

Processamentos de Imagens



Cruzeiro do Sul Virtual
Educação a distância

Material Teórico



**Relacionando o Processamento de
Imagens com Vídeo e Compressão**

Responsável pelo Conteúdo:

Prof. Me. Josivan Pereira da Silva

Revisão Textual:

Prof.^a M.^a Sandra Regina Fonseca Moreira

UNIDADE

Relacionando o Processamento de Imagens com Vídeo e Compressão



- O Conceito da Compressão;
- Compressão e o Conceito de Redundância de Informação;
- Principais Formatos de Imagem;
- Principais Formatos de Vídeo;
- Prática de Compressão com *Python* e *OpenCV*;
- Medida de Qualidade de Imagem PSNR;
- Compressão de Vídeo a partir de Imagens.

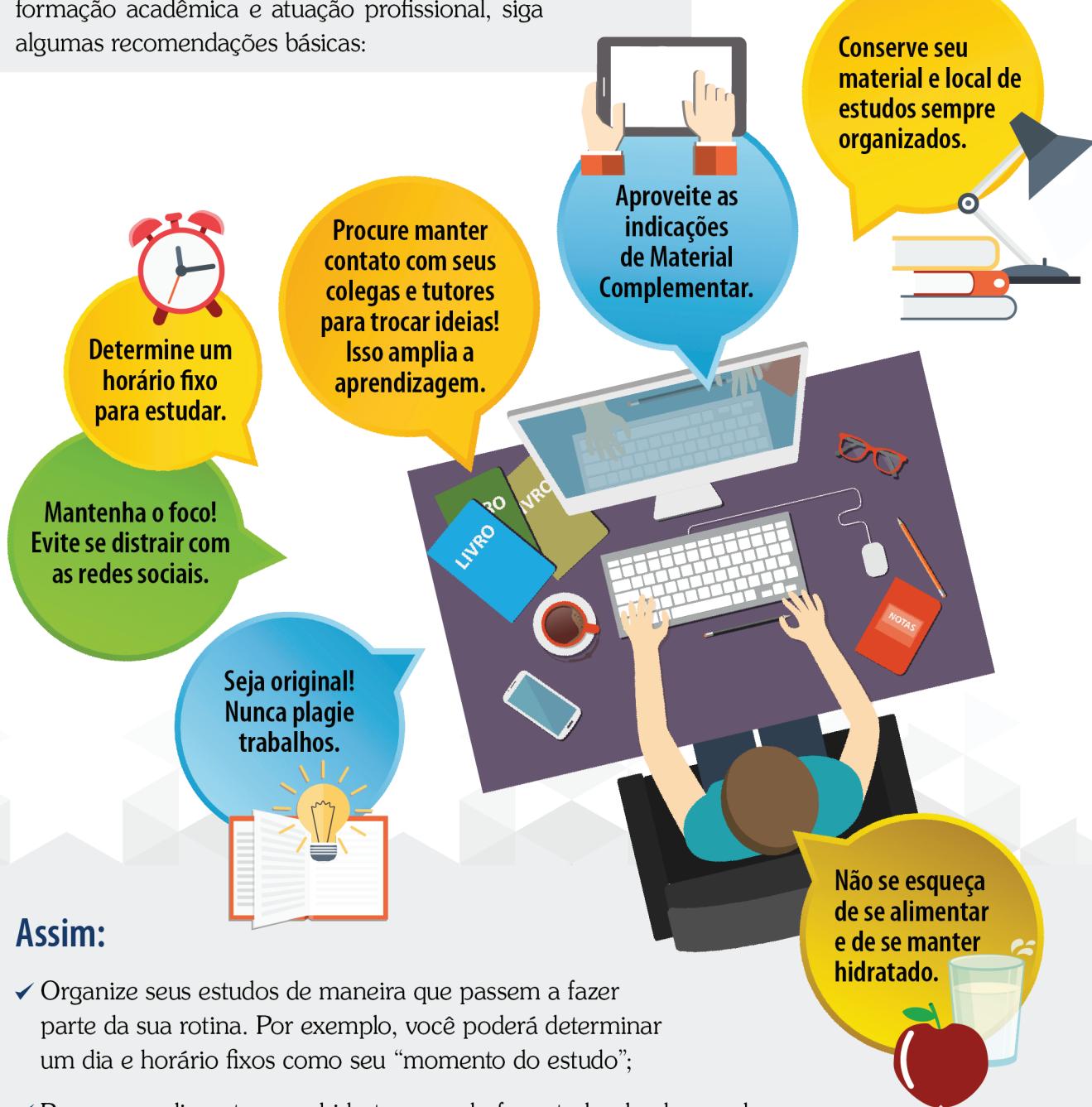


OBJETIVO DE APRENDIZADO

- Entender a compressão de imagem e conseguir economizar memória/espaço com a compressão tanto em imagem quanto em vídeo.

