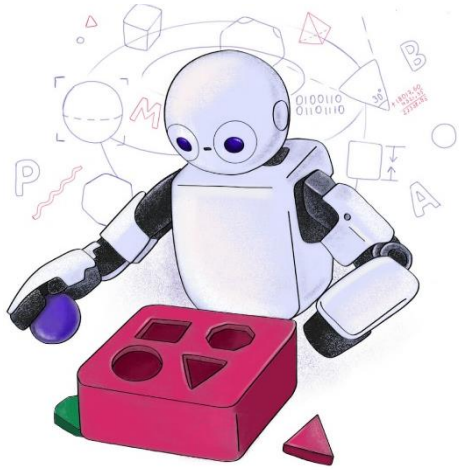
TP558 - Tópicos avançados em Machine Learning:

TripoSR: Fast 3D Object Reconstruction from a Single Image



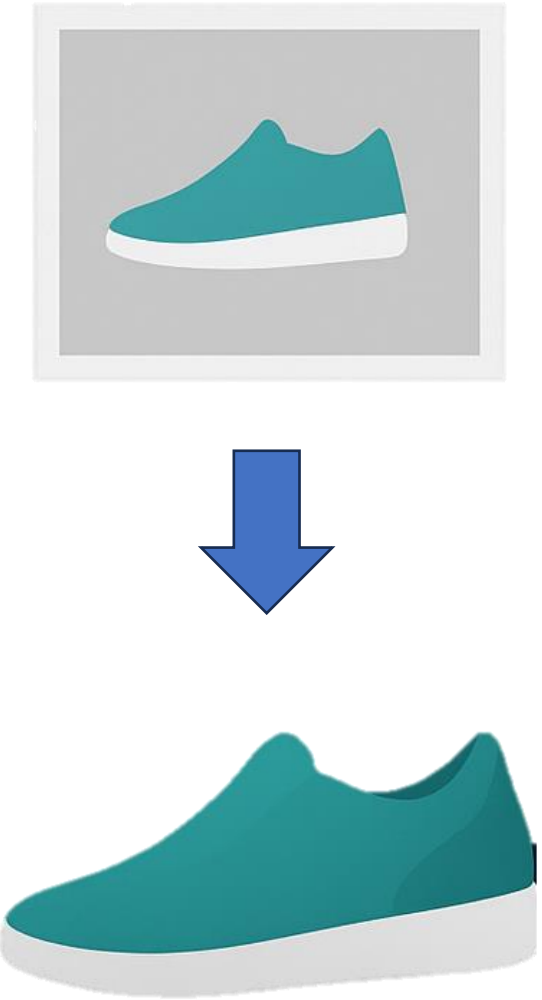
Inatel

Brithany Michelle Oliva Chuquimia
brithany.oliva@mtel.inatel.br

Introdução

- A ideia principal é transformar uma única imagem em um **modelo 3D completo**.
- Essa tarefa é muito importante em áreas como realidade aumentada, jogos, design, animação, robótica e outras.
- Embora já existem grandes bases de dados de objetos 3D, mas nem sempre encontramos o objeto exato que precisamos.

Introdução



- Para superar essa limitação, surgiram métodos que usam **modelos de difusão 2D** para gerar objetos 3D a partir de imagens ou descrições de texto.

- O problema?
  - Esses métodos são lentos, custosos e difíceis de otimizar.

Introdução

- Foi nesse contexto que nasceu o **TripoSR**.
- Ele é um modelo baseado em **transformers** e no **LRM** (Large Reconstruction Model).
- Objetivo: reconstruir objetos 3D de forma rápida (menos de 0,5s) e com alta qualidade, usando apenas uma imagem.

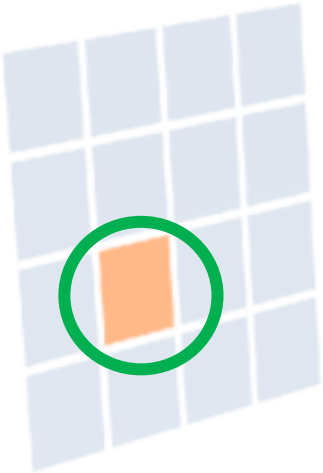
Introdução

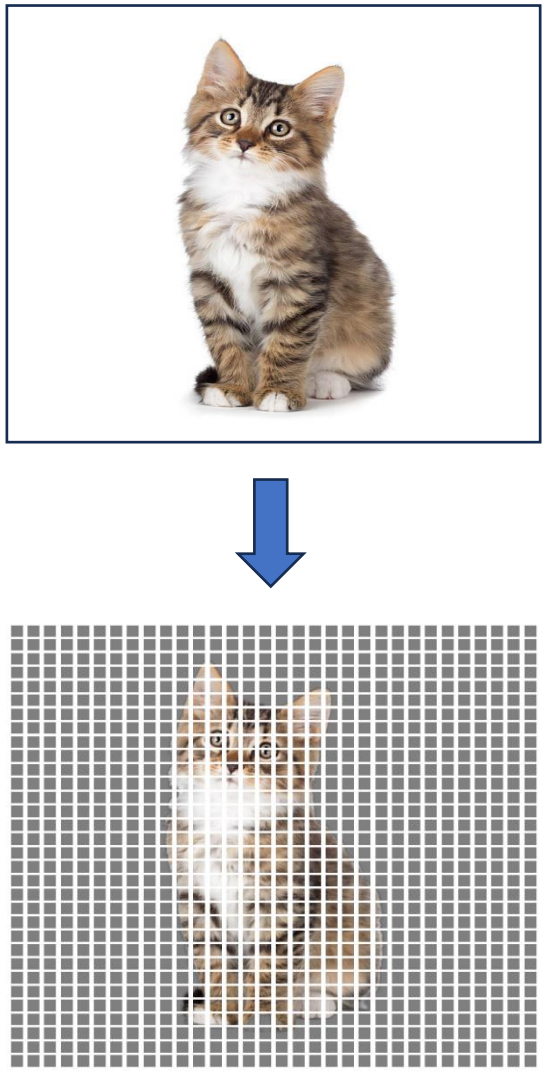
- Ele traz melhorias em dados, arquitetura e treinamento, garantindo mais eficiência e generalização.

- Em resumo: o **TripoSR** resolve as limitações de **tempo** e custo dos métodos anteriores, democratizando a criação de modelos 3D.

