Utilização da Biblioteca OpenCV em Projetos de Computação e Robótica

Visão computacional

Duodécimo Fernandes
duodecimo@gmail.com

Introdução ao Opencv

A biblioteca OpenCV (Open Source Computer Vision Library) foi originalmente desenvolvido pela Intel em torno do ano 2000. É uma biblioteca multiplataforma, livre para uso acadêmico e comercial, e evidentemente para o desenvolvimento de aplicativos na área de Visão Computacional. A biblioteca OpenCV possui módulos de processamento de Imagens e video, estrutura de dados, algebra linear, e GUI (Interface Gráfica do Usuário) básica com sistema de janelas independentes.

Mas o que é visão computacional?

Visão computacional é a ciência e tecnologia das máquinas que processam imagens, ou seja, o que podemos definir como a capacidade dos seres humanos de “enchergarem o mundo em que vivem”, ou mesmo o sentido da visão. A Visão computacional procura desenvolver teoria e prática (tecnologia) para a construção de sistemas artificiais que obtém informação de imagens ou quaisquer dados multi-dimensionais.

Como pode ser utilizada

- Identificação de objetos
- Reconhecimento facial
- Reconhecimento de movimentos (gestos)
- Processamento e produção de vídeos
- Aprendizado de Máquina

Instalação

O primeiro passo é instalar alguns pacotes necessários para funcionamento do opencv. Por exemplo, no Ubuntu, (testado no 14.04 LTS) digite em um terminal de texto:

(compilador)
sudo apt-get install build-essential

(pacotes requeridos)
sudo apt-get install cmake git libgtk2.0-dev pkg-config libavcodec-dev libavformat-dev libswscale-dev

(pacotes opcionais)
sudo apt-get install python-dev python-numpy libtbb2 libtbb-dev libjpeg-dev libpng-dev libtiff-dev libjasper-dev libdc1394-22-dev

Em seguida, obter o código fonte do OpenCV (pode ser no sourceforge: http://sourceforge.net/projects/opencvlibrary ou no repositório git: https://github.com/Itseez/opencv.git).

Instalação (continuação)

Se a partir do sourceforge, obtenha o tarball e expanda em uma pasta (por exemplo, obter o arquivo opencv-3.0.0.zip e expandí-lo na pasta Ferramentas).

Vou deixar o exemplo do código do git de fora, mas é para fazer mais ou menos a mesma coisa, obter o código e colocálo em uma pasta, por exemplo, Ferrramentas).

Em um terminal de texto:

```
cd ~/Ferramentas/opencv-3.0.0
mkdir release
cd release
cmake -D CMAKE_BUILD_TYPE=RELEASE -D
    CMAKE_INSTALL_PREFIX=/usr/local ..
make
sudo make install
```

Instalação (continuação 2)

Compilar as bibliotecas Java:
(garanta que o Java está instalado em seu sistema, o ideal é que a variável
JAVA_HOME aponte para sua instalação, como por exemplo:
export JAVA_HOME=/usr/local/java)

Em um terminal de texto:
cd ~/Ferramentas/opencv-3.0.0
mkdir build
cd build
cmake -DBUILD_SHARED_LIBS=OFF ..
make -j8

