

A person is seen from behind, sitting at a desk in a dimly lit room. They are looking at a large computer monitor that displays a vibrant cityscape at sunset, with the sun low on the horizon and buildings silhouetted against the orange and yellow sky. The desk is cluttered with various items: a white mug of coffee with steam rising from it, a computer mouse, a keyboard, and several cables. To the right of the person, there are server racks with glowing blue lights. The background is filled with a dense network of glowing, colorful lines (red, blue, yellow) that resemble data streams or fiber optic cables, creating a futuristic and high-tech atmosphere. The overall scene suggests a focus on computer science and data visualization.

COMPUTAÇÃO GRÁFICA

Aula 1 – Introdução

Curso de Ciência da Computação
Dr. Rodrigo Xavier de Almeida Leão
Cientista de Dados

UNIDADE 1

Esta unidade vai ajudá-lo a dar respostas positivas às perguntas acima e está dividida de acordo com as seguintes seções: a Seção 1.1 traz uma introdução à computação gráfica, apresentando as subdivisões da computação gráfica e os conceitos de realidade material, virtual e aumentada, além de uma breve descrição de tipos de hardware e software de processamento gráfico. A Seção 1.2 apresenta os fundamentos de imagens vetoriais e matriciais, os formatos de arquivos de imagem, estruturas de dados que representam a imagem na memória, modelos de cor, opacidade e transparência. A Seção 1.3 traz atividades práticas de processamento básico de imagens bidimensionais, vetoriais e matriciais, com a criação e desenho de retas e círculos em imagens vetoriais e matriciais.

Onde a Computação Gráfica se Aplica?

Onde a Computação Gráfica se Aplica?

A computação gráfica não está presente apenas em jogos digitais e filmes de animação, mas também em todas as aplicações do nosso dia a dia que, de alguma forma, lidam com imagens e o sentido da visão do ser humano. Os smartphones e tablets são dispositivos comuns que incorporam diversas dessas aplicações, tais como: leitor de impressão digital, câmera digital com detecção de sorrisos, filtros e reconhecimento facial, interação do usuário feita por tela sensível ao toque, exibindo-se uma interface gráfica de alta resolução, além de funcionalidades de aplicativos que possam ser instalados.

Visão Computacional

Visão Computacional é um campo da inteligência artificial e processamento de imagens que se concentra em capacitar computadores a interpretar e compreender o mundo visual. Aqui estão alguns pontos-chave:

1. **Extração de Informações:** A **Visão Computacional** envolve a extração de informações significativas de imagens ou vídeos. Isso pode incluir detecção de objetos, reconhecimento facial, segmentação de imagens muito mais.
2. **Processamento de Imagens:** Os algoritmos de **Visão Computacional** processam imagens para melhorar a qualidade, remover ruídos e realçar características relevantes.
3. **Aprendizado de Máquina:** Muitas técnicas de **Visão Computacional** utilizam algoritmos de aprendizado de máquina, como redes neurais convolucionais (CNNs), para aprender padrões e representações visuais.
4. **Aplicações Práticas:** A **Visão Computacional** é aplicada em várias áreas, como veículos autônomos, diagnóstico médico, segurança, reconhecimento de gestos e muito mais.

Em resumo, a **Visão Computacional** permite que os computadores “vejam” e interpretem o mundo visual de maneira semelhante aos seres humanos. 😊👁