Fundamentação teórica

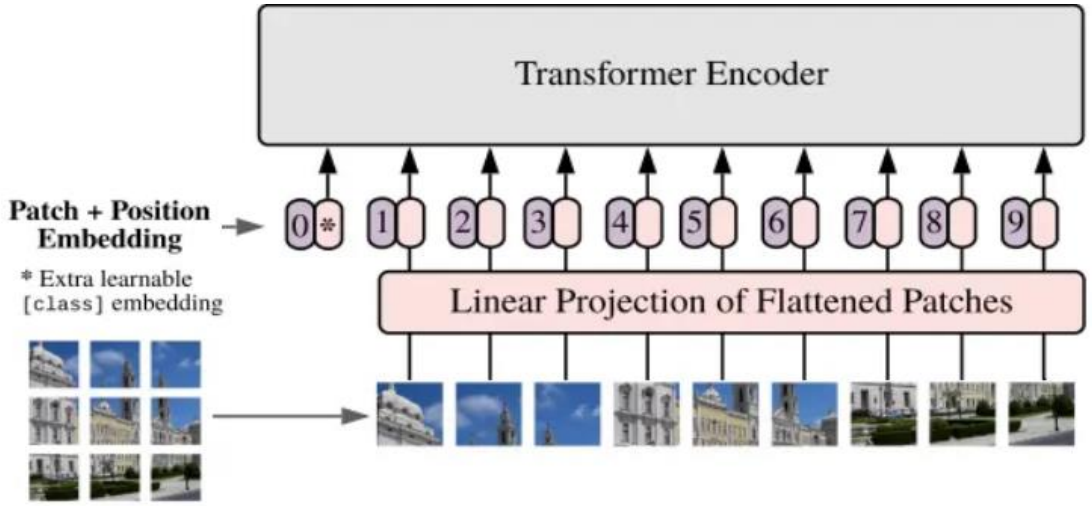
- Em vez de ver uma imagem como milhões de pixels, o computador a divide em pequenos pedaços (**patches**), como se fossem as peças de um quebra-cabeça. Cada pedaço é transformado em um vetor que representa suas características, e esse vetor é o **token** de imagem.





Fundamentação teórica

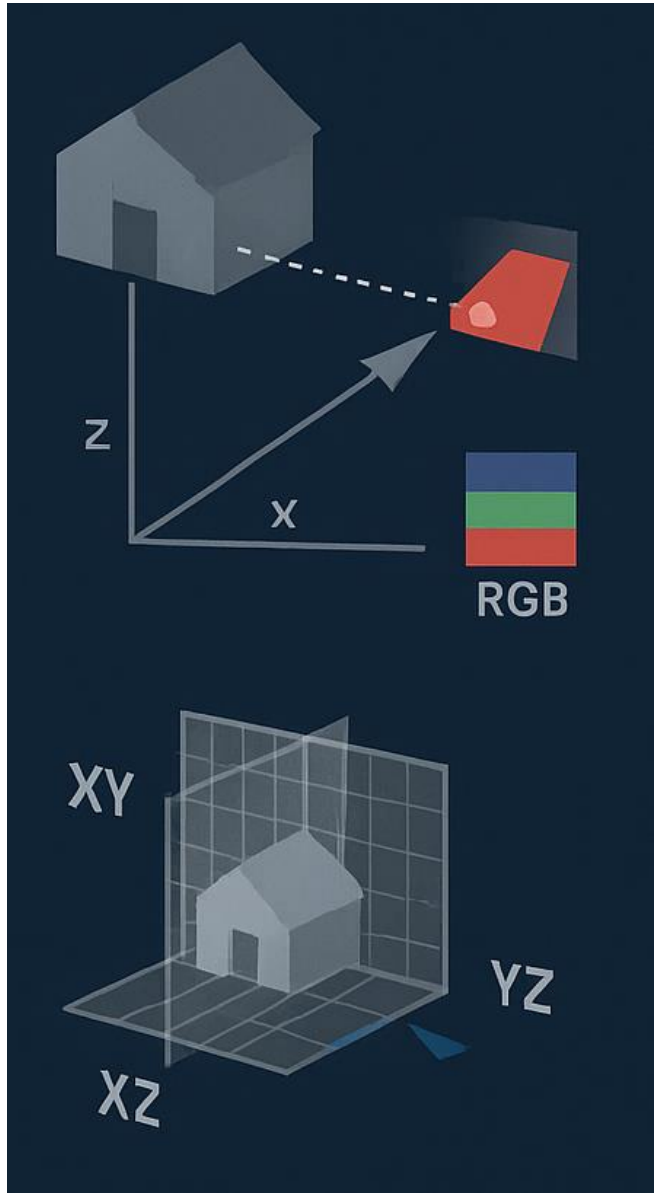
- **Transformers**: ele é o "cérebro" que analisa todos esses pedaços.
- Ele não processa os tokens sequencialmente, mas sim considera todos os tokens de uma vez, descobrindo as relações de dependência entre eles.



- No TripoSR, o Transformer **analisa a relação entre os tokens** de imagem para entender a estrutura e o contexto global da foto (exemplo: *"esses tokens de borda formam a alça da xícara").*

