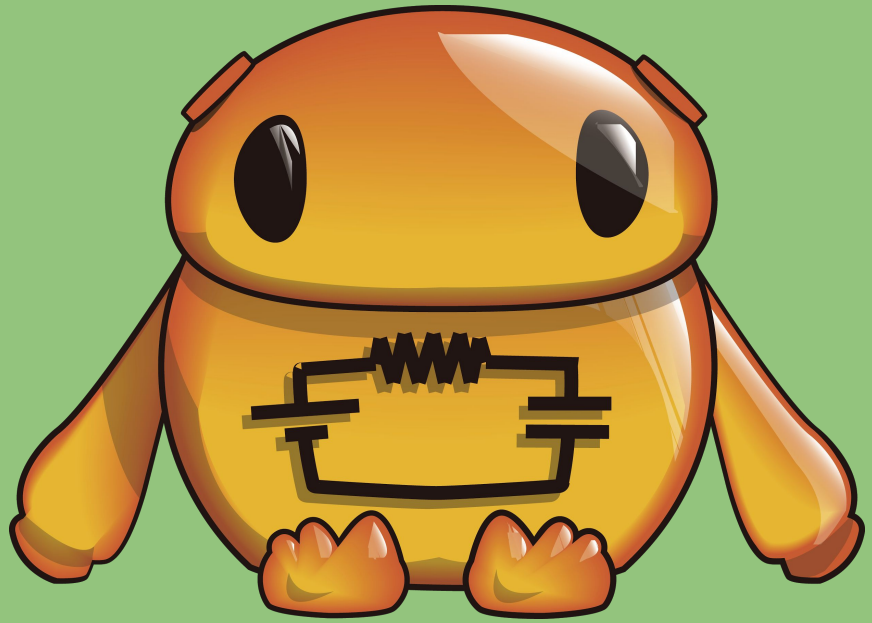


ROBÔCIN

Sobre o que vamos falar?

1 - O que é Visão Computacional?

2 - Visão na VSSL

3 - OpenCV - Instalação e quick start

4 - Introdução a Processamento de Imagem com OpenCV

5 - Tracking

6 - Desafio

ROBÔCIN

O que é Visão Computacional?

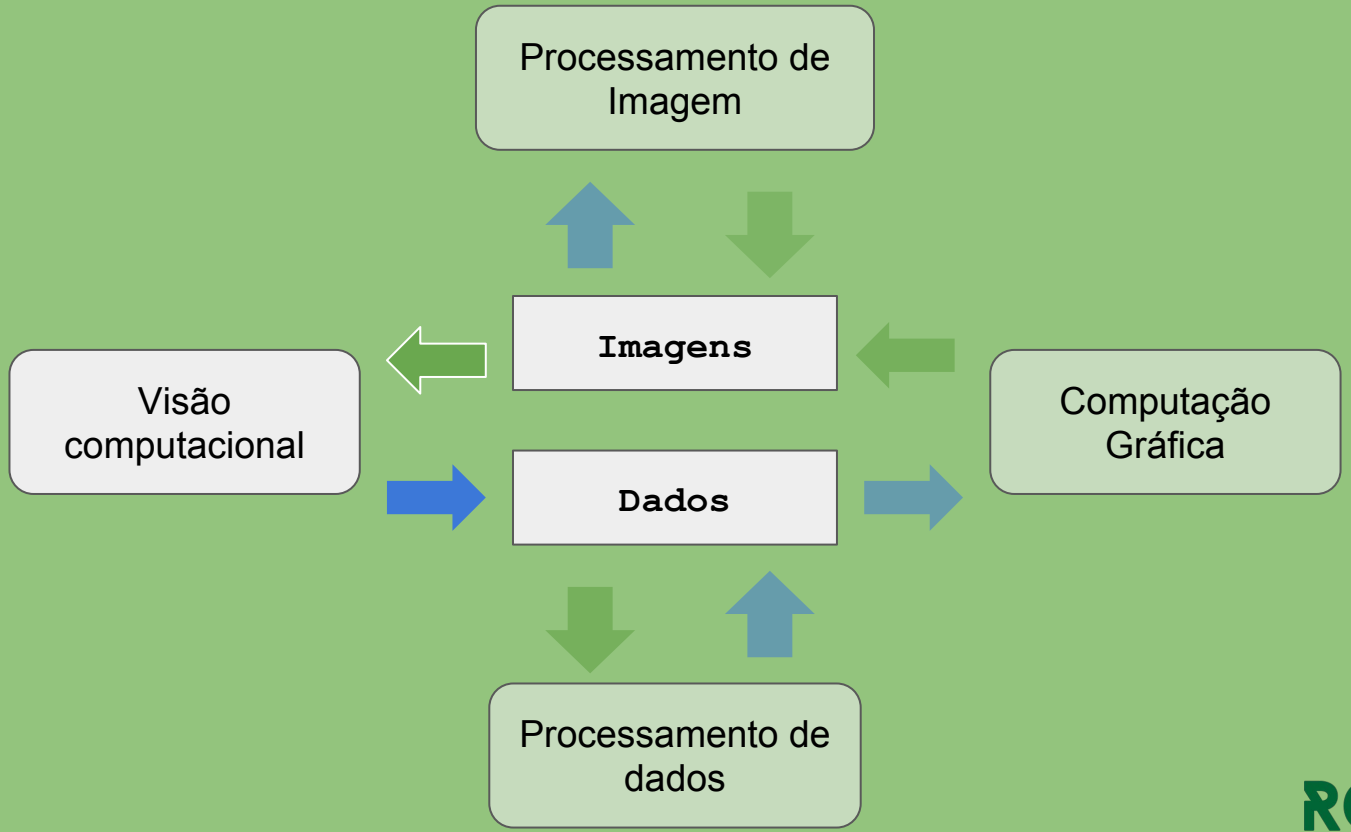
ROBÔCIN

Conceito

"Visão computacional é uma área interdisciplinar que busca extrair informações de alto nível de imagens e vídeos."