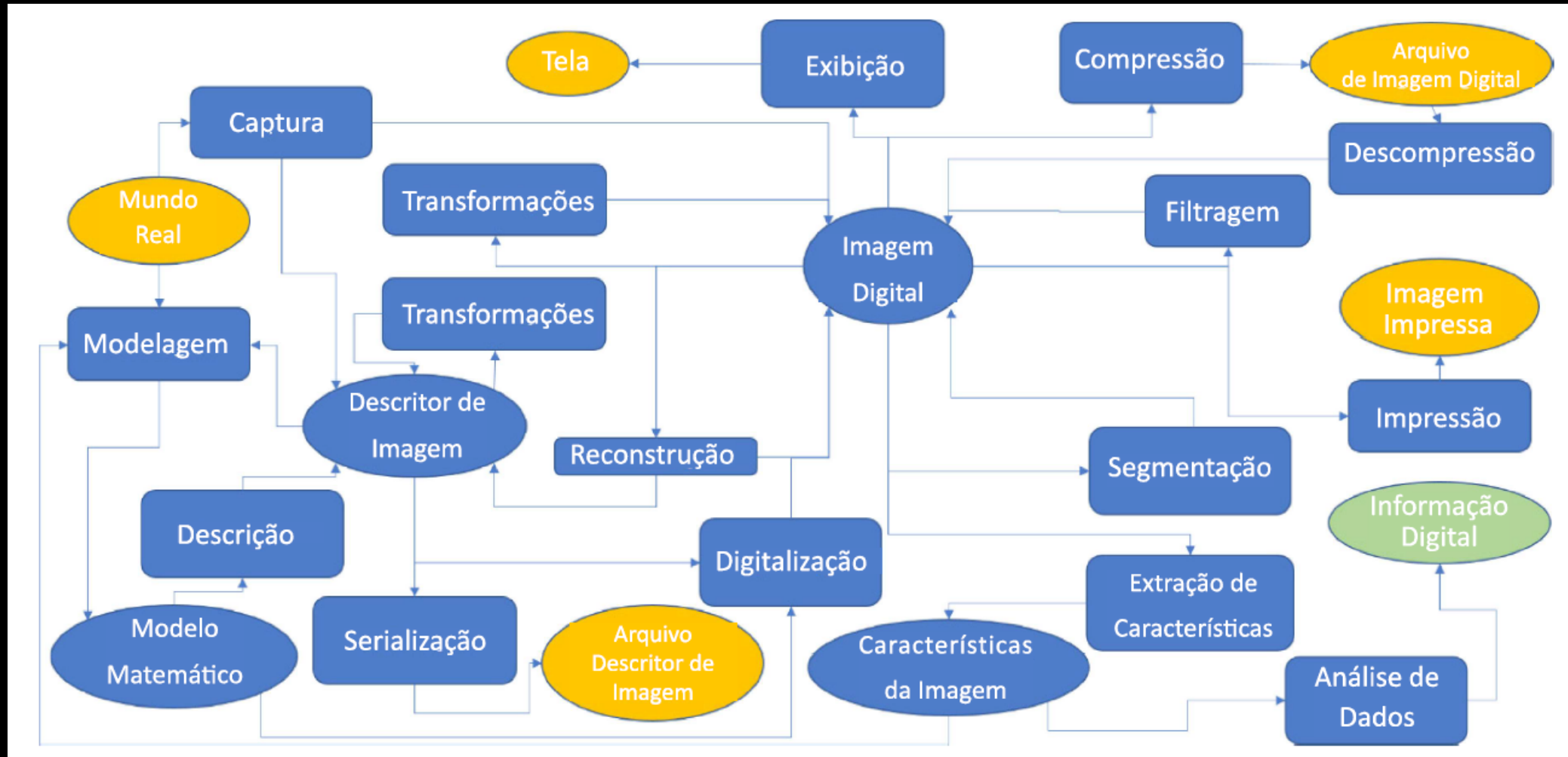
Computação Gráfica

Computação gráfica lida com todos os problemas que envolvem processamento computacional de elementos gráficos, em diversos formatos e para diversas finalidades (MANSSOUR; COHEN, 2006). Inicia sua atuação na captura de imagens ou na modelagem geométrica de um objeto ou cena do mundo real, criando uma representação da cena ou do objeto na memória do computador. Tal representação em memória pode ser um modelo matemático, uma imagem digital ou outro tipo de descritor de imagem que, posteriormente, passará por diferentes tipos de processamento até que possa retornar ao mundo real na forma de imagem exibida em tela, um arquivo de imagem ou imagem impressa.