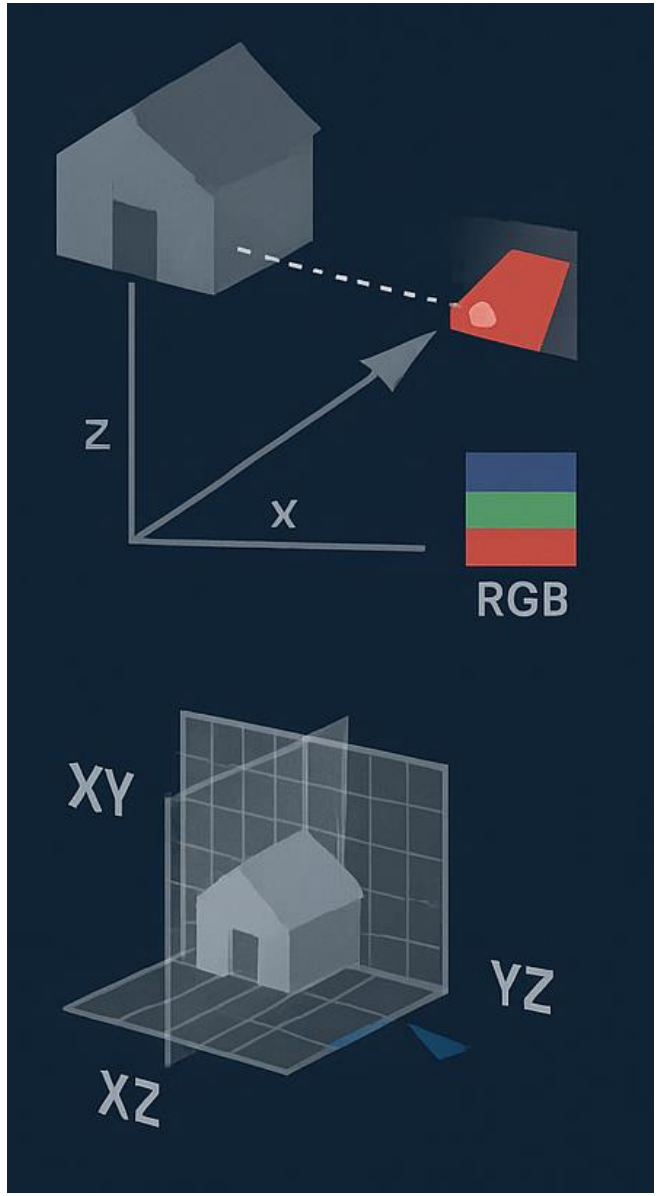
Fundamentação teórica

- O TripoSR se baseia em Neural Radiance Fields (**NeRFs**)
- Pense em um NeRF como uma rede neural que aprende a "desenhar" uma cena 3D inteira.
- Ele pode **prever a cor e a densidade** de qualquer ponto no espaço, a partir de uma visão em 2D.
- O problema é que NeRFs tradicionais são muito lentos para gerar um modelo 3D.



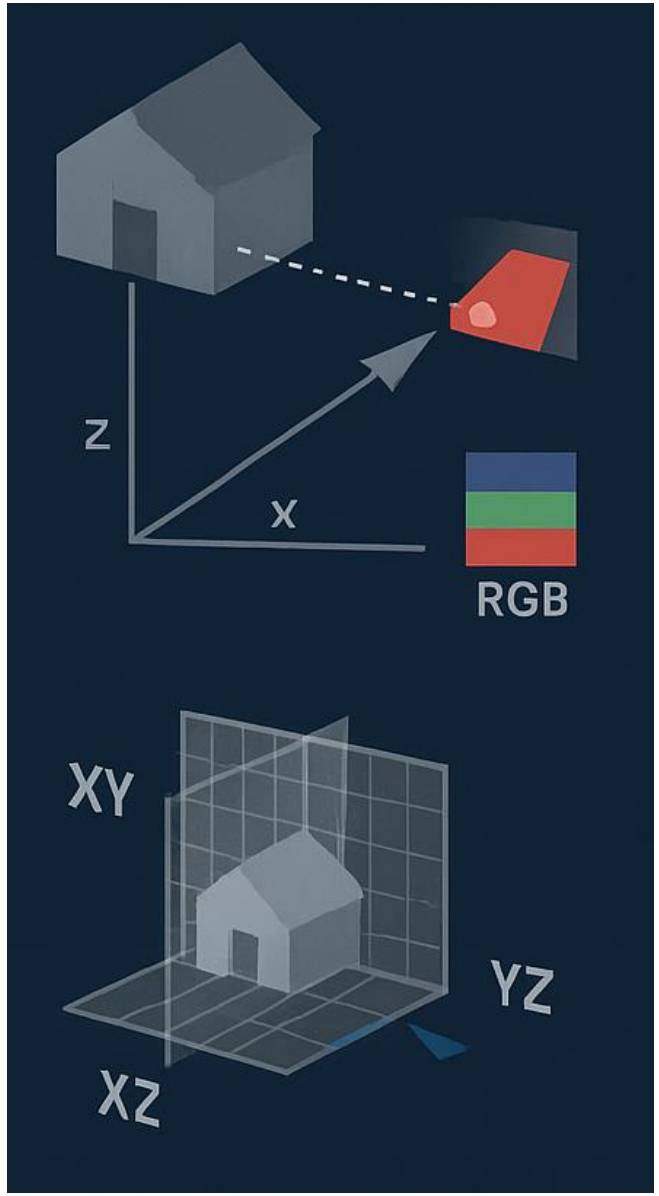
Fundamentação teórica

- Para resolver a lentidão, o TripoSR não usa um NeRF tradicional. Em vez disso, ele usa uma **Representação Triplanar**.
- Um **Triplane** é uma representação 3D compacta.
- Ele **armazena as informações de forma eficiente em três planos** de características ortogonais (XY, XZ, e YZ)
- Isso permite a renderização 3D de forma rápida.