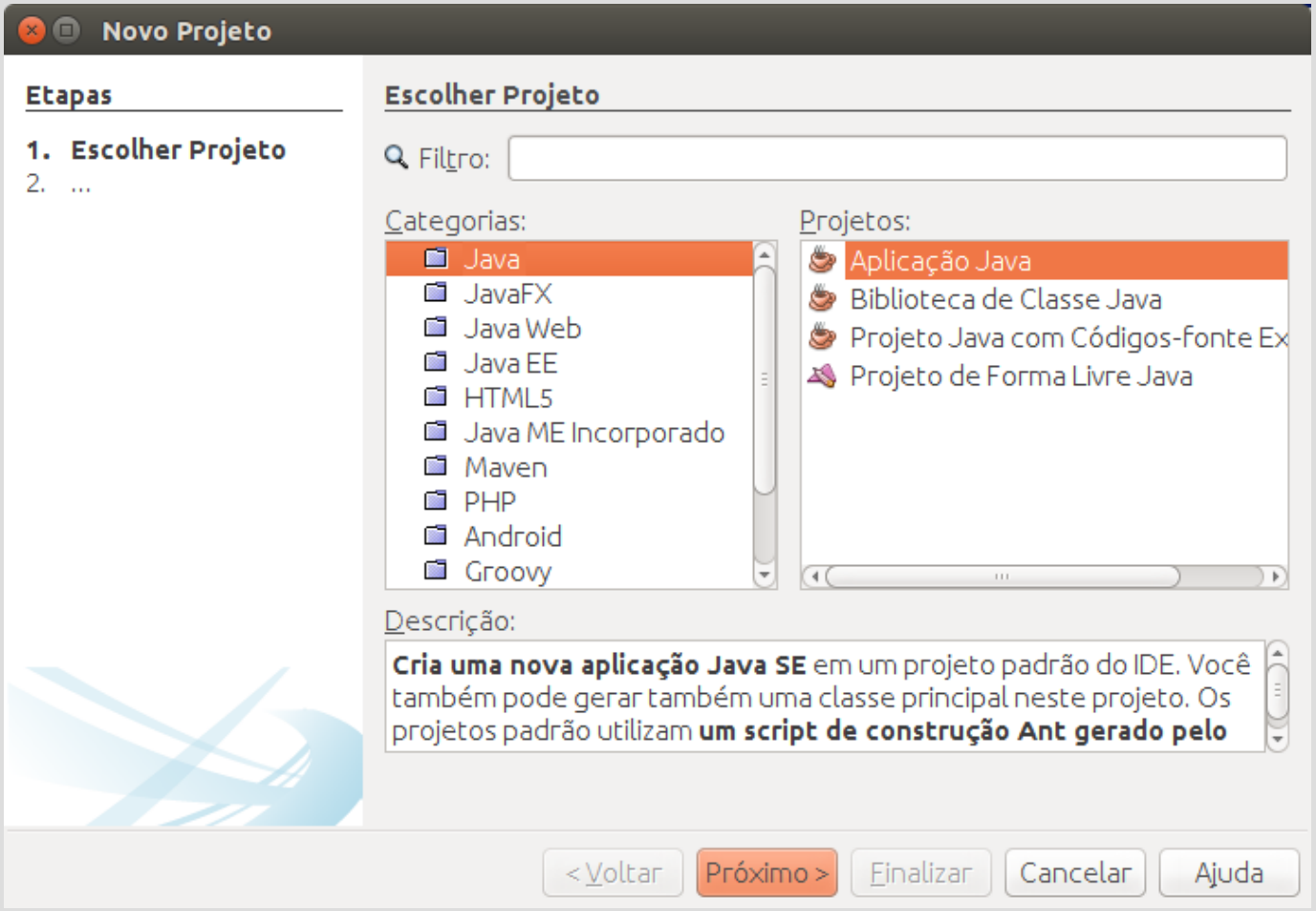
Isto vai gerar dois arquivos fundamentais, dentro da pasta opencv-3.0.0/build:
bin/opencv-300.jar
lib/libopencv_java300.so

Exemplo de uso

Crie um novo projeto no Netbeans IDE (por exemplo, o Netbeans 8.02 completo).
O projeto pode ser da categoria *Java*, *Aplicação Java*.
Chame o projeto de opencvTeste01, use uma pasta dedicada para armazenamento de bibliotecas e nomeie a classe principal opencvteste01.SimpleSample. Veja nas figuras 1 e 2 estes passos.