Alguns dos principais tópicos em computação gráfica incluem:

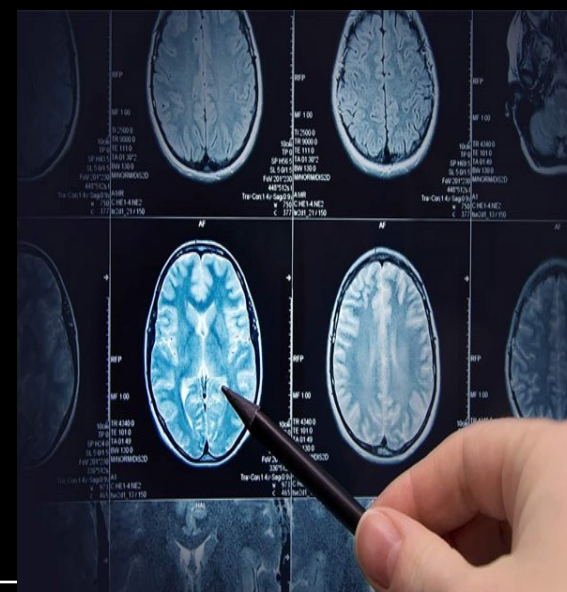
1. **Modelagem 3D:** Criação de modelos tridimensionais de objetos, personagens e ambientes. Isso envolve o uso de software para definir a geometria, texturas e materiais dos modelos.
2. **Renderização:** Processo de gerar uma imagem a partir de um modelo 3D. Existem diferentes métodos de renderização, como renderização em tempo real (usada em jogos) e renderização offline (usada em filmes de animação).
3. **Animação:** Movimento de objetos ou personagens ao longo do tempo. Isso pode incluir animação de personagens, animação de partículas e animação baseada em física.
4. **Computação Gráfica Interativa:** Envolve o uso de gráficos para criar interfaces interativas, como videogames, simulações e visualizações científicas.
5. **Visão Computacional:** Área relacionada que envolve a análise e interpretação de imagens e vídeos por computadores.
6. **Realidade Virtual (VR) e Realidade Aumentada (AR):** Tecnologias que permitem aos usuários interagir com ambientes virtuais ou com informações digitais sobrepostas ao mundo real.

al., 2013). Em azul são apresentados os artefatos (elipses) e processamentos (retângulos) que se encontram ou são realizados no processador ou na memória do computador. Em amarelo, artefatos que podem ser físicos ou arquivos digitais. Em verde é apresentado um artefato resultante do processamento gráfico, mas que será processado fora da computação gráfica.



A modelagem é o processo de representar objetos do mundo real por meio de equações, pontos, retas, superfícies ou outras formas geométricas.

A **captura** é o processo de, literalmente, capturar informações do mundo real com sensores e representar a informação capturada na forma de uma imagem. Há diversos dispositivos de captura de imagem, sendo o mais conhecido a câmera fotográfica, que é um conjunto de lentes e um sensor fotossensível (CCD – *charge-coupled device*). Nem todos os dispositivos



Voltando à Figura 1.1, o processo de captura gera uma **imagem digital** ou um **descriptor de imagem**. Um descriptor de imagem é qualquer estrutura de dados que descreva objetos gráficos ou cenas, mas que não possua um conjunto de pontos em formato de matriz, como na imagem digital. A câmera fotográfica é um exemplo de dispositivo de captura que gera diretamente uma imagem digital. Já a máquina de ressonância magnética gera um conjunto de medições de rádio que não é uma imagem digital, mas sim um descriptor de imagem, que depois passa por um processo de **digitalização** para gerar uma imagem digital.

gráficos, o descriptor de imagens pode sofrer **transformações** geométricas como translação e rotação. Um descriptor de imagem pode ainda passar ou por um processo de **serialização** para ser armazenado em um **arquivo descriptor de imagem**.

SUB-ÁREAS DA COMPUTAÇÃO GRÁFICA

As subáreas da computação gráfica

A computação gráfica pode ser dividida em subáreas com base nos objetivos das aplicações, considerando o que se tem como entrada e o que se busca obter como saída em um fluxo de processamento gráfico. Adotaremos a divisão em quatro subáreas: modelagem tridimensional, síntese de imagens, visão computacional e processamento de imagens (GONZALEZ; WOODS, 2011; HUGHES et al., 2013; MANSSOUR; COHEN, 2006).

TRABALHO

-> Competências para a Vida até dia 09/08
100 pts Oficial 1 e 200 pts Oficial 2

-> GRUPO 4 ALUNOS:

*** Hoje:**

**Pesquisar Artigo sobre Computação Gráfica e apresentar
o RESUMO para turma. (150 pts)**

20/08 -> Apresentação do Artigo (450 pts)

SUBÁREAS DA COMPUTAÇÃO GRÁFICA

Modelagem tridimensional

A modelagem tridimensional é a subárea da computação gráfica cujo objetivo é a criação de modelos matemáticos tridimensionais de objetos ou cenas. A modelagem tridimensional pode ser considerada parte da síntese de imagens, mas o mercado hoje trabalha com equipes dedicadas exclusivamente à modelagem tridimensional, por isso a separação.