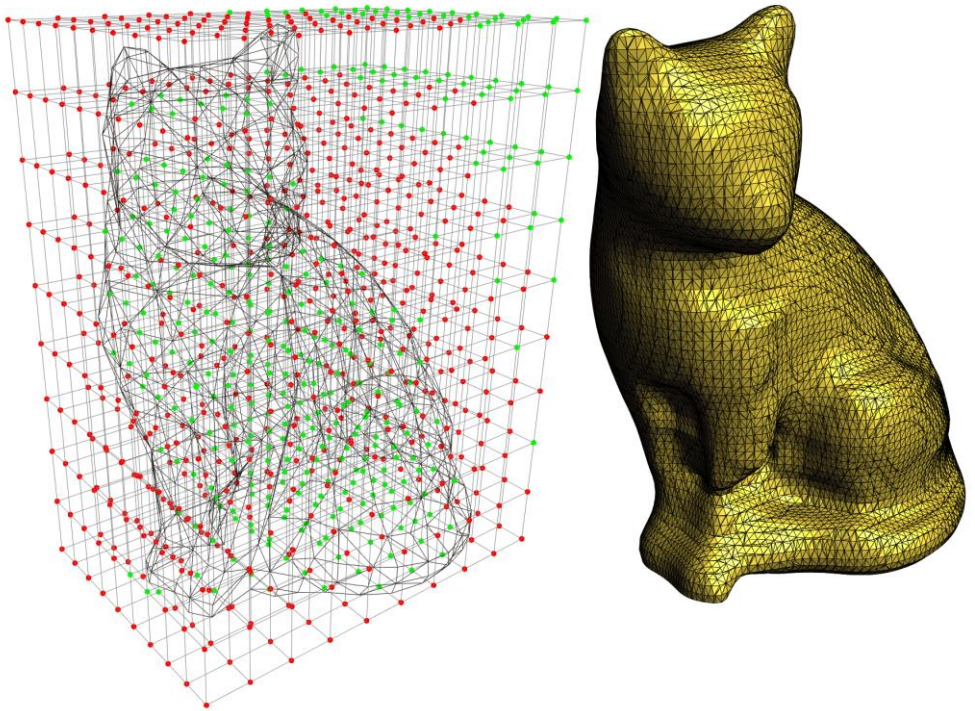
Fundamentação teórica

- Em o NeRF, se faz uso de uma rede neural o qual é o **MLP**
- O MLP é uma **rede neural simples que, faz a mágica de mapear as coordenadas 3D para cor e densidade**, ao receber as coordenadas de um ponto, consulta os triplanes e determina se há algo lá e qual a cor.
- A sigla **MLP** (Multi-Layer Perceptron) significa que ela tem múltiplas camadas, permitindo que ela aprenda relações complexas.



Fundamentação teórica

- Depois de criar a nuvem de informações com o NeRF, o objeto ainda não está em um formato que você possa usar em um software 3D.
- O **Marching Cubes** é um algoritmo que resolve isso.
- Ele transforma a nuvem de informações em uma malha 3D, é esse formato de malha que permite que o objeto seja salvo como um arquivo .obj



Arquitetura e funcionamento

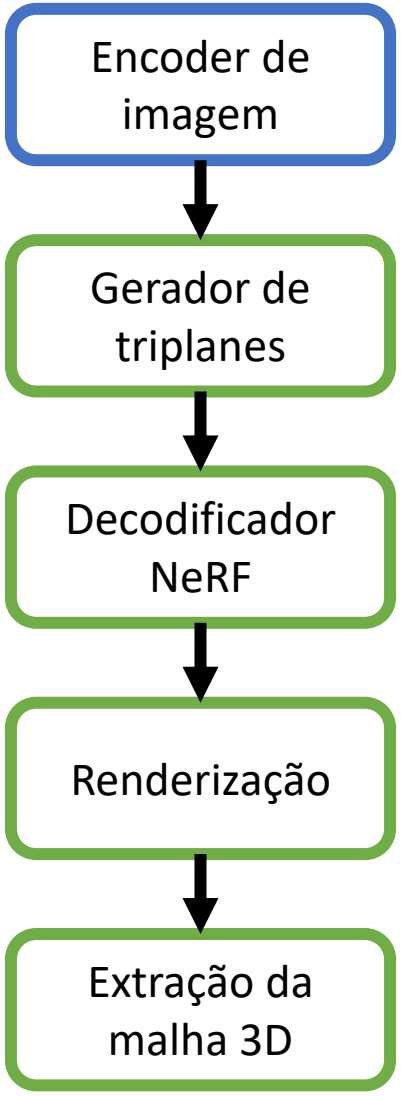
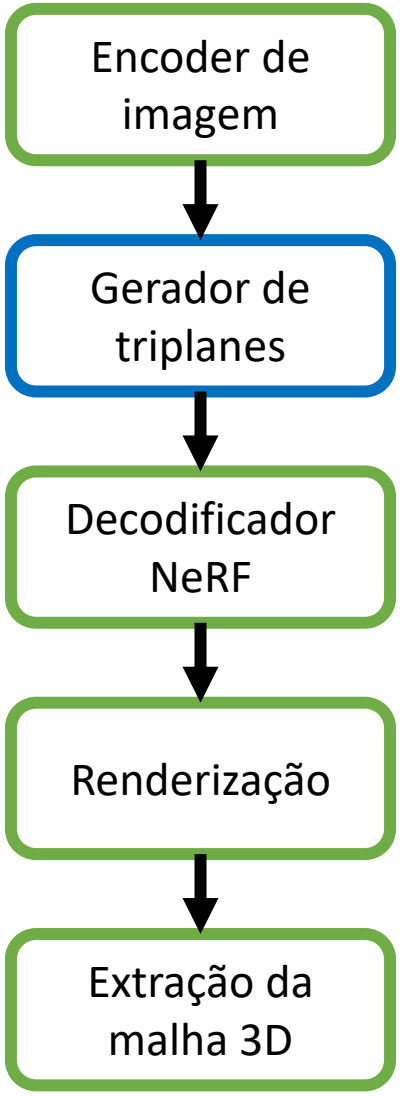


Image Tokenizer	image resolution	512×512
	patch size	16
	# attention layers	12
	# feature channels	768

- Encoder de imagem (transformer):
  - A foto entra no sistema.
  - É cortada em bloquinhos de 16×16 pixels (patches).
  - Cada bloquinho é transformado a uma lista de números (768 características).
  - O Transformer entende como esses bloquinhos se relacionam (*ex: esta curva faz parte desa borda*)

Resultado: uma **descrição matemática da imagem** com informações globais e detalhes finos

Arquitetura e funcionamento

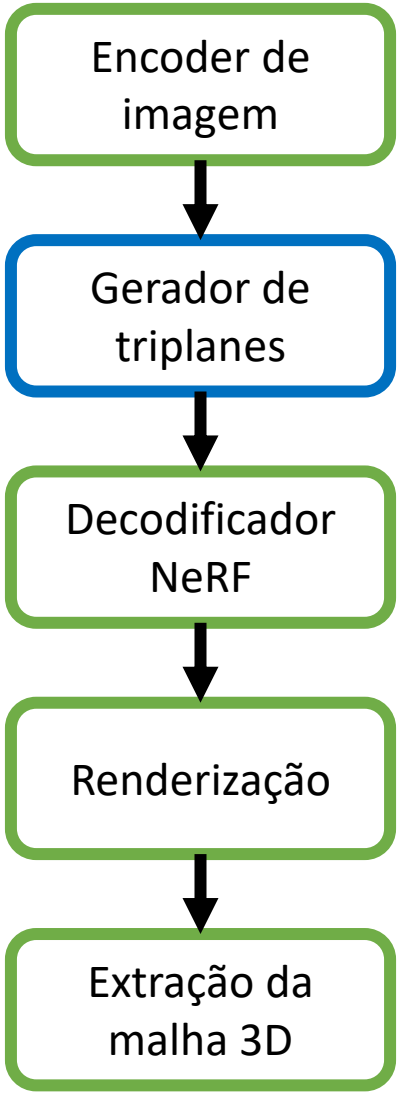


- Gerador de triplanes:
  - Essa descrição vira 3 mapas 2D ortogonais (XY, XZ, YZ).
  - É como olhar o objeto de frente, de cima e de lado ao mesmo tempo.

