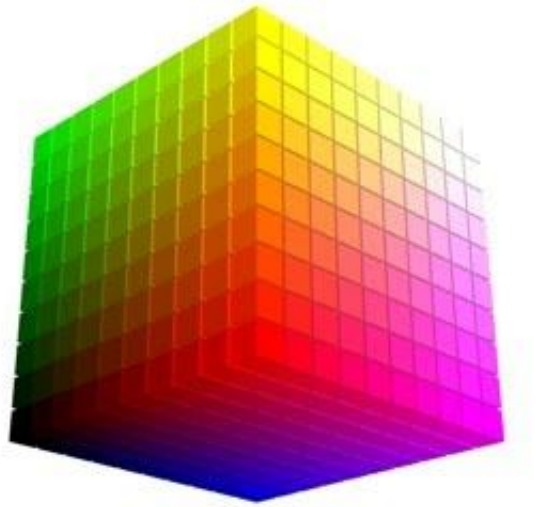
PROCESSAMENTO DIGITAL DE IMAGENS



Projeto de Ensino - Material didático sobre filtros de imagens

Departamento Engenharias e Computação- DCET

Discente - Luciana Roncarati - Ciência da Computação

SUMÁRIO

- Definição borda.
- Definição Filtro Robert Cross.
- Algoritmo Interface *Processing*.
- Referências Bibliográficas.

DEFINIÇÃO

Detecção de bordas: Partindo da definição de borda como uma fronteira entre duas regiões com níveis de cinza relativamente distintos, os algoritmos utilizados para a detecção de bordas são estruturados de forma a detectar as descontinuidades existentes nas transições.

DEFINIÇÃO

Detecção de bordas: A detecção de bordas é uma técnica fundamental no processamento de imagens que visa identificar as transições abruptas de intensidade nos pixels da imagem. Essas transições representam mudanças significativas nas propriedades visuais da imagem, como mudanças de cor, luminosidade ou textura, e podem indicar a presença de objetos, contornos ou padrões importantes.

DEFINIÇÃO

O operador **Roberts Cross** é usado em processamento de imagem e visão computacional para detecção de bordas. Foi um dos primeiros detectores de borda propostos por Lawrence Roberts em 1963. Como um operador diferencial, a ideia por trás do operador Roberts Cross é aproximar o gradiente de uma imagem através de diferenciação discreta, o que é alcançado calculando a soma dos quadrados das diferenças entre pixels diagonalmente adjacentes.

DEFINIÇÃO

O operador realiza uma medição de gradiente espacial 2D em uma imagem. Ele destaca regiões de alta frequência espacial que frequentemente correspondem a bordas. Em seu uso mais comum, a entrada para o operador é uma imagem em escala de cinza, assim como a saída. Os valores de pixel em cada ponto da saída representam a magnitude absoluta estimada do gradiente espacial da imagem de entrada naquele ponto.

DEFINIÇÃO

O operador gradiente é um dos procedimentos utilizados para detectar essas descontinuidades denominadas como bordas

$$\text{Magnitude do Gradiente} = \text{sqrt}(Gx^2 + Gy^2)$$

DEFINIÇÃO

$$I = \begin{pmatrix} a_{x-1\ y-1} & a_{x-1\ y} & a_{x-1\ y+1} \\ a_{x\ y-1} & a_{x\ y} & a_{x\ y+1} \\ a_{x+1\ y-1} & a_{x+1\ y} & a_{x+1\ y+1} \end{pmatrix}$$

Considerando-se uma vizinhança de 3 x 3 pixels em torno de um ponto (x,y).

$$I = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

DEFINIÇÃO

Para realizar a detecção de bordas com o operador Roberts, primeiro convolvemos a imagem original com os seguintes dois kernels:

Kx:

$$\begin{bmatrix} +1 & 0 \\ 0 & -1 \end{bmatrix}$$

Ky:

$$\begin{bmatrix} 0 & +1 \\ -1 & 0 \end{bmatrix}$$

DEFINIÇÃO

Kx:

$$\begin{bmatrix} +1 & 0 \\ 0 & -1 \end{bmatrix}$$

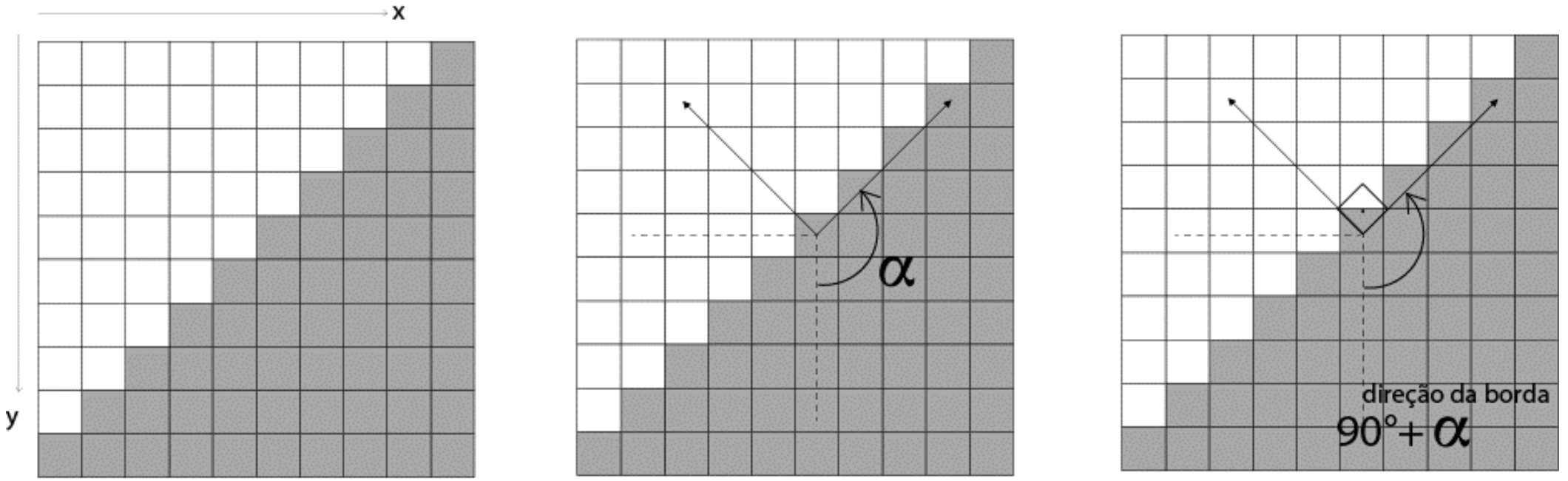


Fig. 1 – definição bordas

DEFINIÇÃO

$$\begin{bmatrix} 0 & +1 \\ -1 & 0 \end{bmatrix}$$

Ky:

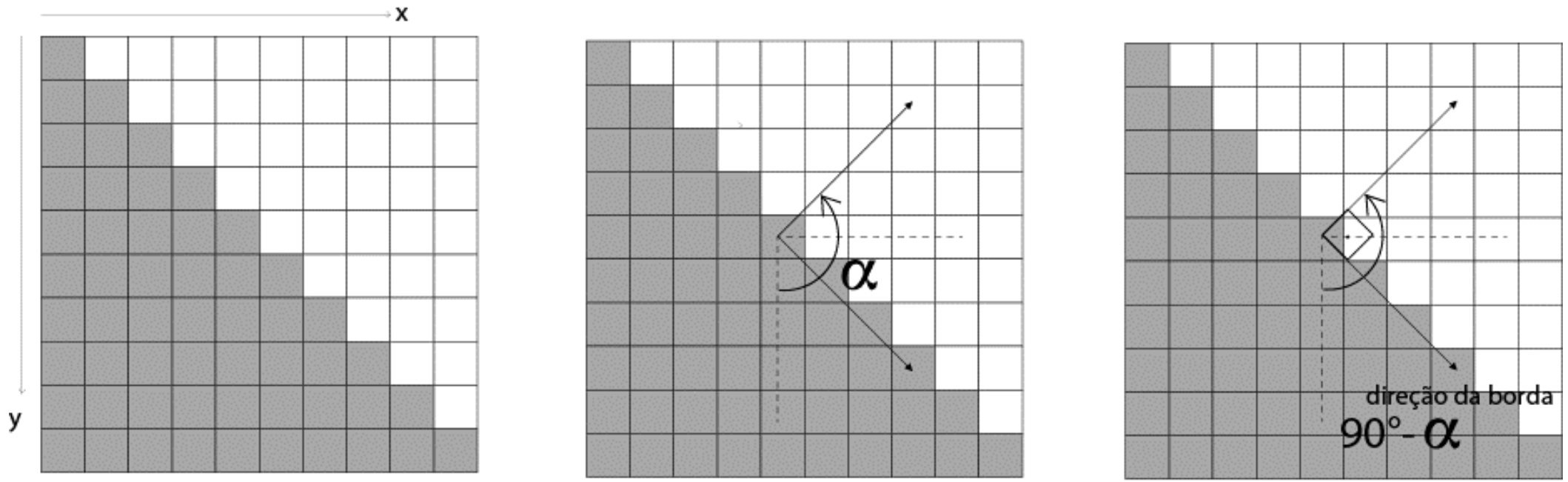


Fig. 2 – definição bordas

ALGORITMO DETECÇÃO DE BORDAS

```
PImage img;
PImage bordasR, bordasG, bordasB;

void setup() {
  size(800, 600);

  // Carregar a imagem de entrada
  img = loadImage("tucano_color.jpg");

  // Criar imagens para armazenar as bordas detectadas nos canais R, G e B
  bordasR = createImage(img.width, img.height, RGB);
  bordasG = createImage(img.width, img.height, RGB);
  bordasB = createImage(img.width, img.height, RGB);

  // Detecta as bordas nos canais R, G e B
  detectarBordasRGB();

  // Exibir a imagem original
  image(img, 0, 0);

  // Salvar as imagens com as bordas detectadas
  bordasR.save("imagem_bordas_R.jpg");
  bordasG.save("imagem_bordas_G.jpg");
  bordasB.save("imagem_bordas_B.jpg");
}
```

Fig. 3 – interface Processing

ALGORITMO DETECÇÃO DE BORDAS

```
}

void detectarBordasRGB() {
  img.loadPixels();
  bordasR.loadPixels();
  bordasG.loadPixels();
  bordasB.loadPixels();

  for (int x = 1; x < img.width - 1; x++) {
    for (int y = 1; y < img.height - 1; y++) {
      // Obter as cores dos pixels vizinhos
      color c1 = img.get(x - 1, y - 1);
      color c2 = img.get(x + 1, y + 1);

      // Calcula a diferença entre os valores dos canais R, G e B dos pixels vizinhos
      int diffR = (int) (red(c1) - red(c2));
      int diffG = (int) (green(c1) - green(c2));
      int diffB = (int) (blue(c1) - blue(c2));

      // Definir o valor dos pixels resultantes nas imagens de bordas
      bordasR.set(x, y, color(abs(diffR), 0, 0));
      bordasG.set(x, y, color(0, abs(diffG), 0));
      bordasB.set(x, y, color(0, 0, abs(diffB)));
    }
  }

  bordasR.updatePixels();
  bordasG.updatePixels();
  bordasB.updatePixels();
}
```

Fig. 4 – interface Processing

Fig. 5 – imagem de entrada

Fig. 6 – imagem bordas em R

Fig. 7 – imagem bordas em G

Fig. 8- imagem bordas em B

REFERÊNCIAS BIBLIOGRÁFICAS

- NUNES L. S, Fátima - Introdução ao processamento de imagens médicas para auxílio ao diagnóstico - uma visão prática, capítulo 2.
- GONZALEZ C, Rafael. e WOODS, Richard - Processamento digital de imagens - 3. Ed. Pearson Prentice hall, São paulo,2010.
- https://homepages.inf.ed.ac.uk/rbf/HIPR2/convolve.htm