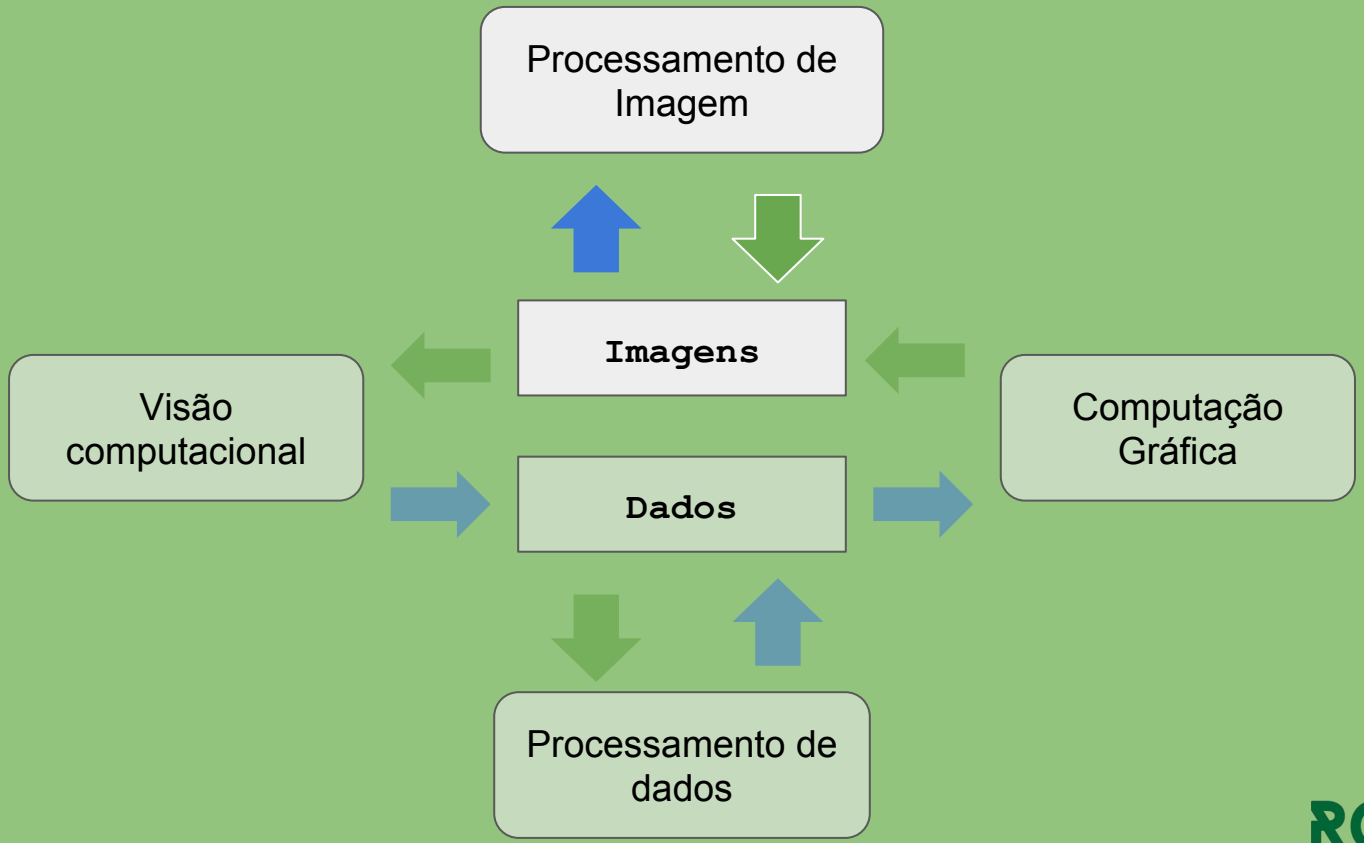
- Dana H. Ballard; Christopher M. Brown (1982)

