Bem-vindo ao Articulação Matemática

ARTICULAÇÃO

MATEMÁTICA

OUTUBRO | 2024 EDIÇÃO Nº 4

IMAGENS DIGITAIS

INSTRUÇÕES DE USO

- Clique nos elementos com o ícone para acessar conteúdos extras.
- Clique nos elementos com o ícone + para acessar as respostas das questões.
- Palavras em **destaque** possuem conteúdo extra que pode ser acessado ao ser clicado.
- Todos os *sites* citados possuem *hiperlink* e podem ser acessados com um *click* sobre ele.
- Você pode navegar pelas páginas seguindo o Sumário ou pela barra superior de navegação. < > ⌂ ⓘ

INICIAR

OUTUBRO | 2024 EDIÇÃO Nº 4

BNCC


DIÁ
LOGO
ABERTO
REFLE
XÃO
NA PRÁTICA
EXPE
DIENTE
FIQUE
SA
BEN
DO!
INFO
GRÁ
FICO
RE
SE
NHA
ARTI
CULA
ÇÃO
MATEMÁTICA
OUTUBRO | 2024 EDIÇÃO Nº 4
BNCC
IMAGENS DIGITAIS

FIQUE SA BEN DO!

SAIBA MAIS SOBRE
O TEMA DESTA EDIÇÃO.

DIÁ LOGO ABERTO

REFLE XÃO

EXPE DIENTE

INFO

A PRÁTICA

RE

A FIQUE SA BEN DO!

Foto real ganha prêmio em categoria de imagens feitas com IA e é desqualificada

B FIQUE SA BEN DO!

Dia histórico para a ciência: revelada a primeira imagem de buraco negro

C FIQUE SA BEN DO!

Imagens digitais são utilizadas para monitoramento de contaminação da água no Rio Capibaribe

D FIQUE SA BEN DO!

O que é uma imagem?

ARTI CULA ÇÃO

MATEMÁTICA

OUTUBRO | 2024 EDIÇÃO Nº 4

BNCC

IMAGENS DIGITAIS

FIQUE SA BEN DO!

SAIBA MAIS SOBRE O TEMA DESTA EDIÇÃO.

A

Foto real ganha prêmio em categoria de imagens feitas com IA e é desqualificada

Com a intenção de defender as fotografias captadas com o olho humano, o fotógrafo Miles Astray [...] inscreveu uma fotografia feita por ele mesmo em uma categoria de imagens geradas por inteligência artificial em uma competição de prestígio.

[...]

Astray [...] diz que ficou motivado a quebrar as regras depois de uma série de imagens geradas por IA vencerem concursos de fotografia convencional. "Ocorreu-me que eu poderia distorcer essa história de dentro para fora, como só um ser humano poderia e faria, enviando uma foto real para uma competição de IA. [...]", disse ele em entrevista ao The Guardian.

[...]

"A IA já pode produzir conteúdo de aparência incrivelmente real e [...] você pode facilmente enganar públicos inteiros. [...] Nunca foi tão importante questionar [...]", defendeu o fotógrafo.

[...]

A vitória de Astray ocorre um ano depois de um artista alemão, Boris Eldagsen, ter ganhado o prêmio *Sony World Photography* com uma imagem gerada por IA. [...]

Foto real ganha prêmio em categoria de imagens feitas com IA e é desqualificada. **Época Negócios**. Disponível em: https://epocanegocios.globo.com/inteligencia-artificial/noticia/2024/06/foto-real-ganha-premio-em-categoria-de-imagens-feitas-com-ia-e-e-desqualificada.ghtml. Acesso em: 1º set. 2024.

IMAGENS DIGITAIS

FIQUE SA BEN DO!

SAIBA MAIS SOBRE O TEMA DESTA EDIÇÃO.

B

Dia histórico para a ciência: revelada a primeira imagem de buraco negro

Se você está lendo este texto, está entre as pessoas de sorte que viveram para ver uma imagem histórica para a ciência. Acaba de ser revelada a primeira foto de um buraco negro supermassivo no centro de Messier 87, uma enorme galáxia no aglomerado de Virgem. Este buraco negro está a 53,5 milhões de anos-luz da Terra e tem uma massa de 6,5 bilhões de vezes a massa do Sol. Esses monstros cósmicos conhecidos como buracos negros são pequenos, considerando a escala universal, mas com uma massa imensa a ponto de gerar um efeito gravitacional gigantesco. E que torna impossível a luz escapar deles — daí o nome que recebem. [...]
[...]

Como isso foi possível?

O *Event Horizon Telescope* tem seus radiotelescópios espalhados pelo planeta, apontando para dois buracos negros supermassivos: Sagitário A*, localizado no centro da Via Láctea, e um buraco negro ainda mais massivo, porém mais distante: 53,5 milhões de anos-luz de distância na galáxia M87 — foi este último que teve sua imagem revelada.
Há exatamente dois anos, em abril de 2017, a rede se uniu para observar o chamado horizonte de eventos desses buracos negros. Trata-se do limite até onde a luz consegue passar próxima ao buraco sem ser sugada por sua força gravitacional extrema. Se nem a luz pode escapar, além do horizonte de eventos tudo é escuridão. Mas junto com gás, poeira e átomos se chocando em velocidades extremas, as micro-ondas do disco de gás que fica em volta do buraco (o chamado disco de acreção) formam um anel de radiação que pode ser captada para nos mostrar os contornos do buraco negro. E é isso que mostra a imagem, obtida a partir do cruzamento dos dados dos observatórios.
Não bastava, no entanto, captar essa radiação. O processamento dos dados obtidos pela interferometria das ondas captadas através de cada um dos telescópios exigiu tecnologia robusta e o trabalho de um grupo enorme de cientistas, por dois anos. Por isso, só agora as imagens foram apresentadas.

O que estamos vendo?

A imagem capturou a sombra do buraco negro no interior do disco de material brilhante que a acompanha. Por causa da gravidade intensa perto de um buraco negro, a trajetória da luz mostrada na imagem, correspondendo à radiação do disco, é deformada em torno do horizonte de eventos e aparece na forma de um anel. [...]

Dia histórico para a ciência: revelada a primeira imagem de buraco negro. **Jornal da USP**. Disponível em: https://jornal.usp.br/ciencias/ciencias-exatas-e-da-terra/dia-historico-para-a-ciencia-revelada-a-primeira-imagem-de-buraco-negro/. Acesso em: 1º set. 2024.

IMAGENS DIGITAIS

FIQUE SA BEN DO!

SAIBA MAIS SOBRE O TEMA DESTA EDIÇÃO.

C

Imagens digitais são utilizadas para monitoramento de contaminação da água no Rio Capibaribe

O uso de imagens digitais para monitoramento de contaminação da água é um tema ao qual a mestranda Helayne Santos tem se dedicado no Programa de Pós-Graduação em Química (PPGQUÍMICA) da Universidade Federal de Pernambuco (UFPE).

Como parte disso, a pesquisadora trabalha para desenvolver um método que agilize e barateie o processo de identificação da presença de poluentes na água: com um *smartphone* e o aplicativo, uma pessoa pode fazer a análise no próprio local da coleta e obter informações sobre o nível de poluição em tempo real.

