Universidade Paulista

Ciência da Computação

PROCESSAMENTO DE IMAGEM

Aleksander Rocha - R.A.: C630IH-0

Daniel Jançanti - R.A.: C630IG-2

Ingrid Oliveira - R.A.: C51791-7

Rafaela Aranas - R.A.: C29CII-0

Araraquara – 2017

1 Introdução

Este trabalho tem por objetivo a implementação das operações de rotacionamento, redirecionamento e aplicação de filtros de imagens. A filtragem aplicada a uma imagem digital é uma operação local que modifica os valores dos níveis digitais de cada pixel da imagem considerando o contexto atual do pixel.

O processo de filtragem é feito utilizando matrizes denominadas máscaras, as quais são aplicadas sobre a imagem. Pela filtragem, o valor de cada pixel da imagem é modificado utilizando-se uma operação de vizinhança, ou seja, uma operação que leva em conta os níveis digitais dos pixels vizinhos e o próprio valor digital do pixel considerado. Nas definições de vizinhança começamos por considerar que: um pixel qualquer da imagem digital é vizinho dele mesmo.

2 Objetivos da filtragem

- Extração de ruídos da imagem;
- Homogeneização da imagem ou de alvos específicos;
- Melhorar na discriminação de alvos da imagem;
- Detecção de bordas entre alvos distintos presentes na imagem;
- Detecção de formas.

3 Descrição

Processamento de imagem é qualquer forma de processamento de dados no qual a entrada e saída são imagens tais como fotografias ou quadros de vídeo. Ao contrário do tratamento de imagens, que se preocupa somente na manipulação de figuras para sua representação final, o processamento de imagens é um estágio para novos processamentos de dados tais como aprendizagem de máquina ou reconhecimento de padrões. A maioria das técnicas envolve o tratamento da imagem como um sinal bidimensional, no qual são aplicados padrões de processamento de sinal.

3.1 Métodos de processamento

Algumas décadas atrás o processamento de imagem era feito majoritariamente de forma analógica, através de dispositivos ópticos. Apesar disso, devido ao grande aumento de velocidades dos computadores, tais técnicas foram gradualmente substituídas por métodos digitais.

O processamento digital de imagem é geralmente mais versátil, confiável e preciso, além de ser mais fácil de implementar que seus duais analógicos. Hardware especializado ainda é usado para o processamento digital de imagem, contando com arquiteturas de computador paralelas para tal, em sua maioria no processamento de vídeos. O processamento de imagens é, em sua maioria, feito por computadores pessoais.

3.2 Técnicas mais usadas

A maioria dos conceitos de processamento de sinais que se aplicam a sinais unidimensionais também podem ser estendidos para o processamento bidimensional de imagens. A transformada de Fourier é bastante usada nas operações de imagem envolvendo uma grande área de correlação.

- Resolução de imagem;
- Limite dinâmico;
- Largura de banda;
- Filtro: Permite a redução de ruídos da imagem para que mais padrões possam ser encontrados;

- Operador diferencial;
- Histograma: Consiste na frequência de um tom específico (seja escala de cinza ou colorido) em uma imagem. Permite a obtenção de informações como o brilho e o contraste da imagem e sua distribuição;
- Detecção de borda;
- Redução de ruído.

3.3 Problemas típicos

Além de imagens bidimensionais estáticas, o campo também abrange o processamento de sinais variados pelo tempo tais como vídeos ou a saída de um equipamento de tomografia. Tais técnicas são especificadas somente para imagens binárias ou em escala de cinza.

- Transformações geométricas tais como escala, rotação e inclinação;
- Correção de cor como ajustes de brilho e contraste, limiarização ou conversão de espaço de cor;
- Combinação de imagens por média, diferença ou composição;
- Interpolação e recuperação de imagem de um formato bruto tal como o filtro bayesiano;
- Segmentação de uma imagem em regiões;
- Edição de imagem e acabamento (retoque) digital;

4 Resultados

4.1 Histograma

O cálculo do histograma é um procedimento simples, onde um vetor que contém a intensidade (luminância) de cada pixel é indexado pelo valor do mesmo e incrementado a cada pixel.

