Universidade de Uberaba
Faculdade de Engenharia da Computação
Professor Marcos Alberto Lopes da Silva

Avaliação 3º Momento
Manipulação de imagens

Samuel Gadiel de Ávila
RA 5158902
samuelgadiel@gmail.com

Uberlândia – Minas Gerais
2023

Introdução

Nessa atividade irei abordar o tratamento de imagem resolvendo os exercícios contidos nos slides das aulas 07 e 08. Para isso utilizei da linguagem de programação Dart, juntamente com a biblioteca Image. As referências se encontram a seguir

- `https://dart.dev/`
- `https://pub.dev/packages/image`

Os exercícios a serem abordados serão os seguintes

- 01 - Manipulação de contraste
- 02 - Manipulação de gama
- 03 - Fatiamento de bits

Serão apresentados o código aplicado para a resolução dos exercícios, bem como a imagem original e a imagem processada.

Exercício 01

Nesse caso foi fixado um valor mínimo e máximo de cinza para o ajuste de contraste. Sendo o valor mínimo de cinza 180 e o valor máximo de cinza 75.

Uma maneira de processar a imagem sem definir valores para o mínimo e máximo e cinza seria aplicar um método chamado "contraste automático", onde são ignorados uma pequena porcentagem dos pixels mais claros e mais escuros para encontrar um intervalo de contraste mais significativo.
Isso seria realizado em três passos:

1. Calcule a Distribuição de Luminância
2. Encontre os Limites de Contraste
3. Aplique o Ajuste de Contraste

Mas como dito, isso não foi abordado no exercício.

Código

```dart
import 'package:image/image.dart';

void main() async {
  // Carregar a imagem do arquivo especificado
  final image = await decodeImageFile('../Fig0241-a.png');

  // Verificar se a imagem foi carregada corretamente
  if (image == null) {
    throw Exception('Imagem nula');
  }

  // Inicializar valores de cinza mínimo e máximo para o ajuste de contraste
  // minGray é inicializado com um valor alto e maxGray com um valor baixo
  // Estes valores podem ser ajustados conforme a necessidade
  num minGray = 180;
  num maxGray = 75;

  // Percorrer todos os pixels da imagem para encontrar os valores
  // reais de cinza mínimo e máximo
  for (final Pixel pixel in image) {
    final grey = pixel.luminance;

    // Atualiza minGray se encontrar um valor de cinza mais baixo
    if (grey < minGray) minGray = grey;

    // Atualiza maxGray se encontrar um valor de cinza mais alto
    if (grey > maxGray) maxGray = grey;
  }

  // Ajustar o contraste de cada pixel na imagem
  for (final Pixel pixel in image) {
    final grey = pixel.luminance;

    // Calcula o novo valor de cinza com base no ajuste de contraste
    final adjustedGrey = ((grey - minGray) * 255 / (maxGray - minGray)).
   round().clamp(0, 255);

    // Atualiza o pixel na imagem com o novo valor de cinza
    image.setPixelRgb(pixel.x, pixel.y, adjustedGrey, adjustedGrey,
   adjustedGrey);
  }

  // Salva a imagem ajustada em um novo arquivo
  await encodePngFile('../output/Fig0241-a_contrast.png', image);

  print('Contraste aplicado e imagem salva com sucesso!');
}
```