[...]

[...] a presença de grandes quantidades de surfactantes em ambientes como rios e estuários dá origem a problemas ambientais por alterar os níveis de oxigênio da água, provocando a morte de várias espécies e riscos à saúde humana como o desenvolvimento de dermatites. [...]

Os pesquisadores têm buscado meios para detectar poluentes como esses, considerando que a rapidez no processo é fundamental para que ações sejam tomadas antes que a contaminação se espalhe e alcance níveis críticos.

Entre as alternativas disponíveis para órgãos de controle ambiental e vigilância sanitária está o método baseado em imagem digital (DIB, do inglês *digital image-based*), semente da proposta desenvolvida por Helayne Santos com a professora do Departamento de Oceanografia da UFPE Eliete Zanardi Lamardo e o professor do Departamento de Química da Universidade Federal de Viçosa (UFV) Willian Toito Suarez.

Conhecido por sua eficiência e praticidade, o Método DIB tem como uma de suas principais vantagens o fato de que pode ser utilizado fora dos laboratórios.

Imagens digitais são utilizadas para monitoramento de contaminação da água no Rio Capibaribe. **Diário de Pernambuco**. Disponível em: https://www.diariodepernambuco.com.br/noticia/vidaurbana/2024/08/imagens-digitais-sao-utilizadas-para-monitorar-poluicao-no-capibaribe.html. Acesso em: 1º set. 2024.

FIQUE SA BEN DO!

SAIBA MAIS SOBRE O TEMA DESTA EDIÇÃO.

DIÁ LOGO ABERTO

INFO GRÁ FICO

REFLE XÃO NA PRÁTICA

RE SE NHA

EXPE DIENTE

D

O que é uma imagem?

Para entender acerca de imagens, assista ao vídeo **O que é uma imagem?**, do canal Programação Dinâmica, em que o autor desta edição do Articulação, Hallison Paz, introduz o estudo de imagens digitais, explicando sobre matrizes, *pixels* e cores.

https://s.ftd.li/0txalv

ARTI CULA ÇÃO

MATEMÁTICA

OUTUBRO | 2024 EDIÇÃO Nº 4

BNCC

IMAGENS DIGITAIS

DIÁLOGO ABERTO

CONHEÇA A OPINIÃO DE QUEM ESTUDA O ASSUNTO.

ORIENTAÇÕES

MATERIAL CONSULTADO

A Matemática das imagens digitais

O que é uma imagem?

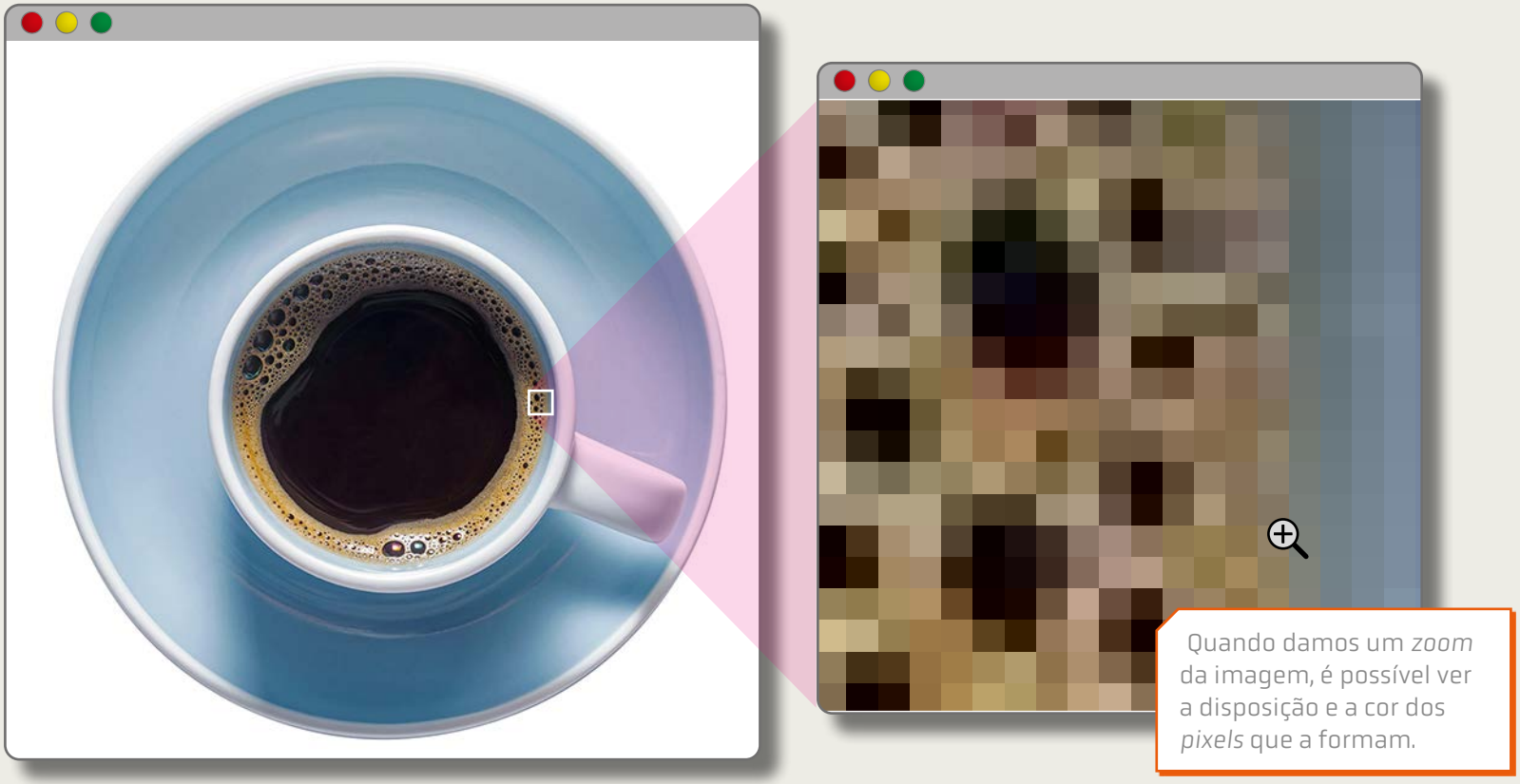
Quando falamos de imagens, uma das primeiras coisas que devemos entender é como enxergamos o mundo ao nosso redor. Na Física, aprendemos que só conseguimos ver os objetos porque eles refletem parte da luz que chega a eles. Essa luz pode vir de fontes naturais, como o Sol, ou artificiais, como uma lâmpada. Por exemplo, quando vemos um objeto vermelho, isso acontece porque ele reflete a luz vermelha e absorve as outras cores. A luz refletida, contudo, precisa ser captada pelos nossos olhos e interpretada pelo nosso cérebro para que possamos ver as coisas. Esse processo, que se inicia na propagação da luz e continua com sua captação pelo nosso sistema visual, é o que nos permite interpretar imagens e ver o mundo como ele é. Na ausência de luz, não enxergamos nada.