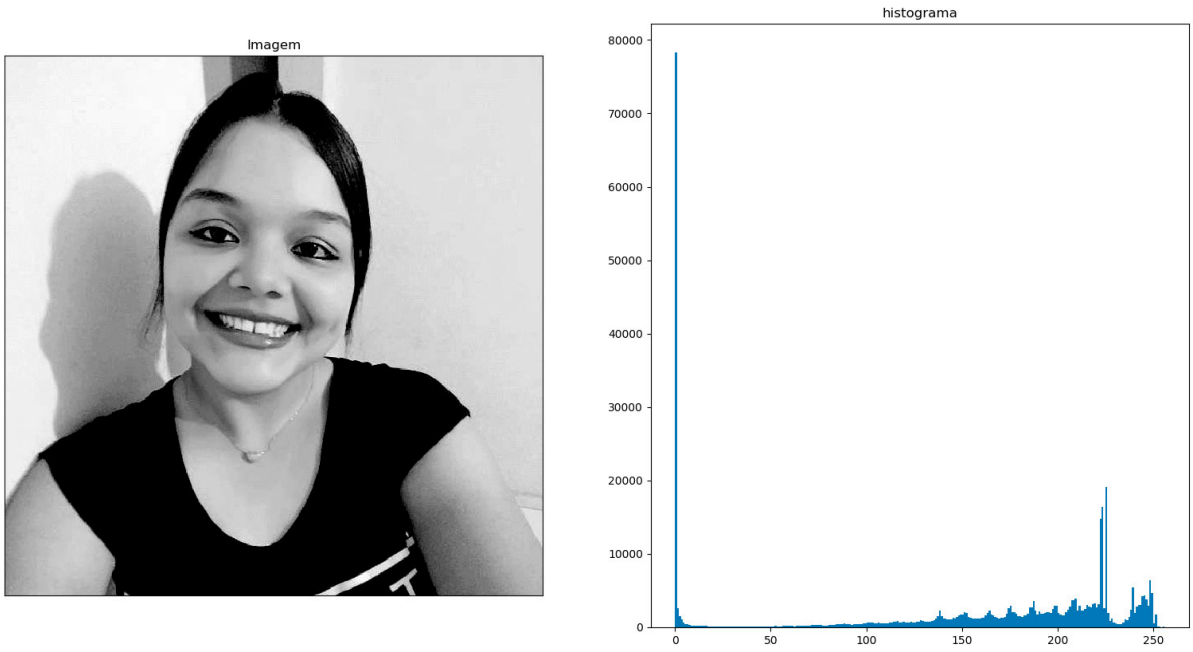

Figura 1: Histograma da imagem

```
void calcula_histograma(IplImage * img) {
  for (int i = 0; i < 256; i++) {
    histograma[i] = 0;
  }
  histMax = 0;
  foreachpixel(img, calc_hist, nothing, nothing);
}

void * calc_hist(unsigned char * w) {
  temp[ * w]++;
  if (temp[ * w] > histMax) {
    histMax = temp[ * w];
  }
}

IplImage * desenha_histograma(int HISTSIZE) {
  int HISTOFFSET;
  HISTOFFSET = HISTSIZE / 256;
  IplImage * ret;
  float a, b, c;
  b = histMax;
  int DrawHist[256];

  for (int i = 0; i < 256; i++) {
    a = (float) histograma[i];
    c = a / b;
    DrawHist[i] = (int)(c * HISTSIZE);
  }
  ret = cvCreateImage(cvSize(HISTSIZE, HISTSIZE), IPL_DEPTH_8U, 3);
```

```
  for (int i = 0; i < 256; i++) {
    cvRectangle(ret, cvPoint(i * HISTOFFSET + 1, HISTSIZE - DrawHist[i]), cvPoint(i * HISTOFFSET + HISTOFFSET, HISTSIZE), cvScalar(255, 127, 127),
  }
  return ret;
}
```

4.2 Equalização do Histograma

O cálculo do histograma é um procedimento simples, onde um vetor que contém a intensidade (luminância) de cada pixel é indexado pelo valor do mesmo e incrementado a cada pixel.

