Universidade de Brasília
PPGI

Relatório de Resultados de Quantização e Difusão de erro em Imagens

03 de Outubro de 2023

Disciplina: Processamento de Imagens
Professor: Ricardo Lopes de Queiroz
Aluno: Henrique Tibério Brandão Vieira Augusto (221101092)

Metodologia

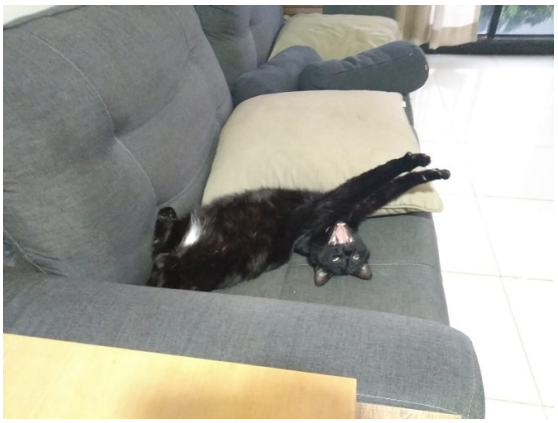
Escolher imagens para realizar os seguintes experimentos:

- Escolher um par de imagens (entre 500 e 700 pixels de largura ou altura) monocromáticas e realizar quantização com 64, 16, 8, 4 e 2 níveis de cinza e, em seguida, realizar difusão de erro com pesos de Floyd e Steinberg.
- Escolher um par de imagens (entre 500 e 700 pixels de largura ou altura) coloridas e realizar quantização com 4 e 2 níveis de cinza para cada canal (R, G, B) e, em seguida, realizar difusão de erro com pesos de Floyd e Steinberg.

Foram utilizadas os seguintes pares de imagens, respectivamente, para cada experimento:

- Primeiro:

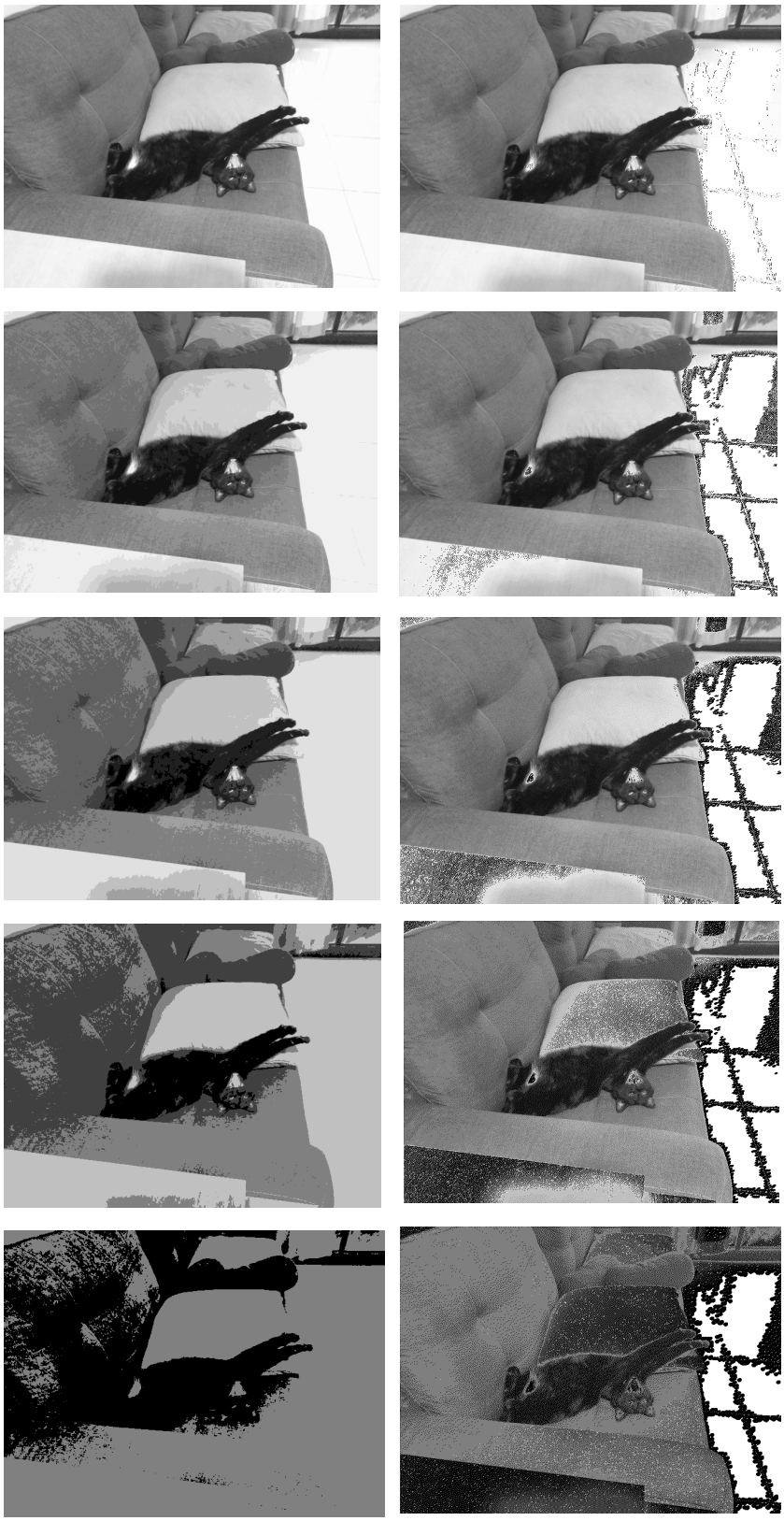

- Segundo:




Para ambos experimentos, o tamanho das imagens (largura e altura, em pixels) é de 666x500 para a imagem da esquerda (gato) e de 600x450 para a imagem da direita (lago).

Resultados

Para o primeiro experimento, foram obtidas as imagens abaixo. Estão dispostas, lado a lado, a imagem quantizada e sua versão com difusão de erro. Em ordem decrescente (64, 16, 8, 4 e 2) de tons de cinza:



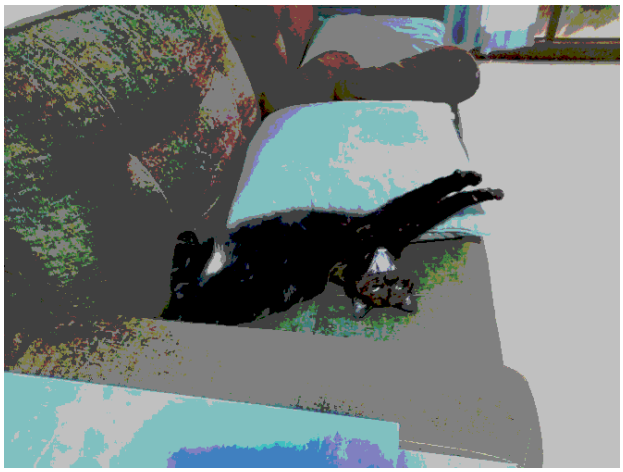


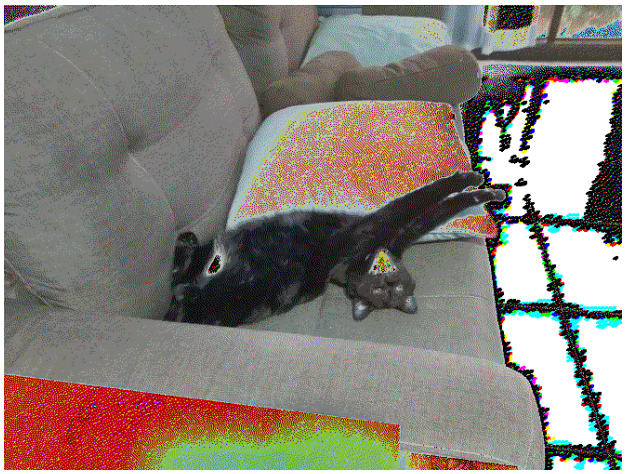


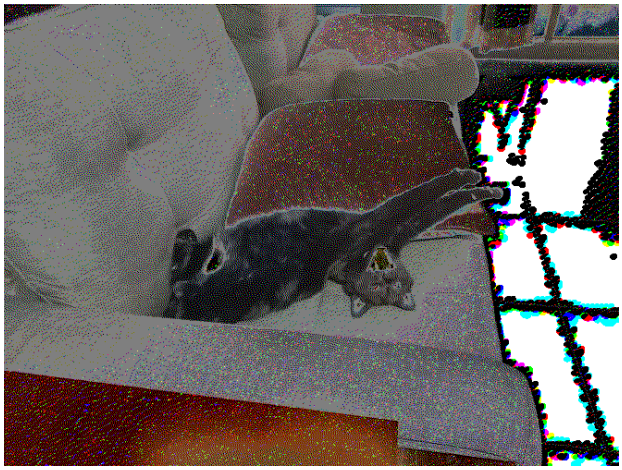


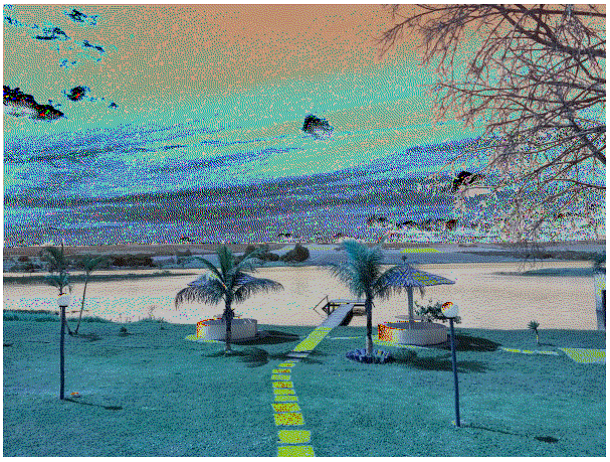


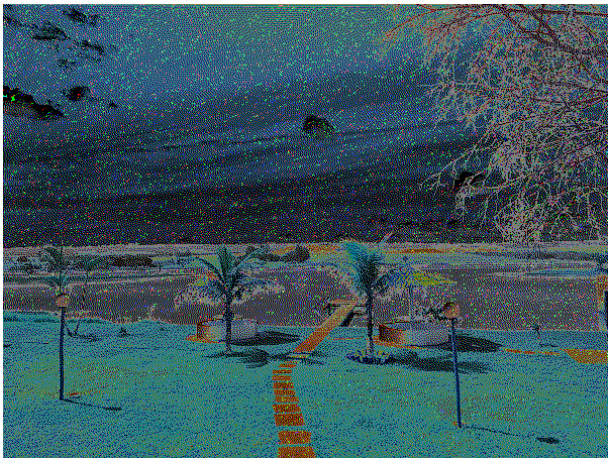

Para o segundo experimento, foram obtidas as imagens abaixo. Estão dispostas, lado a lado, a imagem quantizada e sua versão com difusão de erro. Em ordem decrescente (4 e 2) de tons de cinza:











Discussão dos Resultados

- Primeiro experimento:
  - Primeira imagem: A medida que reduzimos os tons de cinza da imagem, percebemos um aumento do contraste bem como a perda significativa de detalhes. Apesar disso noção de profundidade parece razoavelmente preservada, dado o ângulo de captura da imagem. Entretanto, para a parte inferior da imagem, ao alcançar apenas 2 tons de cinza, a mesa, o braço do sofá e o piso tendem a um mesmo elemento praticamente indistinguível. Ao observar as imagens com difusão de erro, percebemos que o contraste tende a se preservar, bem como o contorno dos objetos. Com isso, conseguimos preservar tanto a profundidade da imagem bem como, apesar da aparência ruidosa, as linhas que separam os azulejos se tornam presentes ao longo de todas as instâncias, e, essas mesmas linhas podem ser percebidas com mais clareza mesmo quando comparamos a última imagem (2 tons de cinza e difusão de erro) com a primeira imagem (64 tons de cinza sem difusão de erro).
  - Segunda imagem: A medida que reduzimos os tons de cinza da imagem, percebemos um aumento do contraste bem como a perda significativa de detalhes do céu, porém não tanto do gramado e do deck. Apesar disso noção de profundidade parece bem preservada, dado o ângulo de captura da imagem. Entretanto, na parte superior da imagem, onde percebíamos nuvens em tons distintos, estes elementos foram perdidos de modo que não há contraste algum que possa indicar sua existência prévia. Ao observar as imagens com difusão de erro, percebemos que o contraste não ficou tão preservado, vide quando comparamos o gramado entre ambas imagens com apenas 2 tons de cinza. A silhueta dos objetos (árvores e deck) ficou bastante difusa com os elementos ao redor. A noção de profundidade também foi prejudicada, vide a percepção de distância entre a beira do lago e as faixas de terra dentro do lago. Entretanto, quando comparamos as últimas imagens (2 tons de cinza), a difusão de erro manteve a presença dos elementos na parte superior da imagem, não presente na sua análoga, tanto para as nuvens mais esparsas quanto para as mais concentradas.
- Segundo experimento:
  - Primeira imagem: A medida que reduzimos os tons da imagem, percebemos uma perda significativa de detalhes bem como da noção de profundidade. Na parte inferior da imagem, ao alcançar apenas 2 tons, a mesa, o braço do sofá e o piso tendem a um mesmo elemento visual praticamente indistinguível. Ao observar as imagens com difusão de erro, percebemos que o contraste tende a se preservar. Não só o contorno dos objetos, como tons próximos aos iniciais, preservam - de certa forma - a iluminação e a noção de profundidade. A almofada e a parte visível da mesa, antes amareladas, são representadas por tons vermelhos, e, apesar da aparência ruidosa, as linhas que separam os azulejos se mantém presentes.
  - Segunda imagem: Assim como no primeiro experimento, a medida que reduzimos os tons da imagem, percebemos um aumento do contraste e a perda significativa de detalhes do céu, porém não tanto do gramado e do deck. A noção de profundidade parece preservada, dado o ângulo de captura da imagem. Na parte superior da imagem, os elementos do c[eu foram perdidos, restando um borrão amarelado que poderia ser confundido com a copa de uma árvore. Ao observar as imagens com difusão de erro, o contraste da grama não ficou tão preservado, embora a sombra das árvores esteja mais nítida em virtude disso. A silhueta dos objetos (árvores e deck) ficou bastante difusa com os elementos ao redor embora o contraste ajude a distinguí-los dos elementos próximos.