O mundo ao nosso redor está em constante movimento. Uma imagem, seja ela natural, seja ela digital, é como um recorte de uma cena específica, em um instante específico de tempo. Na imagem abaixo, por exemplo, não é possível ver o que está para além da borda da imagem, nem é possível saber se o barco ainda estava nessa mesma região do mar cinco minutos depois do momento em que essa fotografia foi tirada. Seriam necessárias outras imagens ou, até mesmo, uma sequência de várias imagens capturadas ao longo do tempo, como um vídeo, para obtermos mais informações. Ainda assim, cada imagem individualmente contém apenas uma parte dessa informação.

Atualmente, muitas imagens que vemos não são naturais, mas, sim, reproduzidas em telas, como as de celulares, televisores e computadores. Para que essas imagens possam ser criadas, armazenadas e exibidas digitalmente, precisamos representá-las de uma forma que um computador entenda. Computadores são máquinas que funcionam fazendo operações aritméticas (contas) e operações lógicas. Portanto, para que possamos lidar com imagens em um computador, precisamos de um modelo para explicar e descrever imagens na forma de números.

Representação digital de imagens

Uma das formas mais comuns de representar imagens no mundo digital é usando uma matriz. Nessa representação, conhecida como matricial ou *raster*, a imagem é formada por uma grade de pequenos pontos chamados *pixels*, e cada *pixel* pode mostrar uma única cor. Observe as imagens abaixo. À direita, temos um região da imagem da xícara de café, vista mais de perto, em *zoom*.



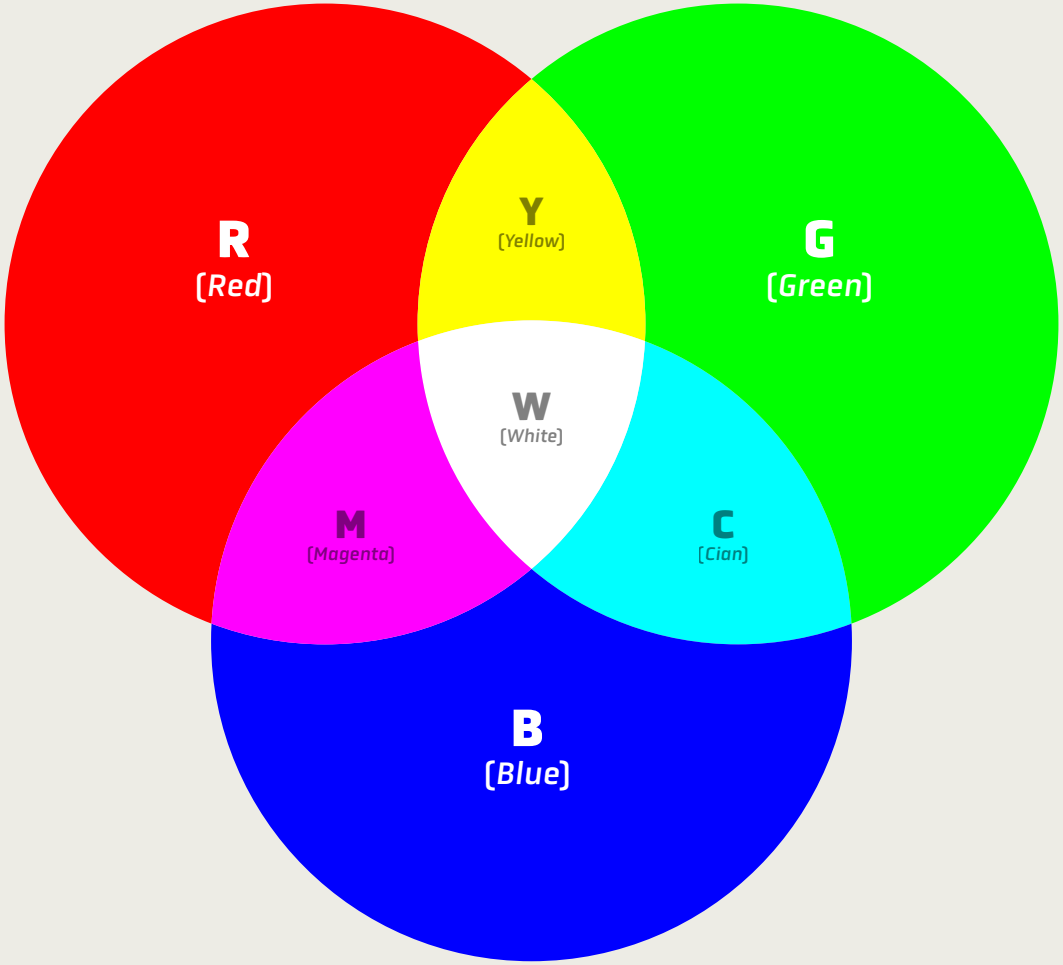
Quando damos um *zoom* da imagem, é possível ver a disposição e a cor dos *pixels* que a formam.

Esse método é bastante utilizado porque combina muito bem com a forma como as telas dos dispositivos que usamos são projetadas. Por exemplo, a tela do seu celular ou do seu monitor é como um grande retângulo cheio de *pixels* organizados em linhas e colunas. Quando você ouve falar que a resolução de uma tela é de 1 920 × 1 080, isso significa que a tela tem 1 920 *pixels* em cada linha e 1 080 *pixels* em cada coluna, ou seja, são 2 073 600 *pixels*. Cada um desses *pixels* pode acender (emitir luz) e mostrar uma cor específica. A tecnologia que faz isso pode variar (LED, OLED etc.), mas a ideia básica é a mesma.

Agora, se o posicionamento dos *pixels* é fácil de entender, como então representamos as cores?
Existem alguns modelos diferentes para representar cores, mas um dos mais usados é o modelo tricromático. Esse modelo foi proposto pelos pesquisadores Thomas Young e Hermann von Helmholtz, que descobriram que qualquer cor pode ser criada combinando diferentes quantidades de luz vermelha, verde e azul. Caso você já tenha mexido com programas de edição de imagem ou de *design* gráfico, é possível que tenha se deparado com o sistema RGB — *Red* (vermelho), *Green* (verde), *Blue* (azul) — para a escolha de cores. Cada um desses componentes de cor é chamado de canal de cor. Caso você queira saber mais sobre o modelo tricromático, você pode assistir ao vídeo **Entendendo cores no RGB**, do canal Programação Dinâmica, disponível no *link* ao lado.

Entendendo cores no RGB | Processamento de Imagens

https://s.ftd.li/ny3aq3



O sistema RGB é aditivo, o que significa que as cores são criadas pela adição de vermelho, verde e azul. Quanto mais de cada cor é combinado, mais clara é a cor resultante. Por exemplo, ao adicionar o máximo de vermelho e o máximo de verde obtemos amarelo. O mesmo acontece adicionando-se verde e azul, para resultar em ciano; vermelho e azul, para resultar em magenta, e as três cores, para resultar em branco.