Triplane Tokenizer	`# tokens`	$32 \times 32 \times 3$
	`# channels`	16

- São criados 3 planos de 32x32 células cada
- Cada célula do triplano armazena um vetor de 16 números

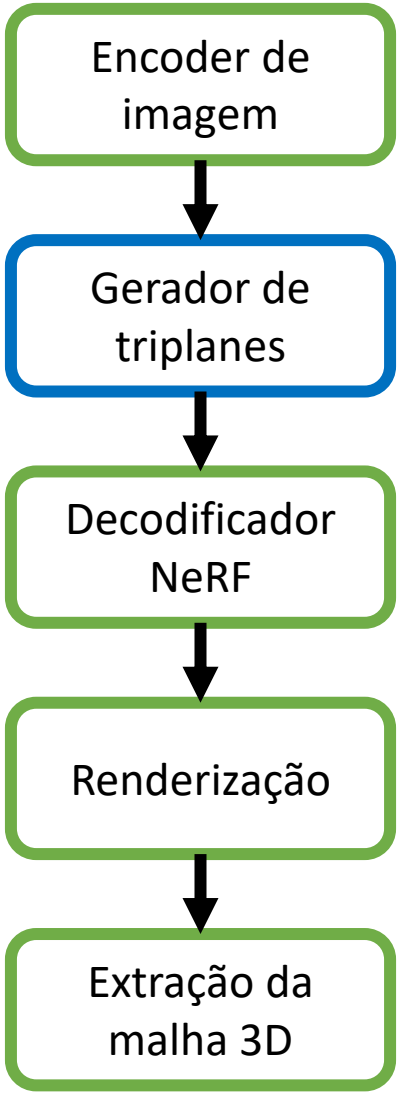
Arquitetura e funcionamento



Backbone	# channels	1024
	attention layers	16
	# attention heads	16
	attention head dim	64
	cross attention dim	768

- O Backbone é o cérebro do Transformer que conecta tudo.
- Os vetores internos têm um tamanho de 1024
- O modelo aplica 16 camadas de cuidados
- Cada camada analisa as informações de 16 perspectivas diferentes.
- Cada cabeça usa vetores de tamanho 64.

Arquitetura e funcionamento

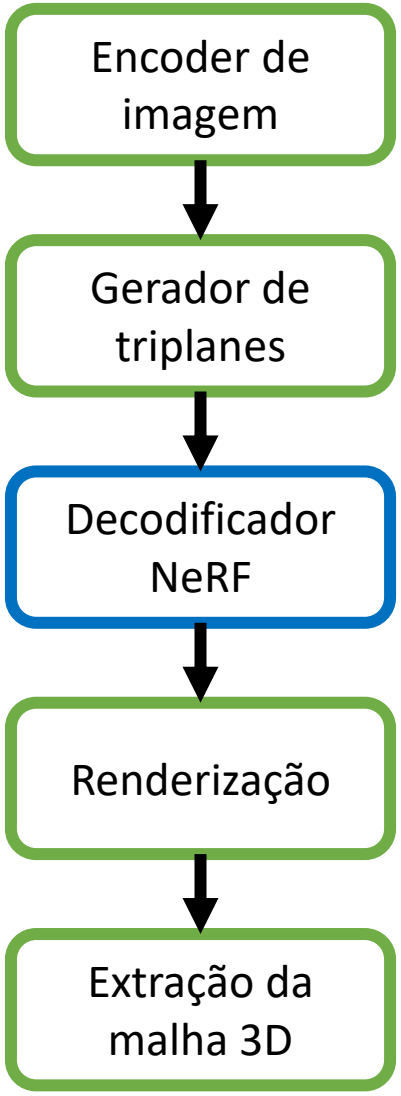


Triplane Upsampler	`factor`	2
	`# input channels`	1024
	`# output channels`	40
	`output shape`	$64 \times 64 \times 40$

- Aumenta a resolução dos triplanos para dar-lhes mais detalhes.
- Duplica a resolução espacial de cada plano.
- Os triplanos finais têm mais detalhes espaciais

Resultado: uma **representação compacta em 3D** que é rápida e eficiente

Arquitetura e funcionamento

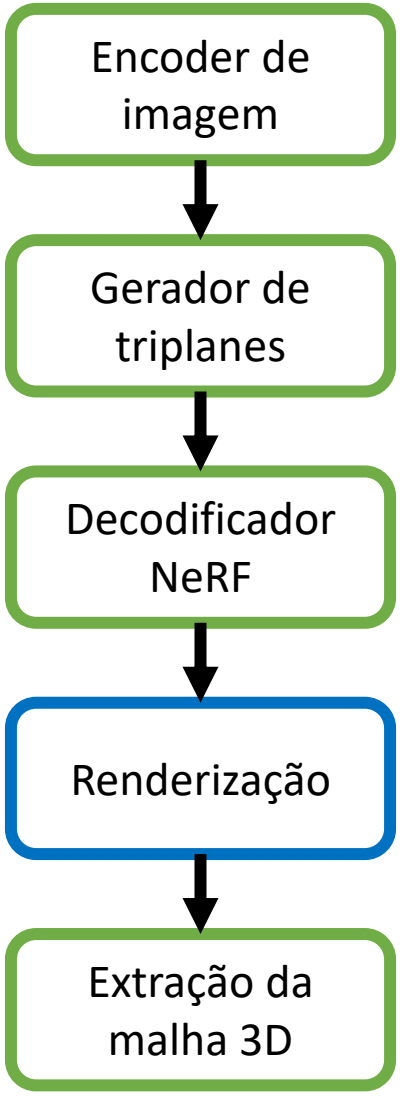


NeRF MLP	width	64
	# layers	10
	activation	SiLU

- Decodificador NeRF (triplane NeRF):
  - Para cada ponto do espaço 3D, o modelo responde:
    - *"Aqui tem superfície ou está vazio?" (densidade).*
    - *"Se tem superfície, de que cor é?" (RGB).*
  - Para isso, usa os triplanos como referência.