SCANNER 3D



RECONSTRUÇÃO



SUBÁREAS DA COMPUTAÇÃO GRÁFICA

Síntese de imagens

A síntese de imagens é a subárea da computação gráfica que visa à produção de imagens sintéticas a partir de modelos matemáticos e transformações do modelo. É também a subárea que trata da criação de interfaces gráficas de usuário e animações (AMMERAAL; ZHANG, 2008;

HUGHES et al., 2013).

A síntese de imagens é fortemente embasada na geometria. Transformações e digitalização são processamentos que envolvem geometria. Movimentações de pontos de observação e fontes de luz são realizadas por transformações geométricas em 3D. A digitalização é realizada por transformações geométricas que levam do espaço 3D para o espaço 2D, a tela de exibição (AMMERAAL; ZHANG, 2008).

SUBÁREAS DA COMPUTÇÃO GRÁFICA

Nos jogos digitais, a equipe de modelagem tridimensional busca construir o modelo mais realístico possível. A equipe de síntese de imagem vai criar, a partir desse modelo, um descritor de imagem para cada tipo de equipamento onde o jogo será processado. Para um console de última geração, será gerado um descritor de alta resolução, com um grande número de pontos descrevendo a superfície do objeto. Para o console da geração anterior será gerado um descritor de mais baixa resolução, com um número menor de pontos descrevendo a superfície do objeto.

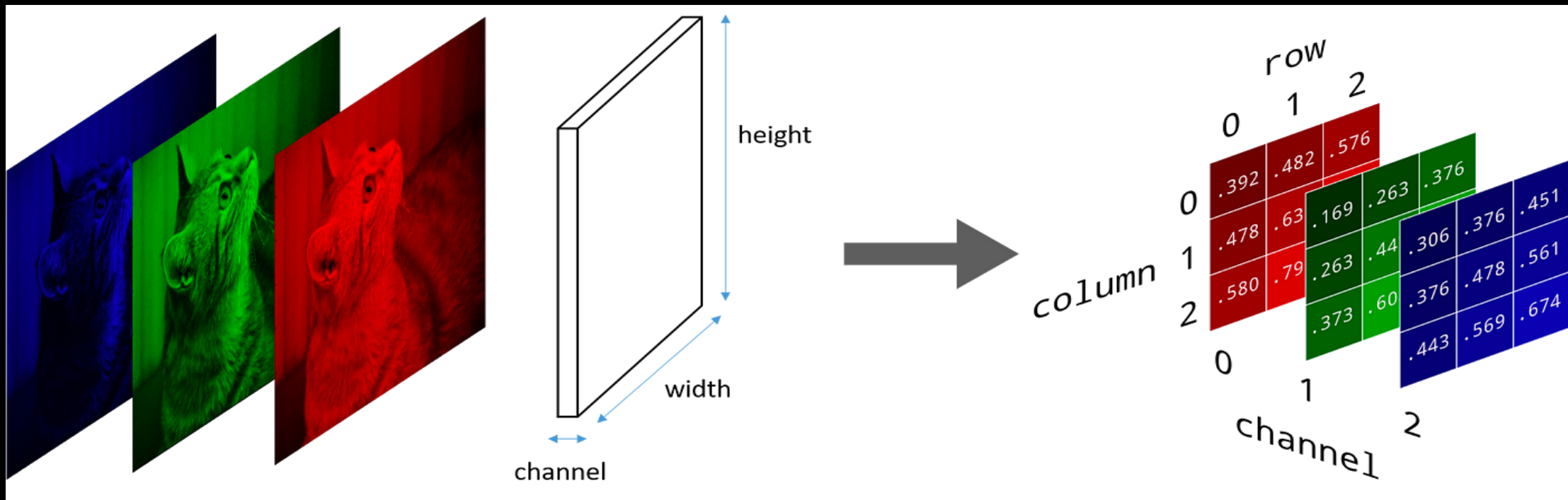


SUBÁREAS DA COMPUTAÇÃO GRÁFICA

Visão computacional

A visão computacional é a subárea da computação gráfica que visa a extração de informações úteis a partir de imagens capturadas do mundo real.