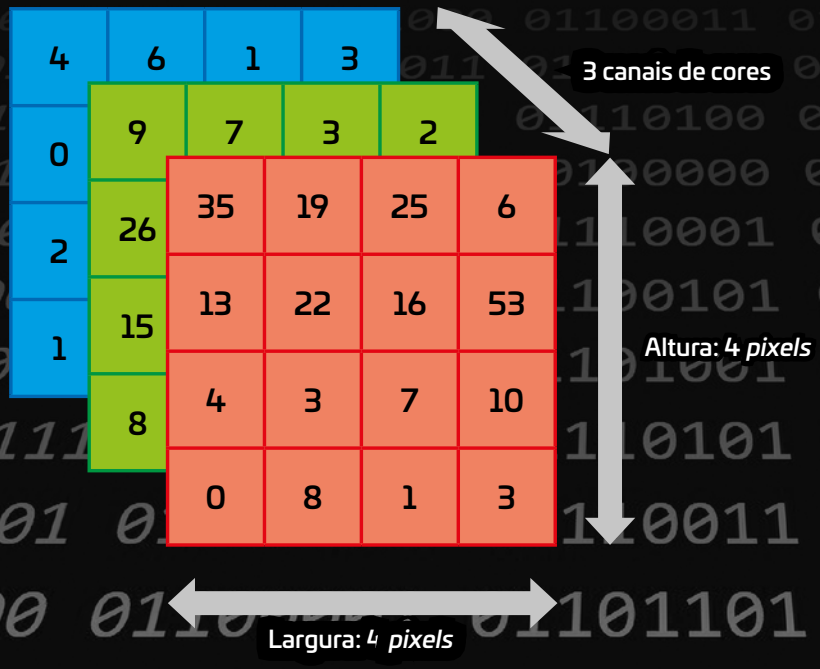
Esse sistema se difere do que ocorre no mundo real, em que objetos não emitem luz, mas a refletem ou a absorvem. Nesse caso, usamos o sistema subtrativo, no qual a cor de um objeto é determinada pela luz que ele reflete enquanto absorve outras. Por exemplo, um objeto verde, reflete apenas o verde e absorve as luzes das outras cores.

Se você fosse responsável por projetar representações de imagens, sabendo que elas devem ser armazenadas na memória de um computador e transmitidas por uma rede de computadores como a *internet*, qual seria uma das suas principais preocupações? Isso mesmo, o tamanho, o espaço ocupado em memória por essa representação!

Para economizar espaço de memória, foi decidido que cada canal de cor seria representado por 1 *byte*, que corresponde a 8 *bits*, e cada *bit* sendo um dígito binário 0 ou 1. Isso significa que podemos representar 256 valores diferentes para cada canal, já que 8 *bits* podem criar 256 combinações diferentes ($2^8 = 256$) de 0s e 1s, segundo o que estudamos na análise combinatória. As combinações, na base 2 (binária, representada por $_2$), variam de 00000000, 00000001, 00000010, ..., até 11111111, que correspondem, na base 10 (decimal, representada por $_{10}$), a 0, 1, 2, ..., 255, respectivamente. O 00000000_2 ou 0_{10} representa a ausência total da cor daquele canal, enquanto 11111111_2 ou 255_{10} representa a intensidade máxima da cor daquele canal.

Combinando as variações de vermelho, verde e azul, podemos representar aproximadamente 16,8 milhões de cores ($256^3 = 16\,777\,216$). Parece muita coisa, certo? Bom, e é. Com essa representação, é possível reproduzir digitalmente mais que todos os tons que o olho humano pode perceber na natureza. Assim, ela é suficiente para criar imagens bem realistas.

Assim, é possível representar uma imagem digital usando uma matriz de *pixels*, em que cada *pixel* é um conjunto de três valores inteiros (entre 0 e 255) que representam as quantidades de vermelho, verde e azul, respectivamente. Geralmente, esses valores estão dispostos em três matrizes alinhadas, tal como ilustrado abaixo. As linhas e colunas dessa matriz determinam a resolução da imagem, que é de 4 × 4. Quanto maior a resolução, maior a capacidade de representar detalhes finos na imagem.



Para criar cores além de vermelho, verde e azul, combinamos as intensidades delas em cada canal. Por exemplo, quando todas as três cores são definidas como 0 (R = 0, G = 0, B = 0), o resultado é preto, representando a ausência de luz. Em contraste, quando todas as três cores são definidas como 255 (R = 255, G = 255, B = 255), a cor resultante é branca, indicando a presença máxima de todas as cores. Variações entre esses extremos, como cinza (R = 128, G = 128, B = 128) ou ciano (R = 255, G = 255, B = 0), demonstram a flexibilidade e a diversidade de cores que o modelo RGB pode produzir.

Criando imagens com programação

Agora que você já entende como as imagens digitais são representadas, pode começar a criar suas próprias imagens usando uma linguagem de programação. Vamos utilizar a linguagem de programação *Python* e começar com um exemplo simples: a bandeira da Colômbia, que é formada por três faixas horizontais nas cores amarela, azul e vermelha.

Para criar e manipular imagens em *Python*, precisamos usar uma biblioteca, que é como uma coleção de ferramentas prontas que facilitam o nosso trabalho. Uma das bibliotecas mais usadas para isso é a *Pillow*. Se você não tem experiência com programação, não se preocupe! Você pode usar o *Google Colab*, uma plataforma *on-line* em que você pode programar diretamente no navegador, sem precisar instalar nada e sem custos. Para utilizar a plataforma, basta seguir o passo a passo abaixo.

1. Acesse o *link* do *Google Colab*, clicando ou acessando o *QR Code* abaixo:

https://s.ftd.li/ifjvr5

2. Em primeiro lugar, é preciso ter uma conta *Gmail* e clicar em "Fazer *login*" na plataforma. Depois, sem a necessidade inicial de criar um *notebook*, basta clicar em "Cancelar" na primeira tela que se abre ao finalizar o *login*.

3. Na página inicial da plataforma, há a introdução "Conheça o *Colab*". Acima desse título, você já pode clicar em "+ Código". Ao fazer isso, uma caixa de texto abrirá, em que você escreverá seu código ou copiará o disponibilizado aqui. Para fazê-lo rodar, basta clicar no botão de *play*, ao lado esquerdo da caixa de texto.

O código para gerar a bandeira da Colômbia é o seguinte:

Ao colar o código no *Google Colab*, substitua cada » por uma tabulação (tecla *tab* do teclado) ou dois espaços.

```
1  from PIL import Image
2
3  altura = 400
4  comprimento = int(3 * altura / 2)
5  colombia = Image.new("RGB", (comprimento, altura))
6
7  AMARELO = (255, 205, 0)
8  AZUL = (0, 48, 135)
9  VERMELHO = (200, 16, 46)
10
11 parte = altura // 4
12
13 for y in range(altura):
14 » for x in range(comprimento):
15 » » if (y < 2 * parte):
16 » » » colombia.putpixel((x, y), AMARELO)
17 » » elif (y < 3 * parte):
18 » » » colombia.putpixel((x, y), AZUL)
19 » » else:
20 » » » colombia.putpixel((x, y), VERMELHO)
21
22 colombia.save("bandeira_colombia.png")
23 colombia
```

Para construir nosso código, primeiro precisamos importar o módulo **Image** da biblioteca *Pillow* (PIL), que nos permitirá trabalhar com imagens.