Orientações de estudo

Para que o conteúdo desta Disciplina seja bem aproveitado e haja maior aplicabilidade na sua formação acadêmica e atuação profissional, siga algumas recomendações básicas:



Assim:

- ✓ Organize seus estudos de maneira que passem a fazer parte da sua rotina. Por exemplo, você poderá determinar um dia e horário fixos como seu “momento do estudo”;
- ✓ Procure se alimentar e se hidratar quando for estudar; lembre-se de que uma alimentação saudável pode proporcionar melhor aproveitamento do estudo;
- ✓ No material de cada Unidade, há leituras indicadas e, entre elas, artigos científicos, livros, vídeos e sites para aprofundar os conhecimentos adquiridos ao longo da Unidade. Além disso, você também encontrará sugestões de conteúdo extra no item **Material Complementar**, que ampliarão sua interpretação e auxiliarão no pleno entendimento dos temas abordados;
- ✓ Após o contato com o conteúdo proposto, participe dos debates mediados em fóruns de discussão, pois irão auxiliar a verificar o quanto você absorveu de conhecimento, além de propiciar o contato com seus colegas e tutores, o que se apresenta como rico espaço de troca de ideias e de aprendizagem.

O Conceito da Compressão

Compressão de Imagem significa representar uma imagem de outra forma que consiga mostrar a imagem com informação semelhante, mas de forma a reduzir o espaço que ela ocupa de armazenamento ou que utiliza em transmissão (ARORA; SHUKLA, 2014).

Neste material serão mostradas imagens de vários aspectos e formatos e também serão explicadas analogias com vídeos, já que um vídeo tem uma forte relação com arquivos de imagens, e a questão da compressão é importantíssima tanto em imagem quanto vídeo.



Para facilitar o trabalho com imagens e com vídeos foi utilizado um vídeo que é um filme aberto com licença *Creative Commons*, chamado *Big Buck Bunny* (*Big Buck Bunny*, 2008). Esse filme está disponível em: <https://bit.ly/3nZZfpz>

A Figura 1 apresenta uma imagem pertencente ao filme *Big Buck Bunny*.



Figura 1 – Imagem integrante do filme *Big Buck Bunny* rodando diretamente do computador do autor (*printscreen* do software *VLC Player*)

Fonte: Acervo do Conteudista

Vamos aproveitar e fazer analogias utilizando esse filme e algumas plataformas da internet que trabalham com filmes e imagens.

A Figura 2 mostra informações sobre um vídeo MP4 que é um trecho de 1 minuto do Filme *Big Buck Bunny*. Como demonstrado na Figura 2, o trecho de apenas 1 minuto ocupa 5,25 MB de espaço no Disco Rígido do Computador.

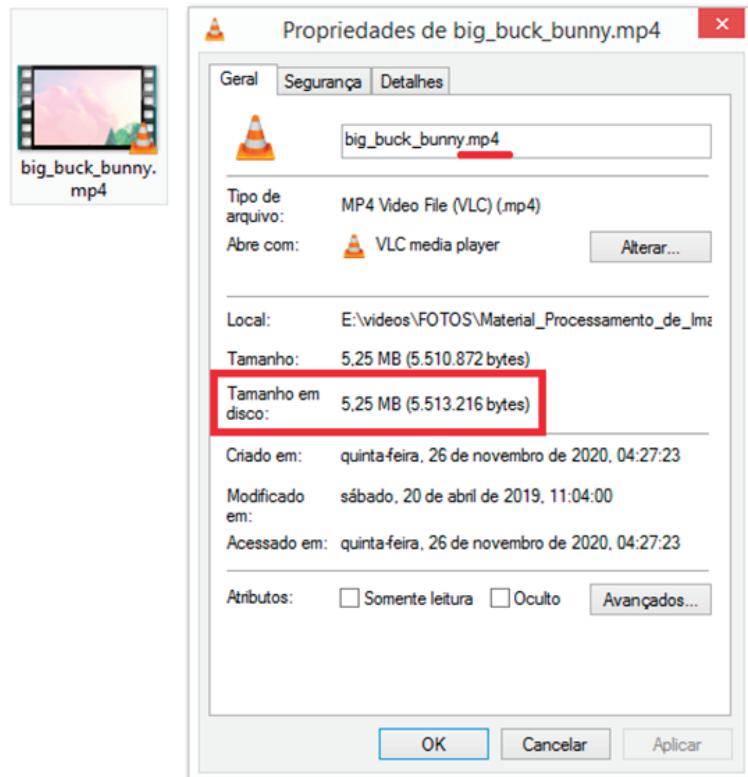


Figura 2 – Demonstração do espaço de armazenamento de 1 minuto do filme *Big Buck Bunny* (printscreen do Sistema Operacional *Windows 8*)

Fonte: Acervo do Conteudista

O Filme *Big Buck Bunny* é um curta-metragem de licença livre (*Creative Commons*) com a duração de 10 minutos. O arquivo MP4 do filme completo consta de 317 MB de espaço de armazenamento em disco. Esse filme é exibido em uma taxa de 24 FPS (*Frames per Second*), ou seja, cada segundo de vídeo é composto de 24 imagens, portanto, o filme completo tem aproximadamente 14.400 imagens.

Pensando que um filme de 10 minutos pode conter 14.400 imagens, imagine o seguinte:

Você vai comprar um HD externo para guardar os vídeos e fotografias que você faz com o seu *smartphone* ou câmera digital. Você vai gastar um dinheiro que é um recurso importante para você e, provavelmente, vai querer que caibam muitas fotografias e vídeos ali naquele equipamento. Contudo, se você chegar no limite da capacidade do HD externo, você terá que comprar outro HD para continuar armazenando seus arquivos, certo?