Resultado: o modelo sabe **onde está a superfície e qual cor ela tem**

Arquitetura e funcionamento

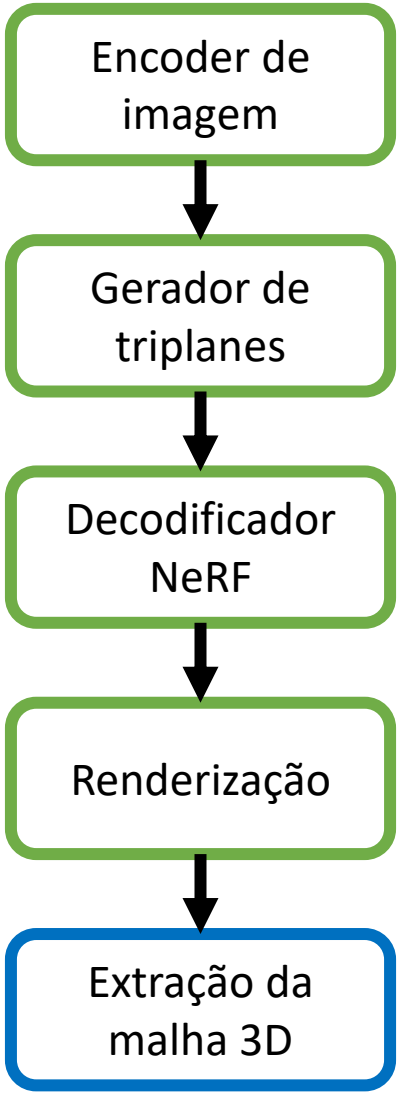


Renderer	# samples per ray	128
	radius	0.87
	density activation	exp
	density bias	−1.0

- Renderização:
  - O modelo dispara raios virtuais como se fosse uma câmera olhando para o objeto.
  - Cada raio verifica 128 pontos no espaço.
  - O raio da esfera 3D onde a geometria é amostrada é 0,87

Resultado: uma **imagem renderizada** a partir da reconstrução 3D.

Arquitetura e funcionamento



- Extração da malha 3D:
  - Até aqui temos só um “campo de densidade” (informação de onde há matéria)
  - O algoritmo **Marching Cubes** transforma isso em uma **malha poligonal** (com vértices e faces)
  - Esse é o formato padrão usado em softwares 3D

Resultado: um **modelo 3D pronto para uso**

Treinamento e otimização

- Dataset: O TripoSR foi treinado com um subconjunto de alta qualidade do dataset **Objaverse**, que contém mais de um milhão de modelos 3D diversos.
- Processo de Treinamento: O modelo aprende a prever um modelo 3D a partir de uma única imagem, comparando-o com o modelo 3D original do dataset. do mercado.

Treinamento e otimização

- Parâmetros de treinamento
  - Otimizador: AdamW
  - Taxa de aprendizado inicial: 4e-4
  - Scheduler (Cosine): diminui essa taxa suavemente ao longo do treino com uma curva cosseno
  - Warm-up (2000 passos): início suave para evitar instabilidade

Training	`learning rate`	$4e{-4}$
	`optimizer`	AdamW
	`lr scheduler`	CosineAnnealingLR
	`# warm-up steps`	$2,000$
	λ_{LPIPS}	2.0
	λ_{mask}	0.05