A função **new** (linha 5), disponível na biblioteca **Image**, é usada para criar uma imagem. Para isso, precisamos definir o sistema de cores (nesse caso, RGB) e a resolução desejada. A proporção oficial entre a altura e o comprimento da bandeira é de 2 : 3. Isso significa que, se definirmos a variável **altura** como 400 *pixels* (linha 3), a variável **comprimento** deve ser 600 *pixels* (linha 4). Podemos escrever uma instrução em *Python* para que o comprimento seja calculado automaticamente com base na altura (linha 4). Vamos armazenar nossa nova imagem na variável chamada **colombia** (linha 5).

Os códigos oficiais RGB das cores usadas na bandeira da Colômbia são:

- Amarelo: (255, 205, 0);
- Azul: (0, 48, 135);
- Vermelho: (200, 16, 46).

Assim, criamos as variáveis **AMARELO**, **AZUL** e **VERMELHO** para guardar esses valores (linhas 7, 8 e 9).

De acordo com as proporções que cada faixa de cor ocupa na bandeira, a altura da bandeira deve ser dividida em quatro partes iguais (linha 11). As duas primeiras partes são amarelas, a terceira é azul, e a quarta é vermelha.

A linha 13 roda, no que chamamos de laço, 400 vezes o código da linha 14, uma para cada linha de *pixels*, começando pela primeira de cima. Por sua vez, a linha 14 roda 600 vezes o código das linhas 15 a 20, uma para cada *pixel* das colunas, da esquerda para a direita. Ou seja, o código das linhas 15 a 20 é rodado 240 mil vezes, uma para cada *pixel* da imagem, verificando se o *pixel*, determinado por uma linha e uma coluna, está nas duas primeiras partes amarelas, na parte azul ou na parte vermelha da bandeira, pintando-o com a cor correspondente por meio da função **putpixel**, que recebe a posição do *pixel*, além da cor.

Na linha 22, você pode salvar a imagem como um arquivo, por exemplo, PNG, e abri-la com o programa de visualização de imagens de sua preferência. Para baixar a imagem, basta clicar na quinta opção, um ícone de pasta, no menu esquerdo, clicar nos três pontinhos no arquivo "bandeira_colombia.png" gerado e fazer o *download*.

Para finalizar, a linha 23 exibe o resultado da bandeira, como na imagem abaixo. Legal, né?

Uma observação importante é que, usando *Python*, as linhas do código precisam ter esse espaçamento no começo da linha para explicitar ao programa qual é a hierarquia entre elas. É o que chamados de indentação. Então, por exemplo, para um *pixel* selecionado na linha 14, o código das linhas 15 a 20 é rodado sobre ele, antes da linha 14 passar para o próximo *pixel*. O mesmo acontece na linha 15.
Se o *pixel* estiver na metade amarela da bandeira, o código da linha 16 é executado, senão o programa passa para a linha 17 e assim por diante.

O resultado é o seguinte:

A Colômbia é um país localizado na região noroeste da América do Sul. Banhada pelo oceano Atlântico ao norte e pelo oceano Pacífico a oeste; a Colômbia é o único país da América do Sul a ter costas nos dois oceanos.

Caso deseje aprofundar-se no uso do *Google Colab*, para, por exemplo, aprender sobre os *notebooks*, você pode assistir ao vídeo **Como usar o Google Colab para analisar dados?**, do canal Programação Dinâmica.

https://s.ftd.li/7olibz

Manipulando imagens

Escala de cinza

Além de criar imagens sintéticas, nós podemos manipular qualquer imagem digital para criar outros efeitos interessantes. Vamos usar a imagem abaixo do Palácio do Congresso Nacional do Brasil como exemplo, que é uma imagem bastante conhecida.

O Palácio do Congresso Nacional, localizado em Brasília, é a sede do Poder Legislativo brasileiro, abrigando a Câmara dos Deputados e o Senado Federal. Projetado pelo arquiteto Oscar Niemeyer, o edifício é um dos marcos da arquitetura modernista brasileira. Suas formas icônicas incluem duas cúpulas, uma convexa e outra côncava, e duas torres gêmeas, que simbolizam a harmonia entre os poderes legislativos.

Antes, é necessário ter a fotografia salva no seu computador. Podemos nomear o arquivo PNG como "congresso". Para abrir o arquivo da imagem e introduzi-la no programa, antes é preciso subir a imagem no *Google Colab*, indo na opção “Arquivos”, no ícone de pasta (como fizemos anteriormente), e clicando no primeiro ícone, uma folha com uma seta para cima (“Fazer *upload* para o armazenamento da sessão”).

Depois de subir a imagem, usamos o seguinte código:

```
1  from PIL import Image
2
3  congresso = Image.open("congresso.png")
4  comprimento, altura = congresso.size
5
6  for y in range(altura):
7  » for x in range(comprimento):
8  » » rgb = congresso.getpixel((x, y))
9  » » media = (rgb[0] + rgb[1] + rgb[2]) // 3
10 » » congresso.putpixel((x, y), (media, media, media))
11
12 congresso.save("congresso_cinza.png")
13 congresso
```

Ao colar o código no *Google Colab*, substitua cada » por uma tabulação (tecla *tab* do teclado) ou dois espaços.

O resultado é o seguinte:

Primeiro, usamos, na linha 2, a função **open** para abrir a imagem que subimos. Criamos uma variável chamada **congresso** para nos referenciarmos à imagem no programa. As variáveis **comprimento** e **altura** da linha 4 guardam o tamanho da imagem para sabermos quantas vezes precisaremos rodar o código dos laços das linhas 6 e 7.

Na linha 8, lemos, com a variável **rgb**, as informações de um *pixel* específico usando a função **getpixel** e passando a coordenada dele (coluna, linha: **x** e **y** das linhas 6 e 7) na matriz. Qualquer código RGB com as três coordenadas iguais representa um tom de cinza. Quanto menores forem os valores, mais escura e mais próxima do preto estará a cor; valores maiores representam tons mais claros, até o branco. Assim, uma forma de transformar uma imagem colorida em sua representação em tons de cinza seria apenas substituir a cor de cada *pixel* pela média dos canais de cores. Para isso, na linha 9, calculamos a média entre os valores dos três canais, acessando cada canal pelos índices 0, 1 e 2 da variável **rgb**. Para finalizar, na linha 10, trocamos a cor original daquele *pixel* pela nova cor.

Essa, contudo, não é a representação mais acurada de tons de cinza, porque o nosso sistema visual é muito mais sensível à iluminação do canal verde, seguido pelo vermelho e, por último, pelo azul. O ideal seria fazer uma média ponderada levando isso em consideração. Caso queira aprender mais sobre esse assunto e experimentar uma versão com coeficientes de uma média ponderada, confira o vídeo **Como converter uma imagem para escala de cinza com *Python*?**, do canal Programação Dinâmica, no *link* ao lado.

Como converter uma imagem para escala de cinza com *Python*?

https://s.ftd.li/3p7ymo

Detector de arestas

Agora, vamos dar um passo além e criar um detector de arestas verticais aplicando um filtro sobre a imagem. O termo filtro pode ter vários significados, mas, no nosso contexto de processamento de imagens, filtrar uma imagem significa modificá-la realizando operações em cada *pixel*, de modo a atingir algum objetivo. Por exemplo, podemos filtrar uma imagem para deixá-la mais nítida, desfocá-la ou, como faremos agora, destacar as bordas de objetos presentes na imagem.

