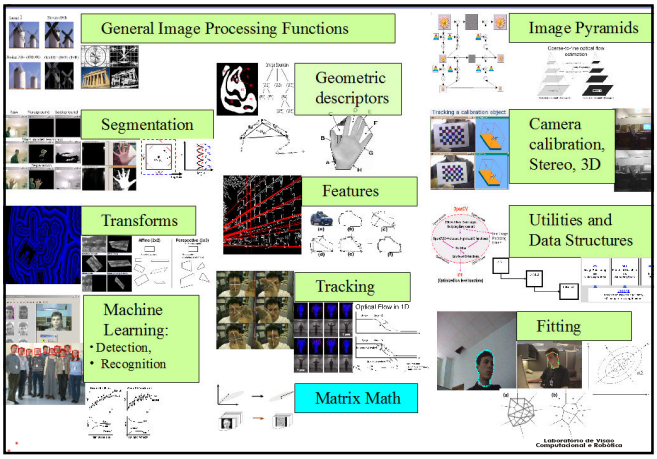
OpenCV
www.opencv.org

Erickson R. Nascimento

erickson@dcc.ufmg.br
www.verlab.dcc.ufmg.br

VERLab
Laboratório de Visão Computacional e Robótica



Características da biblioteca

- Licença BSD
  - Livre para uso comercial e pesquisa
  - Para a maior parte do código
- Quem usa atualmente?
  - Google Maps, Google street view, Google Earth
  - Willow Garage
- Projeto irmão
  - PCL: www.pointclouds.org
  - Manipulação de dados geométricos, e.g. nuvens de pontos 3D, malhas tridimensionais, etc.

Características da biblioteca

- Escrita em C e C++
- Windows, Linux e MacOS
- Android, iOS

- Interfaces para Python e Java

Bibliotecas necessárias

- sudo apt-get install build-essential libgtk2.0-dev libjpeg-dev libtiff4-dev libjasper-dev libopenexr-dev cmake python-dev python-numpy python-tk libtbb-dev libeigen2-dev yasm libfaac-dev libopencore-amrnb-dev libopencore-amrwb-dev libtheora-dev libvorbis-dev libxvidcore-dev libx264-dev libqt4-dev libqt4-opengl-dev sphinx-common texlive-latex-extra libv4l-dev libdc1394-22-dev libavcodec-dev libavformat-dev libswscale-dev

Instalando OpenCV

- sudo apt-get install build-essential
- Cmake
- cd ~/opencv
- mkdir release; cd release
- cmake -D CMAKE_BUILD_TYPE=RELEASE -D CMAKE_INSTALL_PREFIX=/usr/local ..
- Pode usar o ccmake para configurar as opções de instalação também
- make; sudo make install

Instalando OpenCV

- http://miloq.blogspot.com.br/2012/12/install-opencv-ubuntu-linux.html

Hello OpenCV

- CmakeLists.txt

```
PROJECT( helloworld_proj )
FIND_PACKAGE( OpenCV REQUIRED )
ADD_EXECUTABLE( demo demo.cpp )
TARGET_LINK_LIBRARIES( demo ${OpenCV_LIBS} )
```

Hello OpenCV

- hello.c

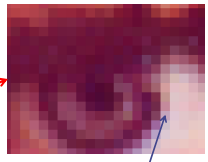
```
#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <iostream>

using namespace cv;
using namespace std;

int main( int argc, char** argv )
{
   int window_width = 640;  int window_height = 480;
   Mat image = Mat::zeros( window_height, window_width, CV_8UC3 );
   string message("Hello OpenCV");
   Point pos_text;  pos_text.x = window_width/2;  pos_text.y = window_height/2;

   putText( image, message, pos_text, 0, 1, Scalar(0,0,255) );
   namedWindow( "Hello", CV_WINDOW_AUTOSIZE );  // Create a window for display.
   imshow( "Hello", image);                     // Show our image inside it.
   waitKey(0);
   return 0;
}
```

Imagens



Pixel = Picture Element

Imagens

We perceive this:

But the camera sees this:

194	210	201	212	199	213	215	195	178	158	182	209
198	189	190	221	209	205	191	167	147	115	129	163
114	126	140	188	176	165	153	140	170	106	78	88
97	103	115	154	143	141	149	152	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	66	99	65	45	68	82	56	76	75	65
49	43	99	75	56	41	91	73	55	70	62	44
50	50	57	69	75	75	73	74	53	68	59	37
72	59	53	66	84	92	94	74	57	72	63	42
67	61	58	65	75	78	76	73	59	75	69	50

Imagens

- Matriz com m linhas, n colunas e geralmente 3 planos de cor: RGB
  - Imagens coloridas
  - Cada um com 256 valores, [0,255]

