Universidade da Beira Interior
Departamento de Informática

Nº 1 - 2022: *Trabalho Prático de Multimédia*

Elaborado por:

45842 Diogo Manuel Paiva dos Santos
39740 José Miguel Reis Castanheira dos Santos

Orientador:

João Alfredo Fazendeiro Dias

17 de janeiro de 2022

Agradecimentos

A conclusão do trabalho prático de Multimédia não seria possível sem a ajuda do Professor Doutor João Alfredo Fazendeiro Dias, tanto pelas bases teóricas e práticas lecionadas nas aulas como também pela ajuda individual, quando pedida, durante as mesmas.

Conteúdo

Lista de Figuras

Acrónimos

UBI Universidade da Beira Interior

Capítulo

1

Introdução

1.1 Enquadramento

Este documento foi feito no contexto da Unidade Curricular de Multimédia da Universidade da Beira Interior (UBI), funcionando como relatório do trabalho prático final da mesma cadeira.

Este trabalho, cotado para 8 valores, tenciona solidifcar os conhecimentos adquiridos tanto nas aulas teóricas como no trabalho autónomo pessoal.

1.2 Motivação UBI

A realização deste projeto pretende a consolidação dos conteúdos lecionados na unidade curricular, nomeadamente aos que são relativos à modificação e compressão de tipos de informação multimédia. Pretende-se também o aprimoramento da capacidade de pesquisa, trabalho de equipa e distribuição de tarefas por parte dos integrantes deste grupo de trabalho.

1.3 Objetivos

Este projeto tem como objetivo a familiarização face aos vários tipos de informação multimédia, assim como relativamente aos melhores métodos de manipular e comprimir essa informação. Para que tal se suceda, será necessário ter em conta os seguintes passos:

- **Pesquisa** dos melhores métodos para a realização do algoritmo, assim como a elaboração do mesmo;
- Conceção de *scripts* alusivos à modificação dos ficheiros multimédia;
- Realização de um programa principal que aloje os *scripts* e um menu navegador do programa;
- Elaboração do **Relatório**.

1.4 Distribuição de tarefas

As tarefas deste projeto foram distribuídas uniformemente, em que os dois elementos integrantes sempre estiveram presentes no seu processo construtivo. Apesar disto, marcam-se abaixo as contribuições mais relevantes que os elementos proporcionaram (individualmente):

- **Pesquisa:** Diogo Santos, José Miguel Santos;
- **Imagem:** Diogo Santos, José Miguel Santos;
- **Vídeo:** Diogo Santos;
- **Audio:** Diogo Santos;
- **Menu:** José Miguel Santos;
- **Relatório:** José Miguel Santos;

Capítulo

2

Algoritmos

2.1 Introdução

Neste capítulo apresentamos o algoritmo escolhido e as suas demais funções. É descrito o código utilizado e as funções incluídas no mesmo.

O programa foi concebido na linguagem de programação ***python***, devido à versatilidade e produtividade que a linguagem proporciona, assim como devido à enorme quantidade de informação relativa à mesma na *internet*.

2.2 Visão geral do algoritmo

A abundância de *scripts* no algoritmo levou-nos à construção de um menu navegador, facilitando o acesso sobre as demais operações do programa:

```
- - - - - Menu Principal - - - - -
    1. Modificacao de Imagem
    2. Modificacao de Audio
    3. Modificacao de Video
    4. Sair
- - - - - - - - - - - - - - - - - -
```

(código)

```
# MENU PRINCIPAL
def mainMenu():
    menuChoice = 0
    while menuChoice < 1 or menuChoice > 4:
        print("- - - - - Menu Principal - - - - -\n\t1. Modificacao de
            Imagens"
              "\n\t2. Modificacao de Audio"
              "\n\t3. Modificacao de Video"
              "\n\t4. Sair"
              "\n- - - - - - - - - - - - - - - - - - -")
        try:
            menuChoice = int(
                input("Introduza o numero relativo    sua escolha aqui:
                    "))
            if menuChoice < 1 or menuChoice > 4:
                raise ValueError("Valor invalido. Introduza novamente.")
        except ValueError as ve:
            print(ve)
    print("\n"*10)
    return menuChoice
```

Excerto de Código 2.1: Menu principal utilizado no projeto.