Matriz de Convolução



SUBÁREAS DA COMPUTAÇÃO GRÁFICA

Processamento de imagens

O processamento de imagens é a subárea da computação gráfica que visa o tratamento e alterações de imagens digitais. O objetivo do processamento de imagens é fazer modificações em imagens digitais para se obterem outras imagens digitais (GONZALEZ; WOODS, 2011).

O processamento de imagens é a subárea que engloba qualquer aplicação de edição de imagens. O processamento gráfico mais comum é a filtragem, mas a segmentação e a reconstrução também são usadas no processamento de imagens. Se um objeto for removido de uma imagem digital, por exemplo, uma lacuna será criada, e a imagem precisará passar por um processo de reconstrução. Se for relevante aplicar um filtro apenas sobre um objeto de interesse da imagem, é preciso antes fazer uma segmentação.

REALIDADE VIRTUAL

A realidade virtual (RV) é uma tecnologia que permite aos usuários interagirem com ambientes digitais tridimensionais simulados, criando a sensação de estar presente fisicamente em um espaço virtual. A RV utiliza dispositivos específicos, como headsets (óculos de realidade virtual), controladores manuais e sensores, para proporcionar uma experiência imersiva. Abaixo estão os principais aspectos da realidade virtual:



APLICAÇÕES

Entretenimento e Jogos: Jogos de realidade virtual são uma das aplicações mais populares, oferecendo experiências imersivas em ambientes interativos.

Educação e Treinamento: Simulações em RV são usadas para treinamento em áreas como medicina, aviação e operações militares, onde os usuários podem praticar em ambientes controlados.

Arquitetura e Design: Arquitetos e designers utilizam RV para visualizar e explorar projetos de edifícios e espaços antes da construção.

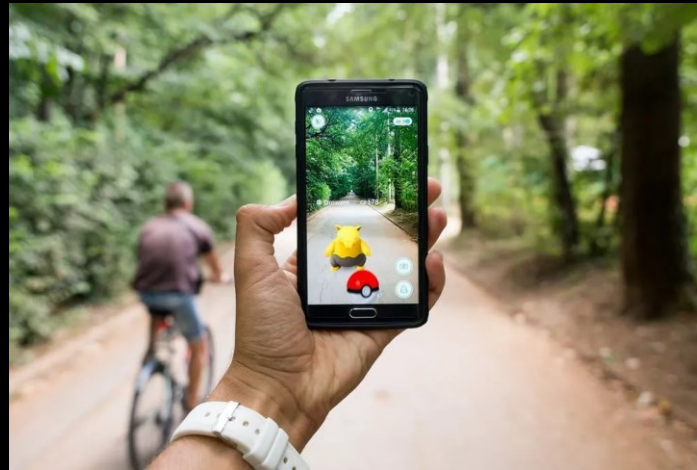
Medicina: Cirurgias simuladas em RV permitem que os médicos pratiquem procedimentos complexos em um ambiente seguro.

Turismo Virtual: Oferece a oportunidade de explorar locais turísticos e históricos em 3D sem a necessidade de viajar fisicamente.

Reabilitação: Usada em terapias para reabilitar pacientes com lesões físicas, oferecendo exercícios controlados em um ambiente virtual.

REALIDADE AUMENTADA

A Realidade Aumentada (RA) é uma tecnologia que combina elementos virtuais com o mundo real, sobrepondo informações digitais (como imagens, sons ou textos) ao ambiente físico que o usuário está visualizando. Ao contrário da Realidade Virtual (RV), que imerge o usuário em um ambiente completamente digital, a RA mantém o usuário no mundo real enquanto adiciona camadas de conteúdo digital.



APLICAÇÃO DE VISÃO PARA CLASSIFICAÇÃO

```
[ ] !pip install tensorflow  
    !pip install keras  
    import keras  
    keras.__version__
```



Mostrar saída oculta

!pip install tensorflow: Instala o TensorFlow, uma biblioteca de código aberto amplamente usada para aprendizado de máquina e redes neurais.

