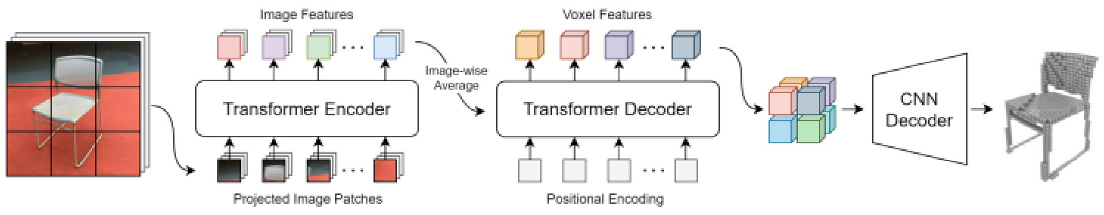
From Multi-Image 2D to 3D Conversion Analysis

1-3D-RETR:

is capable of performing end-to-end single and multi-view 3D Reconstruction with TRansformers.

- ### Architecture:



It consists of :

1. Transformer Encoder
2. Transformer Decoder
3. CNN Decoder

- ### Inputs & Outputs

  3D-RETR first uses a pre-trained Transformer to extract **visual features** from 2D input images. 3D-RETR then uses another Transformer Decoder to obtain **the voxel features**. A CNN Decoder then takes as input the voxel features to obtain the **reconstructed objects**.

  Detailed Explanation:

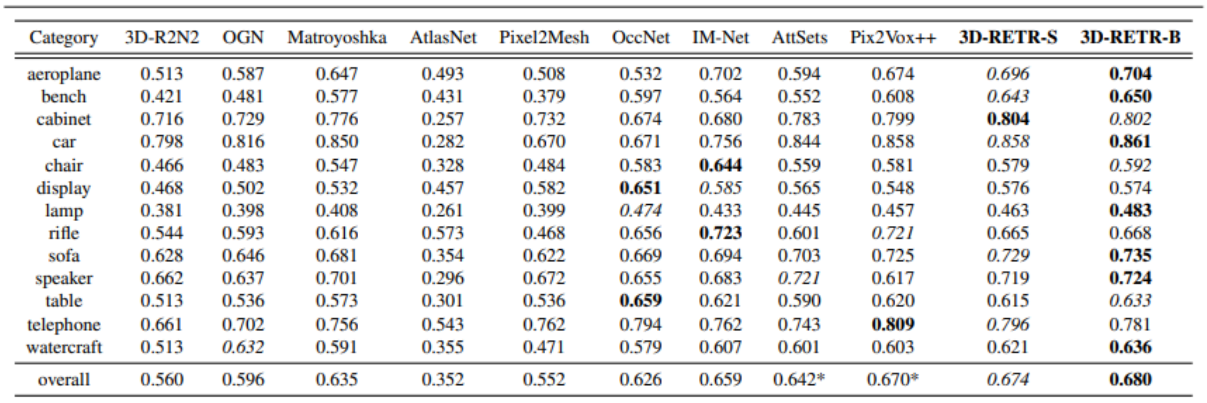
  A Vision Transformer takes as input image xi by splitting the image into patches. At each time step, the corresponding patch is embedded by first linearly transformed into a fixed-size vector, which is then added with positional embeddings. The Transformer takes the embedded patch feature as input and outputs encoded dense image feature vectors. For single-view reconstruction, we keep all the dense image vectors. For multi-view reconstruction, at each time step, we take the average across different views, and keep the averaged dense vectors. We feed all dense image vectors in the next stage into the Transformer Decoder. To enable the Transformer Decoder to understand the spatial relations between voxel features, we create M3 positional embeddings for the Transformer Decoder. The positional embeddings are learnable and are updated during training. The CNN Decoder takes as input the voxel features from the Transformer Decoder and outputs the voxel representation.