	Column 0			Column 1			Column ...			Column m		
Row 0	0,0	0,0	0,0	0,1	0,1	0,1	...	...	...	0, m	0, m	0, m
Row 1	1,0	1,0	1,0	1,1	1,1	1,1	...	...	...	1, m	1, m	1, m
Row ...	...,0	...,0	...,0	...,1	...,1	...,1	...	...	...	..., m	..., m	..., m
Row n	n,0	n,0	n,0	n,1	n,1	n,1	n,...	n,...	n,...	n, m	n, m	n, m

Imagens

- Matriz com m linhas, n colunas de inteiros (intensidades)
  - Imagens em tons de cinza (grayscale)
  - Cada um com 256 valores, [0,255]

	Column 0	Column 1	Column ...	Column m
Row 0	0,0	0,1	...	0, m
Row 1	1,0	1,1	...	1, m
Row ...	...,0	...,1	...	..., m
Row n	n,0	n,1	n,...	n, m

Carregando e salvando uma imagem

- Mat image;
- image = imread("lena.jpg", CV_LOAD_IMAGE_COLOR);
  - CV_LOAD_IMAGE_COLOR: carrega imagem com cor
  - CV_LOAD_IMAGE_GRAYSCALE: carrega imagem em tons de cinza
  - CV_LOAD_IMAGE_UNCHANGED: lê os dados como estão armazenados. Útil para ler imagem de profundidades
- imwrite("processed.jpg", image);
- imshow("Window's name", image);
- waitKey(0);

Region of Interest (ROI)

- Mat image, gray;
- image = imread("lena.jpg", CV_LOAD_IMAGE_COLOR);
- cvtColor(image, gray, CV_BGR2GRAY);
  - CV_BGR2HSV
  - CV_BGR2Lab
  - ...
- Mat ROI(gray, Rect(100, 100, 300, 300));
- Mat img2;
- img2 = ROI; // irá modificar o conteúdo de image
- img2 = ROI.clone(); // não modifica o conteúdo de image

Matrizes e Imagens

- Uma imagem é uma matriz
- Criando uma matriz 480 x 640 x 3
  - Mat img = Mat(480, 640, CV_8UC3);
  - Cada elemento da matriz contém um byte
- Acessando elementos
  - Vec3b color = img.at<Vec3b>(y, x);
  - uchar blue = color.val[0];
  - uchar green = color.val[1];
  - uchar red = color.val[2];

Reshape

- cv::Mat M(7, 7, CV_32FC2, Scalar(1,3));
  - 7x7 2 canais de floats de 32bits preenchido com 1 e 3.
- M.create(100, 60, CV_8UC(15));
  - M agora tem 100x60 com 15 canais cada um com 8 bits. A matriz anterior é desalocada.

Conversão byte para float

- Mat img = Mat(480, 640, CV_8UC3);
- Mat gray;
- cvtColor(img, grey, CV_BGR2GRAY);
- Conversão 8UC1 to 32FC1:
  - gray.convertTo(float_gray, CV_32F);

Operações com matrizes

- Matriz identidade:
  - Mat E = Mat::eye(4, 4, CV_64F);
- Inicialização:
  - Mat C = (Mat_<double>(3,3) << 0, -1, 0,
  - -1, 5, -1,
  - 0, -1, 0);
- Copiando uma linha:
  - Mat RowClone = C.row(1).clone();
- Copiando uma coluna
  - Mat ColClone = C.col(2).clone();

Operações com matrizes

- Mat Row = C.row(1);
- Row.at<double>(0,1) = 3.14;
- M.row(3) = M.row(3) + M.row(5)*3;
  - Multiplica os valores da linha 5 por 3 e soma com os valores da linha 3. O resultado é armazenado na linha 3

Persistência de dados (escrita)

- FileStorage fs("test.yml", FileStorage::WRITE);
- fs << "Matrix" << C;
- fs << "Row" << Row;
- fs << "files" << "[";
- fs << "image1.jpg" << "chair.jpg" << "desk.jpg";
- fs << "]";

Persistência de dados (leitura)

```
FileStorage fs("test.yml", FileStorage::READ);
cv::Mat m;
fs["Matrix"] >> m;
cv::Mat r;
fs["Row"] >> r;
FileNode n = fs["files"];
if (n.type() != FileNode::SEQ)
    cerr << "strings is not a sequence! FAIL" << endl;
} else {
    cout << "Files: " << endl;
    FileNodeIterator it = n.begin(), it_end = n.end();
    for (; it != it_end; ++it)
        cout << (string)*it << endl;
}
```

Outros tipos de dados

- Point2f, Point2d, Point2i
- Point3f, Point3d, Point3i
  - pt1 = pt2 + pt3; pt1 = pt2 - pt3;
  - pt1 = pt2 * a; pt1 = a * pt2;
  - pt1 += pt2; pt1 -= pt2;
  - pt1 *= a;
  - double value = norm(pt); // L2 norm
  - pt1 == pt2;
  - pt1 != pt2;

