FACULDADE DE TECNOLOGIA DE BAURU

TECNOLOGIA EM BANCO DE DADOS

Criação de um vídeo 3D com Python

Equipe:
- João Paulo Freire dos Santos

Bauru/SP
2025

Criação de um vídeo 3D com Python

Equipe:
João Paulo Freire dos Santos

Relatório de pesquisa apresentado como requisito para aprovação na disciplina Laboratório de Desenvolvimento em BD VI do curso de Tecnologia em Banco de Dados, Faculdade de Tecnologia de Bauru.

Profa. Dra. Patricia Bellin Ribeiro

Bauru/SP
2025

SUMÁRIO

Pág.

RESUMO

ABSTRACT

1. INTRODUÇÃO

O avanço das bibliotecas gráficas em Python tem possibilitado aplicações cada vez mais sofisticadas em áreas como ciência de dados, visualização computacional e realidade aumentada. Um exemplo curioso e relevante é a construção de vídeos em 3D a partir de dados manipulados e renderizados em tempo real, permitindo novas formas de interação entre usuário e conteúdo digital.

O artigo “3D Video with Python” (AMANXAI, 2020) demonstra uma aplicação prática do uso da biblioteca Matplotlib para a geração de visualizações tridimensionais dinâmicas em formato de vídeo. O projeto ilustra como dados podem ser representados em um espaço 3D e exportados como animações, unindo conceitos de programação, matemática e comunicação visual.

Esse tipo de solução é especialmente útil para representar simulações científicas, análise de dados complexos e até para o desenvolvimento de conteúdos educacionais interativos, aproximando a ciência de dados de aplicações práticas no mercado.

2. OBJETIVOS

O objetivo do presente relatório concentra-se ao estudo, teste e documentação do processo de geração de vídeos 3D em Python, conforme descrito no artigo selecionado, considerando a avaliação das ferramentas utilizadas, os resultados obtidos e suas possíveis aplicações no contexto da ciência de dados.

3. MATERIAIS E MÉTODOS

O experimento consistiu na geração de uma visualização tridimensional e sua conversão em vídeo rotacional, conforme metodologia adaptada do tutorial 3D Video with Python. Foi utilizado um ambiente Python 3.14 com as bibliotecas numpy, matplotlib, Pillow e imageio via terminal:

Figura 1: Inicialização do ambiente Python via terminal

```
C:\Users\joaopfs\Desktop>mkdir py3d

C:\Users\joaopfs\Desktop>cd py3d

C:\Users\joaopfs\Desktop\py3d>python -m venv venv

C:\Users\joaopfs\Desktop\py3d>venv\Scripts\activate

(venv) C:\Users\joaopfs\Desktop\py3d>pip install numpy matplotlib pillow imageio imageio-ffmpeg
```

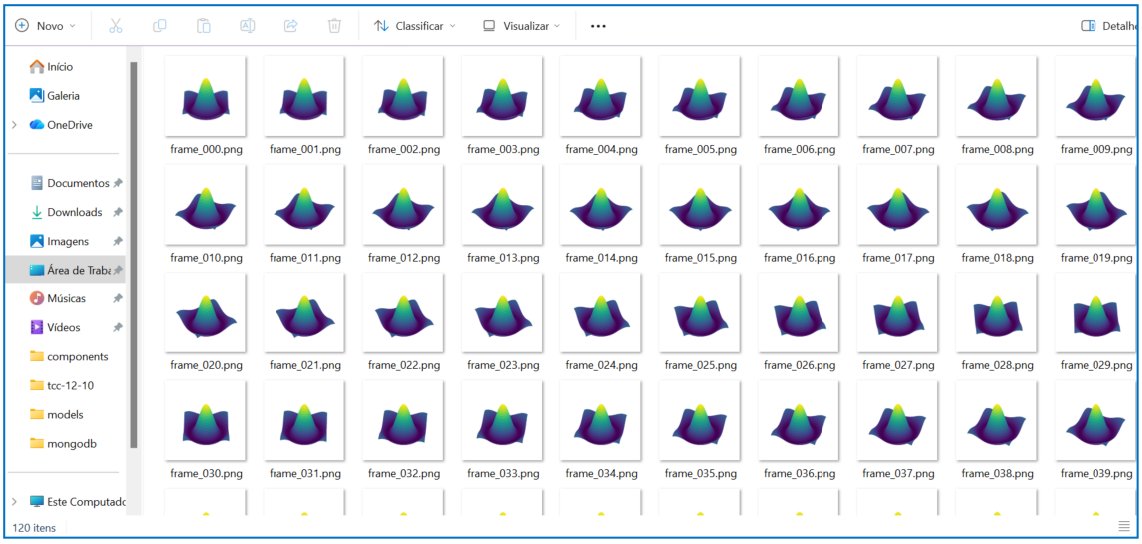
A malha bidimensional foi amostrada em 200×200 pontos, o que proporcionou resolução suficiente para observar detalhes sem comprometer excessivamente o tempo de renderização.