Visão e áreas correlatas



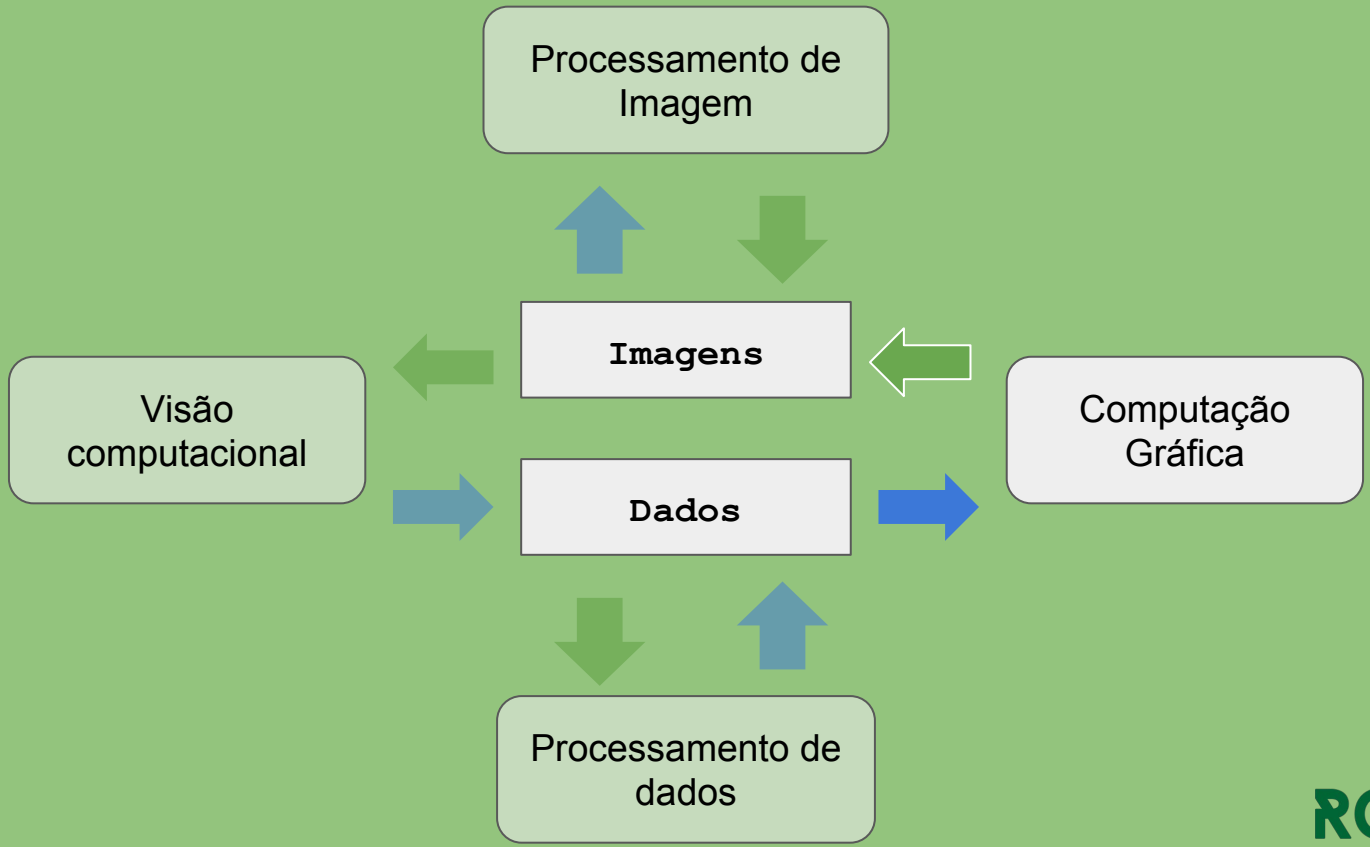
ROBÔCIN

Visão e áreas correlatas



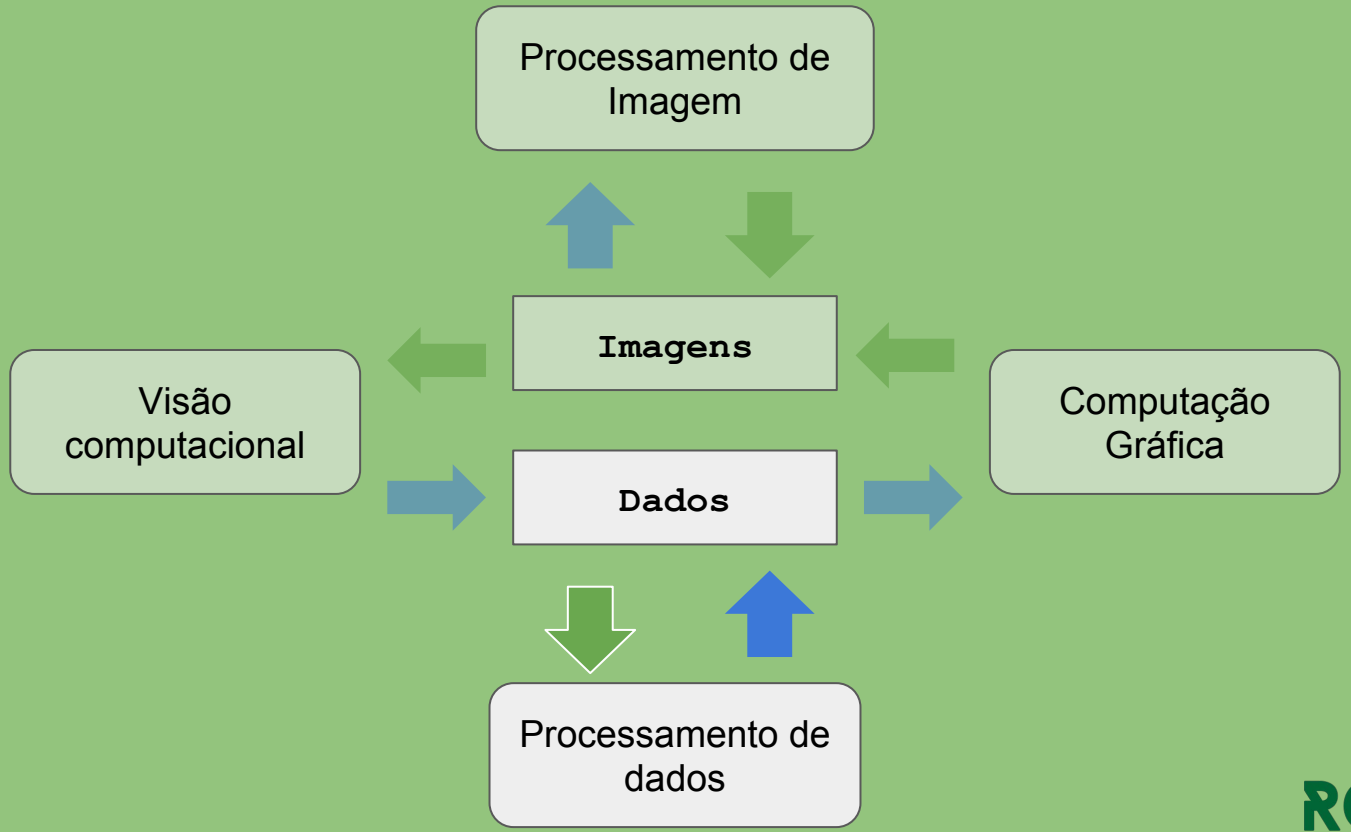
ROBÔCIN

Visão e áreas correlatas



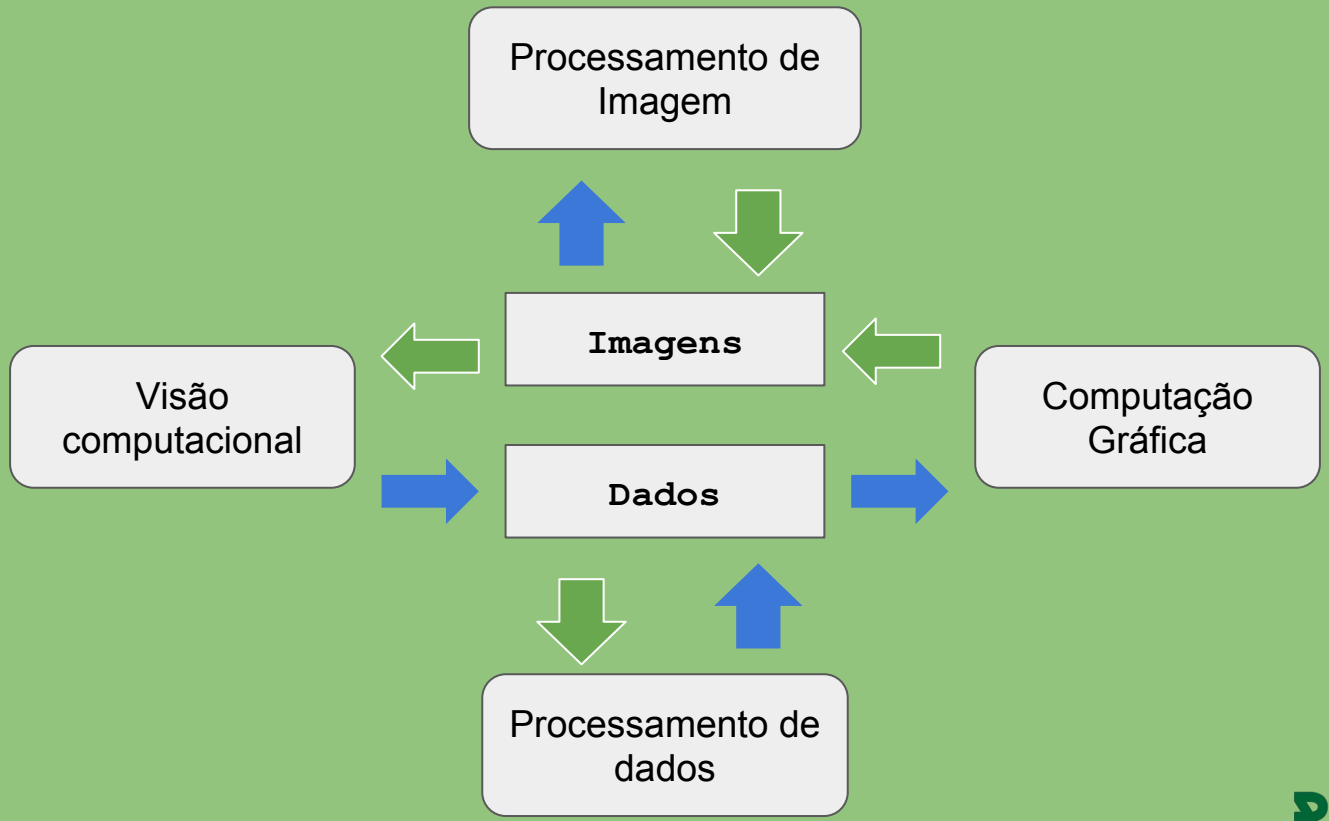
ROBÔCIN

Visão e áreas correlatas



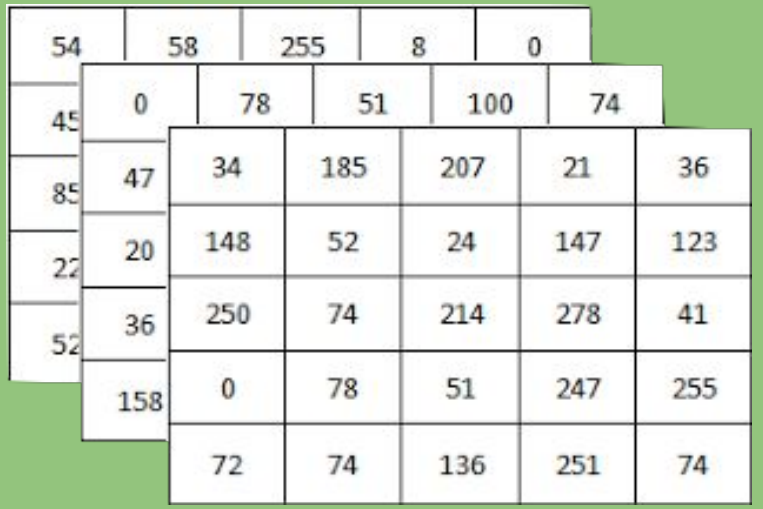
ROBÔCIN

Visão e áreas correlatas



ROBÔCIN

IMAGENS PARA UM COMPUTADOR

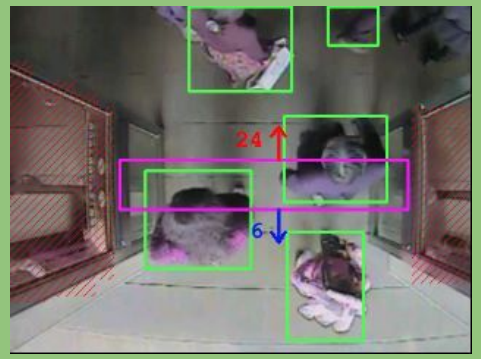


ROBÔCIN

PODER DA VISÃO COMPUTACIONAL

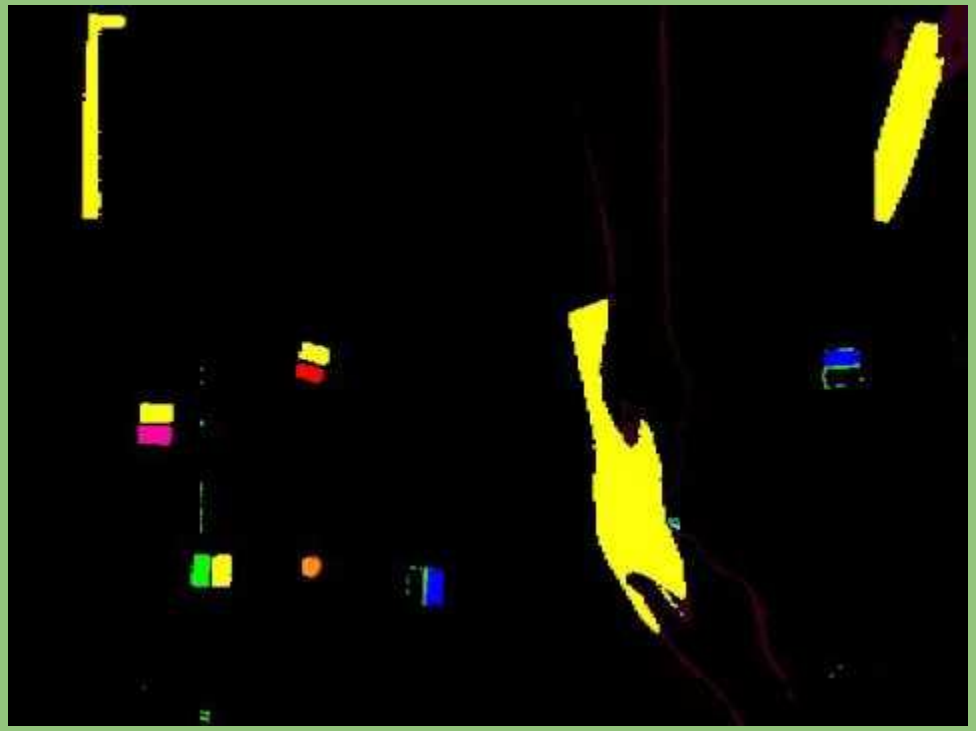
- Reconhecer
- Localizar
- Seguir movimentos
- Reconstruir espaços 3D a partir de imagens 2D
- Reconhecer Ações

Áreas de aplicação



ROBÔCIN

Visão na VSSL



ROBÔCIN

OpenCV

- Open Source Computer Vision Library
- Desenvolvida em C++
- Interfaces completas para diversas linguagens
- Mais de 350 algoritmos implementados
- Ótima documentação e suporte da comunidade

Instalando OpenCV (2.4.11)

Descompactar pasta do Workshop e acessar via terminal.

Para Linux

```
$ chmod +x opencv.sh && ./opencv.sh
```

Compilação:

```