!pip install keras: Instala o Keras, uma API de alto nível usada para construir e treinar modelos de aprendizado profundo. O Keras agora faz parte do TensorFlow, então essa instalação é geralmente redundante.

APLICAÇÃO DE VISÃO PARA CLASSIFICAÇÃO

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout, BatchNormalization
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.preprocessing import image
import numpy as np
```

`from tensorflow.keras.models import Sequential`: Importa a classe Sequential, que é usada para construir um modelo sequencial, onde as camadas são empilhadas linearmente.

`from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout, BatchNormalization`: Importa camadas que serão usadas para construir a rede neural, incluindo:

Conv2D: Camada convolucional 2D.

MaxPooling2D: Camada de pooling máximo para reduzir a dimensionalidade.

Flatten: Achata o tensor em uma única dimensão.

Dense: Camada densa ou totalmente conectada.

Dropout: Camada que aplica a técnica de dropout para evitar overfitting.

BatchNormalization: Normaliza as ativações de saída das camadas anteriores.

`from tensorflow.keras.preprocessing.image import ImageDataGenerator`: Importa a classe ImageDataGenerator para gerar dados de imagem em tempo real com aumento.

`from tensorflow.keras.preprocessing import image`: Importa funções para carregar e pré-processar imagens.

`import numpy as np`: Importa o NumPy, uma biblioteca para operações com arrays.

APLICAÇÃO DE VISÃO PARA CLASSIFICAÇÃO

```
# Exemplo de construção de um modelo simples
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(64, 64, 3)),
    MaxPooling2D(pool_size=(2, 2)),
    BatchNormalization(),
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(1, activation='sigmoid')
])

print(model.summary())
```

`model = Sequential([...])`: Cria um modelo sequencial com várias camadas empilhadas.

`Conv2D(32, (3, 3), activation='relu', input_shape=(64, 64, 3))`: Adiciona uma camada convolucional com 32 filtros, um tamanho de kernel de 3x3, usando a função de ativação relu.

O `input_shape=(64, 64, 3)` define que as imagens de entrada têm 64x64 pixels e 3 canais de cor (RGB).

`MaxPooling2D(pool_size=(2, 2))`: Adiciona uma camada de pooling máximo com um tamanho de pool de 2x2, reduzindo as dimensões da imagem.

`BatchNormalization()`: Normaliza as ativações da camada anterior, acelerando o treinamento e melhorando a estabilidade.

`Flatten()`: Achata a entrada para criar uma única dimensão linear, necessária para a camada densa. `Dense(128, activation='relu')`: Adiciona uma camada densa com 128 neurônios e função de ativação relu.

`Dropout(0.5)`: Aplica dropout, desativando 50% dos neurônios da camada anterior durante o treinamento para prevenir overfitting.

`Dense(1, activation='sigmoid')`: Adiciona uma camada de saída com 1 neurônio e função de ativação sigmoid, usada para problemas de classificação binária.


`print(model.summary())`: Exibe um resumo do modelo, incluindo a configuração das camadas e o número de parâmetros.

APLICAÇÃO DE VISÃO PARA CLASSIFICAÇÃO

```
[ ] # Compilando o modelo
    model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```

model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy']): Compila o modelo com o otimizador adam, a função de perda binary_crossentropy para classificação binária e a métrica accuracy para monitorar a precisão durante o treinamento.

```
[ ] from google.colab import drive
    drive.mount('/gdrive', force_remount=True)
```

 Mounted at /gdrive

model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy']): Compila o modelo com o otimizador adam, a função de perda binary_crossentropy para classificação binária e a métrica accuracy para monitorar a precisão durante o treinamento.

```
[ ] train_local = '/gdrive/My Drive/Colab Notebooks/DATASETS/covimages/train'
    test_local = '/gdrive/My Drive/Colab Notebooks/DATASETS/covimages/test'
```


```
[ ] # Exemplo de uso de ImageDataGenerator
train_datagen = ImageDataGenerator(rescale=1./255)
train_generator = train_datagen.flow_from_directory(
    train_local,
    target_size=(64, 64),
    batch_size=32,
    class_mode='binary'
)

# Ajuste do modelo aos dados
model.fit(train_generator, epochs=10)
```

`train_generator = train_datagen.flow_from_directory(...)`: Gera lotes de dados de imagem a partir do diretório `train_local`. As imagens são redimensionadas para 64x64 pixels, processadas em lotes de 32, e as classes são tratadas como binárias.