- **Model Strengths**:
    1. Experimental results on two datasets show that 3D-RETR reaches state-of-the-art performance on 3D reconstruction. Additional ablation study also demonstrates that 3D-RETR benefits from using Transformers.
    2. It's not vulnerable to rapid changes between views.
    3. It uses much fewer parameters than the previous models and doesn't rely on sophisticated pipelines of different convolutional neural networks.
    4. It's not struggling to output reasonable results when fewer images are available.
    5. It aims to reconstruct the volume without rendering the 2D images.

- **Model Evaluation:**

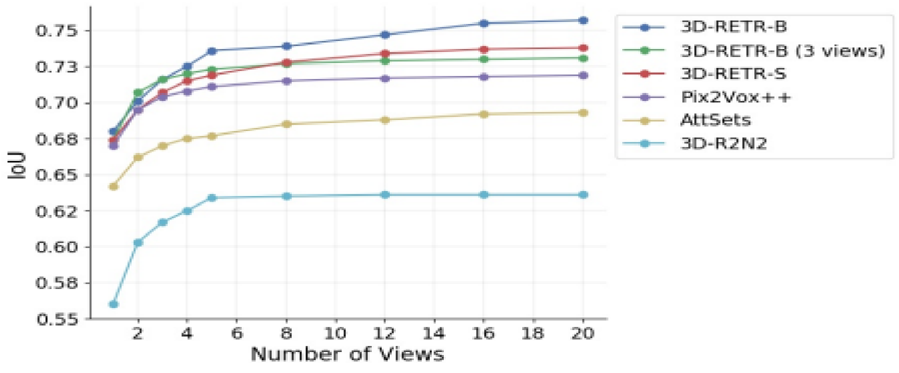
Compared to other Models:

Intersection of Union (IoU) as the evaluation metric.



Category	3D-R2N2	OGN	Matroyoshka	AtlasNet	Pixel2Mesh	OccNet	IM-Net	AttSets	Pix2Vox++	**3D-RETR-S**	**3D-RETR-B**
aeroplane	0.513	0.587	0.647	0.493	0.508	0.532	0.702	0.594	0.674	*0.696*	**0.704**
bench	0.421	0.481	0.577	0.431	0.379	0.597	0.564	0.552	0.608	*0.643*	**0.650**
cabinet	0.716	0.729	0.776	0.257	0.732	0.674	0.680	0.783	0.799	**0.804**	*0.802*
car	0.798	0.816	0.850	0.282	0.670	0.671	0.756	0.844	0.858	*0.858*	**0.861**
chair	0.466	0.483	0.547	0.328	0.484	0.583	**0.644**	0.559	0.581	0.579	*0.592*
display	0.468	0.502	0.532	0.457	0.582	**0.651**	*0.585*	0.565	0.548	0.576	0.574
lamp	0.381	0.398	0.408	0.261	0.399	*0.474*	0.433	0.445	0.457	0.463	**0.483**
rifle	0.544	0.593	0.616	0.573	0.468	0.656	**0.723**	0.601	*0.721*	0.665	0.668
sofa	0.628	0.646	0.681	0.354	0.622	0.669	0.694	0.703	0.725	*0.729*	**0.735**
speaker	0.662	0.637	0.701	0.296	0.672	0.655	0.683	*0.721*	0.617	0.719	**0.724**
table	0.513	0.536	0.573	0.301	0.536	**0.659**	0.621	0.590	0.620	0.615	*0.633*
telephone	0.661	0.702	0.756	0.543	0.762	0.794	0.762	0.743	**0.809**	*0.796*	0.781
watercraft	0.513	*0.632*	0.591	0.355	0.471	0.579	0.607	0.601	0.603	0.621	**0.636**
overall	0.560	0.596	0.635	0.352	0.552	0.626	0.659	0.642*	0.670*	*0.674*	**0.680**

Results of single-view 3D reconstruction on the ShapeNet dataset. Bold is the best performance, while Italic is the second best.



- **Loss Function**

  uses Dice loss as the loss function, which is suitable for 3D reconstruction as the voxel occupancy is highly unbalanced. Formally, we have the Dice loss for the 3D voxels as follows:

$$L_{Dice} = 1 - \frac{\sum_{n=1}^{N^3} p_n y_n}{\sum_{n=1}^{N^3} p_n + y_n} - \frac{\sum_{n=1}^{N^3}(1-p_n)(1-y_n)}{\sum_{n=1}^{N^3} 2 - p_n - y_n}$$

  where p_n is the n-th predicted probability of the voxel occupation and y_n is the n-th groundtruth voxel.