2.3 Funções de Imagem

Foram elaboradas sete funções diferentes que modificam ou relacionam de certa forma duas imagens distintas:

- Função **Soma**: Lê duas imagens, guarda-as em variáveis e cria uma terceira. Ciclos *for* assimilam os pixeis das duas imagens em determinada posição e somam o seu valor RGB - valor resultante que é igualado ao do pixel na mesma posição da terceira imagem criada pelo programa. O valor RGB desses pixéis é sempre limitado entre 0 e 255.

- Função **Subtração**: Realiza operações idênticas à **função soma**, mas subtrai em vez de somar - o valor RGB é sempre limitado entre 0 e 255.

- Função ***OR***: Realiza operações idênticas à **função soma**, mas usa a operação lógica *OR* em vez de somar - o valor RGB é sempre limitado entre 0 e 255;

- Função ***AND***: Realiza operações idênticas à **função soma**, mas usa a operação lógica *AND* em vez de somar - o valor RGB é sempre limitado entre 0 e 255;

- Função ***Greenscreen***: Lê e guarda uma imagem numa variável, cria uma segunda com o mesmo tamanho. Dois ciclos *for* percorrem a imagem, guardando o valor G de determinada posição e igualando o pixel, na mesma posição, da segunda imagem a esse mesmo G, colocando o valor de R e B a zero;

- Função **Negativo**: Processo semelhante à ***função greenscreen***, só que em vez de zerar o R, B e manter o valor de G, subtrai antes os valores encontrados a 255 e armazena o resultado no pixel, da mesma posição, na segunda imagem criada;

- Função **Cinza**: Processo semelhante à ***função negativo***, só que multiplica os valores RGB encontrados por um determinado valor, e armazena o resultado no pixel, da mesma posição, na segunda imagem criada;

```
def somar():
    img1 = Image.open("imagens/foto1.jpg")
    img2 = Image.open("imagens/foto2.jpg")
    img3 = Image.new(img1.mode, img1.size, "white")  # vai ser alterada

    for i in range(0, img1.size[0]):
        for j in range(0, img1.size[1]):

            pixel1 = img1.getpixel((i, j))
            pixel2 = img2.getpixel((i, j))

            # Soma os pixeis
            novopixelR = (pixel1[0] + pixel2[0],
                          255)[pixel1[0] + pixel2[0] > 255]

            novopixelG = (pixel1[1] + pixel2[1],
                          255)[pixel1[1] + pixel2[1] > 255]

            novopixelB = (pixel1[2] + pixel2[2],
                          255)[pixel1[2] + pixel2[2] > 255]

            img3.putpixel((i, j), (novopixelR, novopixelG, novopixelB))

    img3.save("imagens/somar.jpg")
```

Excerto de Código 2.2: Função Soma da secção Modificação de Imagem

```
def subtrair():
    img1 = Image.open("imagens/foto1.jpg")
    img2 = Image.open("imagens/foto2.jpg")
    img3 = Image.new(img1.mode, img1.size, "white")  # vai ser alterada
```

```
    for i in range(0, img1.size[0]):
        for j in range(0, img1.size[1]):

            pixel1 = img1.getpixel((i, j))
            pixel2 = img2.getpixel((i, j))

            novopixelR = (pixel1[0] - pixel2[0], 0)[pixel1[0] - pixel2
                [0] < 0]

            novopixelG = (pixel1[1] - pixel2[1], 0)[pixel1[1] - pixel2
                [1] < 0]

            novopixelB = (pixel1[2] - pixel2[2], 0)[pixel1[2] - pixel2
                [2] < 0]

            img3.putpixel((i, j), (novopixelR, novopixelG, novopixelB))

    img3.save("imagens/subtrair.jpg")
```

Excerto de Código 2.3: Função Subtração da secção Modificação de Imagem

```
def ou():
    img1 = Image.open("imagens/foto1.jpg")
    img2 = Image.open("imagens/foto2.jpg")
    img3 = Image.new(img1.mode, img1.size, "white")  # vai ser alterada
    for i in range(0, img1.size[0]):
        for j in range(0, img1.size[1]):

            pixel1 = img1.getpixel((i, j))
            pixel2 = img2.getpixel((i, j))

            novopixelR = pixel1[0] | pixel2[0]

            novopixelG = pixel1[1] | pixel2[1]

            novopixelB = pixel1[2] | pixel2[2]

            img3.putpixel((i, j), (novopixelR, novopixelG, novopixelB))

    img3.save("imagens/or.jpg")
```