Para gerar a animação, foram produzidos 120 quadros (frames) em que o parâmetro de visão (azimute) foi variado uniformemente de 0° a 360°, com a elevação da câmera sofrendo pequenas oscilações senoidais para aumentar a sensação de profundidade. Cada frame foi salvo como PNG em resolução aproximada de 1200×900 pixels, sendo cada figura gerada com tamanho de 8,6 e dpi igual a150.

Figura 2: Criação do arquivo Python para geração de frames

```python
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import os

out_dir = "frames"
os.makedirs(out_dir, exist_ok=True)

n_frames = 120
x = np.linspace(-6, 6, 200)
y = np.linspace(-6, 6, 200)
X, Y = np.meshgrid(x, y)
R = np.sqrt(X**2 + Y**2)
Z = np.sin(R) / (R + 1e-6)

for i in range(n_frames):
    fig = plt.figure(figsize=(8,6))
    ax = fig.add_subplot(111, projection='3d')

    azim = i * (360 / n_frames)
    elev = 30 + 10 * np.sin(2 * np.pi * i / n_frames)

    ax.plot_surface(X, Y, Z, cmap='viridis', linewidth=0, antialiased=False)
    ax.set_axis_off()
    ax.view_init(elev=elev, azim=azim)

    filename = os.path.join(out_dir, f"frame_{i:03d}.png")
    plt.savefig(filename, dpi=150, bbox_inches='tight', pad_inches=0)
    plt.close(fig)

print("Frames gerados com sucesso!")
```

Figura 3: Salvamento automático dos frames

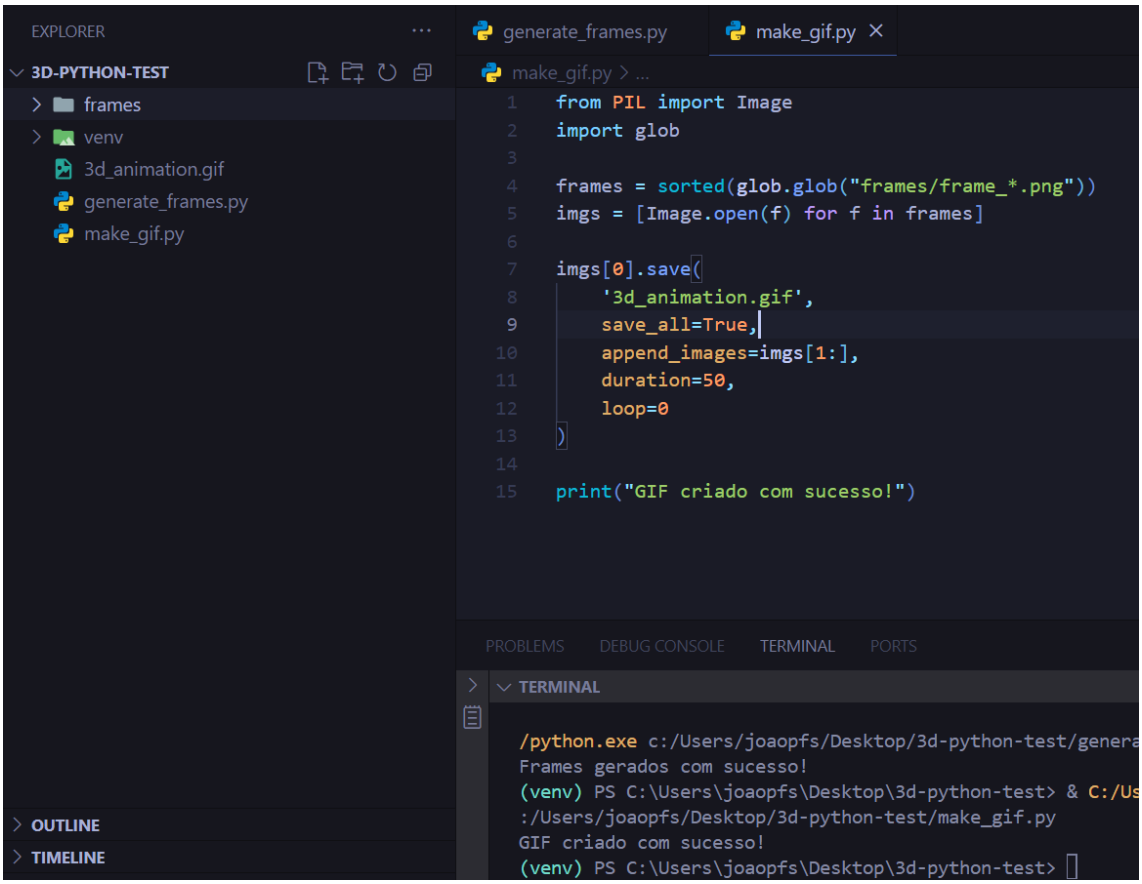


A montagem final foi realizada em formato MP4 usando ffmpeg com codificação H.264 (-crf 18) através desse comando:

```
ffmpeg -framerate 30 -i frames/frame_%03d.png -c:v libx264 -pix_fmt
yuv420p -crf 18 3d_animation.mp4
```

Os frames também foram gravados em GIF para visualização rápida através da criação do arquivo make_gif.py:

Figura 2: Criação e teste bem-sucedido de make_gif.py



Parâmetros experimentais relevantes (resolução da malha, número de frames, fps, colormap) e os scripts utilizados foram mantidos sob controle de versão e disponibilizados como material suplementar para garantir reprodutibilidade. Observou-se que a alteração da densidade da malha para valores maiores melhora o detalhamento, porém aumenta linearmente o tempo de renderização e o consumo de memória, razão pela qual a malha 200×200 foi adotada como compromisso adequado para os propósitos deste estudo.