O filtro Sobel é um tipo de filtro utilizado para detectar bordas, ou seja, aquelas partes da imagem onde há uma mudança brusca de cor ou brilho, como os contornos dos objetos. Quando olhamos para uma imagem, nossos olhos são naturalmente atraídos por essas bordas, porque elas definem as formas dos objetos que vemos. Esse filtro considera não só os *pixels* diretamente à esquerda e à direita de cada ponto, mas também os *pixels* localizados nas diagonais. Ele aplica um conjunto de pesos a uma janela 3×3 ao redor de cada *pixel*, destacando as variações verticais na imagem. Esses pesos, como na matriz abaixo, devem ser aplicados (multiplicados) aos *pixels* no entorno de cada *pixel* da imagem para se obter um novo valor. O nome do filtro é uma homenagem a Irwin Sobel, pesquisador que o inventou.

$$\begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix}$$

Quando aplicamos o filtro Sobel vertical, o que fazemos é passar uma "máscara" (ou um conjunto de regras) sobre cada *pixel* da imagem, levando em consideração os *pixels* ao redor. Essa máscara atribui pesos diferentes aos *pixels* vizinhos; ao somar essas contribuições, o filtro consegue destacar as bordas verticais, que são aquelas em que há uma mudança mais acentuada de cor ou brilho da esquerda para a direita ou vice-versa. Note que, na matriz mencionada, o filtro descreve uma diferença entre vizinhos à esquerda e à direita de cada *pixel*, dando um peso menor para os elementos na diagonal e desconsiderando por completo os vizinhos imediatamente abaixo e acima, bem como o valor do próprio *pixel*. Se quiséssemos aplicar um filtro Sobel horizontal, bastaria calcular a matriz transposta da matriz anterior.

Ao aplicar esse filtro, você verá na imagem abaixo que as arestas verticais estão bem destacadas, enquanto as regiões com transições suaves ou horizontais permanecem mais escuras. Isso ocorre porque o filtro Sobel detecta mudanças bruscas na intensidade dos *pixels* ao longo da direção vertical, evidenciando bordas e detalhes importantes na imagem.

Se quiser saber mais sobre filtro de Sobel, veja o vídeo **Detectando arestas na imagem com filtro Sobel**, do canal Programação Dinâmica, no *link* ao lado.

Detectando arestas na imagem com filtro Sobel

https://s.ftd.li/4fg2r5

A importância da Matemática

A Matemática é essencial na representação e no processamento de imagens digitais. Sem ela, as câmeras digitais, as fotografias e os vídeos que compartilhamos nas redes sociais, e até mesmo as próprias redes sociais como as conhecemos, simplesmente não existiriam. Tudo o que vemos em uma tela, incluindo este texto que você está lendo, precisa ser traduzido em uma linguagem matemática para ser visualizado. Com os conceitos básicos de Matemática que aprendemos na escola, como matrizes, adição e multiplicação, já podemos realizar diversas tarefas fascinantes. Com o entendimento dessas representações, somos capazes de formular e resolver problemas complexos. Agora, o próximo passo é deixar a criatividade guiar novas descobertas.

Na Computação Gráfica 3D, como nas animações da Disney e da Pixar ou em jogos de *videogame*, há ainda mais espaço para aplicar conceitos matemáticos como matrizes e vetores. Essas áreas oferecem inúmeras oportunidades para quem deseja explorar e inovar, visto que as pesquisas e o desenvolvimento de novas técnicas e programas continuam a expandir os limites do que é possível. Um exemplo inspirador é o brasileiro Fernando de Goes, *Principal Scientist* da Pixar, que, em 2022 recebeu um Oscar técnico (*Technical Achievement Award*) da Academia de Artes e Ciências Cinematográficas por sua contribuição à área. Quem sabe um dia, com dedicação e criatividade, você também pode se destacar nesse campo em constante evolução!

Sobre o autor

Hallison Paz é engenheiro de Computação pelo Instituto Militar de Engenharia (IME), mestre e doutorando no Instituto de Matemática Pura e Aplicada (IMPA), onde pesquisa aplicações de Aprendizado de Máquina para novas mídias. Tem experiência em análise, projeto e desenvolvimento de soluções e modelos de Inteligência Artificial e passagem como pesquisador no *Reality Labs Research*, divisão de realidade virtual e aumentada da empresa Meta.

INFO GRÁ FICO

Bandeira do Brasil

Depois de reproduzir a bandeira da Colômbia, imagino que você tenha se perguntado quais outras bandeiras poderíamos fazer em *Python*. E, sim, é possível gerar a bandeira do Brasil com programação usando uma técnica semelhante. Evidentemente, desenhar um losango e um círculo é um pouco mais complexo do que pintar faixas horizontais, mas, com um pouco de conhecimento de Geometria Analítica, como a equação da reta e a equação da circunferência, e uma revisão de inequações, você pode conseguir.

Nesse infográfico, vamos seguir um passo a passo para gerar a bandeira do Brasil. Para simplificar o problema, ignoraremos a faixa branca, as estrelas e a escrita. Você pode tentar reproduzir a bandeira antes de verificar o código. Aceita o desafio?

Resposta do desafio

INFO GRÁ FICO

Bandeira do Brasil

Depois de reproduzir a bandeira da Colômbia, imagino que você tenha se perguntado quais outras bandeiras poderíamos fazer em *Python*. E, sim, é possível gerar a bandeira do Brasil com programação usando uma técnica semelhante. Evidentemente, desenhar um losango e um círculo é um pouco mais complexo do que pintar faixas horizontais, mas, com um pouco de conhecimento de Geometria Analítica, como a equação da reta e a equação da circunferência, e uma revisão de inequações, você pode conseguir.

Nesse infográfico, vamos seguir um passo a passo para gerar a bandeira do Brasil. Para simplificar o problema, ignoraremos a faixa branca, as estrelas e a escrita. Você pode tentar reproduzir a bandeira antes de verificar o código. Aceita o desafio?

Primeiro, vamos definir três funções que nos ajudarão a simplificar a execução: **eq_reta**, **desenhar_quadrilatero** e **desenhar_circulo**. Antes, precisamos chamar a biblioteca PIL para criar imagens.

```
1  from PIL import Image
2
```

A função **eq_reta**, da equação da reta, recebe como parâmetros dois pontos e calcula *m*, o coeficiente angular da reta que passa por esses dois pontos. O retorno da função, ou seja, o resultado que ela devolve, será uma outra função, que chamamos de **lambda** e recebe as coordenadas **x** e **y** de um ponto e devolve um número. Se esse número for positivo, o ponto está acima da reta; se for 0, o ponto pertence à reta; e, se for negativo, está abaixo da reta.

```
3  def eq_reta(p1, p2):
4  » m = (p2[1] - p1[1]) / (p2[0] - p1[0])
5  » return lambda x, y: y - p1[1] - m * (x - p1[0])
6
```

Ao colar o código no *Google Colab*, substitua cada » por uma tabulação (tecla *tab* do teclado) ou dois espaços.