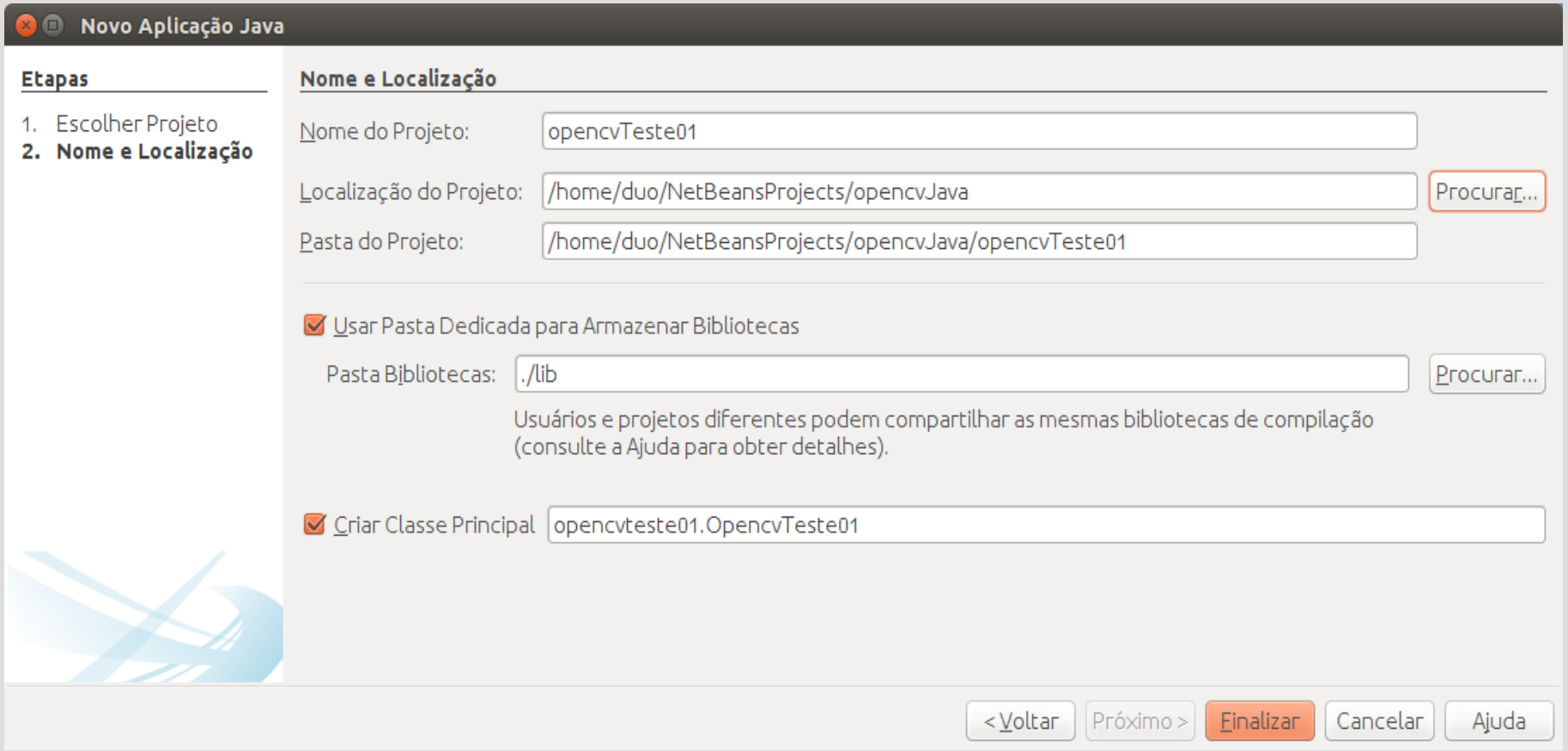
Exemplo de uso (continuação)

Figura 1: Criação de projeto no NetBeans IDE versão 8.02



Exemplo de uso (continuação 2)

Figura 2: Criação de projeto no NetBeans IDE versão 8.02 finalização



Exemplo de uso (continuação 3)

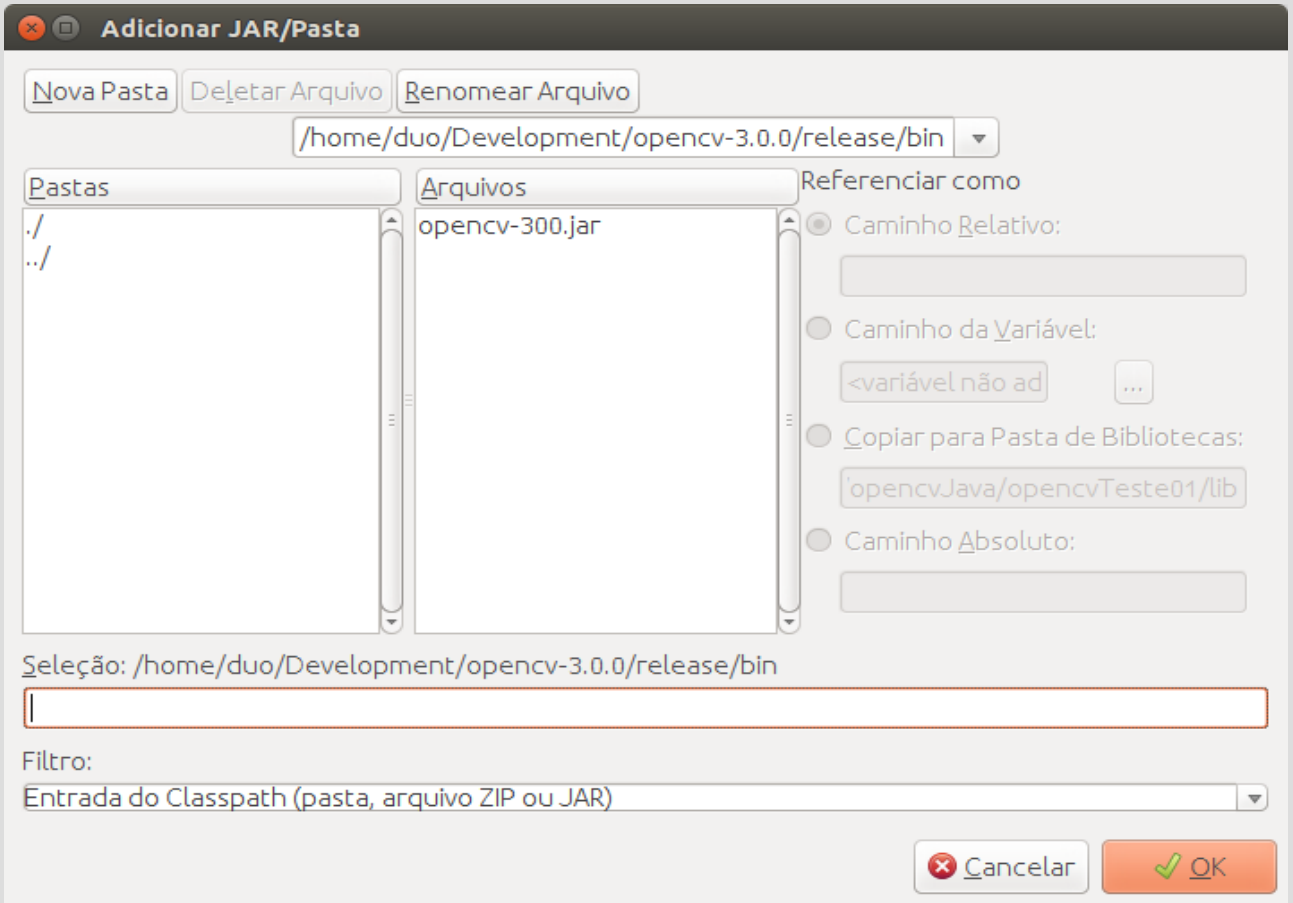
Na aba principal do projeto, clique em biblioteca com o botão direito do mouse, escolha adicionar pasta/arquivo jar, e selecione a biblioteca Opencv-300.jar, conforme mostra a Figura 3 a seguir.