`model.fit(train_generator, epochs=10)`: Treina o modelo por 10 épocas usando os dados fornecidos pelo `train_generator`

```
[ ] # Fluxo de dados de teste
test_datagen = ImageDataGenerator(rescale=1./255)
test_generator = test_datagen.flow_from_directory(
    test_local,
    target_size=(64, 64),
    batch_size=32,
    class_mode='binary'
)
```

 Found 2180 images belonging to 2 classes.

`test_datagen = ImageDataGenerator(rescale=1./255)`: Cria um objeto `ImageDataGenerator` para os dados de teste, escalando os valores dos pixels para o intervalo `[0, 1]`.

`test_generator = test_datagen.flow_from_directory(...)`: Gera lotes de dados de imagem a partir do diretório `test_local`, semelhante ao `train_generator`.


```
[ ] # Exemplo de uso de ImageDataGenerator
train_datagen = ImageDataGenerator(rescale=1./255)
train_generator = train_datagen.flow_from_directory(
    train_local,
    target_size=(64, 64),
    batch_size=32,
    class_mode='binary'
)

# Ajuste do modelo aos dados
model.fit(train_generator, epochs=10)
```

`train_generator = train_datagen.flow_from_directory(...)`: Gera lotes de dados de imagem a partir do diretório `train_local`. As imagens são redimensionadas para 64x64 pixels, processadas em lotes de 32, e as classes são tratadas como binárias.

`model.fit(train_generator, epochs=10)`: Treina o modelo por 10 épocas usando os dados fornecidos pelo `train_generator`

```
[ ] # Fluxo de dados de teste
test_datagen = ImageDataGenerator(rescale=1./255)
test_generator = test_datagen.flow_from_directory(
    test_local,
    target_size=(64, 64),
    batch_size=32,
    class_mode='binary'
)
```



Found 2180 images belonging to 2 classes.

`test_datagen = ImageDataGenerator(rescale=1./255)`: Cria um objeto `ImageDataGenerator` para os dados de teste, escalando os valores dos pixels para o intervalo `[0, 1]`.

`test_generator = test_datagen.flow_from_directory(...)`: Gera lotes de dados de imagem a partir do diretório `test_local`, semelhante ao `train_generator`.

```
▶ # Avaliação do modelo nos dados de teste
test_loss, test_accuracy = model.evaluate(test_generator)
print(f"Test Loss: {test_loss}")
print(f"Test Accuracy: {test_accuracy}")
```

⇒ [Mostrar saída oculta](#)

`test_loss, test_accuracy = model.evaluate(test_generator)`: Avalia o modelo nos dados de teste, retornando a perda e a precisão.

`print(f"Test Loss: {test_loss}")`: Exibe a perda nos dados de teste.

`print(f"Test Accuracy: {test_accuracy}")`: Exibe a precisão nos dados de teste.

```
[ ] print("Mapeamento das classes durante o treinamento:")
    print(train_generator.class_indices)
```

⇒ Mapeamento das classes durante o treinamento:
{'covid': 0, 'non': 1}

```
from tensorflow.keras.preprocessing import image
import numpy as np
import matplotlib.pyplot as plt

# Caminho da nova imagem a ser classificada
image_path = '/gdrive/My Drive/Colab Notebooks/DATASETS/covimages/unica/renata.jpg'

# Carregar a imagem
img = image.load_img(image_path, target_size=(64, 64))

# Converter a imagem em um array numpy
img_array = image.img_to_array(img)

# Adicionar uma dimensão extra para criar um batch (de 1 imagem)
img_array = np.expand_dims(img_array, axis=0)

# Pré-processar a imagem (normalização)
img_array /= 255.0

# Fazer a predição
prediction = model.predict(img_array)
```



```
# Interpretação dos resultados

# Obter o índice da classe prevista
predicted_class_index = int(prediction[0] > 0.5) # Ajuste para um problema binário

# Mapeamento para o nome da classe correspondente
predicted_class_label = class_labels[predicted_class_index]

result = f"A imagem é classificada como '{predicted_class_label}' com uma probabilidade de {max(prediction[0][0], 1-prediction[0][0])*100:.2f}%"

if prediction[0] > 0.5:
    result = f"Classe 1 com uma probabilidade de {prediction[0][0]*100:.2f}%"
else:
    result = f"Classe 0 com uma probabilidade de {(1-prediction[0][0])*100:.2f}%"

# Exibir a imagem junto com o resultado
plt.imshow(image.array_to_img(img_array[0]))
plt.title(result)
plt.axis('off') # Remove os eixos
plt.show()
```

`from tensorflow.keras.preprocessing import image`: Reimporta o módulo de pré-processamento de imagens.

