

14th, March, 2025

Scene Coordinate Reconstruction

Tanks & Temples – Truck

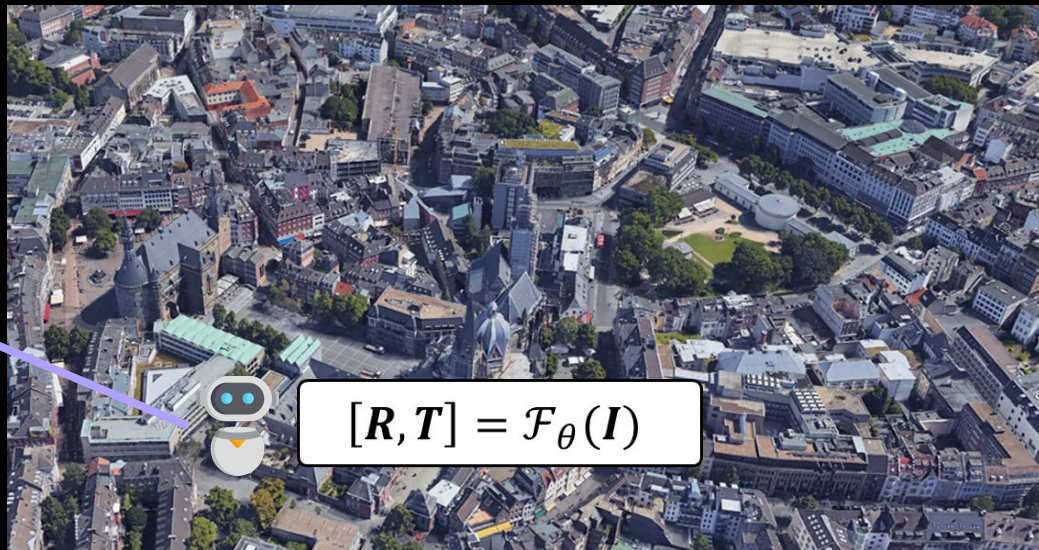
Tanks & Temples – Caterpillar

Presenter
Sihang Li

Background



Query Image

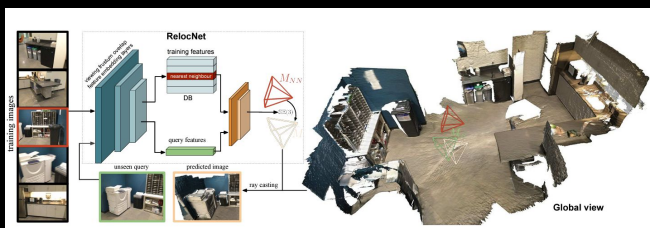
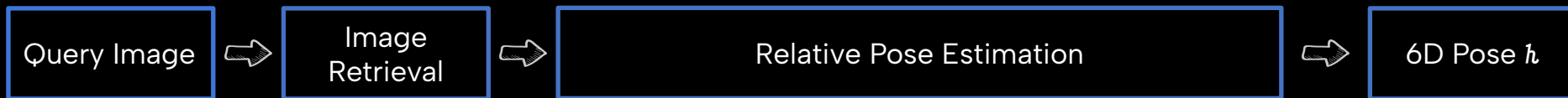


$$[R, T] = \mathcal{F}_\theta(I)$$

Where am I?

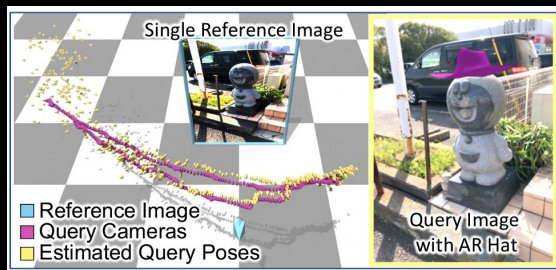
[Rotation, Translation]

Background



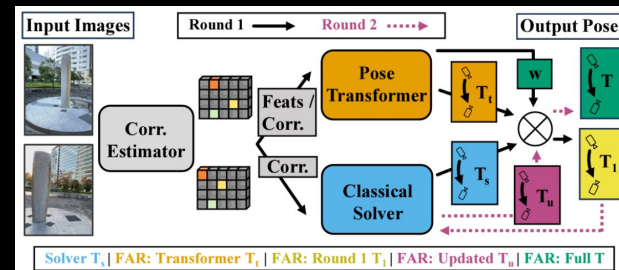
RelocNet: Continuous Metric Learning Relocalisation using Neural Nets, Balntas et al, ECCV18

- ✓ Low or no mapping costs
- ✗ Maintain Large Database



Map-free Visual Relocalization: Metric Pose Relative to a Single Image, Eduardo et al, ECCV22

- ✓ Generalizable to different scenes
- ✗ Scale ambiguity is key challenge



FAR: Flexible, Accurate and Robust 6DoF Relative Camera Pose Estimation, Chris et al, CVPR24

Background

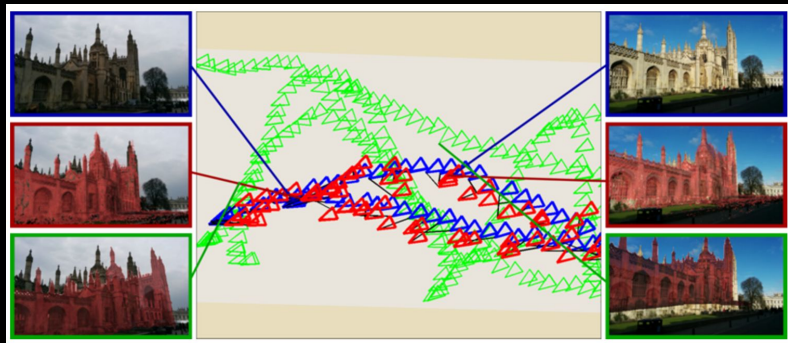
Query Image



Absolute Pose Estimation



6D Pose h



PoseNet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization, Alex et al, ICCV15



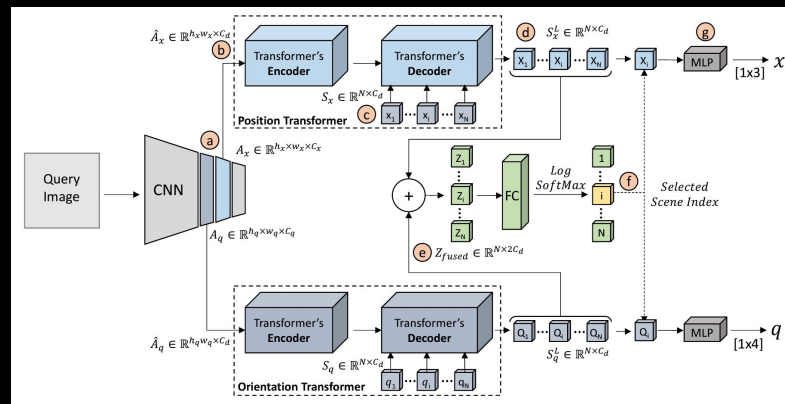
Fast Inference Speed



Scenes with different scales

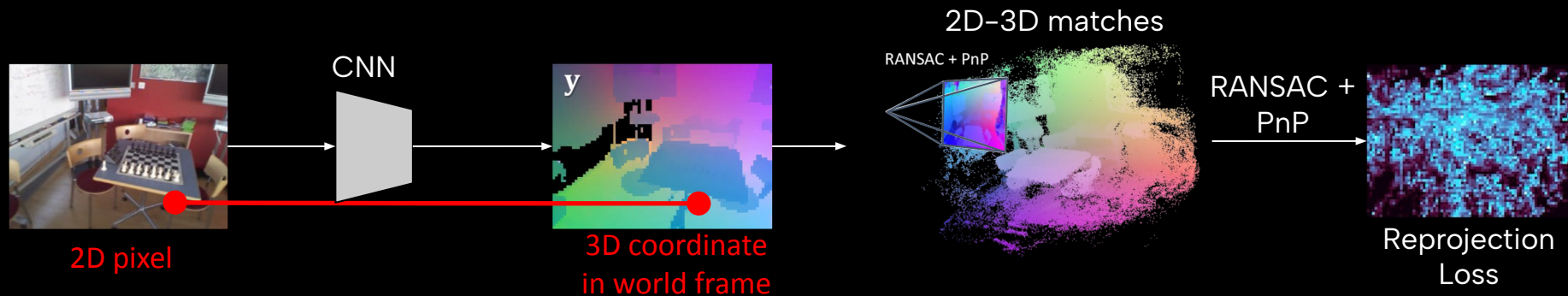


Long Training Time



Ms-Transformer, Yoli et al, ICCV21

Scene Coordinate Regression



✓ High accuracy

✓ Efficiency

Accelerated Coordinate Encoding:

Learning to Relocalize in Minutes using RGB and Poses

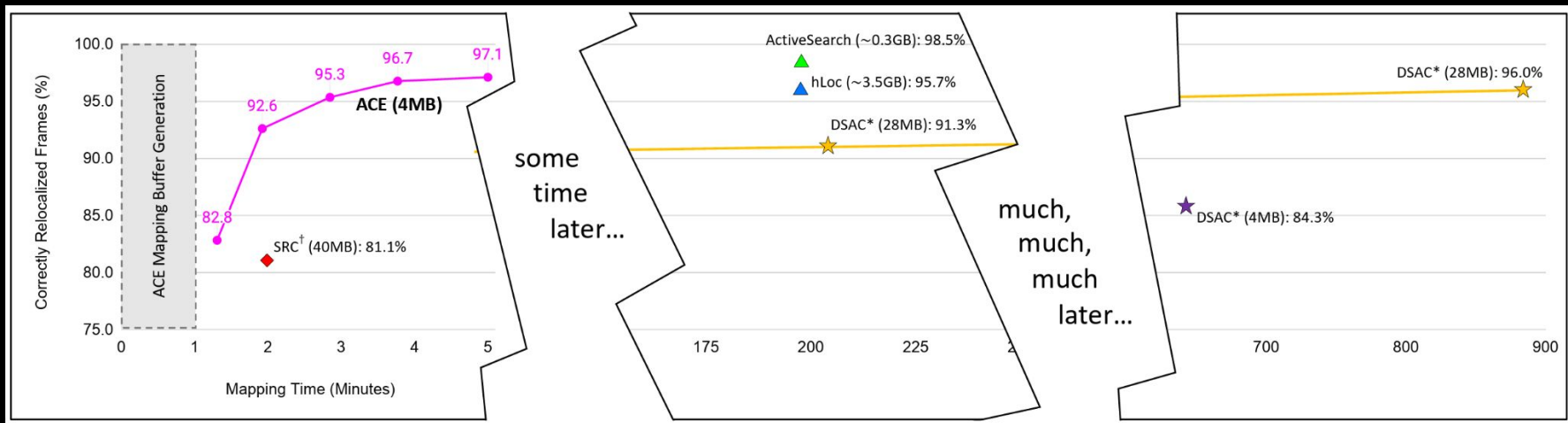
CVPR 2023 (Highlight)