Copie e cole (pode utilizar o navegador de arquivos na pasta de instalação do OpenCV) a biblioteca nativa (libopencv_java300.so) na pasta lib do projeto, pode utilizar a aba de arquivos do projeto para esta tarefa, veja a Figura 4.

Na aba principal do projeto, clique com o botão direito do mouse na raiz do projeto, selecione propriedades, e na janela de edição de propriedades, na categoria *Executar*, adicione na *Opção de VM:* -Djava.library.path=lib, conforme a Figura 5.

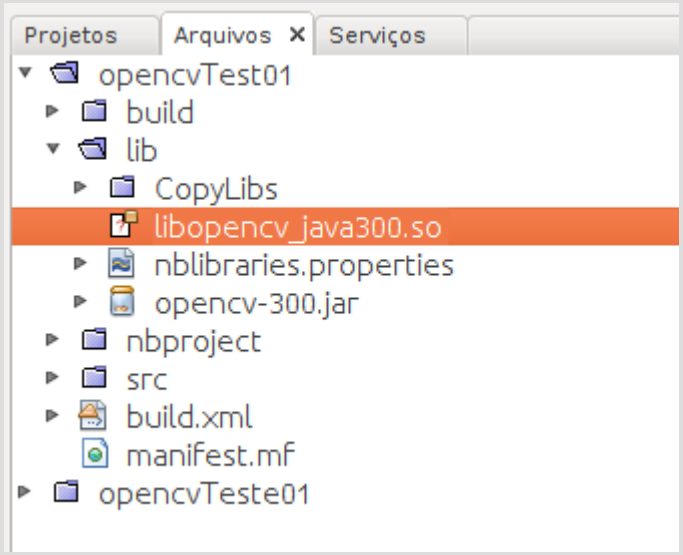
Exemplo de uso (continuação 4)