A função **desenhar_quadrilatero**, que desenha um quadrilátero, recebe como parâmetros uma imagem, os quatro vértices que definem o quadrilátero e uma cor. Entre as **linhas de 8 a 11**, o código calcula as equações das quatro retas que ligam esses vértices. Nas **linhas 14 e 15**, temos dois laços, que passam por cada *pixel* da imagem **img**, de acordo com sua altura e seu comprimento, atribuídos nas **linhas 12 e 13**. Entre as **linhas de 16 a 19**, verificamos se o *pixel* está acima das retas **A** e **B** e abaixo das retas **C** e **D**. Se estiver, pintamos aquele *pixel* com a cor recebida na função. Ao final dos laços, devolvemos a imagem modificada.

```
7  def desenhar_quadrilatero(img, pontos, cor):
8  » ladoA = eq_reta(pontos[0], pontos[1])
9  » ladoB = eq_reta(pontos[1], pontos[2])
10 » ladoC = eq_reta(pontos[2], pontos[3])
11 » ladoD = eq_reta(pontos[3], pontos[0])
12 » altura = img.height
13 » comprimento = img.width
14 » for y in range(altura):
15 »  » for x in range(comprimento):
16 »  »  » if (ladoA(x, y) >= 0 and
17 »  »  »  » ladoB(x, y) >= 0 and
18 »  »  »  » ladoC(x, y) <= 0 and
19 »  »  »  » ladoD(x, y) <= 0):
20 »  »  »  » img.putpixel((x, y), cor)
21
22 » return img
23
```

A função **desenhar_circulo**, que desenha um círculo, recebe uma imagem, o centro e a medida do raio do círculo e uma cor. Nas **linhas 27 e 28**, também temos dois laços, que passam por cada *pixel* da imagem **img**, de acordo com sua altura e seu comprimento, atribuídos nas **linhas 25 e 26**. Na **linha 29**, verificamos se o *pixel* está dentro do círculo de centro e raio especificados nos parâmetros, utilizando a equação da circunferência. Se estiver, pintamos aquele *pixel* com a cor recebida na função. Ao final dos laços, devolvemos a imagem modificada.

```
24 def desenhar_circulo(img, centro, raio, cor):
25 » comprimento = img.width
26 » altura = img.height
27 » for i in range(altura):
28 »  » for j in range(comprimento):
29 »  »  » if ((j - centro[0]) ** 2 + (i - centro[1]) ** 2 <= raio ** 2):
30 »  »  »  » img.putpixel((j, i), cor)
31
32 » return img
33
```

INFO GRÁ FICO

Agora sim, somos capazes de gerar parte da bandeira do Brasil. Primeiro, definimos as cores oficiais nas **linhas de 34 a 36**; calculamos as dimensões, com base na razão oficial entre os lados da bandeira nas **linhas de 38 a 40**, e criamos uma imagem, já com o fundo verde, na **linha 42**.

Na **linha 44**, calculamos a distância oficial em que os quatro vértices do losango amarelo devem estar da borda da bandeira e, entre as **linhas de 46 a 51**, armazenamos esses quatro vértices, em ordem específica: o da esquerda, o debaixo, o da direita e o de cima.

Na **linha 53**, chamamos a função **desenhar_quadrilatero**, que recebe a imagem **bandeira** que criamos, os vértices do losango e a cor amarela. A função pintará o losango amarelo em cima do fundo verde.

Nas **linhas 55 e 56**, calculamos o centro e a medida do raio oficiais do círculo azul, e, na **linha 58**, chamamos a função, que recebe a imagem **bandeira**, o centro e a medida do raio do círculo e a cor azul. A função pintará o círculo azul em cima do losango amarelo.

Para finalizar, salvamos a bandeira e exibimo-la.

```
34  AZUL = (0, 39, 118)
35  AMARELO = (255, 223,  0)
36  VERDE = (0, 156, 59)
37
38  altura = 1000
39  comprimento = int(altura / 14 * 20)
40  dimensoes = (comprimento, altura)
41
42  bandeira = Image.new("RGB", dimensoes, VERDE)
43
44  dist_losango = int(altura / 14 * 1.7)
45
46  vertices = [
47    » (dist_losango, altura // 2),
48    » (comprimento // 2, dist_losango),
49    » (comprimento - dist_losango, altura // 2),
50    » (comprimento // 2, altura - dist_losango)
51  ]
52
53  bandeira = desenhar_quadrilatero(bandeira, vertices, AMARELO)
54
55  centro = (comprimento // 2, altura // 2)
56  raio = int(3.5 * altura / 14)
57
58  bandeira = desenhar_circulo(bandeira, centro, raio, AZUL)
59
60  bandeira.save("bandeira_brasil.png")
61  bandeira
```

O resultado será o seguinte, e podemos comparar com a bandeira completa:

A bandeira do Brasil foi inspirada na bandeira do Império do Brasil (1822–1889). O retângulo verde e o losango dourado, que foram preservados, representavam a Casa de Bragança de Dom Pedro I, o primeiro imperador do Brasil, e a Casa de Habsburgo de sua esposa, a imperatriz Maria Leopoldina, respectivamente. O círculo azul com 27 estrelas, que representam as unidades federativas do país, refletem o céu visto na capital Rio de Janeiro em 15 de novembro de 1889.

REFLEXÃO NA PRÁTICA

ARTICULANDO IDEIAS E PRÁTICAS.

CONTEÚDOS DESTA EDIÇÃO

- O que é uma imagem
- Representação de imagens digitais
- Criando imagens com programação
- Processando imagens com programação

MATERIAL COMPLEMENTAR

Debate e reflexão

Divididos em grupo, produzam um conteúdo multimodal relacionado ao tema da Matemática aplicada à Computação Gráfica e a imagens digitais. Podem ser escolhidas diversas formas de expressão, como texto, vídeo, ilustração, charge, *thread* em redes sociais, documentário, notícia ou apresentação digital. O objetivo é que, ao final, os conteúdos sejam compartilhados com a turma para subsidiar um debate coletivo sobre as implicações e aplicações do tema na vida cotidiana e na tecnologia.

O grupo deve escolher um aspecto específico abordado no texto-base, como a representação matricial de imagens, o modelo RGB ou a aplicação de filtros em imagens digitais. Também é possível abordar temas relacionados que tenham relação com o texto base, mas não foram explicados ou abordados diretamente nele como *chroma key*, animação, efeitos especiais etc. Com o tema definido, o grupo deve realizar uma pesquisa complementar, se necessário, e utilizar os conhecimentos adquiridos para criar o conteúdo escolhido. Deve também buscar ilustrar de forma prática o impacto da Matemática no desenvolvimento de tecnologias digitais.

Após a produção, os conteúdos serão apresentados à turma. Caso tenha optado por vídeos, documentários ou apresentações digitais, o grupo pode exibir seus trabalhos em sala de aula. Se escolheram texto, charge ou ilustração, podem fazer uma exposição ou distribuir as produções.

Para finalizar, a turma deve discutir os temas abordados, destacando a importância da Matemática nas tecnologias do dia a dia, e como diferentes formas de linguagem podem influenciar a compreensão e a comunicação de conceitos complexos.