Eric Brachmann¹ Tommaso Cavallari¹ Victor Adrian Prisacariu^{1,2}

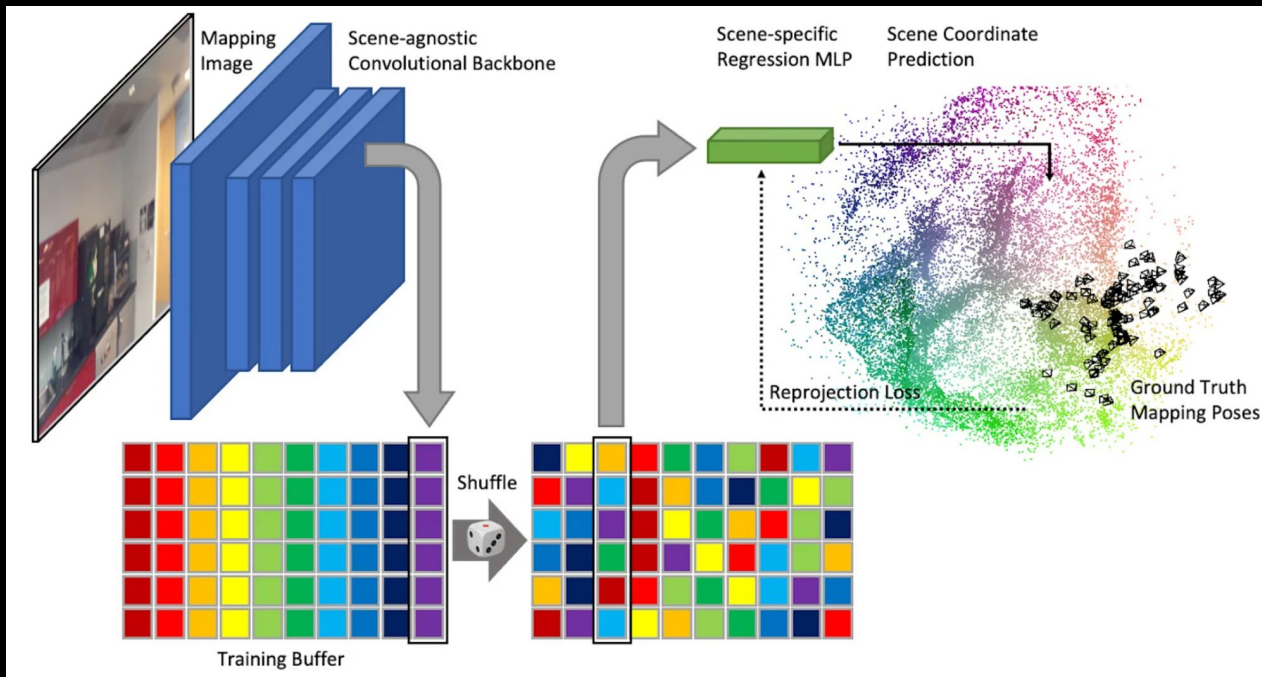
Niantic University of Oxford



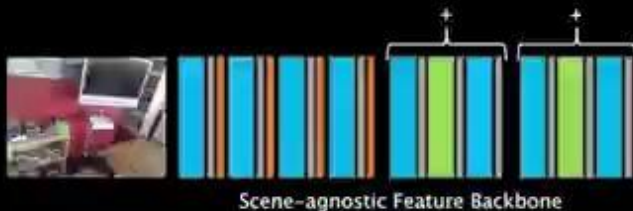
- A scene coordinate regression system that maps a new scene in **5** minutes.
- ACE compiles a scene into **4**MB worth of network weights. Previous scene coordinate regression systems required 7-times more storage.



- Scene-agnostic backbone + Scene-specific MLP



- Scene-agnostic backbone + Scene-specific MLP



Data Preparation (1 Minute)

- Experiment: indoor dataset

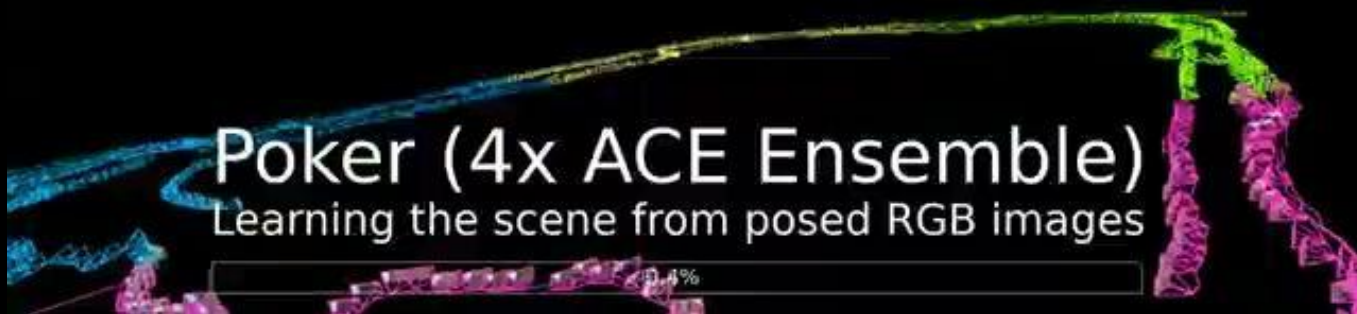
		Mapping w/ Mesh/Depth	Mapping Time	Map Size	7 Scenes		12 Scenes	
					SfM poses	D-SLAM poses	SfM poses	D-SLAM poses
FM	AS (SIFT) [52]	No	~1.5h	~200MB	98.5%	68.7%	99.8%	99.6%
	D.VLAD+R2D2 [26]	No		~1GB	95.7%	77.6%	99.9%	99.7%
	hLoc (SP+SG) [47, 48]	No		~2GB	95.7%	76.8%	100%	99.8%
	pixLoc [49]	No		~1GB	N/A	75.7%	N/A	N/A
SCR (w/ Depth)	DSAC* (Full) [11]	Yes	15h	28MB	98.2%	84.0%	99.8%	99.2%
	DSAC* (Tiny) [11]	Yes	11h	4MB	85.6%	70.0%	84.4%	83.1%
	SANet [67]	Yes	~2.3 min	~550MB	N/A	68.2%	N/A	N/A
	SRC [21]	Yes	2 min [†]	40MB	81.1%	55.2%	N/A	N/A
SCR	DSAC* (Full) [11]	No	15h	28MB	96.0%	81.1%	99.6%	98.8%
	DSAC* (Tiny) [11]	No	11h	4MB	84.3%	69.1%	81.9%	81.6%
	ACE (ours)	No	5 min	4MB	97.1%	<u>80.8%</u>	99.9%	99.6%

- Experiment: Outdoor dataset

		Mapping w/ Mesh/Depth	Mapping Time	Map Size	Cambridge Landmarks					Average (cm / °)
					Court	King's	Hospital	Shop	St. Mary's	
FM	AS (SIFT) [52]	No	~35min	~200MB	24/0.1	13/0.2	20/0.4	4/0.2	8/0.3	14/0.2
	hLoc (SP+SG) [47, 48]	No		~800MB	16/0.1	12/0.2	15/0.3	4/0.2	7/0.2	11/0.2
	pixLoc [49]	No		~600MB	30/0.1	14/0.2	16/0.3	5/0.2	10/0.3	15/0.2
	GoMatch [70]	No		~12MB	N/A	25/0.6	283/8.1	48/4.8	335/9.9	N/A
	HybridSC [13]	No		~1MB	N/A	81/0.6	75/1.0	19/0.5	50/0.5	N/A
APR	PoseNet17 [30]	No	4 – 24h	50MB	683/3.5	88/1.0	320/3.3	88/3.8	157/3.3	267/3.0
	MS-Transformer [55]	No	~7h	~18MB	N/A	83/1.5	181/2.4	86/3.1	162/4.0	N/A
SCR w/ Depth	DSAC* (Full) [11]	Yes	15h	28MB	49/0.3	15/0.3	21/0.4	5/0.3	13/0.4	21/0.3
	SANet [67]	Yes	~1min	~260MB	328/2.0	32/0.5	32/0.5	10/0.5	16/0.6	84/0.8
	SRC [21]	Yes	2 min [†]	40MB	81/0.5	39/0.7	38/0.5	19/1.0	31/1.0	42/0.7
SCR	DSAC* (Full) [11]	No	15h	28MB	34/0.2	18/0.3	21/0.4	5/0.3	15/0.6	19/0.4
	DSAC* (Tiny) [11]	No	11h	4MB	98/0.5	27/0.4	33/0.6	11/0.5	56/1.8	45/0.8
	ACE (ours)	No	5 min	4MB	43/0.2	28/0.4	31/0.6	5/0.3	18/0.6	25/0.4
	Poker (Quad ACE Ensemble)	No	20 min	16MB	28/0.1	18/0.3	25/0.5	5/0.3	9/0.3	17/0.3

- Experiment: Outdoor dataset → Large-scale

>50px ACE 1 Repro. Error 0px >50px ACE 2 Repro. Error 0px >50px ACE 3 Repro. Error 0px >50px ACE 4 Repro. Error 0px