Figura 3: Adicionar biblioteca Opencv-300.jar



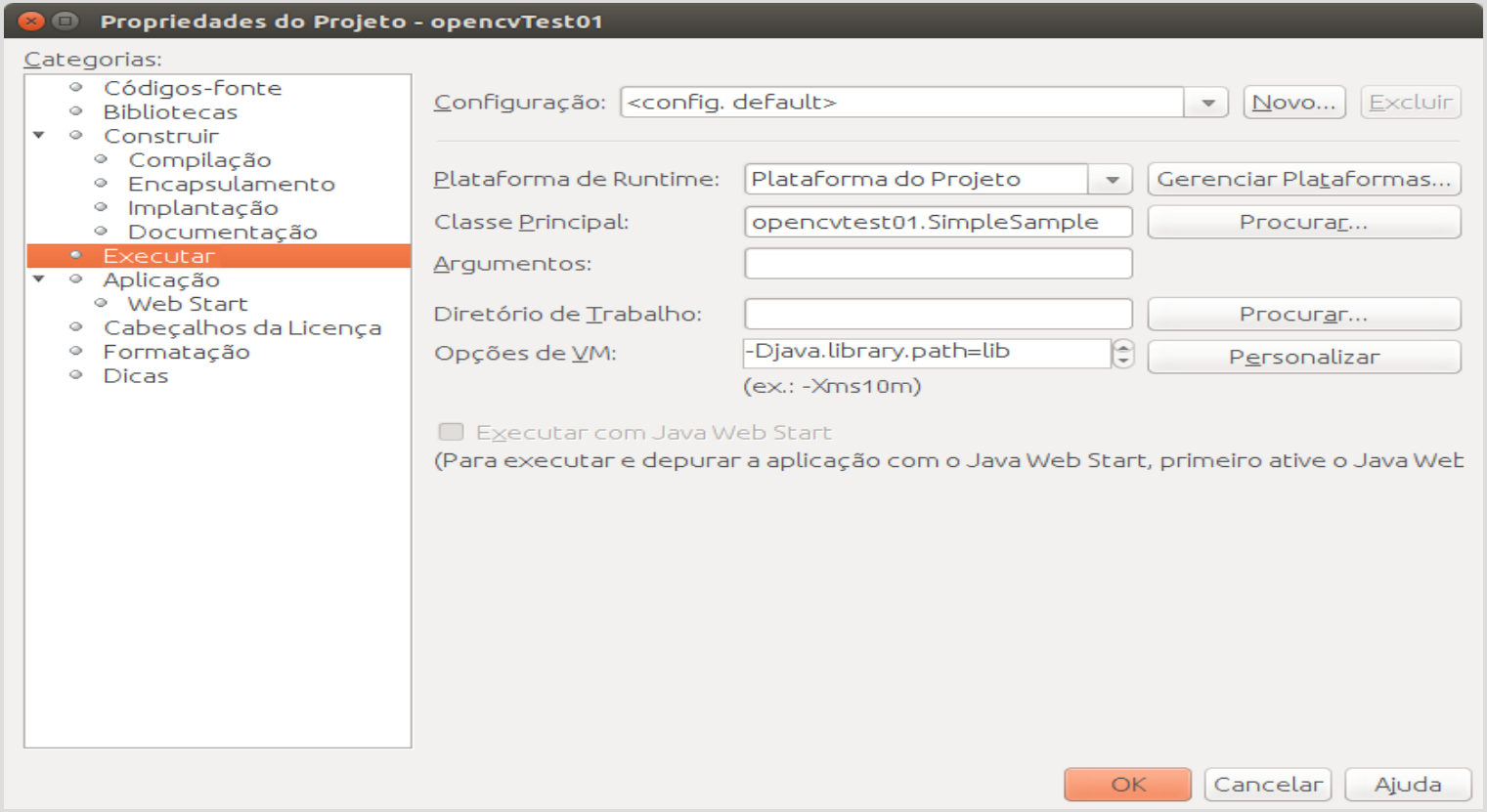
Exemplo de uso (continuação 5)

Figura 4: Copie e cole a biblioteca nativa (libopencv_java300.so) na pasta lib do projeto



Exemplo de uso (continuação 6)

Figura 5: Opção de VM categoria Executar nas propriedades do projeto: -Djava.library.path=lib



Exemplo de uso (continuação 7)

Na Figura 7 a seguir consta o código da classe principal SimpleSample, e na Figura 8 podemos ver um exemplo de execução do projeto.
Uma matriz é criada, populada e impressa. Em seguida, duas matrizes são criadas e populadas, e uma terceira matriz recebe o produto das duas primeiras. As três são impressas.

Através deste simples projeto podemos aprender a adicionar as bibliotecas necessárias para rodar códigos com o OpenCV. Serve como base para adaptar muitos exemplos que podem ser obtidos na internet, mesmo de outras linguagens, como C, CPP, Python e Android.