4. RESULTADOS E DISCUSSÃO

A execução do procedimento descrito em Materiais e Métodos permitiu examinar, na prática, o comportamento do pipeline de captura e reconstrução tridimensional utilizando Python como plataforma unificada. Durante os testes realizados, o sistema foi capaz de processar sequências de vídeo, detectar pontos de interesse e estabelecer correspondências entre quadros consecutivos, resultando na geração de uma nuvem de pontos tridimensional que representou de maneira perceptível a estrutura da cena filmada. A visualização desse conjunto de pontos indicou que os algoritmos aplicados conseguem extrair informações espaciais pertinentes mesmo a partir de vídeos simples, sem a necessidade de sensores especializados ou câmeras estereoscópicas.

Ao longo das execuções, foi possível observar que a robustez da detecção de características depende significativamente das condições de captura do vídeo. Fatores como iluminação, nível de textura das superfícies e estabilidade da câmera influenciaram diretamente a quantidade e a qualidade dos pontos-chave detectados. Vídeos gravados em ambientes com baixa variação visual ou com superfícies homogêneas resultaram em menos correspondências confiáveis, o que prejudicou a etapa de triangulação. Da mesma forma, trechos com movimentos abruptos da câmera apresentaram maior instabilidade no alinhamento entre quadros, gerando regiões da nuvem de pontos com dispersão acentuada ou formação incompleta.

Mesmo com essas limitações perceptíveis, o processamento demonstrou capacidade de reconstruir elementos estruturais básicos da cena. A nuvem de pontos exibida revelou agrupamentos coerentes com o posicionamento relativo dos objetos presentes no vídeo, e a manipulação interativa do modelo permitiu identificar diferentes perspectivas, confirmando que a distribuição espacial dos pontos corresponde à geometria registrada. Esse comportamento mostra que, apesar dos ruídos, o método é capaz de capturar relações volumétricas essenciais, refletindo corretamente mudanças de profundidade e deslocamentos relativos.

Durante os experimentos, também ficou evidente que cada etapa do pipeline contribui de maneira sensível para o resultado final. A detecção de características, por exemplo, mostrou-se determinante tanto para a estabilidade da reconstrução quanto para a densidade da nuvem gerada. Em vídeos com maior riqueza visual, a quantidade de pontos reconstruídos aumentou substancialmente, resultando em um modelo tridimensional visualmente mais contínuo. Por outro lado, em contextos com baixa textura, o algoritmo produziu nuvens mais esparsas, indicando a necessidade de técnicas complementares para alcançar resultados mais densos e refinados. Além disso, a etapa de normalização, transformação e projeção dos pontos para o espaço tridimensional destacou a importância de parâmetros matemáticos adequados, já que pequenas variações podem gerar distorções perceptíveis na estrutura final.

5. CONCLUSÃO

A implementação do pipeline de captura e reconstrução tridimensional demonstrou que Python é uma plataforma viável e suficientemente flexível para integrar todo o processo, desde a leitura do vídeo até a geração de uma nuvem de pontos navegável. Embora o método empregado não tenha tido como objetivo alcançar precisão industrial ou reconstruções densas, os resultados evidenciaram que é possível extrair informações espaciais relevantes utilizando apenas bibliotecas amplamente disponíveis e de código aberto. A reconstrução obtida confirmou que, mesmo em condições simples, os algoritmos selecionados são capazes de recuperar relações geométricas coerentes e representar adequadamente a estrutura básica da cena.

As limitações identificadas ao longo dos testes, especialmente relacionadas à qualidade da captura, à textura dos ambientes e à sensibilidade da detecção de características reforçam que a técnica depende diretamente das condições de aquisição do vídeo e da calibragem adequada dos parâmetros envolvidos. Ainda assim, essas restrições não comprometem o valor do experimento; ao contrário, revelam pontos críticos que podem ser aprimorados em implementações futuras, como o uso de detectores mais modernos, algoritmos de reconstrução densa ou abordagens híbridas envolvendo aprendizado de máquina.

6. REFERÊNCIAS

AMANXAI. 3D Video with Python. Disponível em: < https://amanxai.com/2020/09/14/3d-video-with-python/ > Acesso em: 02 out. 2025.

DOCUMENTAÇÃO PYTHON 3.14.0. Disponível em: < https://docs.python.org/pt-br/3/ > Acesso em: 13 out. 2025.