- Comparison with the previous work



Scene Coordinate Reconstruction

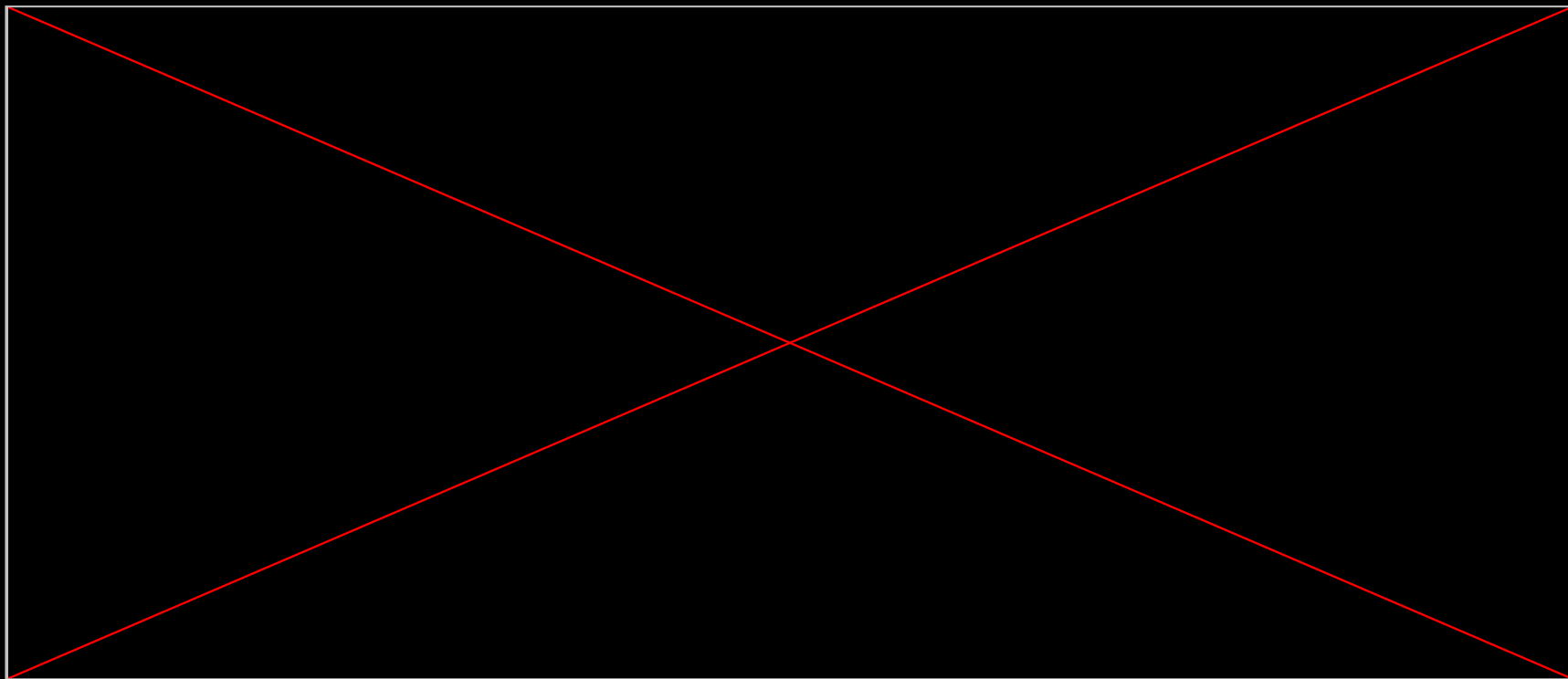
Posing of Image Collections via Incremental Learning of a Relocalizer

ECCV 2024 (Oral)

Eric Brachmann Jamie Wynn Shuai Chen Tommaso Cavallari Áron Monszpart
Daniyar Turmukhambetov Victor Adrian Prisacariu

Niantic University of Oxford

ACE Zero



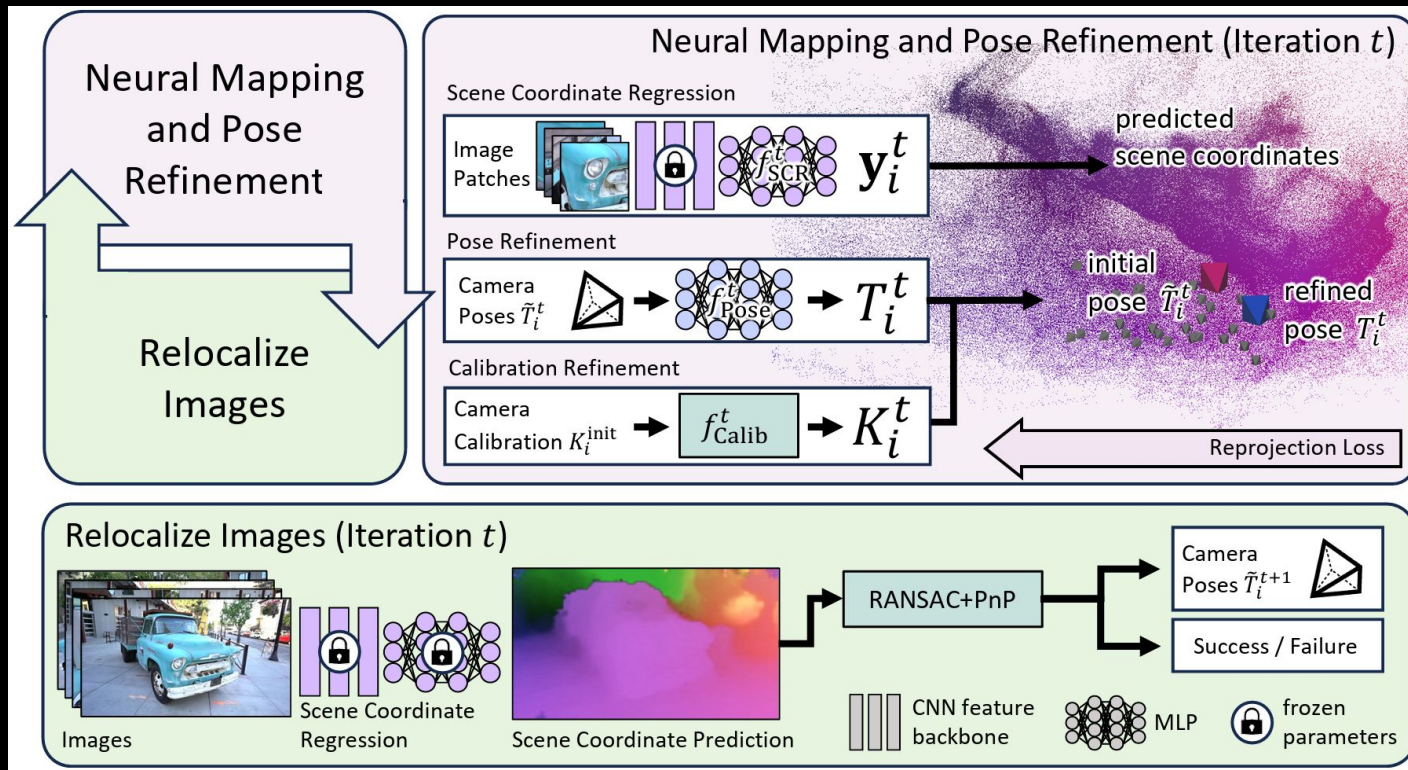
ACE Zero

- A structure from motion framework for 3D reconstruction
- Incremental SfM can be re-interpreted as a **loop** of:
 - do visual relocalization to register new views to the reconstruction,
 - and refine/extend the reconstruction using the newly registered views.
- What do we need:
 - a relocalizer with high accuracy and good generalization.
 - Training of the relocalizer has to be swift.
 - We need to be able to bootstrap relocalization without ground truth poses.
 - we need to be able to tell whether registration of a new image was successful.

ACE Zero

- Scene coordinate reconstruction:
 - Rather than optimizing over image-to-image matches, like feature-based SfM, scene coordinate reconstruction regresses image-to-scene correspondences directly.
 - Rather than representing the scene as a 3D point cloud with high dimensional descriptors, we encode the scene into a lightweight neural network.

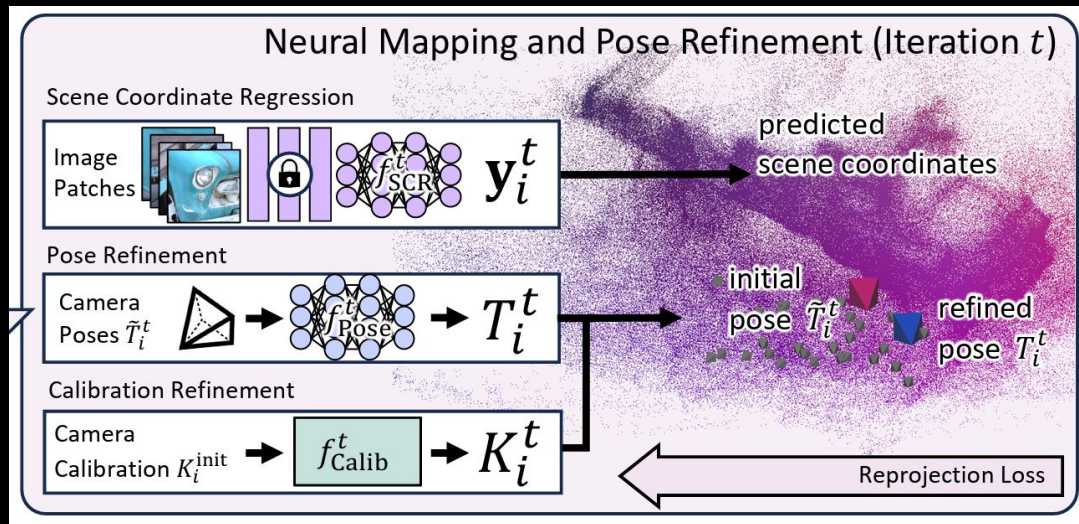
ACE Zero



ACE Zero

- Neural mapping
 - train the scene model by minimizing the pixel-wise reprojection error:

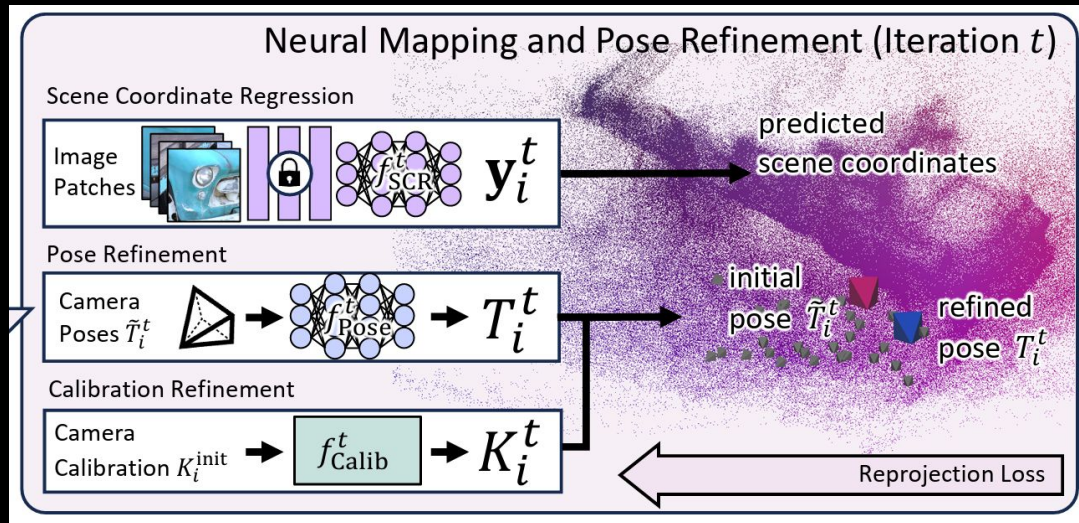
$$\sum_{I_i \in \mathcal{I}_{\text{Reg}}^t} \sum_{j \in I_i} \|\mathbf{p}_{ij} - \pi(K_i^t, T_i^t, \mathbf{y}_{ij}^t)\|$$



ACE Zero

- Neural mapping
 - Pose refinement: jointly optimize the Scene coordinate MLP and pose refine MLP by the reprojection loss

$$T_i^t = f_{\text{Pose}}^t(\tilde{T}_i^t)$$

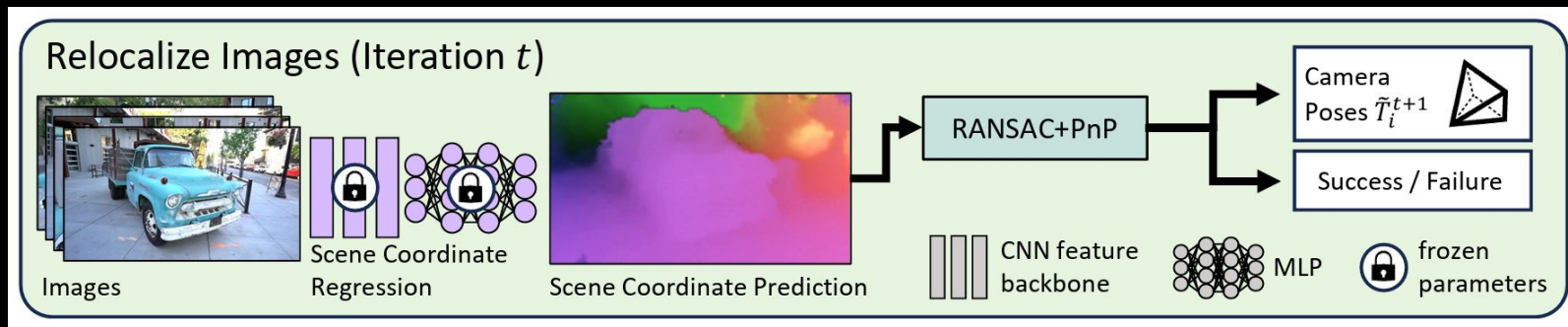


$$\sum_{I_i \in \mathcal{I}_{\text{Reg}}^t} \sum_{j \in I_i} \|\mathbf{p}_{ij} - \pi(K_i^t, T_i^t, \mathbf{y}_{ij}^t)\|$$

ACE Zero

- Relocalization
 - We pass all images to the scene coordinate regressor to gather 2D–3D correspondences, and solve for their poses using RANSAC and PnP.

$$\tilde{T}_i^{t+1}, s_i^{t+1} = g(K_i^t, \{(p_{ij}, y_{ij}^t)\})$$



ACE Zero

- Initialization
 - One image + depth (Zoe-depth):

$$\hat{\mathbf{y}}_{ij} = d_{ij} (K_{\text{seed}}^{\text{init}})^{-1} \mathbf{p}_{ij}$$

$$\sum_{j \in I^0} \|\hat{\mathbf{y}}_{ij} - \mathbf{y}_{ij}\|$$

ACE Zero

- Evaluation: Novel-view Synthesis (NVS) → image quality
 - Train a NeRF model based on the estimated poses

	Frames	Pseudo Ground Truth		All Frames				200 Frames			50 Frames	
		Kinect Fusion	COLMAP (default)	COLMAP (fast)	DROID-SLAM [†] [90]	ACE0 (ours)	KF+ACE0 (ours)	BARF [54]	NoPE-NeRF [†] [10]	ACE0 (ours)	DUST3R [97]	ACE0 (Ours)
Chess	6k	19.6	23.6	23.5	19.3	23.3	23.0	12.8	12.6	22.7	18.9	19.2
Fire	4k	19.2	22.6	22.6	13.0	22.3	22.3	12.7	11.8	22.1	18.8	19.5
Heads	2k	17.0	18.8	18.9	17.6	18.8	19.1	10.7	11.8	19.9	18.4	21.3
Office	10k	18.9	21.4	21.6	failed	21.1	21.5	11.9	10.9	19.8	12.5	13.7
Pumpkin	6k	19.9	24.1	23.8	18.3	24.1	23.8	19.6	14.2	24.7	21.7	22.3
RedKitchen	12k	17.6	21.4	21.4	10.9	20.8	20.9	11.6	11.2	18.9	13.8	13.7
Stairs	3k	19.0	16.7	21.0	13.0	17.7	19.9	15.8	15.9	18.8	15.3	15.4
Average		18.7	21.2	21.8	N/A	21.2	21.5	13.6	12.6	21.0	17.1	17.9
Avg. Time		realtime	38h	13h	18min	1h	7min	8.5h	47h	27min	4min*	16min

Table 1: 7-Scenes. We show the pose accuracy via view synthesis with Nerfacto [89] as PSNR in dB, and the reconstruction time. Results for *All Frames* are color coded w.r.t. similarity to the COLMAP pseudo ground truth: > 0.5 dB better within ± 0.5 dB > 0.5 dB worse > 1 dB worse. For some competitors, we had to sub-sample the images due to their computational complexity (right side). [†]Method needs sequential inputs. *Results on more powerful hardware.

ACE Zero



ACE Zero



ACE Zero

- vs Colmap



7-Scenes - Chess



Tanks & Temples - Caterpillar



Mip-NeRF 360 - Bonsai



Mip-NeRF 360 - Bicycle



ACE Zero

- More evaluation for visual localization and pose quality in PSNR

	ACE [12]	ACE [12]	ACE0 (ours)
Supervision	KinectFusion	COLMAP	<i>self-supervised</i>
Chess	<u>96.0 %</u>	100.0 %	100.0 %
Fire	98.4 %	99.5 %	<u>98.8 %</u>
Heads	100.0 %	100.0 %	100.0 %
Office	36.9 %	100.0 %	<u>99.1 %</u>
Pumpkin	47.3 %	100.0 %	<u>99.9 %</u>
Redkitchen	47.8 %	98.9 %	<u>98.1 %</u>
Stairs	<u>74.1 %</u>	85.0 %	61.0 %
Average	71.5 %	97.6 %	<u>93.8 %</u>

Table 2 (a): Relocalization on 7-Scenes. % poses below 5cm, 5° error, computed w.r.t. COLMAP pseudo GT.

	Pseudo GT	DROID-SLAM [†]	BARF	NoPe-NeRF [†]	ACE0
	(COLMAP)	[94]	[55]	[10]	(ours)
Bicycle	21.5	10.9	11.9	12.2	18.7
Bonsai	27.6	10.9	12.5	14.8	25.8
Counter	25.5	12.9	11.9	11.6	24.5
Garden	26.3	16.7	13.3	13.8	25.0
Kitchen	27.4	13.9	13.3	14.4	26.1
Room	28.0	11.3	11.9	14.3	19.8
Stump	16.8	13.9	15.0	13.7	20.5
Average	24.7	12.9	12.8	13.5	22.9

Table 2 (b): Mip-NeRF 360. Pose quality in PSNR, higher is better. Best in **bold**. [†]Method needs sequential inputs.

ACE Zero

- Failure cases / limitation:
 - Extreme view change / repetitive structure / lightning change

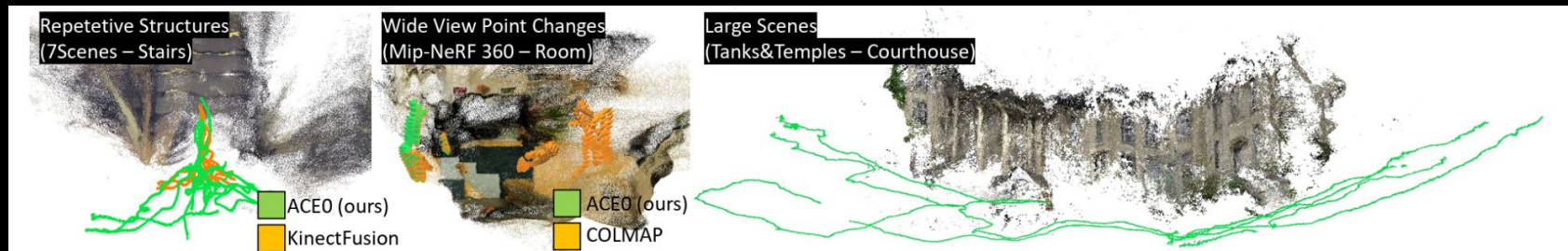
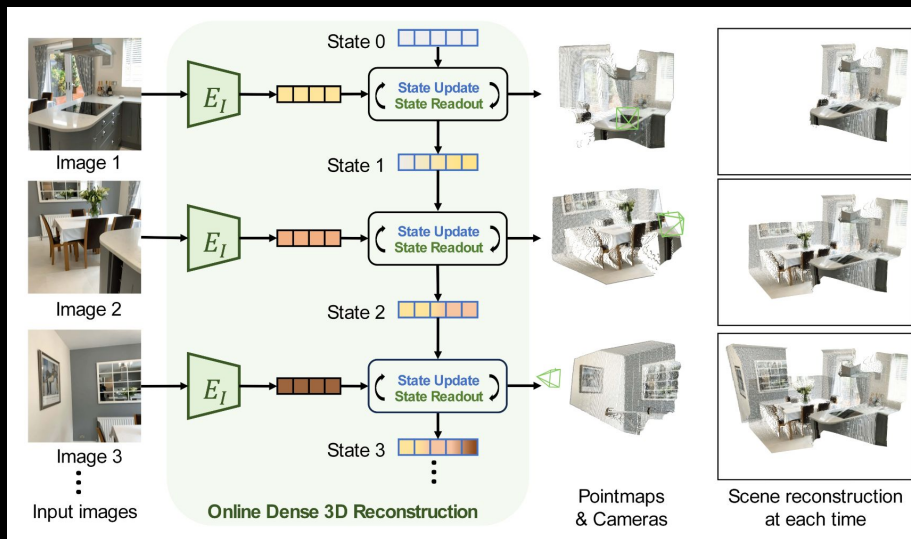


Fig. 6: Failure Cases. **Stairs:** Parts of the reconstruction collapse due to visual ambiguity. **Room:** ACE0 fails to register the views on the right due to low visual overlap. **Courthouse** is too large to be represented well by a single MLP.