$ g++ nomeDoPrograma.cpp -o nomeDoPrograma `pkg-config --cflags --libs opencv`
$ ./nomeDoPrograma
```

ROBÔCIN

Carregando uma imagem usando OpenCV

```
// cria uma Mat para guardar a imagem carregada
cv::Mat imagem;
// Carrega a imagem1
imagem = cv::imread("image1.jpg", CV_LOAD_IMAGE_COLOR);

if(! imagem.data ) {
        cout <<  "Não foi possível abrir a imagem" << std::endl ;
        return -1;
}

// cria uma janela
cv::namedWindow( "window", CV_WINDOW_AUTOSIZE );
// mostra a imagem na janela "window"
cv::imshow( "window", imagem );
//Salva a imagem num arquivo de nome result.jpg
cv::imwrite("result.jpg",imagem);
// Espera por input do teclado
cv::waitKey(0);
```

ROBÔCIN

Acessando os pixels da imagem (unidimensional)

```
// criando um ponteiro legal
uchar * pointer;

for (int i=0; i < imagem.rows; i++) {
    // ponteiro recebe o endereço da linha i da imagem
    pointer = imagem.ptr<uchar>(i);

    for (int j=0; j < imagem.cols; j++) {
        //acessando a posiçao j do ponteiro com o operador padrão de C
        pointer[j] = pointer[j]/10;

    }
}
```