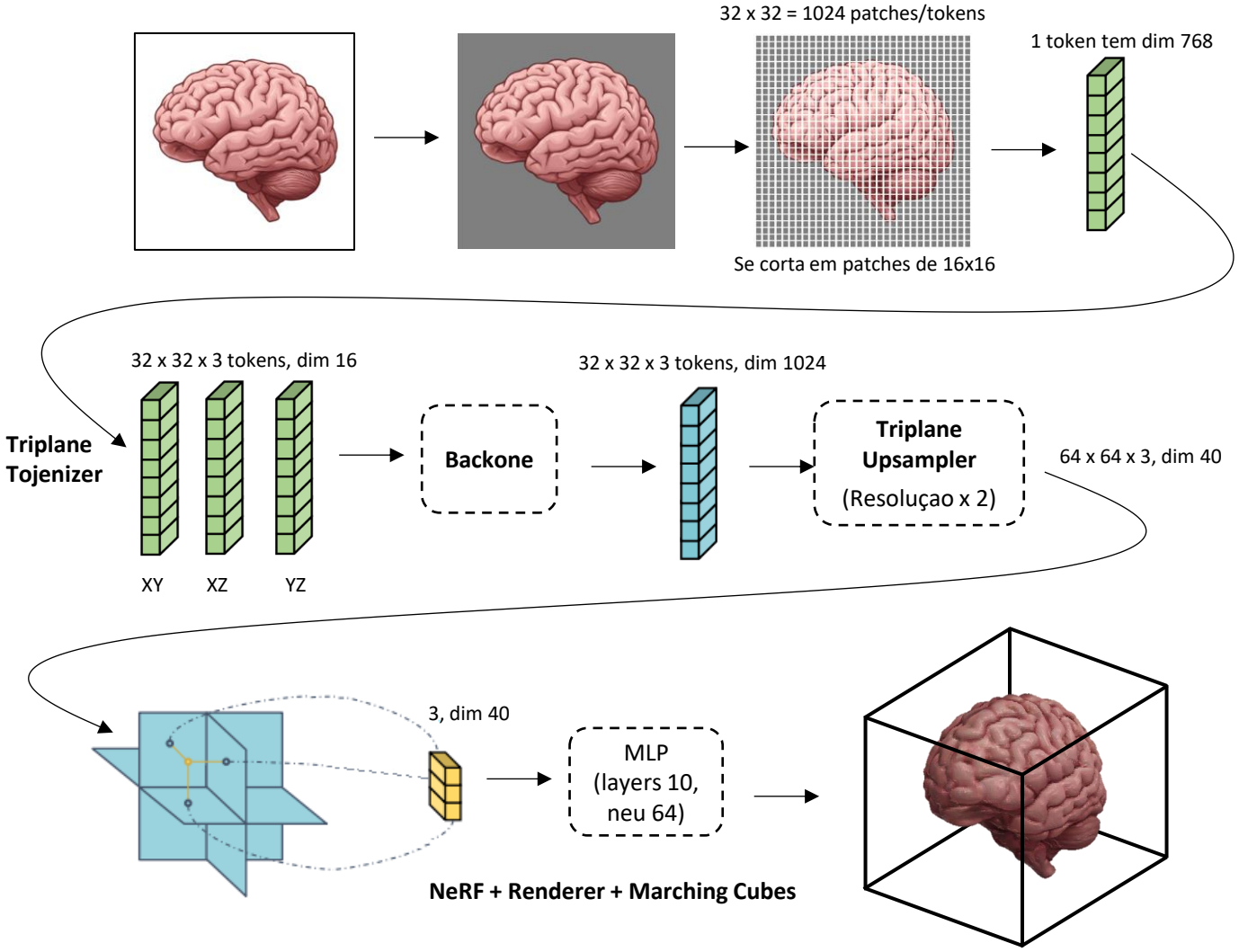
Vantagens e desvantagens

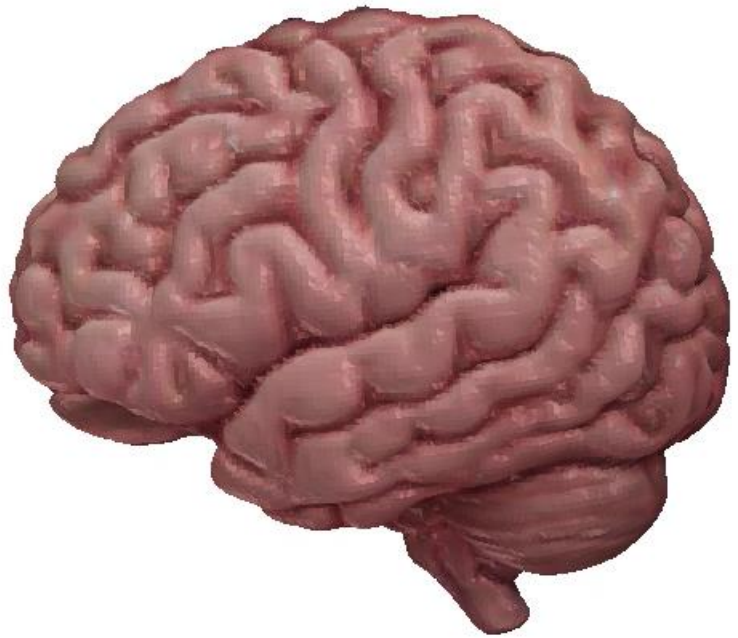
- Vantagens
  - **Velocidade:** Consegue gerar modelos 3D em menos de 0.5 segundos em uma GPU A100.
  - **Alto Desempenho:** O TripoSR supera outros modelos de código aberto tanto na qualidade (reconstruindo melhor a forma e a textura) quanto na velocidade de inferência.
  - **Acessibilidade:** É um modelo de código aberto (licença MIT) , o que permite que desenvolvedores e criativos o utilizem livremente.

Vantagens e desvantagens

- Desvantagens
  - **Qualidade da Imagem:** O resultado final é altamente dependente da qualidade da foto de entrada.
  - **Detalhes Finos:** Embora seja muito bom, o modelo pode ter dificuldades em reconstruir detalhes muito pequenos ou texturas complexas.
  - **Objetos Ocultos:** As partes do objeto que não aparecem na foto precisam ser "imaginadas" pelo modelo, o que pode gerar imprecisões
  - **Dependência de hardware:** para melhor desempenho é bom ter uma bom GPU
  - **Compatibilidade de versões:** O algoritmo depende de versões específicas de Python e bibliotecas

Exemplo(s) de aplicação



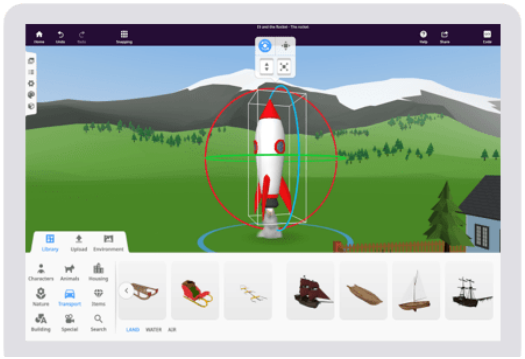
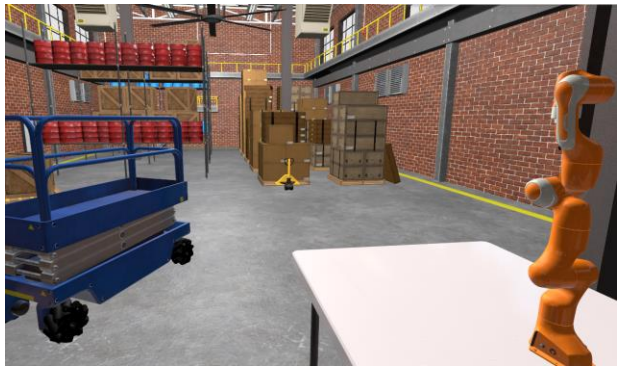