Future Work

- Lightning / appearance change
- Large-scale scene \rightarrow accumulated error
- Inspiration for Cut3R, inference time self-calibration is important



Thank you for listening



NeRF : Neural Radiance Fields

Representing Scenes as Neural Radiance Fields
for View Synthesis

PRESENTED BY
Kanishkha Jaisankar
kj2675
03/13/2025

Presentation Overview

Content:

- NeRF Fundamentals: What are Neural Radiance Fields?
- How is it better than existing models
- Technical Foundation: How NeRF works (step-by-step)
- Evolution: From basic NeRF to advanced variations
- Applications: S-NeRF, City-Scale NeRF and Self-Driving
- Future Directions & Conclusion

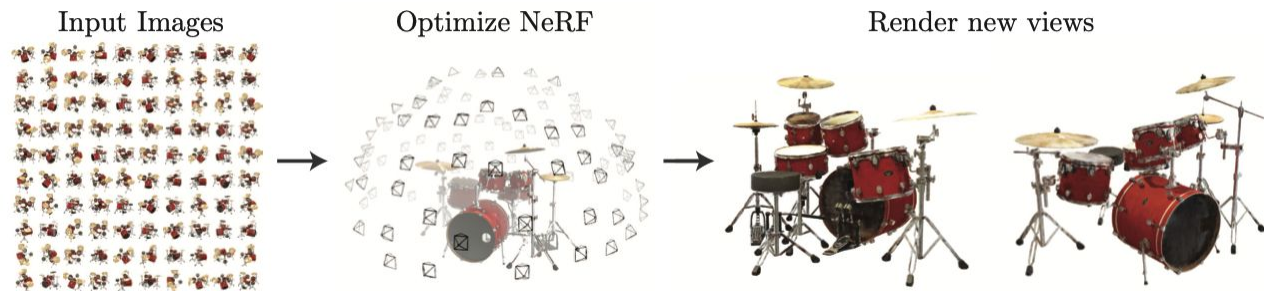
What is NeRF ?

NeRF is a technique developed to represent 3D scenes as a 5D continuous function representation (3D spatial location and 2D viewing direction) using a neural network.

Neural : It uses neural network MLP as its core component.

Radiance : The network learns to predict the radiance at any point in 3D space from any direction.

Fields : Creates a field of radiance not just discrete points



The 3D Reconstruction Challenge

Problem: Creating photorealistic 3D models from 2D images

Traditional Approaches

1. Mesh-Based
2. Voxel Based
3. Point Cloud

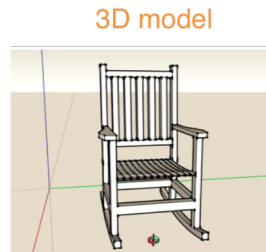
Limitations

1. Not continuous
2. Computational Cost
3. Memory Usage
4. Realism

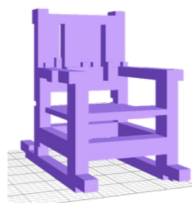


1	44	33	12	20	23	35	14
51	16	40	32	46	48	28	17
29	60	3	63	49	55	36	7
52	22	26	41	38	10	61	53
2	24	19	11	34	43	5	8
57	9	37	42	25	21	27	18
30	56	50	64	4	59	6	13
58	47	45	31	39	15	62	54

Pixel



Polygonal
mesh



Voxel
(volumetric pixel)

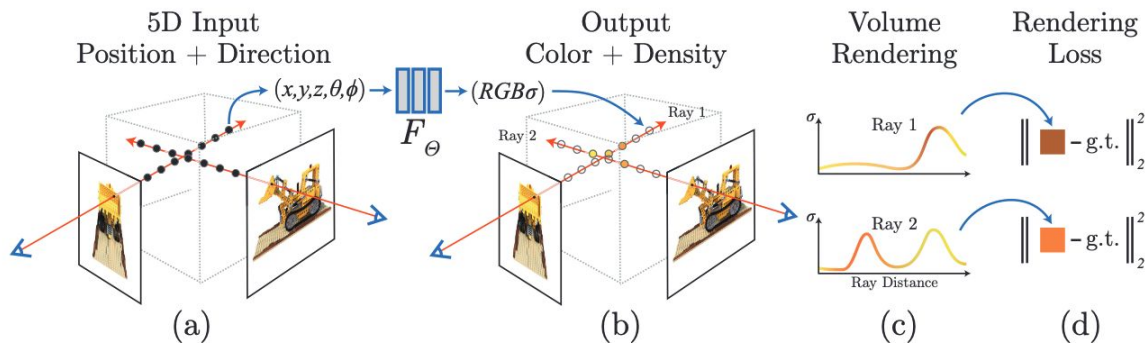


Point cloud

How NeRF Solves the Problem

NeRF Approach

1. NeRF uses math functions to represent scenes as continuous neural functions instead of discrete elements by passing in spatial and directional information (Infinite resolution).
2. NeRF models geometry and appearance both the components at once at every point in space by implicitly encoding the entire scene within the neural network weights.
3. NeRF performs volume rendering and can capture view dependent effects like reflections, transparency and specular highlights.



How does NeRF Work - Breakdown

Step 1 - Multi-View Input Images

- **Objective:** To reconstruct a 3D scene, NeRF requires multiple 2D images captured from various viewpoints, each accompanied by precise camera pose information.
- **Why ?:** These diverse perspectives enable the model to learn the scene's depth, geometry, and appearance. Accurate camera intrinsic and extrinsic parameters ensure that each image's viewpoint is correctly understood.



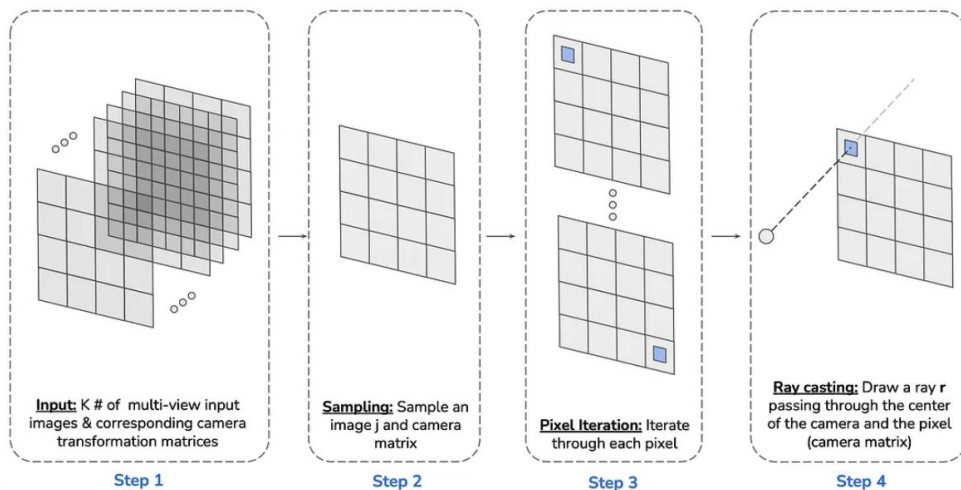
How does NeRF Work - Breakdown

Step 1: Multi-View Input Images - To reconstruct a 3D scene, NeRF requires multiple 2D images captured from various viewpoints, each accompanied by precise camera pose information.

Step 2: Sampling - To process each image independently by selecting an image and its associated camera matrices for further analysis.

Step 3: Pixel Iteration - To iterate over each pixel in the selected image, preparing for ray tracing.

Step 4: Ray Tracing - To trace a ray from the camera center through each pixel and extend it into the scene, mathematically represented as $r(t) = o + t \cdot d$, where o = camera center, d = viewing direction, and t is the distance from the camera center.

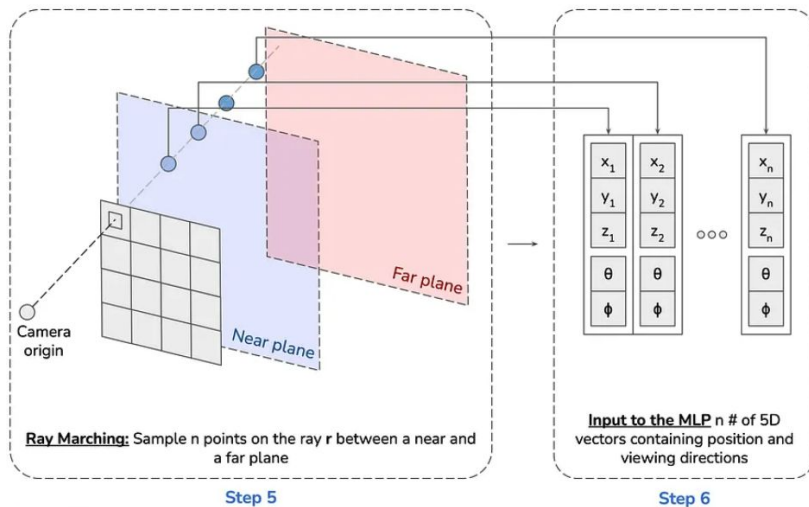


How does NeRF Work - Breakdown

Step 5: Ray Marching - After casting a ray from the camera through a pixel into the scene, we sample n points along this ray. To limit the extent of the ray, we define a near plane at t_n and a far plane at t_f , which are distances from the camera center. These planes confine our sampling to a specific region of interest.

Step 6: Multi Layer Perceptron input preparation - To transform each sampled point along a ray into a format suitable for the MLP, enabling the prediction of color and density at that point.

