage* img16_y = cvCreateImage(sz, IPL_DEPTH_16S, 1);
  cvSobel(src, img16_x, 1, 0, aperture_size);
  cvSobel(src, img16_y, 0, 1, aperture_size);

  IplImage* imgF_x = cvCreateImage(sz, IPL_DEPTH_32F, 1);
  IplImage* imgF_y = cvCreateImage(sz, IPL_DEPTH_32F, 1);
  cvScale(img16_x, imgF_x);
  cvScale(img16_y, imgF_y);

  IplImage* magnitude = cvCreateImage(sz, IPL_DEPTH_32F, 1);
  cvCartToPolar(imgF_x, imgF_y, magnitude);
  double res = cvSum(magnitude).val[0];

  cvReleaseImage(&magnitude );
  cvReleaseImage(&imgF_x);
  cvReleaseImage(&imgF_y);
  cvReleaseImage(&img16_x);
  cvReleaseImage(&img16_y);

  return res;
}
```

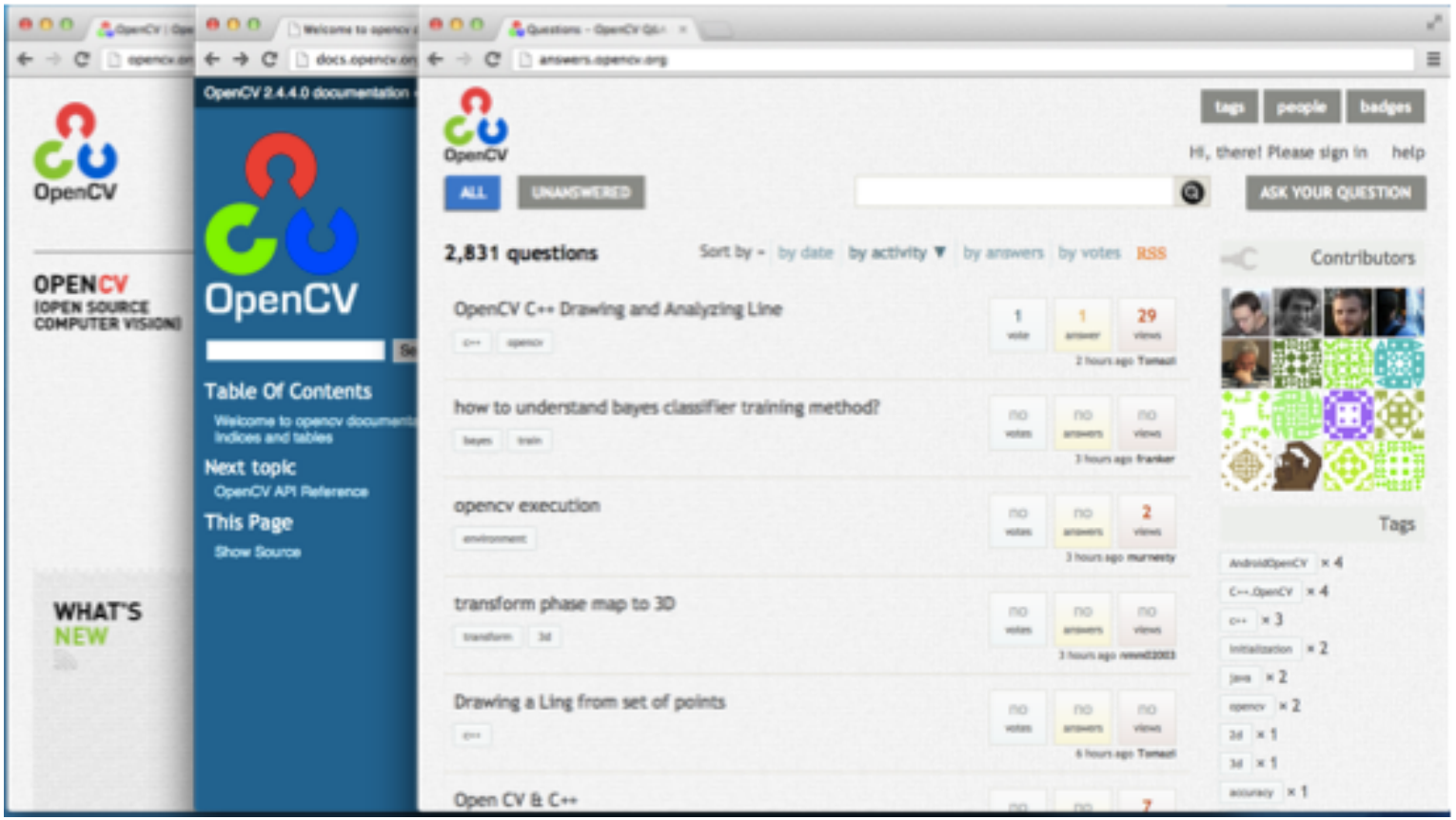
C++

```
double contrast_measure(Mat& img)
{
  Mat dx, dy;

  Sobel(img, dx, 1, 0, 3, CV_32F);
  Sobel(img, dy, 0, 1, 3, CV_32F);
  magnitude(dx, dy, dx);

  return sum(dx)[0];
}
```

Suporte ao OpenCV



Suporte ao OpenCV

- Homepage: http://opencv.org
- Online docs: http://docs.opencv.org
- Q&A forum: http://answers.opencv.org
- Dev zone: http://code.opencv.org

Recomendação

- Exercitar com os tutoriais do OpenCV
  - http://docs.opencv.org/doc/tutorials/tutorials.html

Próxima aula...

- Câmeras ...