Excerto de Código 2.4: Função OR da secção Modificação de Imagem

```
def e():
    img1 = Image.open("imagens/foto1.jpg")
    img2 = Image.open("imagens/foto2.jpg")
    img3 = Image.new(img1.mode, img1.size, "white")  # vai ser alterada
    for i in range(0, img1.size[0]):
        for j in range(0, img1.size[1]):
```

```
            pixel1 = img1.getpixel((i, j))
            pixel2 = img2.getpixel((i, j))

            novopixelR = pixel1[0] & pixel2[0]

            novopixelG = pixel1[1] & pixel2[1]

            novopixelB = pixel1[2] & pixel2[2]

            img3.putpixel((i, j), (novopixelR, novopixelG, novopixelB))

    img3.save("imagens/and.jpg")
```

Excerto de Código 2.5: Função AND da secção Modificação de Imagem

```
def filtroverde():
    img1 = Image.open("imagens/foto2.jpg")
    img2 = Image.new(img1.mode, img1.size, "white")  # vai ser alterada

    for i in range(0, img1.size[0]):
        for j in range(0, img1.size[1]):

            pixel1 = img1.getpixel((i, j))
            # Soma os pixeis
            novopixelR = 0

            novopixelG = pixel1[1]

            novopixelB = 0

            img2.putpixel((i, j), (novopixelR, novopixelG, novopixelB))

    img2.save("imagens/verde.jpg")
```

Excerto de Código 2.6: Função Greenscreen da secção Modificação de Imagem

```
def negativo():
    image = Image.open("imagens/foto1.jpg")

    # criar a nova imagem

    negativo = Image.new(image.mode, image.size, "white")

    for i in range(0, image.size[0]):
        for j in range(0, image.size[1]):

            pixel = image.getpixel((i, j))

            # Inverter a cor
```

```
            r = 255 - pixel[0]
            g = 255 - pixel[1]
            b = 255 - pixel[2]

            negativo.putpixel((i, j), (r, g, b))

    negativo.save("imagens/negativo.jpg")
```

Excerto de Código 2.7: Função Negativo da secção Modificação de Imagem

```
def cinzentaImagem():
    img = Image.open('imagens/foto1.jpg')
    cinzenta = Image.new(img.mode, img.size, "white")

    for i in range(0, img.size[0]):
        for j in range(0, img.size[1]):

            pixel = img.getpixel((i, j))

            # mete a cinzenta

            corCizenta = int((pixel[0]*0.2125+pixel[1]*0.7174+pixel
                [2]*0.0721))

            cinzenta.putpixel((i, j), (corCizenta, corCizenta,
                corCizenta))

    cinzenta.save("imagens/cinzenta.jpg")
```

Excerto de Código 2.8: Função Cinza da secção Modificação de Imagem

2.4 Funções de Audio

Foram elaboradas três funções diferentes que modificam ou relacionam ficheiros de audio:

- Função **Cortar**: Lê um *audio1.wav*, divide a duração em duas, guardando cada uma numa variável diferente, ficando ambas guardadas em formato *.wav*;
- Função **Juntar**: Lê dois *audio.wav*, guarda-os em variáveis distintas, e junta-as numa de seguida através de uma operação de soma, colocando o resultado numa nova variável, sendo depois exportada em *.wav*.
- Função ***Eco***: É lida a faixa de audio em que se quer pôr o efeito de eco. São criadas duas variáveis de audio, cada uma com um segundo de si-

lêncio, uma ao início e outra ao fim - tendo a faixa com o segundo de silêncio ao início os decibeis reduzidos. Juntam-se ambas as faixas com a operação de multiplicação e exporta-se o resultado (*em .wax*).

- Função **Normalizar**: Lê uma faixa de audio, reduzem-se os decibéis em 10 e exporta-se em *.wav*.

```
def cortar():
    audio1 = AudioSegment.from_wav("audio/audio1.wav")
    duracao_audio1 = len(audio1)
    metade = duracao_audio1/2
    primeira_parte = audio1[:metade]
    ultima_parte = audio1[-metade:]

    primeira_parte.export("audio/audio1_1m.wav", format="wav")
    ultima_parte.export("audio/audio1_2m.wav", format="wav")
```