- For each pixel in the input image, determine the viewing direction, denoted as (θ, ϕ) .
- Along the ray corresponding to this pixel, sample n 3D points, represented as (x, y, z) .
- Combine the spatial coordinates of each sampled point with the viewing direction to form n five-dimensional input vectors: $[(x, y, z, \theta, \phi)]$.

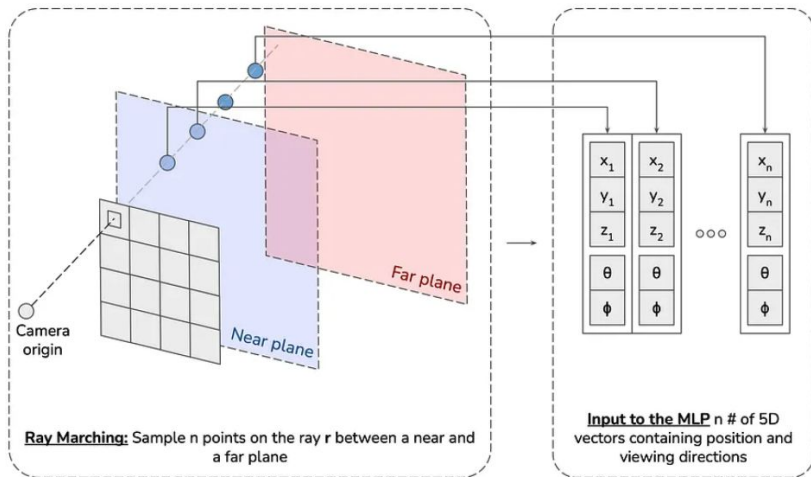
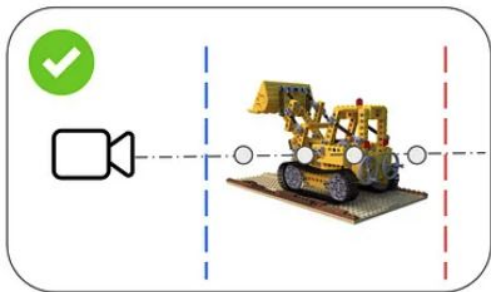


How does NeRF Work - Breakdown

Step 5: Ray Marching - After casting a ray from the camera through a pixel into the scene, we sample n points along this ray. To limit the extent of the ray, we define a near plane at t_n and a far plane at t_f , which are distances from the camera center. These planes confine our sampling to a specific region of interest.

Step 6: Multi Layer Perceptron input preparation - To transform each sampled point along a ray into a format suitable for the MLP, enabling the prediction of color and density at that point.

- For each pixel in the input image, determine the viewing direction, denoted as (θ, ϕ) .
- Along the ray corresponding to this pixel, sample n 3D points, represented as (x, y, z) .
- Combine the spatial coordinates of each sampled point with the viewing direction to form n five-dimensional input vectors: $[(x, y, z, \theta, \phi)]$.



Technical Foundations

Ray Marching Implementation

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^N T_i (1 - \exp(-\sigma_i \delta_i)) \mathbf{c}_i, \text{ where } T_i = \exp\left(-\sum_{j=1}^{i-1} \sigma_j \delta_j\right)$$

NeRF discretize the continuous integral into N sample points.

- $T_i = \exp(-\sum_{j=1}^{i-1} \sigma_j \delta_j)$ (accumulated transparency)
- δ_i : Distance between samples
- \mathbf{c}_i : RGB color at sample i
- σ_i : Density at sample i

How does NeRF Work - Breakdown

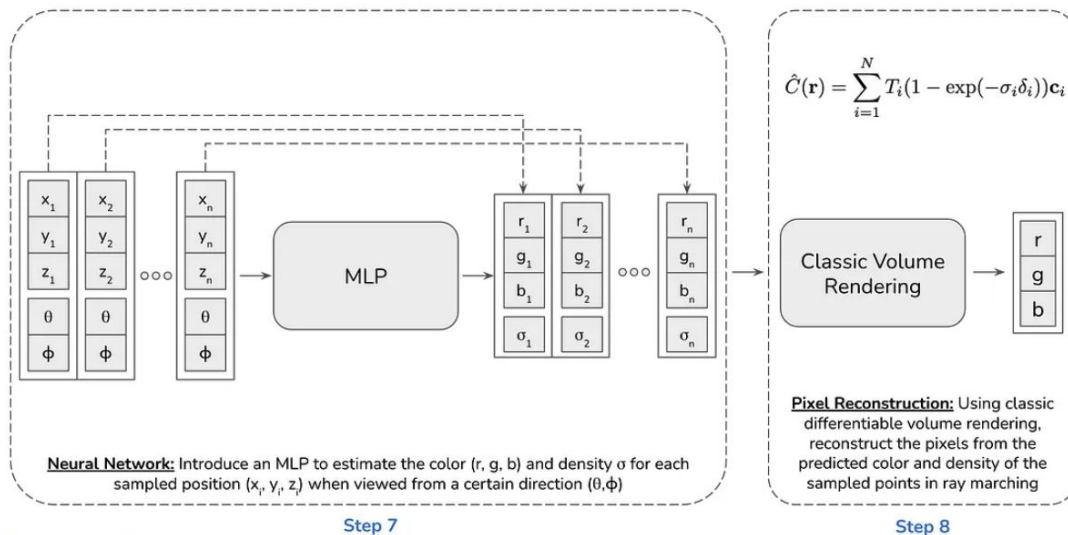
Step 7: Multi Layer Perceptron prediction - The MLP outputs a four-dimensional vector for each input, comprising:

Directional Emitted Color \mathbf{c} : The RGB color $(\mathbf{r}, \mathbf{g}, \mathbf{b})$ that the point contributes when viewed from direction (θ, ϕ) .

Volumetric Density σ : A scalar value indicating the opacity or density at the point

Step 8: Pixel Reconstruction -

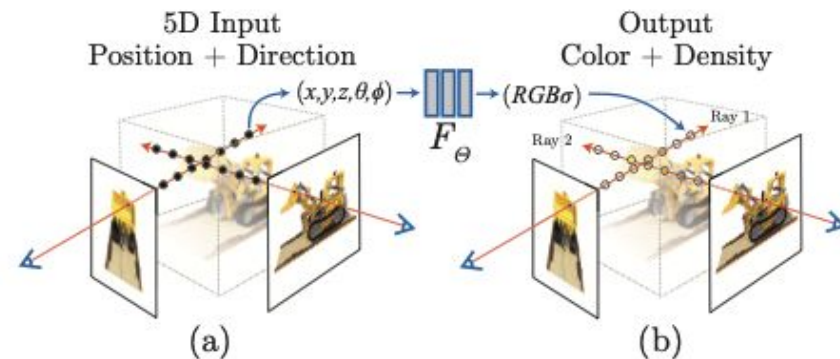
- **Volume Rendering Integration:** Aggregate the color & density predictions from sampled points along each ray.
- **Differentiable Process:** The volume rendering equation differentiability allows for efficient training of the MLP through backpropagation, minimizing the difference between predicted and actual pixel colors.



Technical Foundations

NeRF Mathematical Foundation

- NeRF represents a scene as a continuous 5D function:
- $F(\mathbf{x}, \mathbf{y}, \mathbf{z}, \theta, \phi) \rightarrow (\mathbf{r}, \mathbf{g}, \mathbf{b}, \sigma)$
- (x, y, z) : 3D position in space
- (θ, ϕ) : Viewing direction (angles)
- (r, g, b) : RGB color values
- σ (sigma): Volume density (how "solid" a point is)



Technical Foundations

Volume Rendering Equation

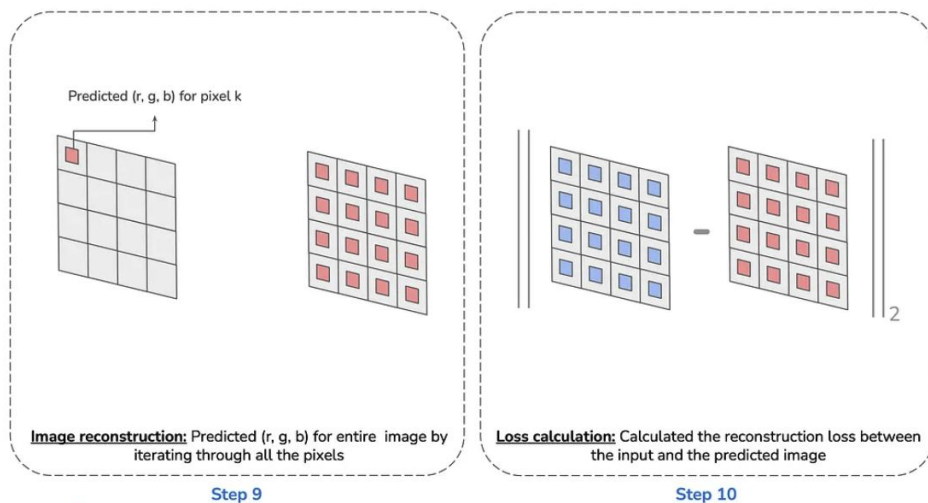
$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t) \sigma(\mathbf{r}(t)) \mathbf{c}(\mathbf{r}(t), \mathbf{d}) dt, \text{ where } T(t) = \exp\left(-\int_{t_n}^t \sigma(\mathbf{r}(s)) ds\right)$$

- To render a pixel, NeRF uses the volume rendering equation.
- $C(\mathbf{r})$: Color of ray \mathbf{r} (final pixel color)
- $T(t)$: Accumulated transmittance (how much light reaches this point)
- $\sigma(\mathbf{r}(t))$: Density at point $\mathbf{r}(t)$
- $\mathbf{c}(\mathbf{r}(t), \mathbf{d})$: Color at point $\mathbf{r}(t)$ from direction \mathbf{d}
- t : Distance along the ray

How does NeRF Work - Breakdown

Step 9: Image Reconstruction - To synthesize the entire image by predicting the color for each pixel based on the trained NeRF model. For each pixel cast a ray from the camera through the pixel into the scene. Sample points along the ray and input coordinates into trained MLP. Aggregate MLP output using volume rendering equation for pixel final color.

Step 10: Loss Calculation - To quantify the difference between reconstructed image and actual input image guiding the optimization process. Compute the pixel-wise Mean Squared Error (MSE) loss between the predicted pixels colors and true pixel colors.