Agora imagine uma situação semelhante, agora você quer guardar roupas em uma gaveta. Se você armazenar as roupas de qualquer forma, logo a gaveta estará cheia e pode ser que algumas roupas fiquem do lado de fora. Se você conseguir dobrar essas roupas de forma mais estratégica e ir colocando as roupas na gaveta de forma cuidadosa, tentando encaixar as roupas nos espaços distribuídos, a chance de você conseguir armazenar cada vez mais roupas nessa mesma gaveta vai aumentar.

Nos exemplos anteriores, das fotografias e filmes armazenados no HD externo e das roupas armazenadas na gaveta, a técnica que te ajudaria seria uma compressão, ou seja, uma forma de deixar os objetos a serem armazenados de uma forma mais compacta para o armazenamento, o que gera uma economia de recursos importante.

Compressão e o Conceito de Redundância de Informação

As Técnicas de compressão de imagens digitais tratam da diminuição da quantidade de informação necessária para armazenar imagens. Atualmente, transmite-se e armazena-se uma quantidade de imagens cada vez maior (PAWE; MANDAOGADE, 2015).

Um exemplo prático: para transmitir imagens em HD para o padrão exigido em transmissão de TV, necessita-se de imagens com pelo menos 1280x720 de resolução (*pixels*), com 24 *bits* para descrever os *pixels* a 60 quadros por segundo (FPS), o que daria 1,32 Gbits por segundo, exigindo uma compressão de imagens para gerar uma economia valiosa (CONCI *et al.*, 2008).

Redundante é algo em que há excesso, com repetição. É fato que muitos conjuntos de dados contêm informações redundantes, que podem ou precisam ser eliminados de alguma forma. Comprimir informação pode ser entendido como uma maneira de retirar redundâncias da informação.

Por exemplo: a sequência B1B1B1B1 ocupa 4 *Bytes* em sistema Hexadecimal, mas, se representada como 4B1 pode ocupar 2 *Bytes*, o que já é uma boa redução de espaço (CONCI *et al.*, 2008).

Vários tipos de redundância de informação podem ser encontradas em imagens, por exemplo, relacionadas à codificação de cores.

Quanto mais cores forem colocadas para representar uma imagem, mais *bits* serão necessários, porém, há pequenas variações de cores que não são tão percebidas pelo olho humano. Essas variações podem ser agrupadas em uma única cor para tentar diminuir o número de cores utilizadas para representar uma imagem e, assim, tentar diminuir a quantidade de informações.

As imagens contidas na Figura 3 demonstram uma imagem de 16 *Bits* para representação das cores à esquerda, e outra, com 8 *bits*, para representação das cores à direita. Aos olhos humanos não dá para perceber muita diferença de cores entre elas.

```
Type "help", "copyright", "credits" or "license()" for more information
>>>
= RESTART: E:\videos\FOTOS\Material_Processamento_de_Imagens_EaD\Técn
_4\Basico\Bits_.py
img1 = uint16
img2 = uint8
```

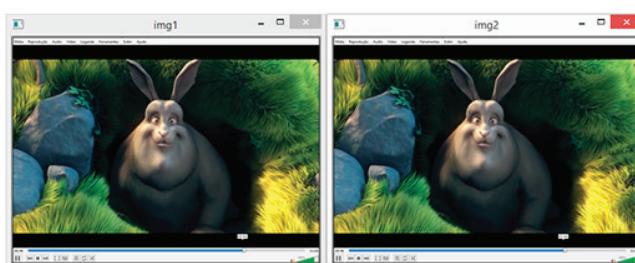


Figura 3 – Imagens com 16 e 8 *bits*, respectivamente.

Trecho do filme *Creative Commons Big Buck Bunny*

Fonte: Acervo do Conteudista

Além de compressão relacionada às cores, outra forma de comprimir os arquivos de imagens é reduzindo-os de forma espacial em relação aos *pixels*. Por exemplo, se os objetos presentes na imagem puderem ser identificados ou representados somente pelo seu contorno, o volume de informação será muito menor, para isso, um filtro de Canny, que é um filtro Passa-Altas detector de bordas, pode ser utilizado. A Figura 4 demonstra esse exemplo.

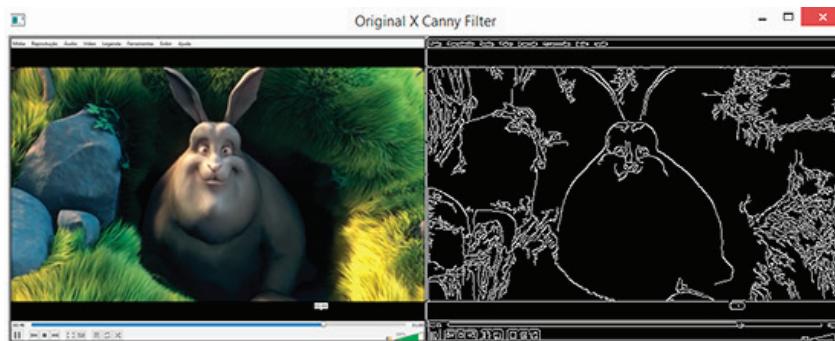


Figura 4 – Imagem colorida e imagem somente com os contornos
 Trecho do filme *Creative Commons Big Buck Bunny*

Fonte: Acervo do Conteudista

Nesse exemplo, a imagem da esquerda (Figura 4) consome 260Kb de espaço em disco, enquanto a imagem da direita consome apenas 24 Kb (10 vezes menos espaço em disco).