Imagem original

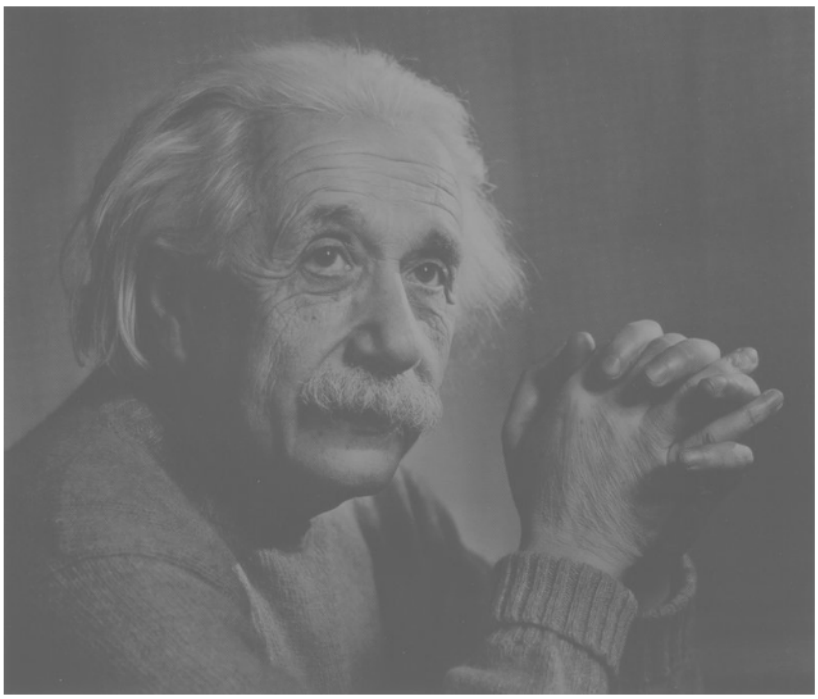
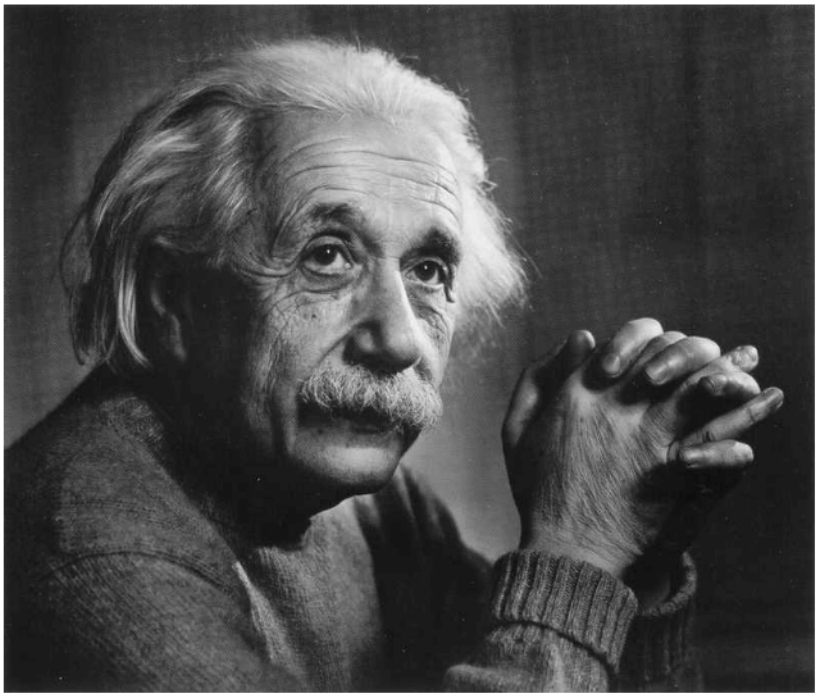


Imagem processada



— Exercício 02 ————————————————————

Veja que neste caso foi configurado o gamma para 0,3. Esse valor também é passivel de ajustes.

Código

```dart
import 'dart:math';

import 'package:image/image.dart';

// Função para aplicar a correção de gama a um pixel
Color applyGamma(Pixel pixel) {
  // Valor de gama utilizado na correção
  double gamma = 0.3; // Este valor pode ser ajustado conforme necessário

  // Calcular o inverso do valor de gama para aplicar a correção
  double gammaCorrection = 1 / gamma;

  // Aplicar a correção de gama a cada canal de cor do pixel
  int r = (pow(pixel.r / 255, gammaCorrection) * 255).round();
  int g = (pow(pixel.g / 255, gammaCorrection) * 255).round();
  int b = (pow(pixel.b / 255, gammaCorrection) * 255).round();

  // Criar um novo pixel com as cores corrigidas
  final pixelCorrigido = pixel.clone();
  pixelCorrigido.setRgb(r, g, b);

  return pixelCorrigido;
}

void main() async {
  // Carregar a imagem do arquivo especificado
  final image = await decodeImageFile('../Fig0309-a.png');

  // Verificar se a imagem foi carregada corretamente
  if (image == null) {
    throw Exception('Imagem nula');
  }

  // Percorrer todos os pixels da imagem e aplicar a correção de gama
  for (final Pixel pixel in image) {
    // Aplicar a correção de gama ao pixel atual
    final gammaCorrectedPixel = applyGamma(pixel);

    // Atualizar o pixel na imagem com o valor corrigido
    image.setPixel(pixel.x, pixel.y, gammaCorrectedPixel);
  }

  // Salvar a imagem ajustada em um novo arquivo
  await encodePngFile('../output/Fig0309-a_gama.png', image);

  print('Gama aplicado e imagem salva com sucesso!');
}
```

Imagem original

Imagem processada

Exercício 03

Código

```dart
import 'package:image/image.dart';

void main() async {
  final image = await decodeImageFile('../Fig0115-a.png');

  if (image == null) {
    throw Exception('Imagem nula');
  }

  int bitsPerChannel = image.bitsPerChannel;

  for (int bitIndex = 0; bitIndex < bitsPerChannel; bitIndex++) {
    Image bitSliced = Image.from(image);

    for (final Pixel pixel in image) {
      int gray = pixel.luminance.toInt();

      // Divide o valor de cinza pelas potências de 2 para acessar cada bit
      for (int i = 0; i < bitsPerChannel - 1 - bitIndex; i++) {
        gray = gray ~/ 2; // Divisão inteira por 2
      }

      // Verifica se o bit menos significativo é 1 ou 0
      int bitValue = (gray % 2) != 0 ? 255 : 0;

      bitSliced.setPixelRgb(pixel.x, pixel.y, bitValue, bitValue,
    bitValue);
    }

    // Salva a imagem resultante
    String outputPath = '../output/Fig0115-a_bit${bitIndex + 1}.png';
    await encodePngFile(outputPath, bitSliced);
  }

  print('Fatiamento de bit aplicado e imagens salvas com sucesso!');
}
```

Imagem original

Imagem processada