Excerto de Código 2.9: Função Cortar da secção Modificação de Imagem

```
def juntar():
    audio1 = AudioSegment.from_wav("audio/audio1.wav")
    audio2 = AudioSegment.from_wav("audio/audio2.wav")

    audio3 = audio1 + audio2

    audio3.export("audio/juncao.wav", format="wav")
```

Excerto de Código 2.10: Função Juntar da secção Modificação de Imagem

```
def eco():
    um_segundo_silencio = AudioSegment.silent(duration=1000)
    audio1 = AudioSegment.from_wav("audio/audio1.wav")
    musica_por_cima = audio1 + um_segundo_silencio
    musica_de_fundo = um_segundo_silencio + audio1
    musica_de_fundo -= 4
    eco = musica_por_cima * musica_de_fundo
    eco.export("audio/eco.wav", format="wav")
```

Excerto de Código 2.11: Função Eco da secção Modificação de Imagem

```
def normaliza():
    audio1 = AudioSegment.from_wav("audio/audio1.wav")
    normalizado = audio1 - 10  # REDUZIR 10 decib  is

    normalizado.export("audio/norma.wav", format="wav")
```

Excerto de Código 2.12: Função Normalizar da secção Modificação de Imagem

2.5 Modificação de Vídeo

Esta secção apenas possui uma função, que transforma determinado vídeo na sua integridade numa versão a preto e branco.

- Função **Preto e Branco**: Um vídeo é lido e colocado numa variável, de seguida uma nova variável de vídeo é criada, com as mesmas propriedades que o vídeo lido (exceto FPS). Um ciclo while percorre o vídeo original, frame a frame, no qual os valores BGR de cada pixel são multiplicados por determinado valor e armazenados no segundo vídeo - efetivamente transformando-o em preto e branco.

```
import cv2

# pega numa imagem e mete em preto e branco e devolve
def gray(img, w, h):
    for i in range(w):
        for j in range(h):
            #BGR
            pixel = int((img[j,i][0]*0.2125+img[j,i][1]*0.7174+img[j,i
                ][2]*0.0721))
            img[j,i] =[pixel, pixel, pixel]
    return img
# Funcao do video
def video():
    # le o video
    original = cv2.VideoCapture('imagens/video.mp4')
    # Resolu  o do video - converter de float para int
    frame_width = int(original.get(3))
    frame_height = int(original.get(4))

    size = (frame_width, frame_height)
    result = cv2.VideoWriter('imagens/Preto_E_Branco.mp4',
                             cv2.VideoWriter_fourcc(*'m','p','4','v'),
                             30, size)  # o numero s o os frames por
                                 segundo

    ret = bool(True)
    # corre o loop
    while ret:

        # extrariar o frame
        ret, img = original.read()
        if ret == True:
            h, w, canais = img.shape
            gray1 = gray(img, w, h)
            # guardar no ficheiro em preto e branco
```

```
            result.write(gray1)

    original.release()
    result.release()
```

Excerto de Código 2.13: Função Preto e Branco da Modificação de vídeo

Capítulo

3

Resultados

3.1 Introdução

Neste capítulo pretende-se demonstrar a aplicação do algoritmo feito nos variados ficheiros de informação multimédia.

3.2 Aplicação dos Algoritmos em Imagem

3.2.1 Imagens Originais

As funções e operações do algoritmo foram realizadas sobre as duas imagens abaixo, havendo até operações que relacionam ambas entre si *(Soma, Subtração, OR, AND)*.

Figura 3.1: Imagens utilizadas

3.2.2 Operações Aritméticas

O primeiros algoritmos a serem construídos foram referentes às funções de soma e subtração de imagem. São operações aritméticas simples, idênticas a nível de código, mas claramente diferentes a nível visual, conforme evidenciado abaixo:

Figura 3.2: Soma

Figura 3.3: Subtrair

3.2.3 Operações Lógicas

As imagens abaixo foram alvo de uma operação lógica *(OR e AND, respetivamente)*:

Figura 3.4: *OR*

Figura 3.5: *AND*

3.2.4 Aplicação de filtros

Nos dias de hoje a aplicação de filtros a imagens é uma prática bastante comum. Abaixo seguem imagens nas quais foram aplicados algoritmos de filtro, um diferente para cada imagem:

Figura 3.6: *Filtro da cor verde*

Figura 3.7: *Negativo da Imagem*

Figura 3.8: Filtro Cinzento

3.3 Aplicação dos Algoritmos em Audio

Nesta secção abordam-se os resultados gerados pelo algoritmo em ficheiros de áudio, tendo utilizado audios anteriormente fornecidos pelo professor, durante uma aula prática. Os ficheiros audio estão em formato *".wav"*.