Outros tipos de dados

- Vec3b, Vec3f, Vec2b, etc.
  - v1 = v2 + v3
  - v1 = v2 - v3
  - v1 = v2 * scale
  - v1 = scale * v2
  - v1 = -v2
  - v1 += v2
  - v1 == v2, v1 != v2
  - norm(v1)

Outros tipos de dados

- Scalar: vetor com 4 elementos do tipo double
  - Usado para passar valores de pixels:
    - RED, GREEN, BLUE e ALPHA
  - Scalar(0)
  - Scalar(0,255,0)
  - Scalar::all(0)

Templates em C++

- Usar templates torna possível criar funções e classes genéricas.
- using namespace std;
- vector, set;

Templates em C++

- Suponha uma função para encontrar o máximo entre dois inteiros:

```
int max(int a, int b) {
    if (a > b)
        return a;
    return b;
}
```

Templates em C++

- Como seria para o caso de floats?

```
float max(float a, float b) {
    if (a > b)
        return a;
    return b;
}
```

Templates em C++

- E para uchar?
- Templates torna a vida mais fácil para casos genéricos

```
template <class X> X max(X a, X b){
    if (a > b) return a;
    return b
}
```

Templates em C++

- int a, b, im;
- a = 2; b = 3;
- im = max(a, b);

- float fa, fb, fm;
- fa = 3.14; fb = 2.17;
- fm = max(fa, fb);

Templates em C++

```
Template <class T>
Class ponto2D {
Private:
        T x;
        T y;
Public:
        void setCoord(T x_, T y_);
}
```

Templates em C++

```
int main(int argc, char **argv) {
        ponto2D<float> fp;

        fp.setCoord(3.14, 2.17);

        ponto2D<int> ip;
        ip.setCoord(1, 2);
}
```

Templates em C++

- Standard Template Library (STL)
  - vector
  - set
  - map
- #include<vector>
- std::vector<int> iv;
- std::vector<Mat> mV;

Templates em C++

- Vector:
  - size()
  - push_back()
    - im.push_back(1);
  - pop_back()
  - im[2] = 3;
  - first_element = im[0];
  - clear()
  - empty()

Passagem por referência em C++

```
void foo1(cv::Mat img) {
        img = Mat:zeros(3, 3, CV_32F);
}

void foo2(cv::Mat &img) {
        img = Mat:zeros(3, 3, CV_32F);
}
```

Passagem por referência em C++

```
void main() {
        cv::Mat m = cv::Mat::ones(4, 4, CV_32F);
        foo1(m);
        cout << m << endl;

        foo2(m);
        cout << m << endl;
}
```

Passagem por referência em C++

```
void foo2(cv::Mat &img) {
      img = Mat:zeros(3, 3, CV_32F);
}

void foo3(const cv::Mat &img, cv::Mat &out) {
      ...
}
```

Smart Pointers

- Gerência automática de memória
- int *ip, array_int;
- ip = new int;
- delete ip;

- array_int = new int[10];
- delete [] array_int;

Smart Pointers

- SurfFeatureDetector *detector = new SurfFeatureDetector();
- delete detector;

- SIFT, SURF, FAST
- Star, ORB
- Etc.

Smart Pointers

- **Ptr**<FeatureDetector> detector = FeatureDetector::create("FAST");

- Vector<KeyPoint> keypoints;
- detector->detect(image, keypoints);

- Det = detector;

```
Main(){

if ( param < 0)
{
        Ptr<FeatureDetector> Det2 = det;
}
det->detect(…)
}
```

Medindo tempo na OpenCV

```
double t = (double)getTickCount();
...
t = ((double)getTickCount() - t)/getTickFrequency();

cout << "Tempo em secondos: " << t << endl;
```

Lendo frames de um vídeo

```
#include <opencv\cv.h>
#include <opencv\highgui.h>

using namespace cv;
 void main(){
   Mat image;

   VideoCapture cap;
   cap.open(0);
   namedWindow("VideoCaptureTutorial", 1);
    while(true)   {
      cap>>image;
      imshow("VideoCaptureTutorial", image);
      waitKey(33);
   }
}
```

Lendo frames de um vídeo

- Para capturar de um arquivo de vídeo:
  - VideoCapture cap("C:/Users/Desktop/SampleVideo.avi");
- Do Kinect

```
VideoCapture capture(CV_CAP_OPENNI );
for(;;) {
        Mat depthMap, rgbImage
        capture.grab();
        capture.retrieve( depthMap, OPENNI_DEPTH_MAP );
        capture.retrieve( bgrImage, OPENNI_BGR_IMAGE );
}
```

Facilidades para depuração