Principais Formatos de Imagem

BMP

BMP é uma abreviação de “*Bitmap*”, ou mapa de bits. O formato .bmp é basicamente um arquivo que descreve quantos *pixels* a imagem tem, e qual é a cor de cada *pixel*. Foi desenvolvido pela Microsoft para ser compatível com o Windows. Para cada *pixel* da imagem são descritos três valores: um para a cor vermelha, um para a cor verde e um para a cor azul. Cada um desses valores pode ser de 0 até 255. Um *pixel* no qual os três valores são 0 será preto, e um no qual os três sejam 255 será branco. Alterando esses valores, é possível que cada *pixel* da imagem tenha uma dentre 16.777.216 cores diferentes (OLHAR DIGITAL, 2017).

É um tipo de arquivo muito pesado. Imagine uma imagem de 800 por 600 *pixels*, ela terá 480 mil *pixels*. Se cada um dele tiver 3 *bytes*, a imagem final terá cerca de 1,44 megabyte, o que é espaço demais para uma imagem desse tamanho.

JPG

JPG vem da sigla *JPEG*, que significa “*Joint Photographic Experts Group*” – o nome de uma organização de fotógrafos que criou o formato. É um dos formatos mais usados na *internet*, pois, em comparação com o .bmp, ele economiza muito espaço.

Para fazer isso, ele divide a imagem em blocos de oito por oito *pixels* e compara cada um a um dos 64 padrões representados na imagem abaixo. Em seguida, determina qual o “peso” de cada um desses padrões em cada bloco.

Dependendo da qualidade de compressão escolhida pelo usuário, blocos com padrões como esse são substituídos por blocos mais simples. Isso permite que o arquivo final seja muito mais leve, com pouquíssima perda de qualidade.

Assim, o tamanho reduzido é a principal vantagem dos arquivos JPG. A desvantagem é que o tipo de compressão que ele usa funciona melhor para fotos (é muito utilizado em câmeras digitais e *smartphones*). Uma foto em .bmp tem muito poucas vantagens sobre essa mesma foto em .jpg, mas é muito mais pesada, e por isso .jpg é mais recomendado (OLHAR DIGITAL, 2017).

PNG

PNG significa *Portable Network Graphics*. O PNG é um formato bom para gráficos na internet por habilitar transparências com eficiência e elegância. O PNG oferece suporte à cor 8-bit, mas também à cor RGB 24-bit. Ele faz isso sem perdas, comprimindo as imagens fotográficas sem degradar a qualidade. O PNG tende a ser o formato de imagem mais pesado entre os três (BMP, JPG e PNG). Além de ser um formato excelente para transparências, a natureza sem perdas do PNG 24-bit é ideal para software de *screenshot*, permitindo uma reprodução *pixel* por *pixel* do seu ambiente *desktop*.

São duas as vantagens dos arquivos .png: o bom equilíbrio entre peso e qualidade e o suporte a transparências. Contudo, esse primeiro ponto também pode acabar sendo uma fraqueza: se a imagem exige muita qualidade, pode ser melhor usar outro formato, e se ela precisa ser leve, pode ser melhor usar JPG ao invés de PNG (GIZMODO, 2019).

WebP

Criado em 2010 pela Google, o formato WebP representa imagens com muita qualidade em um tamanho menor que os formatos mais utilizados até então. Ou seja, economiza espaço, aumenta a velocidade de uma página web e não perde em qualidade. Na prática, tem como principal objetivo compactar imagens de forma mais eficiente para oferecer uma experiência mais rápida ao usuário.

Quando você faz uma pesquisa de imagem no Google, por exemplo, os mais variados tamanhos e formatos aparecem entre os resultados, certo? Após escolher a foto ou o gráfico que procura, você pode tentar diminuir o seu tamanho, porém, isso geralmente representa uma perda de qualidade. A ideia do WebP é solucionar esse problema.

Para se ter uma ideia da sua eficiência, as imagens WebP reduzem, em média, 26% do tamanho de arquivos em PNG e entre 25% e 34% em comparação com figuras em JPEG de mesma qualidade. Portanto, na prática, é possível notar uma redução sensível no espaço que uma foto, infográfico ou ilustração vai ocupar (CAMARGO, 2019).

Principais Formatos de Vídeo

WMV

O formato WMV (*Windows Media Video*) é um formato de compressão de vídeo criado pela *Microsoft Corporation* para ser usado com conteúdo de vídeo e aplicações de reproduções na *Internet* e em computadores pessoais com o sistema *Windows*. Ele permite que arquivos enormes sejam comprimidos ou reduzidos em tamanho, para gerar economia de espaço e facilidade de transmissão, mantendo a qualidade. Atualmente, o WMV é considerado um formato aberto padrão. O próprio *player* da *Microsoft*, o *Windows Media Player*, reconhece automaticamente esse formato (RODRIGUES, 2013).

MP4

Esse é um dos formatos de vídeo mais conhecidos. O MP4 é típico dos dispositivos móveis e, por isso, está presente no dia a dia de muita gente.

Também compatível com o *YouTube* e outras plataformas de *streaming*, a vantagem do MP4 é que ele armazena em um mesmo arquivo as faixas de áudio, de vídeo, imagens estáticas e legendas.