Exemplo de uso (continuação 8)

Figura 6: Código SimpleSample

```
package opencvtest01;

import org.opencv.core.Core;
import org.opencv.core.Mat;
import org.opencv.core.CvType;
import org.opencv.core.Scalar;

/**
 *
 * @author duo
 */
class SimpleSample {


static{ System.loadLibrary(Core.NATIVE_LIBRARY_NA
ME); }

  public static void main(String[] args) {
    System.out.println("Welcome to OpenCV " +
Core.VERSION);
    Mat m = new Mat(5, 10, CvType.CV_8UC1, new
Scalar(0));
    System.out.println("OpenCV Mat: " + m);
    Mat mr1 = m.row(1);
    mr1.setTo(new Scalar(1));
    Mat mc5 = m.col(5);
    mc5.setTo(new Scalar(5));
    System.out.println("OpenCV Mat data:\n" + m.dump());
    Mat m1 = Mat.ones(4, 2, CvType.CV_64FC1);
    Mat m2 = Mat.ones(2, 4, CvType.CV_64FC1);
    Core.randn(m2, 4.0, 2.0);
    System.out.println("\n\nMultiplicação de Matrizes:\n");
    System.out.println("m1 =\n" + m1.dump() + "\n");
    System.out.println("m2 =\n" + m2.dump() + "\n");
    Mat m3 = Mat.zeros(4, 3, CvType.CV_64FC1);
    Core.gemm(m1, m2, 1, new Mat(), 0, m3);
    //Core.multiply(m1, m2, m3);
    System.out.println("m1 X m2 =\n" + m3.dump());
  }

}
```

Exemplo de uso (continuação 9)

Figura 7: Execução do código SimpleSample

```
Welcome to OpenCV 3.0.0
OpenCV Mat: Mat [ 5*10*CV_8UC1, isCont=true,
isSubmat=false, nativeObj=0x7faf700ebc50,
dataAddr=0x7faf700ebd30 ]
OpenCV Mat data:
[  0,   0,   0,   0,   0,   5,   0,   0,   0,   0;
   1,   1,   1,   1,   1,   5,   1,   1,   1,   1;
   0,   0,   0,   0,   0,   5,   0,   0,   0,   0;
   0,   0,   0,   0,   0,   5,   0,   0,   0,   0;
   0,   0,   0,   0,   0,   5,   0,   0,   0,   0]
```

```
Multiplicação de Matrizes:

m1 =
[1, 1;
 1, 1;
 1, 1;
 1, 1]

m2 =
[1, 1, 1, 1;
 1, 1, 1, 1]

m1 X m2 =
[2, 2, 2, 2;
 2, 2, 2, 2;
 2, 2, 2, 2;
 2, 2, 2, 2]
CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)
```

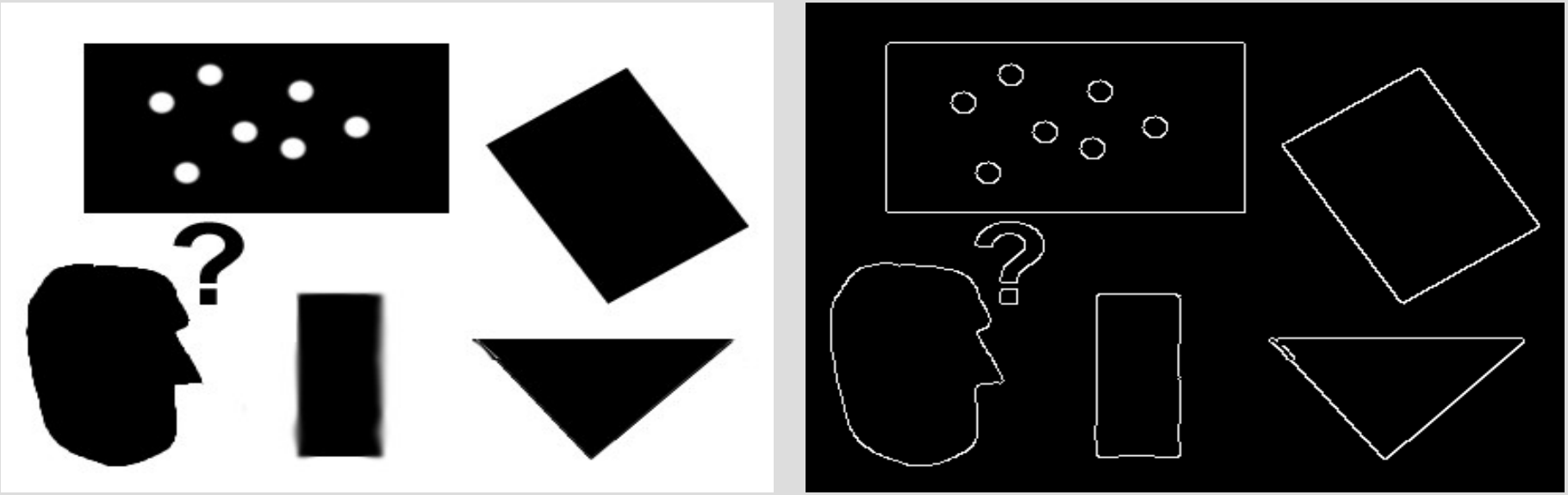
Detecção de contornos

```
# -*- coding: utf-8 -*-
from opencv.cv import *
from opencv.highgui import *

img = cvLoadImage("figura.bmp")
im_gray = cvCreateImage(cvGetSize(img), 8, 1)#cria a imagem que #receberá a imagem original convertida em tons de cinza
im_canny = cvCreateImage(cvGetSize(img), 8, 1)#cria a imagem que #receberá os contornos da imagem original
cvCvtColor(img, im_gray, CV_BGR2GRAY)#converte a imagem original em escala de cinza
cvCanny(im_gray, im_canny, 100, 150, 3)#detecta os contornos na imagem
cvNamedWindow("teste")
cvShowImage("teste", im_canny)
cvWaitKey(0)
```