- **Optimizer**

  AdamW as the optimizer with a learning rate of $1e-4$, $\beta1 = 0.9$, $\beta2 = 0.999$, and a weight decay of $1e-2$. The batch size is set to 16 for all the experiments.

- **Problem it aims to solve:**

  Reaches state-of-the-art performance on 3D reconstruction under both synthetic and real-world settings. 3D-RETR is more efficient than previous models as 3D-RETR reaches better performance with much fewer parameters.

- **Challenges that make it difficult to use 3D-RETR**

  1. **Lack of Pretrained Weights:**
     One of the challenges faced with 3D-RETR is the absence of pretrained weights. This means that the model does not come with pre-learned parameters, which are crucial for effective and efficient use in tasks such as transfer learning or fine-tuning.

  2. **Unmet Dataset Requirements:**
     Another obstacle is the specific dataset prerequisites that are essential for training or inference. If these datasets are not available or accessible, it becomes challenging to utilize the model effectively due to incompatible or missing data.

  3. **Unclear Training Process:**
     The training methodology for 3D-RETR may not be well-defined or documented. This ambiguity can impede the ability to properly train the model, affecting performance and adaptability.

  4. **Limited Dataset Compatibility:**
     3D-RETR might be designed to work only with particular datasets. This restriction limits its applicability to broader use cases and restricts users to specific datasets, potentially limiting flexibility.

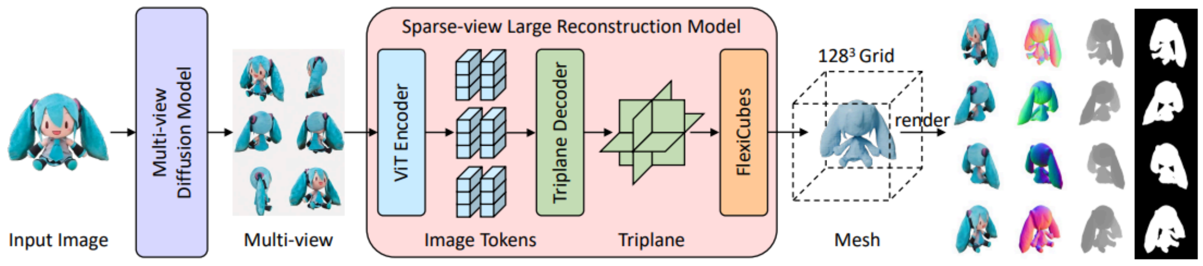
  5. **Outdated Requirements and Dependencies:**
     The model's requirements may not be updated to align with the latest software libraries or dependencies. This outdated configuration makes it challenging to integrate and use the model in modern environments, requiring additional effort to resolve compatibility issues.

By addressing these challenges through improved documentation, providing pretrained weights, ensuring dataset flexibility, clarifying the training process, and updating software dependencies, the usability and accessibility of 3D-RETR can be significantly enhanced.

2- InstantMesh

Efficient 3D Mesh Generation from a Single Image with Sparse-view Large Reconstruction Models. a feed-forward framework for high-quality 3D mesh generation from a single image.

- **Architecture**



It consists of

1. Multi-view Diffusion Model

   Zero123++ is used due to its reliable multi-view consistency and tailored viewpoint distribution that covers both the upper and lower parts of a 3D object.

2. Sparse-view Large Reconstruction Model (Transformer-based)

   The model is initialized using the pre-trained weights of OpenLRM, an open-source implementation of LRM

   - ☐ Vit Encoder
   - ☐ Triplane Decoder
   - ☐ iso-surface extraction module (i.e., FlexiCubes)

Improvements on model architecture

To address inconsistencies in background colour and improve reconstruction stability, fine-tuning Zero123++ is employed to generate uniform white-background images. Originally, Zero123++ produces 960x640 grey-background images displaying 6 multi-view images in a 3 × 2 grid. However, variations in background colour and RGB values across different regions introduce artifacts like floaters and cloud-like formations in reconstructions. While LRMs typically train on white-background images, using third-party tools for background removal risks segmentation inconsistencies between views. Therefore, fine-tuning Zero123++ ensures consistent white backgrounds, enhancing stability for subsequent sparse-view reconstruction.