Acessando os pixels da imagem (3D)

```
typedef struct RGB
{
    uchar blue;
    uchar green;
    uchar red;
} RGB;
```

```
RGB* pixelColor;

for (int i=0; i < imagem.rows; i++) {

    for (int j=0; j < imagem.cols; j++) {

        pixelColor = &(imagem.ptr<RGB>(i)[j]);

        pixelColor->red = pixelColor->red/10;

    }

}
```

ROBÔCIN

Acessando os pixels da imagem (3D)

```
typedef struct RGB
{
    uchar blue;
    uchar green;
    uchar red;
} RGB;
```

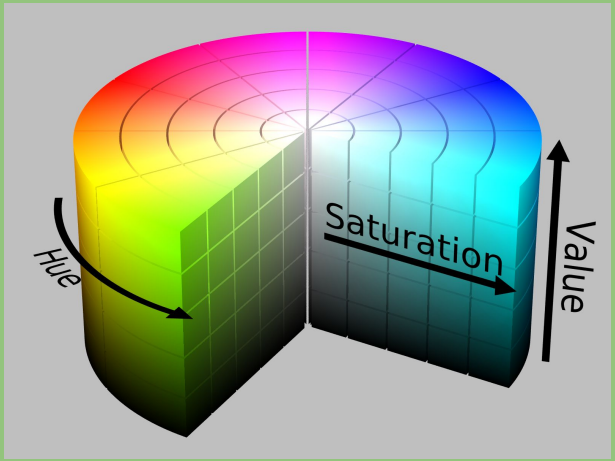
```
RGB* pointer;

for (int i=0; i < imagem.rows; i++) {

    pointer = imagem.ptr<RGB>(i);

    for (int j=0; j < imagem.cols; j++) {


        pointer[i].red = pointer[i].red/10;

    }

}
```

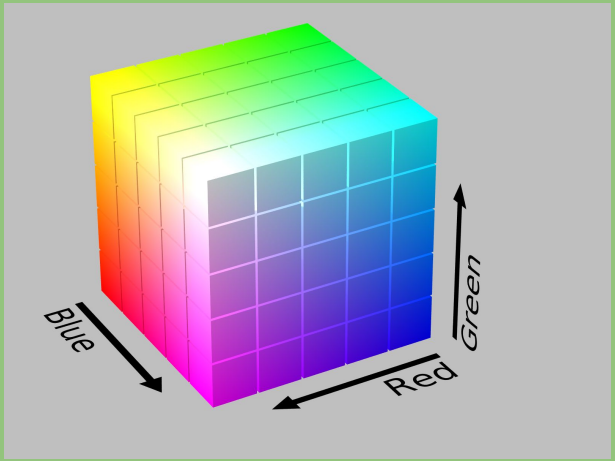
ROBÔCIN

Introdução ao Processamento Digital de Imagens

ROBÔCIN

Espaço de Cores

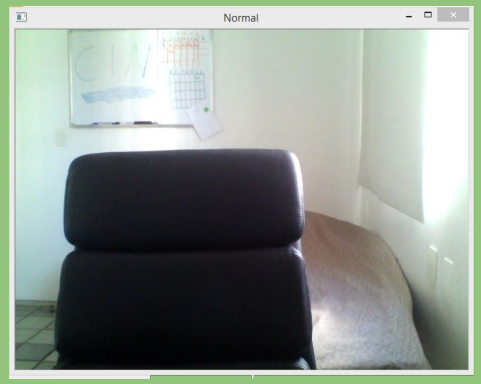


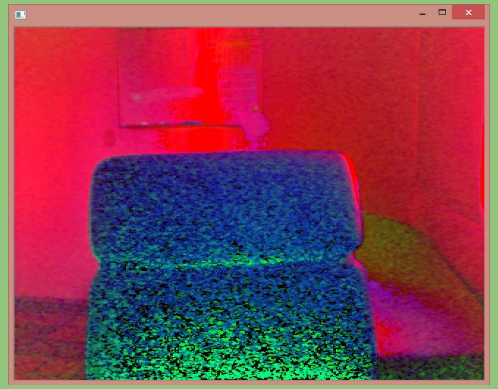


Espaço de Cores

Dado que temos uma imagem em BGR (similar ao RGB), para converter a imagem para outros espaços de cores basta utilizar a função:

```
void cvtColor(cv::Mat src, cv::Mat dst, int code, int dstCn=0 )
```





ps: para mais informações entre no link

ROBÔCIN

Espaço de Cores

Dado que temos uma imagem em BGR (similar ao RGB), para converter a imagem para tons de cinza basta utilizar a função:

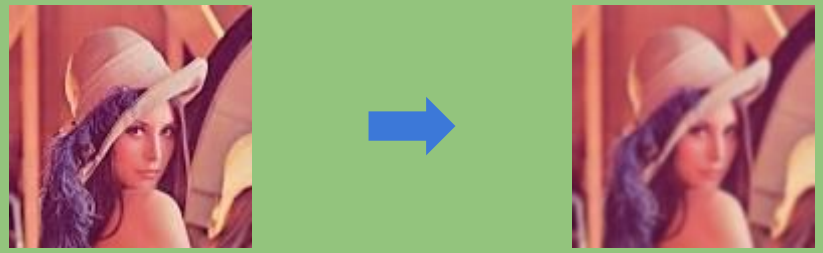
```
void cvtColor(cv::Mat src, cv::Mat dst, int code, int dstCn=0 )
```

```
// Converter bgr (rgb) para tons de cinza
cvtColor(matrizOrigemImagem, matrizDestinoImagem, CV_BGR2GRAY);

// Mostrando as imagens
namedWindow( "Imagem Colorida", CV_WINDOW_AUTOSIZE );
imshow( "Imagem Colorida", matrizOrigemImagem );

namedWindow( "Imagem em Tons de Cinza", CV_WINDOW_AUTOSIZE );
imshow( "Imagem em Tons de Cinza", matrizDestinoImagem );

waitKey(0);
```

ROBÔCIN

Filtros no Domínio Espacial: Filtro Gaussiano

- Filtro passa-baixa
- Redução de ruídos em imagens
- Suavização de imagens
- Borramento de imagens
  - Gaussian Blur



ps: para mais informações entre nolink

ROBÔCIN

Filtros no Domínio Espacial: Filtro Gaussiano

```
void GaussianBlur(cv::Mat src, cv:Mat dst, Size kSize,
                  double sigmaX, double sigmaY=0, int borderType=BORDER_DEFAULT )
```