Resultado

Figura 9: Rodando código em python, á esquerda a imagem original e à direita a imagem com os contornos identificados.

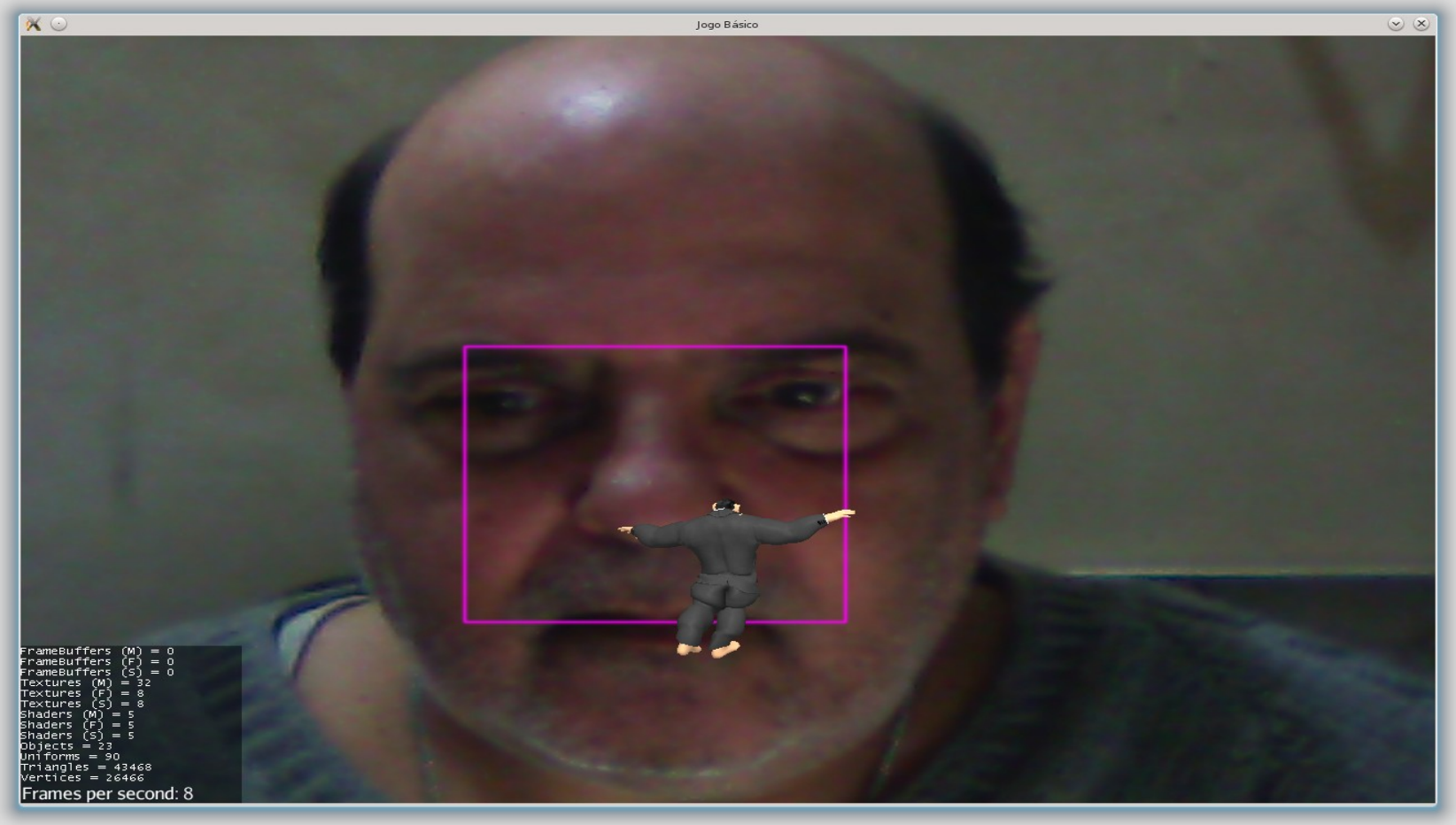


Aplicação em projeto de Computação

O projeto JMCV - AMBIENTE DE PROGRAMAÇÃO PARA JOGOS 3D COM PROCESSAMENTO DE IMAGENS é um exemplo do uso do OpenCV em projetos de computação.
JMCV é um ambiente de programação para jogos 3D com processamento de imagens, obtido através da união de duas plataformas livres: O jMonkeyEngine 3.0 e o JavaCV 0.9, uma implementação em linguagem Java do OpenCV.
Pode ser utilizado para implementar e/ou aprimorar programas como de jogos 3D e realidade aumentada com fins de entretenimento, educacionais, simulações e treinamento entre outros. Veja na Figura 10 a seguir uma imagem do projeto em ação.

Aplicação em projeto de Computação (continuação)

Figura 10: O projeto JMCV.



Aplicação em projeto de Robótica

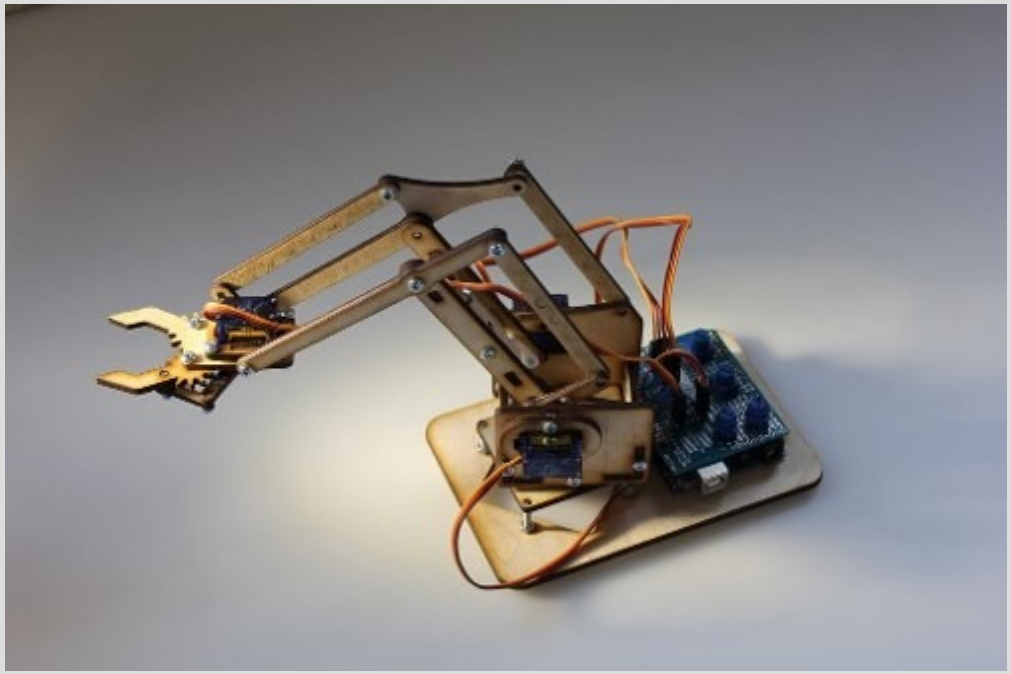
O projeto GARRA ROBÓTICA SELECIONADORA POR IMAGEM é um exemplo do uso do OpenCV em robótica.

Robôs fazem uso de várias peças externas de software e hardware para estender suas capacidades.
Neste experimento objetos colocados ao alcance de um braço robótico são observados através de uma câmera e descartados em um recipiente positivo ou negativo, respectivamente se foram reconhecidos ou não através do processamento da imagem obtida.

Na Figura 11 pode-se ver a garra robótica a ser utilizada no projeto. A classificação de imagens pode ser feita de várias formas. Uma maneira interessante de fazê-lo seria treinando uma Suport Vector Machine (SVM). O OpenCV implementa SVM's.

Aplicação em projeto de Robótica (continuação)

Figura 11: A garra robótica.



Aplicação em projeto de Robótica (continuação 2)

Para aprender mais sobre SVM e outras técnicas de aprendizagem de máquina, um curso gratuito e de qualidade pode ser encontrado em:
https://www.coursera.org/learn/machine-learning/

Agradeço a atenção.

Caso hajam comentários ou dúvidas, estou a disposição.