- **Inputs & Outputs**

  Given an input image, we first utilize a multi-view diffusion model to synthesize **6 novel views at fixed camera poses**. Then we feed the generated multi-view images into a transformer-based sparse-view large reconstruction model to reconstruct a **high-quality 3D mesh**.

- **Model Strengths**
  1. Zero123 fine-tunes Stable Diffusion to generate new views based on camera poses, improving 3D consistency and enabling realistic shape generation from diverse images, as demonstrated by recent image-to-3D methods.
  2. InstantMesh generates 3D consistent multi-view images swiftly using a multi-view diffusion model, providing diverse perspectives of the input image.
  3. The model predicts a 3D mesh directly from the generated multi-view images using a sparse-view large reconstruction model, achieving rapid processing within seconds.
  4. Utilizing a differentiable iso-surface extraction module, InstantMesh applies geometric supervisions directly on the mesh surface, ensuring efficient training and high-quality mesh generation.
  5. Built upon an LRM-based architecture, InstantMesh offers superior training scalability suitable for large-scale datasets, facilitating robust model performance.
  6. Experimental results demonstrate that InstantMesh surpasses the performance of other state-of-the-art image-to-3D approaches significantly, showcasing its effectiveness and potential in 3D generative AI.

- ## Training Process

Stage 1: Training on NeRF

- Model Setup: Train using the triplane NeRF representation, leveraging prior knowledge from pre-trained OpenLRM.
- Model Modifications:
  - Add AdaLN (Adaptive Layer Normalization) camera pose modulation layers to the ViT image encoder for pose-aware output image tokens, inspired by Instant3D.
  - Remove source camera modulation layers from the triplane decoder of LRM.
- **Loss Function:**
  - Use combined image loss (L1 and LPIPS) and mask loss

$$L1 = \sum_i \|\hat{I}_i - I_{gt_i}\|_2^2 + \lambda_{lpips} \sum_i L_{lpips}(\hat{I}_i, I_{gt_i}) + \lambda_{mask} \sum_i \|\hat{M}_i - M_{gt_i}\|_2^2$$

where $\hat{I}_i, I_{gt_i}, \hat{M}_i$, and M_{gt_i} are rendered images, ground truth images, rendered masks, and ground truth masks for the i-th view.

- **Training Parameters:**
  - Set $\lambda lpips = 0.2, \lambda mask = 1.0$.
  - Use a cosine-annealed learning rate schedule from 4.0×10^{-4} to 4.0×10^{-5}.
  - Render 192 × 192 patches supervised by cropped ground truth patches ranging from 192 × 192 to 512 × 512 for high-resolution training.

Stage 2: Training on Mesh

- Model Transition: Switch to training with mesh representation for efficiency and additional geometric supervisions.
- Integration of FlexiCubes:
  - Integrate FlexiCubes into the reconstruction model to extract mesh surfaces from triplane implicit fields.
- Model Adaptation:
  - Reuse the density MLP from the original triplane NeRF renderer to predict Signed Distance Function (SDF).
- Add two additional MLPs to predict deformation and weights required by FlexiCubes.

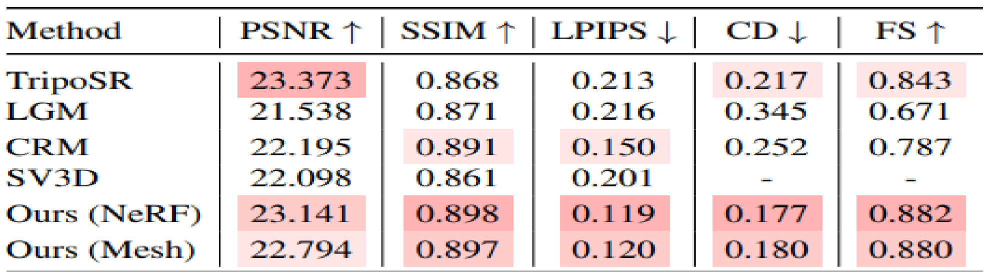
- ## Model Evaluation

Metrics Used for Evaluation:

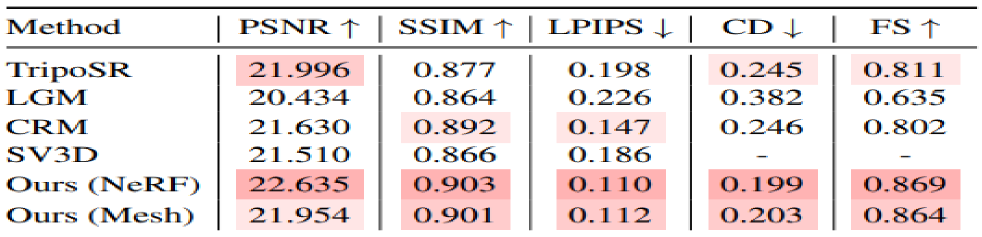
1. **2D Visual Quality Metrics:**
   - **PSNR** (Peak Signal-to-Noise Ratio): Measures fidelity of rendered novel views compared to ground truth views.
   - **SSIM** (Structural Similarity Index): Evaluates similarity in structure between rendered and ground truth views.
   - **LPIPS** (Learned Perceptual Image Patch Similarity): Assesses perceptual similarity between generated and ground truth views.
2. **3D Geometric Quality Metrics:**
   - **Chamfer Distance (CD):** Measures average distance between points on the generated mesh surface and ground truth mesh surface.
   - **F-Score (FS):** Computes harmonic mean of precision and recall based on surface points sampled uniformly.

Main Quantitative Results:

- Comparison Across Evaluation Sets (Tables 1 and 2):
  - Highlight top three results per metric, with deeper colours indicating superior performance.
  - InstantMesh variants (using "NeRF" and "Mesh" sparse-view reconstruction models) are compared against baselines.



Method	PSNR ↑	SSIM ↑	LPIPS ↓	CD ↓	FS ↑
TripoSR	23.373	0.868	0.213	0.217	0.843
LGM	21.538	0.871	0.216	0.345	0.671
CRM	22.195	0.891	0.150	0.252	0.787
SV3D	22.098	0.861	0.201	-	-
Ours (NeRF)	23.141	0.898	0.119	0.177	0.882
Ours (Mesh)	22.794	0.897	0.120	0.180	0.880



Method	PSNR ↑	SSIM ↑	LPIPS ↓	CD ↓	FS ↑
TripoSR	21.996	0.877	0.198	0.245	0.811
LGM	20.434	0.864	0.226	0.382	0.635
CRM	21.630	0.892	0.147	0.246	0.802
SV3D	21.510	0.866	0.186	-	-
Ours (NeRF)	22.635	0.903	0.110	0.199	0.869
Ours (Mesh)	21.954	0.901	0.112	0.203	0.864

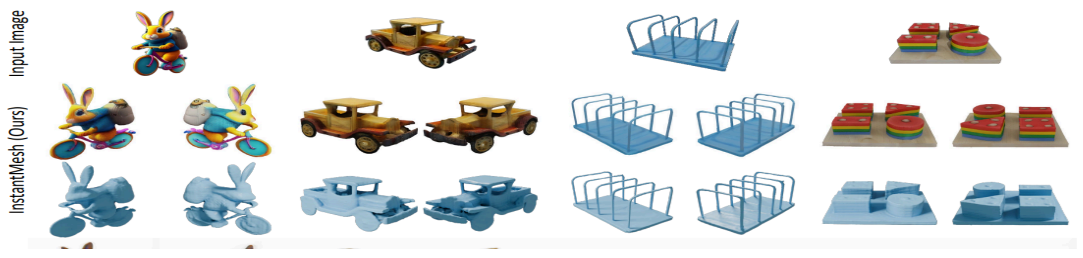
Observations from Evaluation Metrics:

1. **2D Novel View Synthesis:**
    - InstantMesh excels in SSIM and LPIPS metrics, indicating superior perceptual viewing quality compared to baselines.
    - Although InstantMesh has slightly lower PSNR, its perceptual quality is emphasized as novel views are "dreamed" by the multi-view diffusion model.
2. **3D Geometric Evaluation:**
    - InstantMesh outperforms baselines significantly in both CD and FS, indicating higher fidelity in generated shapes.
    - InstantMesh presents more reliable geometries based on qualitative observations.