- Src - Imagem de entrada
- Dst - Imagem de destino
- Size(x,x) - tamanho do kernel, Size(5,5) é um filtro com matriz 5x5 para passa na imagem pixel a pixel.
- desvioX, desvioY - desvio padrão em cada direção (normalmente se coloca 0 como padrão)
- BorderType - Método de extrapolação de pixels

ROBÔCIN

Filtros no Domínio Espacial: Filtro Gaussiano

```
// Aplicando o Gaussian Blur
GaussianBlur( matrizOrigemImagem, matrizDestinoImagem, Size( 5, 5 ), 0, 0 );

// Mostrando as imagens
namedWindow( "Imagem Original", CV_WINDOW_AUTOSIZE );
imshow( "Imagem Original", matrizOrigemImagem );

namedWindow( "Imagem com filtro Gaussiano", CV_WINDOW_AUTOSIZE );
imshow( "Imagem com filtro Gaussiano", matrizDestinoImagem );

waitKey(0);
```

Filtros no Domínio Espacial: Filtro Laplaciano

- Filtro passa-alta
- Detectar bordas
- Segunda derivada
- Realce das características

Filtros no Domínio Espacial: Filtro Laplaciano

```
// Removendo ruído através do filtro Gaussiano
GaussianBlur( matrizOrigemImagem, matrizOrigemImagem, Size(3,3), 0, 0);

// Converter bgr (rgb) para tons de cinza
cvtColor(matrizOrigemImagem, matrizOrigemImagem, CV_BGR2GRAY);

Laplacian( matrizOrigemImagem, matrizDestinoImagem, CV_16S, 3, 1, 0, BORDER_DEFAULT );

// Convertendo a imagem para 8-bits para poder ser mostrada
convertScaleAbs( matrizDestinoImagem, abs_dst );

// Mostrando as imagens
imshow( "Imagem com Filtro Laplaciano", abs_dst );
```

ROBÔCIN

Limiarização

$$g(x,y) = \begin{cases} 1 & \text{se } f(x,y) > T \\ 0 & \text{se } f(x,y) \leq T \end{cases}$$


DEPARTAMENTO DE ORDEM POLITICA E SOCIAL

VERIFICAÇÃO DE JORNAIS

HISTORICO

Publicação:	Jornal do Brasil
Data:	09-10-73
Notas:	
Folha nº	

P 007-RJ

Auditoria julga 14 da Ação Marxista


VERIFICAÇÃO DE JORNAIS

HISTORICO

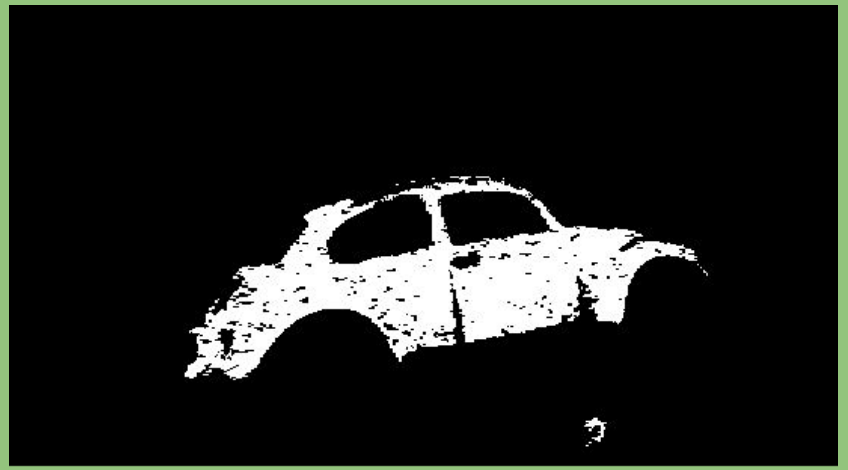
Publicação:	Jornal do Brasil
Data:	09-10-73
Notas:	
Folha nº	

P 007-RJ

Auditoria julga 14 da Ação Marxista

ROBÔCIN

Limiarização



Limiarização

```
double threshold(InputArray src, OutputArray dst, double thresh, double maxval, int type)
```

```
void inRange(InputArray src, InputArray lowerb, InputArray upperb, OutputArray dst)
```

Limiarização

```
src = cv::imread( "image.jpg", CV_LOAD_IMAGE_GRAYSCALE );

if (!src.data) {

  std::cout << "Não foi possível abrir ou encontrar a imagem" << std::endl;
  return -1;
}

cv::imshow("original",src);

cv::namedWindow( "tela", CV_WINDOW_AUTOSIZE );

cv::createTrackbar( "Trackbar Value", "tela", &thresholdValue, 255, applyThreshold);

applyThreshold( 0, 0 );
```

ROBÔCIN

Limiarização

```
cv:: Mat src, dst;
int thresholdValue = 0;

void applyThreshold( int, void* )
{
  cv::threshold( src, dst, thresholdValue, 255, cv::THRESH_BINARY);

  cv::imshow( "tela", dst );
}
```

ROBÔCIN

Limiarização

```
while(true)
{
    int c;
    c = cv::waitKey( 20 );
    if( (char)c == 27 ){
        break;
    }
 }
```

Detecção de Bordas

ps: para mais informações entre no link

ROBÔCIN

Detecção de Bordas

```
src = imread( "image.jpg" );

if( !src.data )
{ return -1; }


dst.create( src.size(), src.type() );


cvtColor( src, src_gray, CV_BGR2GRAY );


namedWindow( window_name, CV_WINDOW_AUTOSIZE );


createTrackbar( "Min Threshold:", window_name, &lowThreshold, max_lowThreshold, CannyThreshold );


CannyThreshold(0, 0);
```

ps: para mais informações entre nolink

ROBÔCIN

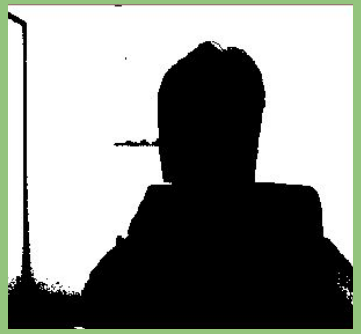
Detecção de Bordas

```
void CannyThreshold(int, void*)
{

  blur( src_gray, detected_edges, Size(3,3) );


  Canny( detected_edges, detected_edges, lowThreshold, lowThreshold*ratio, kernel_size );


  dst = Scalar::all(0);

  src.copyTo( dst, detected_edges);
  imshow( window_name, dst );
 }
```