Exemplo(s) de aplicação

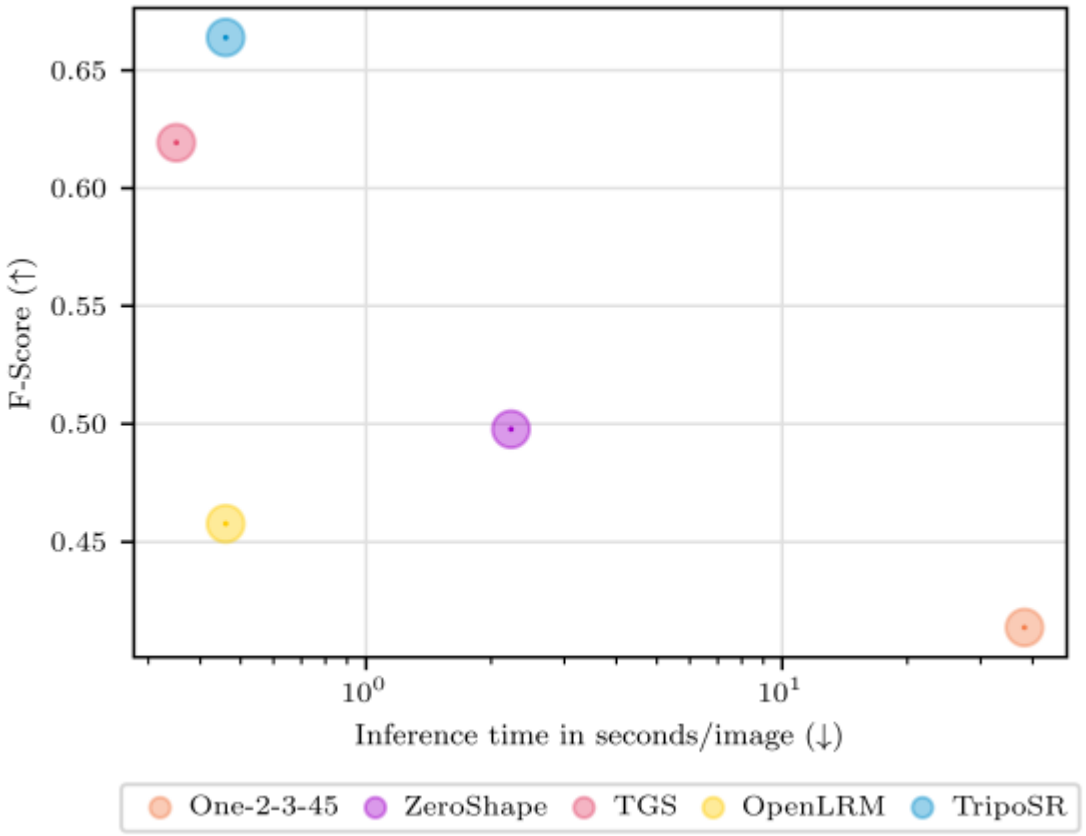
Exemplo(s) de aplicação

- **E-commerce**: geração automática de modelos 3D de produtos a partir de uma única foto.
- **Realidade aumentada e VR:** criação rápida de ativos digitais.
- **Jogos digitais:** prototipagem de objetos para ambientes virtuais.
- **Design industrial/arquitetura:** reconstrução de peças e maquetes.



Comparação com outros algoritmos



Comparação quantitativa de diferentes técnicas em GSO

Method	CD↓	FS@0.1↑	FS@0.2↑	FS@0.5↑
One-2-3-45 [16]	0.227	0.382	0.630	0.878
ZeroShape [13]	0.160	0.489	0.757	0.952
TGS [35]	0.122	0.637	0.846	0.968
OpenLRM [10]	0.180	0.430	0.698	0.938
TripoSR (ours)	**0.111**	**0.651**	**0.871**	**0.980**

Comparação quantitativa de diferentes técnicas no OmniObject3D

Method	CD↓	FS@0.1↑	FS@0.2↑	FS@0.5↑
One-2-3-45 [16]	0.197	0.445	0.698	0.907
ZeroShape [13]	0.144	0.507	0.786	0.968
TGS [35]	0.142	0.602	0.818	0.949
OpenLRM [10]	0.155	0.486	0.759	0.959
TripoSR (ours)	**0.102**	**0.677**	**0.890**	**0.986**

Comparação com outros algoritmos

- CD (Chamfer Distance): Um valor mais baixo indica maior precisão geométrica e uma forma mais próxima do objeto original.
  - TripoSR alcança a menor distância, provando ser o mais preciso geometricamente.
- FS (F-score): Uma pontuação mais alta indica um melhor desempenho geral na reconstrução da forma do objeto.
  - TripoSR tem a pontuação F mais alta de todas, o que o torna o modelo com o melhor desempenho

Comparação quantitativa de diferentes técnicas em GSO

Method	CD↓	FS@0.1↑	FS@0.2↑	FS@0.5↑
One-2-3-45 [16]	0.227	0.382	0.630	0.878
ZeroShape [13]	0.160	0.489	0.757	0.952
TGS [35]	0.122	0.637	0.846	0.968
OpenLRM [10]	0.180	0.430	0.698	0.938
TripoSR (ours)	**0.111**	**0.651**	**0.871**	**0.980**

Comparação quantitativa de diferentes técnicas no OmniObject3D

Method	CD↓	FS@0.1↑	FS@0.2↑	FS@0.5↑
One-2-3-45 [16]	0.197	0.445	0.698	0.907
ZeroShape [13]	0.144	0.507	0.786	0.968
TGS [35]	0.142	0.602	0.818	0.949
OpenLRM [10]	0.155	0.486	0.759	0.959
TripoSR (ours)	**0.102**	**0.677**	**0.890**	**0.986**

GSO e OmniObject3D são datasets públicos utilizados para validar e comparar o desempenho de modelos de reconstrução 3D

Comparação com outros algoritmos

Método	Qualidade da Forma	Qualidade da Textura	Funcionalidades
TripoSR	Captura melhor a estrutura geral do objeto e detalhes intrincados.	Gera malhas com textura diretamente.	Rápido, preciso, e se generaliza a diversos objetos.
ZeroShape	Tende a prever formas muito suavizadas.	Não gera malhas texturizadas diretamente.	Não é tão preciso e o tempo de inferência é similar.
TGS	Reconstrói mais detalhes, mas nem sempre se alinham com a imagem de entrada.	Não gera malhas texturizadas diretamente.	Mais rápido que TripoSR, mas menos preciso.
One-2-3-45	As formas são frequentemente imprecisas.	Gera malhas texturizadas diretamente.	Muito mais lento que os outros modelos.
OpenLRM	As formas são frequentemente imprecisas.	Gera malhas texturizadas diretamente.	O desempenho é inferior ao de TripoSR.

Comparação com outros algoritmos

Perguntas?

Referências

- Dmitry Tochilkin, et al., “TripoSR: Fast 3D Object Reconstruction from a Single Image”, https://arxiv.org/pdf/2403.02151.
- Repositório do GitHub: https://github.com/VAST-AI-Research/TripoSR
- Mohamed Debbagh, “Neural Radiance Fields (NeRFs): A Review and Some Recent Developments”, https://arxiv.org/abs/2305.00375.
- Yicong Hong, et al.,“LRM: LARGE RECONSTRUCTION MODEL FOR SINGLE IMAGE TO 3D”, https://arxiv.org/pdf/2311.04400.
- Alexey Dosovitskiy, et al., “AN IMAGE IS WORTH 16X16 WORDS: TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE”, https://arxiv.org/pdf/2010.11929.

Links

- GitHub: https://github.com/Mish0404/TP558/tree/main/Semin%C3%A1rio_TripoSR_Fast_3D_Object_Reconstruction_from_a_Single_Image
  - Para reproduzir o exemplo, consulte o repositório GitHub do seminário. Os códigos são *“prueba_TripoSR.py”*, *“prueba_TripoSR_app.py”*, *“prueba_TripoSR_Notebooks_Jupyter.ipynb”*.
- Quiz: TripoSR

Obrigado!