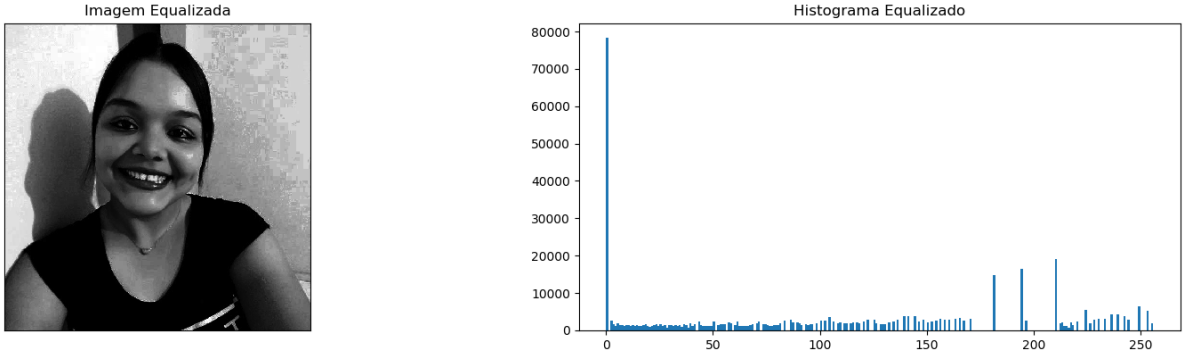

Figura 2: Histograma equalizado

```
void foreachpixel(IplImage * input, void * ( * red)(unsigned char * ), void * ( * green)(unsigned char * ), void * ( * blue)(unsigned char * )) {
  unsigned char * pixel;
  pixel = (unsigned char * ) input -> imageData;

  for (int i = 0; i < input -> width * input -> height * input -> nChannels; i += 3) {

    ( * red)(pixel);
    pixel += 3;
  }
  pixel = (unsigned char * ) input -> imageData + 1;
  for (int i = 0; i < input -> width * input -> height * input -> nChannels; i += 3) {

    ( * green)(pixel);
    pixel += 3;
  }

    pixel = (unsigned char * ) input -> imageData + 2;
  for (int i = 0; i < input -> width * input -> height * input -> nChannels; i += 3) {

    ( * blue)(pixel);
    pixel += 3;
  }
}
```

4.3 Conversão para escala cinza

Algoritmo de escala de cinza baseado na luminosidade do pixel pela visão humana usando a fórmula: `L = R*0.3 + B*0.59 + G*0.11`. Dado o resultado o algoritmo salva o pixel na forma LLL. Primeiro convertemos a imagem em JPEG para PPM

(formato simples e sem compressão, sendo mais fácil a manipulação), então obtemos um buffer dos pixels, na classe Image.

Figura 3: Imagem convertida para escala cinza

```
for i in range(img.width):
    for j in range(img.height):
        pix = image.Pix(img.getPixel(i, j))
        lum = int(pix[0]*0.3 + pix[1]*0.59 + pix[2]*0.11)

        img.setPixel(i, j, image.Pix((lum, lum, lum)))
```

4.4 Rotacionamento de imagens

O método de rotação consiste em usar a matriz de pixel e deslocá-las de lugar como na imagem abaixo praticamente especificado.

Figura 4: Imagem rotacionada em sentido anti-horário

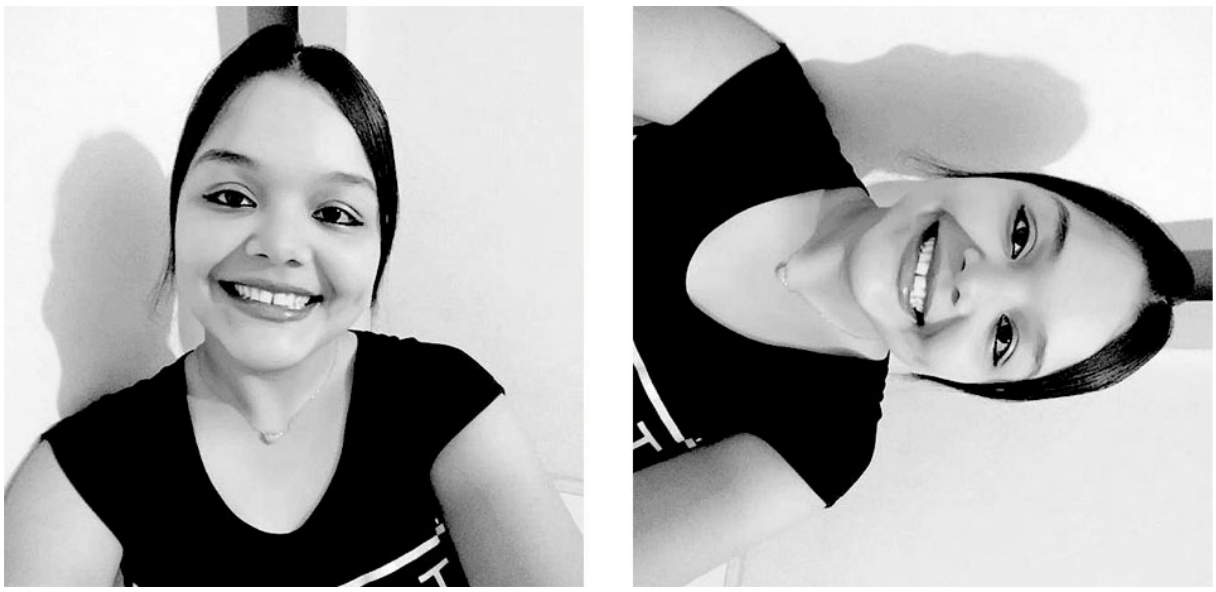

Figura 5: Imagem rotacionada em sentido horário

```
for x in xrange(xoffset,nw):
        for y in xrange(yoffset,nh):
            ox, oy = affine_t(x-cx, y-cy, *mrotate(-r, cx, cy))
            if ox > -1 and ox < ow and oy > -1 and oy < oh:
                pt = bilinear(bmp, ox, oy) if interpol else nn(bmp, ox, oy)
                draw.point([(x+mx,y+my),], fill=pt)
```

4.5 title

kkkk