ps: para mais informações entre nolink

ROBÔCIN

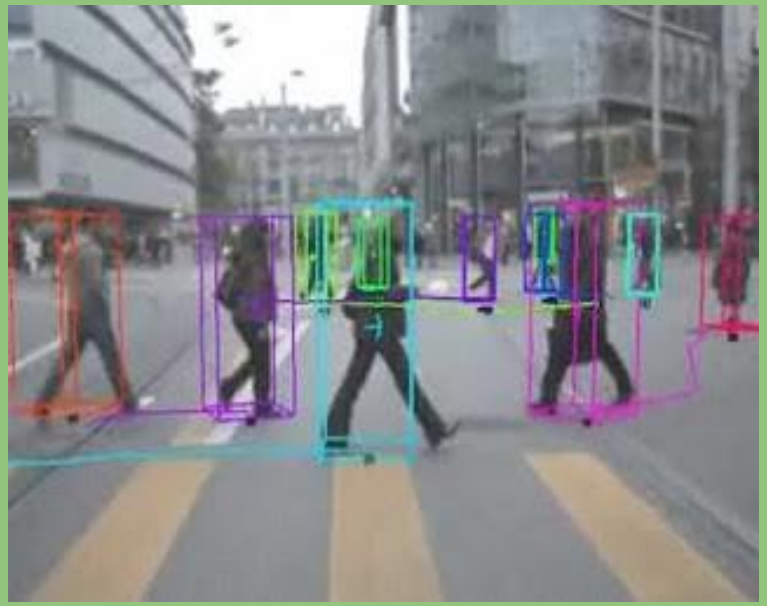
Composição de filtros

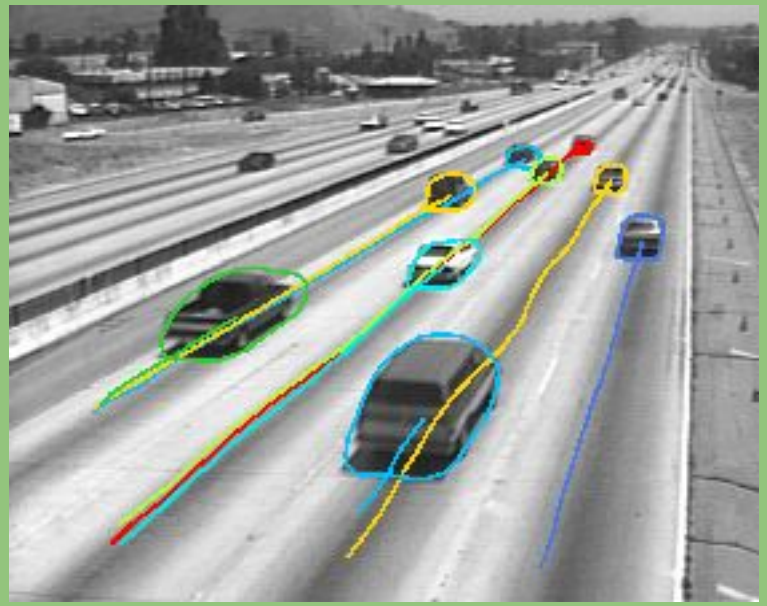


ROBÔCIN

Tracking

O que é Tracking?





ROBÔCIN

Métodos de tracking

Simple Blob Detector

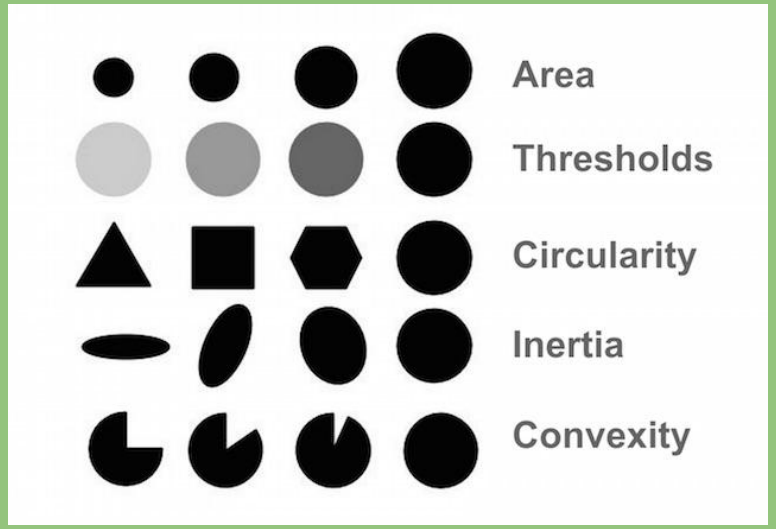
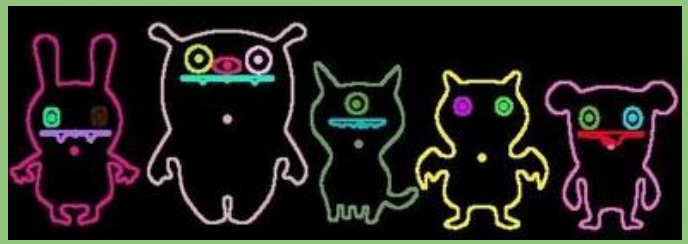


Image Moments



ROBÔCIN

Image Moments

```
/// Load source image and convert it to gray
src = cv::imread( "a.jpg", 1 );

/// Convert image to gray and blur it
cv::cvtColor( src, src_gray, CV_BGR2GRAY );
cv::blur( src_gray, src_gray, cv::Size(3,3) );

/// Create Window
cv::namedWindow( "Source", CV_WINDOW_AUTOSIZE );
cv::imshow( "Source", src );

cv::createTrackbar( " Canny thresh:", "Source", &thresh, max_thresh, thresh_callback );
thresh_callback( 0, 0 );

cv::waitKey(0);
```

Image Moments

```
void thresh_callback(int, void* )
{
  Mat canny_output;
  vector<vector<Point> > contours;
  vector<Vec4i> hierarchy;
  /// Detect edges using canny
  Canny( src_gray, canny_output, thresh, thresh*2, 3 );
  /// Find contours
  findContours( canny_output, contours, hierarchy, CV_RETR_TREE, CV_CHAIN_APPROX_SIMPLE, Point(0, 0) );
```

Image Moments

```
/// Get the moments
vector<Moments> mu(contours.size() );
for( int i = 0; i < contours.size(); i++ )
   { mu[i] = moments( contours[i], false ); }

///  Get the mass centers:
vector<Point2f> mc( contours.size() );
for( int i = 0; i < contours.size(); i++ )
   { mc[i] = Point2f( mu[i].m10/mu[i].m00 , mu[i].m01/mu[i].m00 ); }
```