Technical Foundations

Loss Function

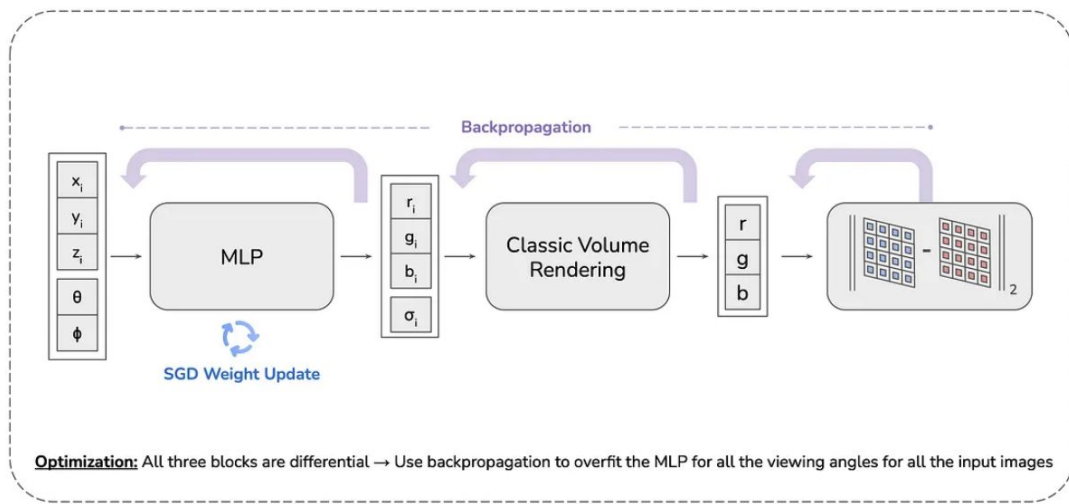
$$\mathcal{L} = \sum_{\mathbf{r} \in \mathcal{R}} \left[\left\| \hat{C}_c(\mathbf{r}) - C(\mathbf{r}) \right\|_2^2 + \left\| \hat{C}_f(\mathbf{r}) - C(\mathbf{r}) \right\|_2^2 \right]$$

- NeRF is trained to minimize the difference between rendered and actual images:
- \mathcal{L} : Loss function
- $C(\mathbf{r})$: Rendered pixel color
- $\hat{C}(\mathbf{r})$: Ground truth pixel color
- \mathcal{R} : Set of all rays (pixels) in training images

How does NeRF Work - Breakdown

Step 11: Optimization - To adjust the MLP's parameters to minimize the loss, thereby improving the accuracy of the scene representation.

- Utilize the differentiable nature of both the MLP and the volume rendering process to perform backpropagation.
- Compute gradients of the loss with respect to the MLP's weights.
- Update the MLP's weights iteratively using an optimization algorithm like stochastic gradient descent) to minimize the loss.



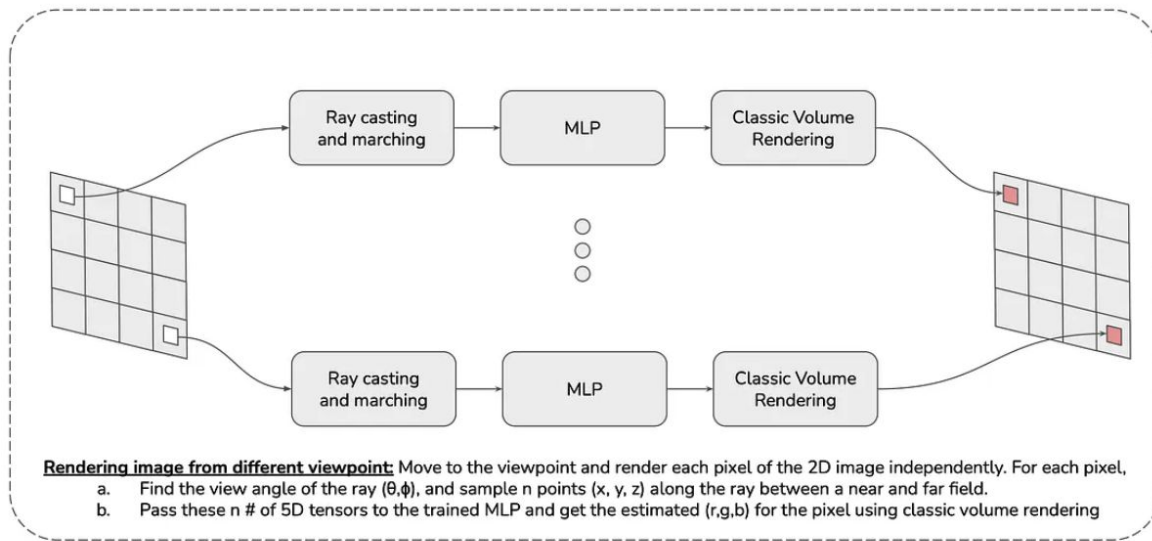
Step 11

How does NeRF Work - Breakdown

Step 12: Rendering image from a novel viewpoint (inference) - After training the Multi-Layer Perceptron (MLP) to represent a specific scene, NeRF enables the generation of photorealistic images from new, unseen viewpoints.

Process flow:

1. Ray Casting -> 2. Sampling Along Rays -> 3. MLP Evaluation -> 4. Volume Rendering -> 5. Image Reconstruction



Step 12

How does NeRF Work - Summary

1. **Input:** NeRF requires multi-view images of a scene, along with their corresponding camera matrices.
2. **Sampling:** Start by selecting an image and its camera matrix to begin the process.
3. **Pixel Iteration:** For each pixel in the image, repeat the following steps.
4. **Ray Casting:** Cast a ray r from the camera center through the pixel, as defined by the camera matrix.
5. **Ray Marching:** Sample n points along the ray r , between a near and a far plane.
6. **Input to the MLP:** Construct n 5D vectors, each containing the sampled position (x,y,z) and viewing direction (θ,ϕ) , and feed them into the MLP.
7. **MLP Output:** The MLP predicts the color (r, g, b) and volumetric density σ for each sampled point.
8. **Pixel Reconstruction:** Use differentiable volume rendering to combine the predicted color and density of the sampled points to reconstruct the pixel's color.
9. **Image Reconstruction:** Iterate over all pixels to predict the entire image.
10. **Loss Calculation:** Compute the reconstruction loss between the predicted image and the ground truth input image.
11. **Optimization:** Leverage the differentiable nature of all components to use backpropagation, training the MLP to overfit the scene for all input views.
12. **Rendering from Novel Viewpoints:** Query the trained MLP to generate pixel colors for a new viewpoint and reconstruct the image.

Datasets

The paper uses both synthetic and real-world datasets to evaluate the performance of NeRF.

Synthetic Datasets:

- **Diffuse Synthetic 360° (from DeepVoxels [41]):** This dataset contains renderings of four Lambertian objects (objects with diffuse, non-shiny surfaces) with simple geometry. The objects are rendered from viewpoints sampled on the upper hemisphere. There are 479 images used as input and 1000 for testing, all at 512x512 resolution.
- **Realistic Synthetic 360° (created by the authors):** This dataset contains renderings of eight objects with more complex geometry and realistic, non-Lambertian materials. Six objects are rendered from viewpoints sampled on the upper hemisphere, and two are rendered from viewpoints sampled on a full sphere. There are 100 views used as input and 200 for testing, all at 800x800 resolution. The scenes include objects like a ship, chair, drums, ficus, hotdog, lego, microphone, and materials.

Real-World Datasets:

- **Real Forward-Facing (taken from Local Light Field Fusion (LLFF) [28] and augmented by the authors):** This dataset contains handheld, forward-facing captures of eight real-world scenes. The scenes are captured with a cellphone, with each scene having between 20 to 62 images. 1/8 of the images are held out for testing. All images are 1008x756 resolution. The scenes include objects like plants (Fern, Orchid), a T-Rex model, and indoor scenes (Room, Leaves).

Results Discussion - Metrics

The paper uses three metrics to quantitatively evaluate the performance of NeRF and compare it to other view synthesis methods:

- **PSNR (Peak Signal-to-Noise Ratio):** Measures the similarity between the rendered image and the ground truth image. Higher PSNR indicates better image quality.
- **SSIM (Structural Similarity Index Measure):** Measures the perceived structural similarity between the rendered image and the ground truth image. Higher SSIM indicates better structural similarity.
- **LPIPS (Learned Perceptual Image Patch Similarity):** Measures the perceptual similarity between the rendered image and the ground truth image, using deep features. Lower LPIPS indicates better perceptual similarity.

The papers uses three view synthesis methods to compare NeRFs performance

- **SRN (Scene Representation Networks) [42]:** SRN represents a scene as a continuous, opaque surface. It uses an MLP to map a 3D coordinate (x, y, z) to a feature vector.
- **NV (Neural Volumes) [24]:** NV synthesizes novel views of objects that lie entirely within a bounded volume in front of a distinct background. The algorithm renders novel views by marching camera rays through the warped voxel grid.
- **LLFF (Local Light Field Fusion) [28]:** LLFF is designed for producing photorealistic novel views for well-sampled forward-facing scenes. It uses a trained 3D convolutional network to directly predict a discretized frustum-sampled RGBa grid for each input view.

Results Discussion - Quantitative

Synthetic Dataset

	PSNR \uparrow							
	Chair	Drums	Ficus	Hotdog	Lego	Materials	Mic	Ship
SRN [42]	26.96	17.18	20.73	26.81	20.85	18.09	26.85	20.60
NV [24]	28.33	22.58	24.79	30.71	26.08	24.22	27.78	23.93
LLFF [28]	28.72	21.13	21.79	31.41	24.54	20.72	27.48	23.22
Ours	33.00	25.01	30.13	36.18	32.54	29.62	32.91	28.65

	SSIM \uparrow							
	Chair	Drums	Ficus	Hotdog	Lego	Materials	Mic	Ship
SRN [42]	0.910	0.766	0.849	0.923	0.809	0.808	0.947	0.757
NV [24]	0.916	0.873	0.910	0.944	0.880	0.888	0.946	0.784
LLFF [28]	0.948	0.890	0.896	0.965	0.911	0.890	0.964	0.823
Ours	0.967	0.925	0.964	0.974	0.961	0.949	0.980	0.856

	LPIPS \downarrow							
	Chair	Drums	Ficus	Hotdog	Lego	Materials	Mic	Ship
SRN [42]	0.106	0.267	0.149	0.100	0.200	0.174	0.063	0.299
NV [24]	0.109	0.214	0.162	0.109	0.175	0.130	0.107	0.276
LLFF [28]	0.064	0.126	0.130	0.061	0.110	0.117	0.084	0.218
Ours	0.046	0.091	0.044	0.121	0.050	0.063	0.028	0.206

Table 4: Per-scene quantitative results from our realistic synthetic dataset. The “scenes” in this dataset are all objects with more complex geometry and non-Lambertian materials, rendered using Blender’s Cycles pathtracer.