Foi criado com o objetivo de ser revolucionário da preservação da qualidade, mesmo após compressão, e é eficiente nesse sentido (HOTMART, 2019).

AVI

AVI significa *Audio Video Interleave*, que é um formato de vídeo criado pela *Microsoft*. Serve como contêiner para faixas de vídeo e áudio. Um arquivo AVI contém tanto um arquivo de áudio quanto um arquivo de vídeo, ambos encapsulados, de forma que, quando o arquivo é reproduzido, as faixas de áudio e vídeo são executadas de forma sincronizada. O formato AVI costuma ser compatível nativamente com as versões do *Windows*, exceto quando a compressão dos arquivos de áudio e/ou vídeo utilizam um *codec* específico. A vantagem desse formato está no fato de ser reconhecido por aparelhos de DVD e *Blu-Ray* que são compatíveis com o *codec* DivX (APOWERSOFT, 2020).

MOV

O MOV é um formato de arquivo de vídeo criado pela *Apple*, para ser utilizado no software *QuickTime* (também disponível para *Windows*). Esse formato, basicamente agrega sequências de vídeo produzidas no *QuickTime* utilizando *codecs* específicos desenvolvidos pela própria *Apple*. Sua vantagem é a compatibilidade com o *iPod*, a *Apple TV*, o *iPhone* e o *iPad*. Além disso, o formato MOV pode ser utilizado para *streaming* de vídeo.

Seu diferencial positivo é que, embora compacte os vídeos em arquivos bem pequenos, a perda de qualidade do som e imagem é pequena (RODRIGUES, 2013).

Prática de Compressão com *Python* e *OpenCV*

Compressão JPG

A seguir, iremos comparar uma imagem BMP e sua versão com compressão JPG com 100% de qualidade.

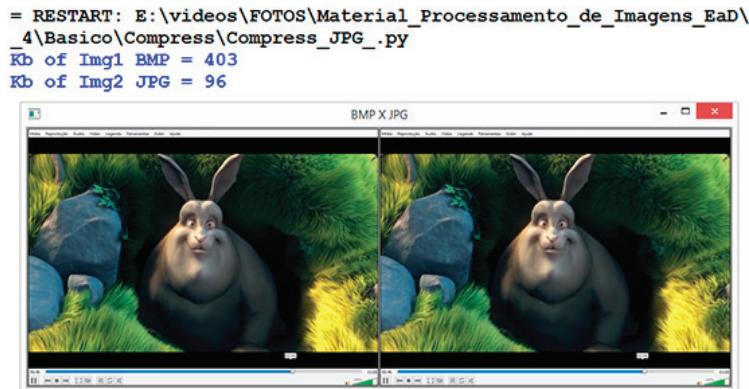


Figura 5 – Imagem em BMP com 403 Kb à esquerda, e imagem comprimida em JPG com 100% de qualidade e 96 Kb à direita. Trecho do filme *Creative Commons Big Buck Bunny*

Fonte: Acervo do Conteudista

O código a seguir é o código que possibilitou gerar a imagem da Figura 5:

```
import cv2
import numpy as np
import os

img = cv2.imread("Bunny_Movie.bmp")
print("Kb of Img1 BMP =", int(os.path.getsize("Bunny_Movie.bmp")/1024))

cv2.imwrite('B_M_Comp_JPG.jpg',img,[int(cv2.IMWRITE_JPEG_QUALITY),100])

img2 = cv2.imread("B_M_Comp_JPG.jpg")
print("Kb of Img2 JPG =", int(os.path.getsize("B_M_Comp_JPG.jpg")/1024))

imgs_concat = np.concatenate((img, img2), axis=1)

cv2.imshow("BMP X JPG", imgs_concat)
cv2.waitKey(0)
```

A linha de código que contém a instrução para comprimir a imagem BMP em JPG com 100% de qualidade foi:

```
cv2.imwrite('B_M_Comp_JPG.jpg',img,[int(cv2.IMWRITE_JPEG_QUALITY),100])
```

Se comprirmos a imagem em JPG com qualidade menor que 100%, a imagem fica menor, mas a qualidade diminui também. Se utilizarmos uma qualidade de até 70%, dificilmente perceberemos que a imagem diminuiu de qualidade ao olharmos para

imagem, mas com taxas menores de qualidade de compressão, conseguiremos visualizar a imagem em menor qualidade (com *pixels* borradinhos).

Para poder visualizarmos essa perda de qualidade, faremos a compressão JPG com qualidade de 5%, utilizando a seguinte linha para compressão:

```
cv2.imwrite('B_M_Comp_JPG.jpg', img,[int(cv2.IMWRITE_JPEG_QUALITY),100])
```

O código completo para a compressão JPG com qualidade de apenas 5%, fica assim:

```
import cv2
import numpy as np
import os

img = cv2.imread("Bunny_Movie.bmp")
print("Kb of Img1 BMP =", int(os.path.getsize("Bunny_Movie.bmp")/1024))

cv2.imwrite('B_M_Comp_JPG.jpg',img,[int(cv2.IMWRITE_JPEG_QUALITY),5])

img2 = cv2.imread("B_M_Comp_JPG.jpg")
print("Kb of Img2 JPG =", int(os.path.getsize("B_M_Comp_JPG.jpg")/1024))

imgs_concat = np.concatenate((img, img2), axis=1)

cv2.imshow("BMP X JPG", imgs_concat)
cv2.waitKey(0)
```