Conclusão

Nas imagens monocromáticas, a difusão de erro desempenhou um papel crucial na melhoria da qualidade visual das imagens quantizadas. Principalmente por trazer de volta alguns contornos e contrastes perdidos na imagem com apenas 2 tons de cinza.

Nas imagens coloridas, também houve uma grande perda de elementos, e a difusão de erro acarretou, novamente, na melhoria da qualidade visual das imagens quantizadas, embora alguns elementos tenham assumido tons distintos do original.

Com base nos resultados deste experimentos, podemos concluir que tanto a quantização quanto a difusão de erro têm um impacto significativo na qualidade visual das imagens. São técnicas são úteis para reduzir a quantidade de bits, mas é importante escolher a abordagem adequada para cada imagem.

Anexo

Detalhes da implementação e resultados estão disponíveis no repositório https://github.com/htbrandao/ppgi-proc-img/tree/master/trab01.

- Código em Python para processamento de imagem monocromática:

```
import cv2
import numpy as np

def processa_monocromatica(fp: str, quant: int):
    imagem_cinza = cv2.imread(fp, cv2.IMREAD_GRAYSCALE)
    niveis_quantizacao = quant
    imagem_cinza_quantizada = np.uint8(np.floor(imagem_cinza / 256.0 * niveis_quantizacao) *
(256.0 / niveis_quantizacao))
    def apply_dithering(image, quant):
        h, w = image.shape
        for y in range(h):
            for x in range(w):
                old_pixel = image[y, x]
                new_pixel = np.uint8(np.floor(old_pixel / 255.0 * quant) * (255.0 / quant))
                error = old_pixel - new_pixel
                image[y, x] = new_pixel
                if x < w - 1:
                    image[y, x + 1] += error * 7 / 16
                if x > 0 and y < h - 1:
                    image[y + 1, x - 1] += error * 3 / 16
                if y < h - 1:
                    image[y + 1, x] += error * 5 / 16
                if x < w - 1 and y < h - 1:
                    image[y + 1, x + 1] += error * 1 / 16
        return image
    imagem_cinza_dithered = apply_dithering(imagem_cinza, quant)
    return imagem_cinza_quantizada, imagem_cinza_dithered
```

- Código em Python para processamento de imagem colorida:

```
import cv2
import numpy as np

def processa_colorida(fp: str, quant: int):
    imagem = cv2.imread(fp)
    canal_azul, canal_verde, canal_vermelho = cv2.split(imagem)
    niveis_quantizacao = quant
    canal_azul_quantizado = np.uint8(np.floor(canal_azul / 256.0 * niveis_quantizacao) * (256 / niveis_quantizacao))
    canal_verde_quantizado = np.uint8(np.floor(canal_verde / 256.0 * niveis_quantizacao) * (256 / niveis_quantizacao))
    canal_vermelho_quantizado = np.uint8(np.floor(canal_vermelho / 256.0 * niveis_quantizacao) * (256 / niveis_quantizacao))
    imagem_quantizada = cv2.merge((canal_azul_quantizado, canal_verde_quantizado, canal_vermelho_quantizado))
    def apply_dithering(channel, quant):
        h, w = channel.shape
        for y in range(h):
            for x in range(w):
                old_pixel = channel[y, x]
                new_pixel = np.uint8(np.floor(old_pixel / 255.0 * quant) * (255.0 / quant))
                error = old_pixel - new_pixel
                channel[y, x] = new_pixel
                if x < w - 1:
                    channel[y, x + 1] += error * 7 / 16
                if x > 0 and y < h - 1:
                    channel[y + 1, x - 1] += error * 3 / 16
                if y < h - 1:
                    channel[y + 1, x] += error * 5 / 16
                if x < w - 1 and y < h - 1:
                    channel[y + 1, x + 1] += error * 1 / 16
        return channel
    canal_azul_dithered = apply_dithering(canal_azul, quant)
    canal_verde_dithered = apply_dithering(canal_verde, quant)
    canal_vermelho_dithered = apply_dithering(canal_vermelho, quant)
    imagem_dithered = cv2.merge((canal_azul_dithered, canal_verde_dithered, canal_vermelho_dithered))
    return imagem_quantizada, imagem_dithered
```