3.3.1 Operações de Edição

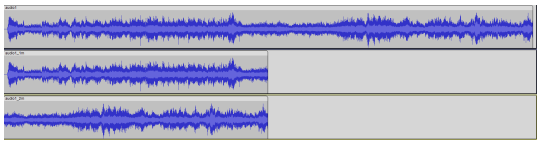

Figura 3.9: Resultado ilustrado da função Cortar Audio

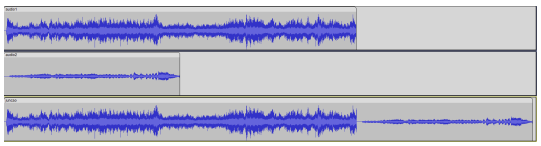

Figura 3.10: Resultado ilustrado da função Juntar Audio

3.3.2 Aplicação de Filtros

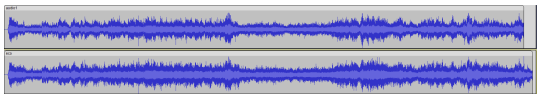

Figura 3.11: Resultado ilustrado da função Eco

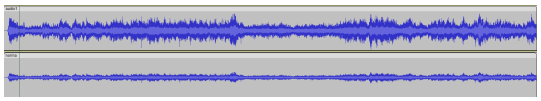

Figura 3.12: Resultado ilustrado da função Normalizar (-10db)

3.4 Aplicação dos Algoritmos em Video

De seguida, colocamos duas *Screenshots*, uma delas do video original(com cores), outra do video alterado, isto é, a preto e branco.

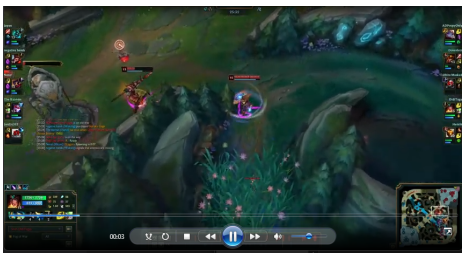

Figura 3.13: *Screenshot* do Video Original

Figura 3.14: *Screenshot* do Video em Preto e Branco

Capítulo

4

Conclusões

4.1 Introdução

Neste capítulo será apresentada uma reflexão sobre o trabalho realizado, dificuldades encontradas, e sobre o que ficou por realizar no projeto desta Unidade Curricular.

4.2 Conclusões e Trabalho Futuro

Neste trabalho foi possível aos integrantes do grupo não só consolidar os conhecimentos adquiridos durante as aulas teóricas e práticas, como também desenvolver *soft-skills* importantes, nomeadamente o trabalho de equipa e competências de investigação.

Na fase inicial deste trabalho não foram encontradas dificuldades por parte dos integrantes - a construção do algoritmos foi inicializada pelas secções mais fáceis, de forma a agilizar a elaboração do projeto. Numa fase mais final o mesmo não se verificou, visto até nem ter sido possível ir de encontro a todas as operações pedidas pelo Professor. Entre as que foram efetivamente realizadas, a elaboração do código para a secção de vídeo foi a mais problemática, na qual o grupo teve dificuldade em recuperar cada frame do vídeo de forma a fazer alterações visuais ao mesmo. O grupo não foi capaz de realizar a operação que comprime/descomprime ficheiros multimédia.

Num trabalho futuro, tendo em conta que os prazos não foram cumpridos, procuraremos ter mais eficácia no concebimento tanto do código como do relatório. Pretende-se também, para um trabalho futuro, uma investigação mais aprofundada, e uma melhor organização de tempo de forma a realizar todos os objetivos soliticados pelo Professor.

Bibliografia

[1] Pydub
https://github.com/jiaaro/pydub/blob/master/API.markdown

[2] Medium
https://medium.com/@brunobastos_1998/opera%C3%A7%C3%B5es-aritm%C3%A9ticas-e-l%C3%B3gicas-no-processamento-digital-de-imagem-5c5016c525f8

[3] GeeksforGeeks
urlhttps://www.geeksforgeeks.org/converting-color-video-to-grayscale-using-opencv-in-python/

[4] W3schools
urlhttps://www.w3schools.com/python/