E o resultado em imagens desse processamento é o seguinte:

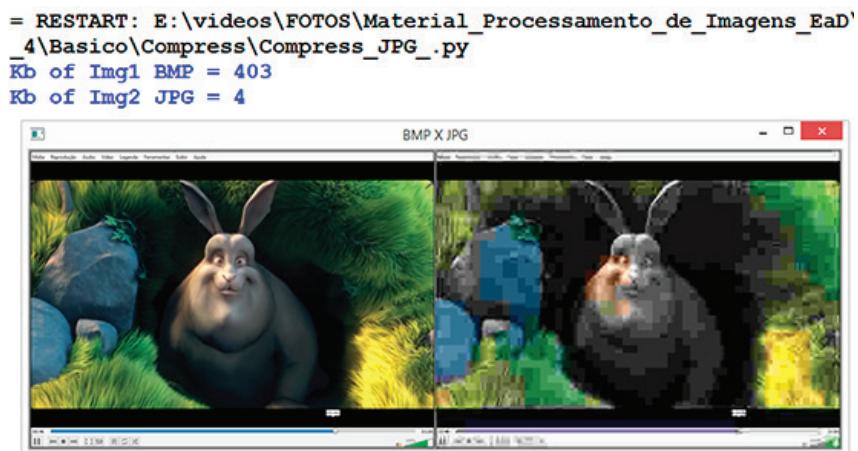


Figura 6 – Imagem em BMP com 403 Kb à esquerda, e imagem comprimida em JPG com 5% de qualidade e 4 Kb à direita. Trecho do filme *Creative Commons Big Buck Bunny*

Fonte: Acervo do Conteudista

No tamanho de armazenamento, a compressão JP com 5% de qualidade caiu de 96 Kb para apenas 4 kb, mas o resultado visual ficou ruim. Uma boa opção é ir subindo a qualidade para 25%, 50%, até que a imagem fique visualmente boa, mas o espaço de armazenamento fique menor. A maior vantagem é que cair de 96 Kb em BMP para 96Kb para JPG com 100% qualidade é uma boa economia de espaço, ainda assim, é sempre bom economizar mais, contanto que não comprometa muito a qualidade visual.

Compressão PNG

Com compressão PNG em 100% de qualidade, o arquivo ficou com 175 Kb, contra apenas 96 Kb do JPG, também em 100% de qualidade.

```
= RESTART: E:\videos\FOTOS\Material_Processamento_de_Imagens_EaD\'  
_4\Basico\Compress\Compress_PNG.py  
Kb of Img1 BMP = 403  
Kb of Img2 PNG = 175
```

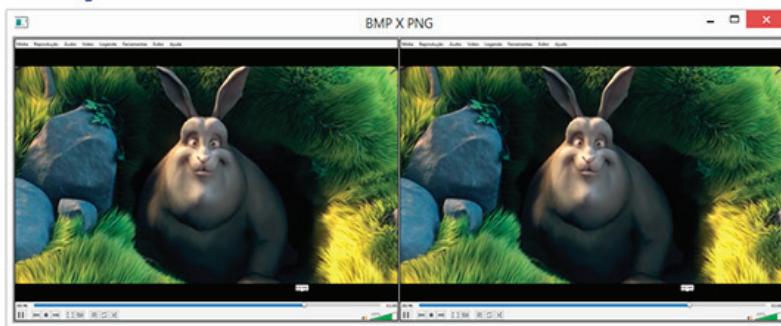


Figura 7 – Imagem em BMP com 403 Kb à esquerda, e imagem comprimida em JPG com 5% de qualidade e 4 Kb à direita. Trecho do filme *Creative Commons Big Buck Bunny*

Fonte: Acervo do Conteudista

Testando com qualidades menores de compressão em PNG, a qualidade visual da imagem não mudou, mas o tamanho em Kb também não diminuiu.

O código que faz a compressão PNG é o seguinte:

```
import cv2  
import numpy as np  
import os  
  
img = cv2.imread("Bunny_Movie.bmp")  
print("Kb of Img1 BMP =", int(os.path.getsize("Bunny_Movie.bmp")/1024))  
cv2.imwrite('B_M_PNG.png',img,[int(cv2.IMWRITE_PNG_COMPRESSION),100])  
img2 = cv2.imread("B_M_PNG.png")  
print("Kb of Img2 PNG =", int(os.path.getsize("B_M_PNG.png")/1024))  
imgs_concat = np.concatenate((img, img2), axis=1)  
  
cv2.imshow("BMP X PNG", imgs_concat)  
cv2.waitKey(0)
```

Compressão WebP

A seguir, iremos comparar uma imagem BMP com versões em compressão WebP com 100% e 5% de qualidade.

```
= RESTART: E:\videos\FOTOS\Material_Processamento_de_Imagens_EaD\Téc
  4\Basico\Compress\Compress_WebP_.py
Kb of Img1 BMP = 403
Kb of Img2 PNG = 49
```



Figura 8 – Imagem em BMP com 403 Kb à esquerda, e imagem comprimida em WebP com 100% de qualidade e 49 Kb à direita. Trecho do filme *Creative Commons Big Buck Bunny*

Fonte: Acervo do Conteudista

```
= RESTART: E:\videos\FOTOS\Material_Processamento_de_Imagens_EaD\Téc
  4\Basico\Compress\Compress_WebP_.py
Kb of Img1 BMP = 403
Kb of Img2 PNG = 3
```



Figura 9 – Imagem em BMP com 403 Kb à esquerda, e imagem comprimida em WebP com 5% de qualidade e 3 Kb à direita. Trecho do filme *Creative Commons Big Buck Bunny*

Fonte: Acervo do Conteudista

Medida de Qualidade de Imagem PSNR

Na seção 5, acabamos avaliando a qualidade visual da imagem com compressão de modo subjetivo, ou seja, simplesmente olhando as imagens e comparando visualmente com a imagem original.