`import matplotlib.pyplot as plt`: Importa o Matplotlib para exibição de gráficos e imagens.

`image_path`: Define o caminho para uma nova imagem a ser classificada.

`img = image.load_img(image_path, target_size=(64, 64))`: Carrega a imagem e a redimensiona para 64x64 pixels.

`img_array = image.img_to_array(img)`: Converte a imagem em um array NumPy.

`img_array = np.expand_dims(img_array, axis=0)`: Adiciona uma dimensão extra ao array para representar um lote com uma única imagem.

`img_array /= 255.0`: Normaliza a imagem, dividindo os valores dos pixels por 255

Existem várias fontes onde você pode encontrar datasets adequados para treinar modelos de classificação de imagens. Abaixo estão algumas opções, juntamente com exemplos de datasets populares para tarefas de classificação de imagens, incluindo conjuntos específicos para categorias como animais, objetos, rostos, etc.

1. Kaggle

Descrição: Kaggle é uma plataforma de ciência de dados que oferece uma ampla gama de datasets gratuitos. Você pode encontrar datasets de classificação de imagens para diversas categorias.

Exemplos de Datasets:

Dogs vs Cats: Um dataset popular para classificação binária onde as classes são "cães" e "gatos".

CIFAR-10: Um dataset com 60.000 imagens em 10 classes diferentes, útil para tarefas mais complexas.

Como Acessar: Crie uma conta no Kaggle e acesse a seção [Datasets](#).

2. TensorFlow Datasets (TFDS)

Descrição: TensorFlow Datasets é uma coleção de datasets prontos para uso com TensorFlow. Muitos desses datasets são de imagens e são fáceis de carregar diretamente em um ambiente de treinamento.


Exemplos de Datasets:

MNIST: Um dataset com dígitos manuscritos, ideal para começar com a classificação de imagens.

Fashion MNIST: Semelhante ao MNIST, mas com imagens de roupas.

Como Acessar: Use a biblioteca TensorFlow para carregar os datasets.


python

 Copiar código

```
import tensorflow as tf
import tensorflow_datasets as tfds

(train_ds, test_ds), ds_info = tfds.load(
    'mnist',
    split=['train', 'test'],
    shuffle_files=True,
    as_supervised=True,
    with_info=True
)
```



 Continuar gerando

3. Public Datasets from Universities and Research Institutions

Descrição: Muitas universidades e instituições de pesquisa publicam datasets como parte de seus trabalhos acadêmicos ou de projetos de pesquisa.

Exemplos de Datasets:

[Oxford Pets Dataset](#): Um dataset com 37 categorias de raças de gatos e cães.

[LFW \(Labeled Faces in the Wild\)](#): Um dataset de faces rotuladas, usado para reconhecimento de rostos.

4. ImageNet

Descrição: ImageNet é um dos maiores e mais populares datasets de classificação de imagens, contendo milhões de imagens categorizadas em milhares de classes.

Exemplos de Datasets:

[ImageNet](#): Usado frequentemente em competições de visão computacional como o ILSVRC.

Como Acessar: Devido ao tamanho, ImageNet geralmente é acessado através de uma solicitação de uso para pesquisas acadêmicas.

5. Google Datasets Search

Descrição: Uma ferramenta de pesquisa para encontrar datasets públicos em várias categorias, incluindo visão computacional.

Como Acessar: Visite [Google Datasets Search](#) e procure por datasets específicos.

6. GitHub Repositories

Descrição: Muitos projetos open-source em GitHub incluem datasets ou links para datasets em seus repositórios.

Como Acessar: Busque por repositórios relacionados a visão computacional e classificação de imagens.