Image Moments

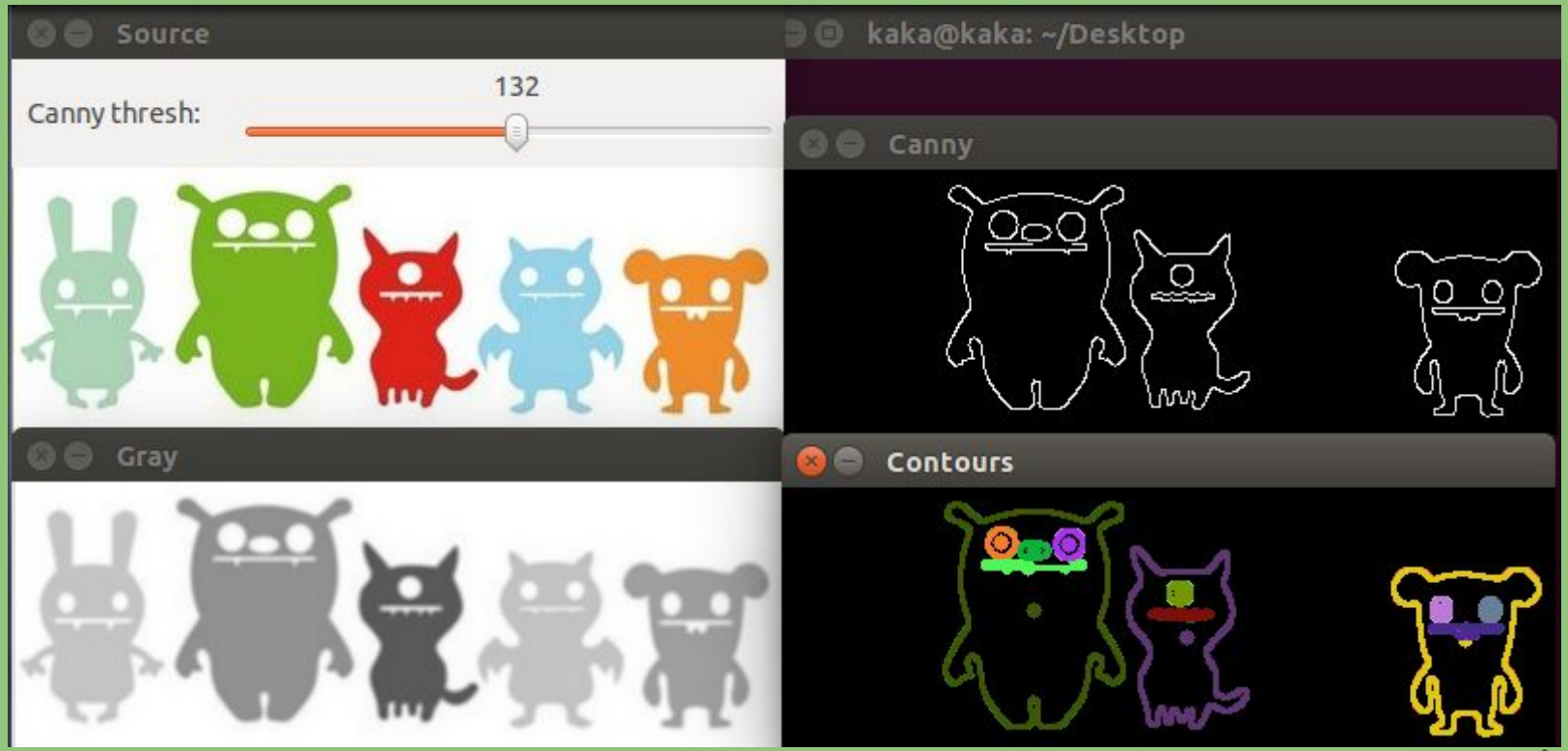
```
/// Draw contours
Mat drawing = Mat::zeros( canny_output.size(), CV_8UC3 );
for( int i = 0; i< contours.size(); i++ )
    {
      Scalar color = Scalar( rng.uniform(0, 255), rng.uniform(0,255), rng.uniform(0,255) );
      drawContours( drawing, contours, i, color, 2, 8, hierarchy, 0, Point() );
      circle( drawing, mc[i], 4, color, -1, 8, 0 );
    }

/// Show in a window
namedWindow( "Contours", CV_WINDOW_AUTOSIZE );
imshow( "Contours", drawing );
```

Image Moments

```
  /// Calculate the area with the moments 00 and compare with the result of the OpenCV function
  printf("\t Info: Area and Contour Length \n");
  for( int i = 0; i< contours.size(); i++ )
     {
       printf(" * Contour[%d] - Area (M_00) = %.2f - Area OpenCV: %.2f - Length: %.2f \n", i, mu[i].m00,
       Scalar color = Scalar( rng.uniform(0, 255), rng.uniform(0,255), rng.uniform(0,255) );
       drawContours( drawing, contours, i, color, 2, 8, hierarchy, 0, Point() );
       circle( drawing, mc[i], 4, color, -1, 8, 0 );
     }
}
```

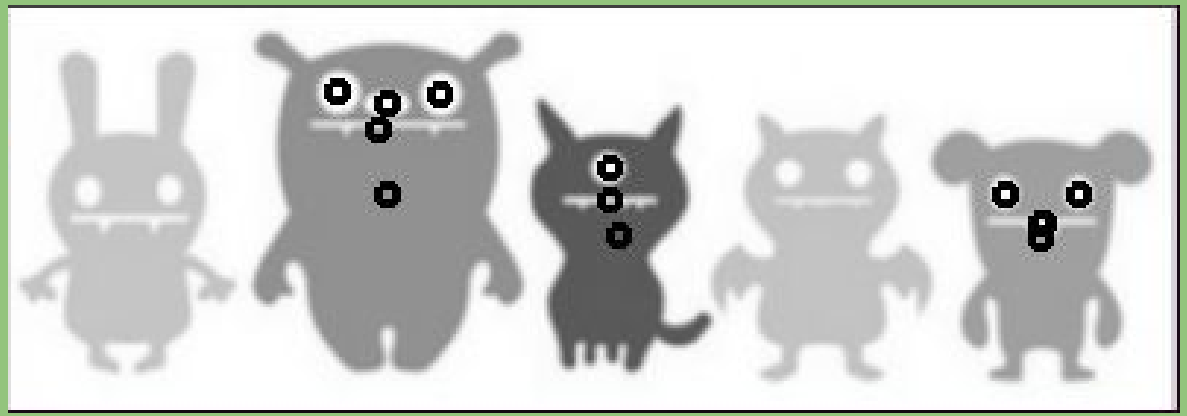
ROBÔCIN

Image Moments



ROBÔCIN

Image Moments



ROBÔCIN

Image Moments

Atividades

1. Segmentar objetos por cores

    a. Escolha uma imagem qualquer, escolha um objeto de fácil visualização devido a sua cor e segmente (separe do restante da imagem).

2. Tracking do objeto segmentado no vídeo:

    a. Escolha um dos robôs do vídeo e faça um algoritmo de tracking baseado na cor, pode se usar o código da atividade 1 como base.

Referências

Wikipedia: Espaço de Cores

CABM/CIn - Visão Computacional

Filtros UFRJ NCE

OpenCV Examples Blogspot

OpenCV - srf Blogspot

ROBÔCIN

Obrigado!

www.cin.ufpe.br/~robocin/

www.facebook.com/robocin

www.instagram.com/robocinufpe

ROBÔCIN