Existem modos objetivos de mensurar essa qualidade, no caso, a qualidade de compressão pode ser avaliada procurando se há um ruído em relação à imagem original na imagem com compressão. Quanto menos ruído houver, mais qualidade há na imagem com compressão.

O ruído é medido por uma diferença calculada para cada *pixel* da imagem original em relação à imagem com compressão. A medida que iremos utilizar aqui chama-se PSNR (KLASSEN; PEDROSO, 2013).

A relação sinal-ruído de pico (PSNR) é a relação entre a potência máxima possível de uma imagem e a potência de ruído corruptor que afeta a qualidade de sua representação. Para estimar o PSNR de uma imagem, é necessário comparar essa imagem à uma imagem limpa ideal com a potência máxima possível.

Se não houver ruído na comparação entre a imagem original e a imagem comprimida, não haverá diferença nenhuma entre elas, o que resultará em um valor de PSNR igual a 0 (zero). Conforme esse valor cresce, isso significa que a qualidade da compressão foi caindo. Não é necessário que esse valor seja realmente zero na prática, mas quanto mais próximo de zero ele for, melhor a qualidade da compressão.

Para testar isso, podemos comparar uma imagem original com ela mesma para saber se realmente o PSNR pode chegar a 0 (Zero) quando não houve perda, e depois compará-la com uma imagem comprimida, para verificar se o valor do PSNR aumentou.

A Figura 10 demonstra o resultado do programa que calcula o valor PSNR para uma única imagem comparada com ela mesma. É possível observar que as imagens são idênticas e que o valor PSNR entre elas é 0 (zero).

```
= RESTART: E:\videos\FOTOS\Material_Processamento_de_Imagens_
4\Basico\PSNR_0_.py
Valor PSNR entre duas imagens idênticas = 0
```

Figura 10 – Cálculo do valor PSNR entre duas imagens idênticas (exatamente a mesma imagem). A cena utilizada foi criada pelo autor no *software Blender 3D*

Fonte: Acervo do Conteudista

O código *Python* com *OpenCV* que resultou na imagem da Figura 10 é mostrado a seguir:

```
import cv2
import numpy as np

def getPSNR(I1, I2):
    s1 = cv2.absdiff(I1, I2)
    s1 = np.float32(s1)
    s1 = s1 * s1
    sse = s1.sum()
    if sse <= 1e-10:
        return 0
    else:
        shape = I1.shape
        mse = 1.0 * sse / (shape[0] * shape[1] * shape[2])
        psnr = 10.0 * np.log10((255 * 255) / mse)
        return psnr

img = cv2.imread('sala.bmp')
img2 = cv2.imread('sala.bmp')
res1 = np.hstack((img, img2))
psnr = getPSNR(img, img2)
```

```

print("Valor PSNR entre duas imagens idênticas = ", psnr)

cv2.imshow('Imagen Entrada, Imagem mediana', res1)
cv2.waitKey(0)
cv2.destroyAllWindows()
  
```

Na Figura 10, o valor do PSNR foi 0 (Zero), pois as duas imagens utilizadas no cálculo foram a mesma, e isso implica em nenhuma diferença de qualidade visual, ou seja, zero ruído entre as imagens.

A medida em que você compara uma imagem original com compressão, a tendência é esse valor se distanciar de zero. Então, você pode usar o PSNR para decidir que algoritmo de compressão utilizar e qual o valor da qualidade de compressão desejada, pois você pode comprimir em dois formatos distintos e comparar os dois com a imagem original, depois verificando qual deles tem o valor PSNR menor (mais próximo de zero).

A Figura 11 demonstra o resultado do programa que calcula o valor PSNR para uma imagem original comparada a sua versão em compressão JPG com qualidade 50%.

```

= RESTART: E:\videos\FOTOS\Material_Processamento_de_Imagens\
  4\Basico\PSNR_1_.py
29.96392326454035


  
```

Figura 11 – Cálculo do valor PSNR entre a imagem original e sua versão com compressão JPG e qualidade 50%. A cena utilizada foi criada pelo autor no software *Blender 3D*

Fonte: Acervo do Conteudista

O código *Python* com *OpenCV* que resultou na imagem da Figura 11 é mostrado a seguir:

```

import cv2
import numpy as np

def getPSNR(I1, I2):
    s1 = cv2.absdiff(I1, I2)
    s1 = np.float32(s1)
    s1 = s1 * s1
    sse = s1.sum()
    if sse <= 1e-10:
        return 0
    else:
        shape = I1.shape
        mse = 1.0 * sse / (shape[0] * shape[1] * shape[2])
        psnr = 10.0 * np.log10((255 * 255) / mse)
        return psnr
  