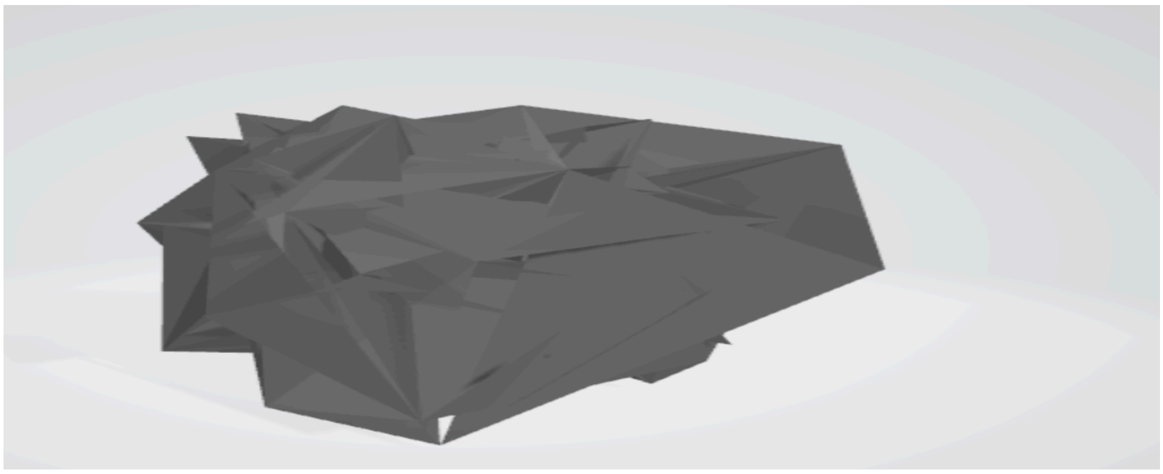
Qualitative Comparison Results:

- **Visualization of Image-to-3D Generation:**
    - InstantMesh demonstrates significantly more plausible geometry and appearance compared to baselines.
    - TripoSR performs well on style-consistent images but lacks imagination ability for free-style images.
    - InstantMesh achieves sharper textures and reliable geometry due to high-resolution supervision.
    - Other methods like LGM and CRM show similar frameworks but exhibit distortions, multi-view inconsistency, or challenges in generating smooth surfaces.

- **Sample Output**



- **Work Challenges**
    1. Transitioning the model to a multi-image version posed significant challenges, resulting in inferior performance compared to the single-image version (the resulting 3D output quality is unsatisfactory compared to the single-image version).



    2. Attempting to use Pix2Vox++ for 3D reconstruction, but encountering limitations due to required libraries not being open source.

- **Leveraging InstantMesh Output (.obj Format) for Web, Blender, and Unity Integration**

The output of the InstantMesh model, which is a 3D object in .obj format, is inherently compatible with integration into websites, Blender, and Unity due to the universal support for the .obj file format in these platforms.

- Integration with Websites:
    - Upload the .obj file along with any associated texture files (e.g., .mtl file and texture images) to your web server.
    - Use WebGL libraries like Three.js to load the .obj model in a web page. Three.js provides functions to load and display 3D models in the browser using WebGL rendering.
- Importing into Blender:
    - Simply import the .obj file directly into Blender by selecting File > Import > Wavefront (.obj).
    - Blender will import the 3D model with its geometry and material information intact, allowing you to further edit, animate, or render the model within Blender.
- Using with Unity:
    - Drag and drop the .obj file into your Unity project's assets folder.