Organizando ideias

- A resolução de uma imagem é definida pelo número de *pixels* que a compõem, organizados em linhas e colunas. Quanto maior a resolução, mais detalhada será a imagem. Dadas três versões de uma mesma imagem com resoluções diferentes: uma de 100 *pixels* × 100 *pixels*, outra de 500 *pixels* × 500 *pixels* e a última de 1 000 *pixels* × 1 000 *pixels*, quais seriam as diferenças entre as três versões da imagem, especialmente em relação à nitidez e aos detalhes visíveis? Calcule as quantidades totais de *pixels* em cada uma das resoluções fornecidas e compare-as, reflitindo sobre como isso impacta na qualidade visual da imagem.
- Suponha que você tenha uma matriz que representa uma imagem em tons de cinza, em que cada elemento da matriz é um valor entre 0 (preto) e 255 (branco). Responda às seguintes questões:
  - **a)** Se você somar 50 a todos os elementos da matriz, o que acontecerá com a imagem resultante?
  - **b)** O que aconteceria se dividíssemos todos os valores da matriz por 2? Como isso afetaria a imagem?
  - **c)** Se uma pessoa quiser inverter as cores da imagem (transformar o branco em preto e vice-versa, por exemplo), como ela poderia fazer isso usando a matriz de valores?

No vestibular

(ENEM/MEC)

Uma construtora, pretendendo investir na construção de imóveis em uma metrópole com cinco grandes regiões, fez uma pesquisa sobre a quantidade de famílias que mudaram de uma região para outra, de modo a determinar qual região foi o destino de maior fluxo de famílias, sem levar em consideração o número de famílias que deixaram a região. Os valores da pesquisa estão dispostos em uma matriz $A = [a_{ij}]$, $i, j \in \{1, 2, 3, 4, 5\}$, em que o elemento a_{ij} corresponde ao total de famílias (em dezena) que se mudaram da região i para a região j durante um certo período, e o elemento a_{ii} é considerado nulo, uma vez que somente são consideradas mudanças entre regiões distintas. A seguir, está apresentada a matriz com os dados da pesquisa.

$$A = \begin{pmatrix} 0 & 4 & 2 & 2 & 5 \\ 0 & 0 & 6 & 2 & 3 \\ 2 & 2 & 0 & 3 & 0 \\ 1 & 0 & 2 & 0 & 4 \\ 1 & 2 & 0 & 4 & 0 \end{pmatrix}$$

Qual região foi selecionada para o investimento da construtora?

- **a)** 1
- **b)** 2
- **c)** 3
- **d)** 4
- **e)** 5

Na hora da redação

Em uma redação, é possível a cobrança de uma discussão, em um texto dissertativo-argumentativo, sobre a importância da Matemática na interpretação da realidade concreta e a transposição de ideias da dimensão física do mundo para a dimensão digital, em que já passamos boa parte de nossas vidas. Este tópico pode ser especialmente relevante à medida que discussões sobre metaverso e humanos digitais avançarem no debate público.

E possível inspirar-se no processo de construir uma representação digital de imagens, que tem por base a interpretação de um fenômeno psicofísico envolvendo a propagação da luz e a interpretação dos sinais luminosos pelo sistema visual humano.

RESENHA

BATE-PAPO COM ESPECIALISTA.

RESENHA

Acesse ao Programa Resenha correspondente a este Articulação.

Cada Programa é pensado como uma extensão do tema aqui proposto.

Pensando na Competência Geral 5 da BNCC – Cultura Digital, o Resenha traz a visão de um especialista. Nossos convidados são professores e profissionais das áreas abordadas e, na maioria das vezes, comunicadores ativos nas redes sociais, tratando cada tema com muita propriedade, mas num bate-papo, de forma leve.

Além do Programa, se quiser acessar os nossos vídeos curtos, eles podem te dar uma prévia do que você vai encontrar.

CONHEÇA O RESENHA

https://s.ftd.li/vm7h4a

PROGRAMA

No *videocast* Resenha de Matemática, o editor Lucas de Souza conversa com o doutorando do IMPA Hallison Paz sobre imagens digitais e qual é a importância da Matemática na área de Tecnologia. Não perca a chance de aprender mais sobre esse assunto!

SHORTS

Que a Matemática está muito presente na Computação, a gente já sabe. Mas como ela aparece quando estamos olhando para a tela do celular ou do computador?

Quer saber mais sobre como a Matemática está envolvida nisso? Venha com a gente para descobrir.

Os filtros são muito usados nas redes sociais para modificar imagens ou vídeos.

Utilizando matrizes, uma das operações que podemos fazer com uma imagem digital são os filtros de imagem. Será que é possível manipular imagens para borrá-las, deixá-las em preto e branco ou evidenciar suas arestas?

#IMAGENSDIGITAIS #MATRIZES #TECNOLOGIA

EXPEDIENTE

ARTICULAÇÃO MATEMÁTICA

FTD educação

Diretor-geral
Ricardo Tavares de Oliveira

Diretor de Conteúdo e Negócios
Cayube Galas

Diretor Adjunto de Sistemas de Ensino
Júlio Ibrahim

Gerente de Conteúdo
Alessandra Naomi Oskata

Gerente de Produção e Fornecedores
Cláudio Espósito Godoy

Editora
Amanda Bonuccelli Voivodic

Editor Assistente
Rodolfo da Silva Campos

Colaborador
Lucas de Souza Santos

Coordenador de Eficiência e *Analytics*
Marcelo Henrique Ferreira Fontes

Supervisora de Fluxo e Qualidade
Letícia Bovolon Bezerra

Assistente de Fluxo
Carolini Fulop

Coordenadora de Preparação, Revisão e Qualidade
Adriana Soares de Souza

Supervisora de Preparação e Revisão
Luciana Duarte Baraldi

Assistente Editorial
Carolina Genúncio

Preparadora de textos
Solange de Araújo Gonçalves

Revisora
Eliana Medina

Coordenadora de Imagem e Texto
Marcia Berne

Imagem e Licenciamento
Equipe FTD

Coordenador de Produção e Arte
Fabiano dos Santos Mariano

Supervisor de Produção e Arte
Pedro Gentile

Projeto Gráfico
Bruno Attili
Carlos Feitosa Ferreira

Editor de Arte
Carlos Feitosa Ferreira

Créditos das imagens e vídeos
(capa) GoodStudio/Shutterstock.com; (p.2) Nicole Piepgras/Shutterstock.com, PeopleImages.com – Yuri A/Shutterstock.com; (p.3) Vadim Starling/Shutterstock.com; (p.4) yingko/Shutterstock.com; (p.6) Ryan DeBerardinis/Shutterstock.com; (p.8) Crédito original/Editoria de Arte; (p.9) vitormarigo/Shutterstock.com; (p.10) vitormarigo/Shutterstock.com; (p.11) vitormarigo/Shutterstock.com, Crédito original/Editoria de Arte; (p.12) Frame Stock Footage/Shutterstock.com, Acervo pessoal; (p.13) Mint Fox/Shutterstock.com, one studio 900/Shutterstock.com, Images Products/Shutterstock.com; (p.15) Owlie Productions/Shutterstock.com; (p.16) kanyanat wongsa/Shutterstock.com, Primakov/Shutterstock.com