```

```
img = cv2.imread('sala.bmp')
cv2.imwrite('test.jpg',img,[int(cv2.IMWRITE_JPEG_QUALITY),50])
img2 = cv2.imread('test.jpg')
res1 = np.hstack((img, img2))
psnr = getPSNR(img, img2)
print(psnr)

cv2.imshow('Imagen Entrada, Imagen mediana', res1)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Como é possível ver na Figura 11, o valor PSNR subiu para aproximadamente 30.

Compressão de Vídeo a partir de Imagens

Um vídeo é uma sequência de imagens em um determinado formato, com uma taxa de quadros específica (imagens por segundo), uma resolução específica e áudio sincronizado (ou não, caso o vídeo não tenha áudio).

Com *Python* e *OpenCV* podemos pegar cada imagem contida em um vídeo, uma a uma, de modo que o *OpenCV* consegue rodar o vídeo, e é possível processar cada imagem dele.

Com *Python* e *OpenCV* podemos pegar um conjunto de imagens que possuam mesma resolução, estabelecer uma taxa de imagens por segundo, um formato de vídeo e gravar um vídeo a partir desse conjunto de imagens.

Para abrir um vídeo e processar imagem por imagem podemos utilizar um código semelhante ao que segue:

```
import cv2
vidcap = cv2.VideoCapture('big_buck_bunny.mp4')
success,image = vidcap.read()
count = 0
while success:
    cv2.imwrite("Images\\frame%d.jpg" % count, image)
    success,image = vidcap.read()
    print('Read a new frame: ', success)
    count += 1
```

O código anterior lê um vídeo MP4 e salva cada *frame* dele em uma pasta chamada Imagens, cada frame dessa imagem poderia ser processado antes de ser salvo e poderia ser utilizada uma compressão JPG ou PNG, por exemplo, nessas imagens.

O código a seguir pega um conjunto de imagens separadas e as transforma em um vídeo de formato AVI:

```
import cv2
import numpy as np
import glob

img_array = []

for filename in glob.glob('Images/*.jpg'):
    img = cv2.imread(filename)
    height, width, layers = img.shape
    size = (width,height)
    img_array.append(img)

out = cv2.VideoWriter('v.avi',VideoWriter_fourcc(*'MP4V'), 15, size)

for i in range(len(img_array)):
    out.write(img_array[i])
out.release()
```

E, dessa forma, é possível comprimir um vídeo utilizando o processamento de imagens, ou seja, comprimir imagem por imagem de um vídeo e depois formar um vídeo a partir de todas as imagens comprimidas.

Material Complementar

Indicações para saber mais sobre os assuntos abordados nesta Unidade:

 Leitura

Learning for Video Compression

<https://bit.ly/395BxE4>

A Systematic Review on Real Time Video Compression and Enhancing Quality Using Fuzzy Logic

<https://bit.ly/2LSWIAv>

Image Compression Algorithm and JPEG

<https://bit.ly/3p3nXqs>

A Study of Image Compression Techniques

<https://bit.ly/2XXmFkY>

Referências

ARORA, K.; SHUKLA, M. A Comprehensive Review of Image Compression Techniques. *International Journal of Computer Science and Information Technologies*, Vol. 5 (2), 2014, 1169-1172.

Big Buck Bunny. The results of the Peach open movie project has been licensed under the Creative Commons Attribution 3.0 license, 2008. Disponível em: <<https://peach.blender.org/download/>>. Acesso em: 29/11/2020.

CAMARGO, G. Entenda o que é o formato de imagem WebP e como ele pode beneficiar os seu SEO, 2019. Disponível em: <<https://rockcontent.com/br/blog/webp/>>. Acesso em: 29/11/2020.

CONCI, A.; AZEVEDO, E.; LETA, F. **Computação Gráfica Teoria e Prática**, Volume 2, Rio de Janeiro: Elsevier, 2008.

GIZMODO. Qual a diferença entre JPG, PNG e GIF? 2011. Disponível em: <<https://rockcontent.com/br/blog/webp/>>. Acesso em: 29/11/2020.

KLASSEN, D. M.; PEDROSO, C. M. Aplicação da Suavização de Tráfego para Melhoria da Qualidade de Experiência em Sistemas IPTV, 2013. **XXXI Simpósio Brasileiro de Telecomunicações**, Fortaleza – CE, 2013.

PARWE, P. R.; MANDAOGADE, N. N. A review on image compression techniques. *International Journal of Computer Science and Mobile Computing*, 2015.

RODRIGUES, L. Entenda as diferenças entre os formatos de arquivos de vídeo. Disponível em: <<https://www.techtudo.com.br/artigos/noticia/2013/04/entenda-diferencias-entre-os-formatos-de-arquivos-de-video.html>>. Acesso em: 29/11/2020.

Sites Visitados

HOTMART, Quais formatos de vídeo existem e como escolher o melhor para seu negócio? 2019. Disponível em: <<https://blog.hotmart.com/pt-br/formatos-de-video/>>. Acesso em: 29/11/2020.

OLHAR DIGITAL. JPG, PNG, GIF e BMP – quais as diferenças entre os principais formatos de imagens? 2017. Disponível em: <<https://olhardigital.com.br/2017/06/08/noticias/jpg-png-gif-e-bmp-quais-as-diferencas-entre-os-principais-formatos-de-imagens/>>. Acesso em: 29/11/2020.



Cruzeiro do Sul
Educacional