Real Image Dataset

	PSNR \uparrow							
	Room	Fern	Leaves	Fortress	Orchids	Flower	T-Rex	Horns
SRN [42]	27.29	21.37	18.24	26.63	17.37	24.63	22.87	24.33
LLFF [28]	28.42	22.85	19.52	29.40	18.52	25.46	24.15	24.70
Ours	32.70	25.17	20.92	31.16	20.36	27.40	26.80	27.45

	SSIM \uparrow							
	Room	Fern	Leaves	Fortress	Orchids	Flower	T-Rex	Horns
SRN [42]	0.883	0.611	0.520	0.641	0.449	0.738	0.761	0.742
LLFF [28]	0.932	0.753	0.697	0.872	0.588	0.844	0.857	0.840
Ours	0.948	0.792	0.690	0.881	0.641	0.827	0.880	0.828

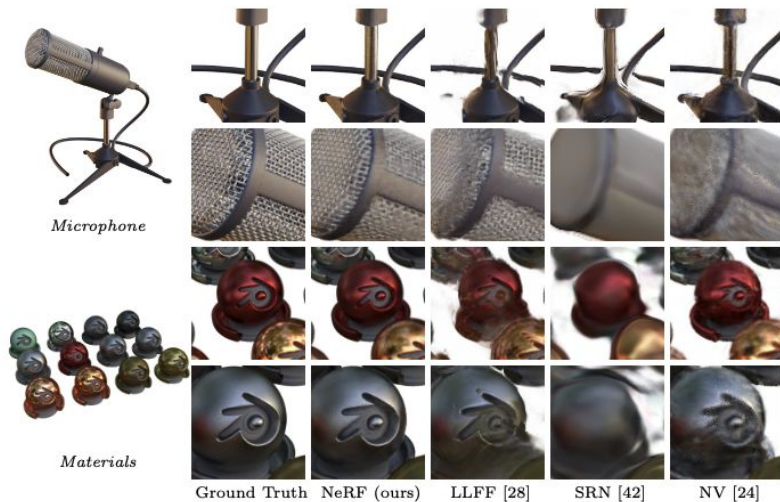
	LPIPS \downarrow							
	Room	Fern	Leaves	Fortress	Orchids	Flower	T-Rex	Horns
SRN [42]	0.240	0.459	0.440	0.453	0.467	0.288	0.298	0.376
LLFF [28]	0.155	0.247	0.216	0.173	0.313	0.174	0.222	0.193
Ours	0.178	0.280	0.316	0.171	0.321	0.219	0.249	0.268

Table 5: Per-scene quantitative results from our real image dataset. The scenes in this dataset are all captured with a forward-facing handheld cellphone.

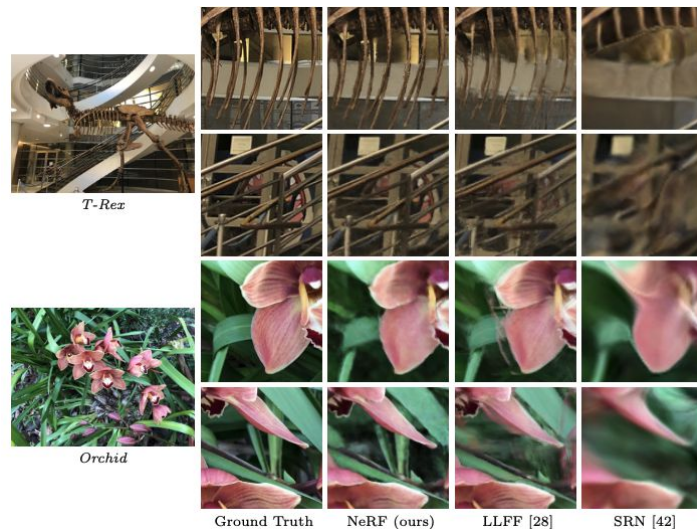
Insights : NeRF outperforms baselines and performs well in both synthetic and real world datasets

Results Discussion - Qualitative

Test-set views of real world datasets



Test-set views of real world datasets



NeRF Ablation Studies

Insights :

1. Positional Embeddings is Crucial
2. View Dependence is important
3. Hierarchical Sampling improves performance

	Input	#Im.	L	(N_c, N_f)	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow
1) No PE, VD, H	xyz	100	-	(256, -)	26.67	0.906	0.136
2) No Pos. Encoding	$xyz\theta\phi$	100	-	(64, 128)	28.77	0.924	0.108
3) No View Dependence	xyz	100	10	(64, 128)	27.66	0.925	0.117
4) No Hierarchical	$xyz\theta\phi$	100	10	(256, -)	30.06	0.938	0.109
5) Far Fewer Images	$xyz\theta\phi$	25	10	(64, 128)	27.78	0.925	0.107
6) Fewer Images	$xyz\theta\phi$	50	10	(64, 128)	29.79	0.940	0.096
7) Fewer Frequencies	$xyz\theta\phi$	100	5	(64, 128)	30.59	0.944	0.088
8) More Frequencies	$xyz\theta\phi$	100	15	(64, 128)	30.81	0.946	0.096
9) Complete Model	$xyz\theta\phi$	100	10	(64, 128)	31.01	0.947	0.081

An ablation study is a technique used to understand the contribution of different components of a model or system. It involves systematically removing parts of a model to see how performance changes.

Key Ideas and Contributions

- **Neural Radiance Field (NeRF):** Representing a scene as a continuous volumetric function, rather than discrete voxels or meshes. This allows for potentially infinite resolution and avoids discretization artifacts.
- **5D Coordinate Input:** The network takes both spatial location and viewing direction as input, enabling it to model view-dependent effects like reflections and specularities.
- **Volume Rendering:** Novel views are synthesized by marching rays through the scene and accumulating the color and density values using classical volume rendering techniques.
- **Differentiable Rendering:** The entire rendering process is differentiable, allowing the network to be trained end-to-end using gradient descent. The only input needed is a set of images with known camera poses.
- **Positional Encoding:** A high-frequency positional encoding is applied to the input 5D coordinates before feeding them into the MLP. This helps the network to represent high-frequency details in the scene.
- **Hierarchical Sampling:** A hierarchical sampling strategy is used to efficiently sample the volume along each ray. A "coarse" network is first used to estimate the density distribution, and then a "fine" network is used to sample more points in regions of high density.

NeRF Limitations

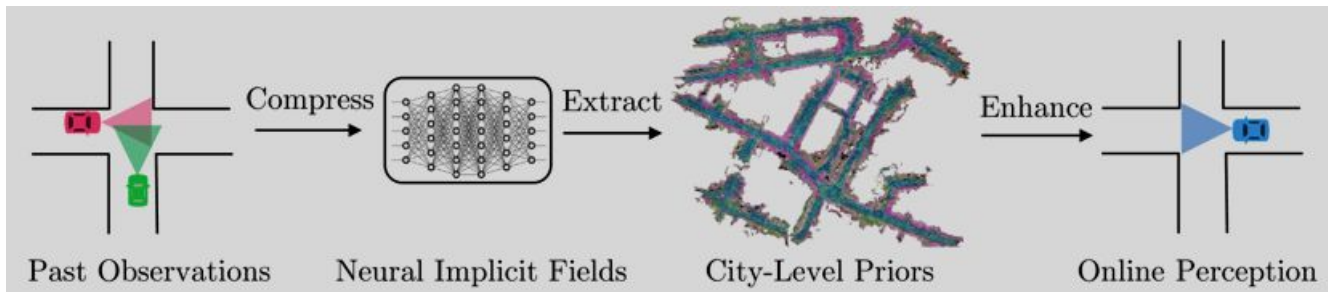
- Computational Cost
- Rendering Time
- Static Scenes
- Memory Requirements
- Difficulties with Large-Scale Scenes
- View Dependence Artifacts
- Lack of Interpretability
- Limited Generalization
- Sensitivity to Input Data
- Difficulty in Handling Transparent or Translucent Objects

NeRF Evolution

- instant-NGP (2022) - Uses multi-resolution hash encoding to speedup training by 100x.
- D-NeRF (2021) - Adds time as input dimension for moving objects.
- Nerfies (2021) - Captures non-rigid deformations like facial expressions.
- Semantic-Nerf (2022) - Adds class prediction point in 3D space.
- LERF (2023) - Connects language descriptions to 3D locations.
- PixelNeRF (2023) - Renders novel views from single image.
- MeRF (2024) - On device rendering for AR Applications.

NeRF in Self Driving Cars

1. S-NeRF (Scene-Specific Neural Rendering with Motion)
 - S-NeRF extends the original NeRF framework to handle dynamic scenes, where objects move and change over time.
 - Self-driving cars need to understand the motion of pedestrians, vehicles, and other obstacles. S-NeRF's ability to model scene flow enables more accurate prediction of future object positions.
2. City-Scale NeRF
 - City-Scale NeRF addresses the challenge of scaling NeRF to large, complex environments like urban areas.
 - City-Scale NeRF can generate highly detailed 3D maps of urban environments, providing self-driving cars with precise and accurate spatial information.

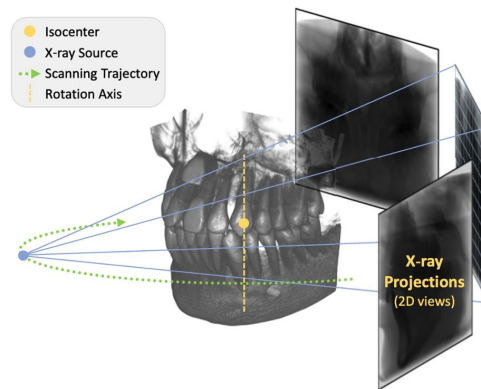


Future Directions

- Develop faster algorithms, optimization techniques and sampling strategies
- Handling Dynamic Scenes (Time varying scenes with moving objects)
- Improve scalability for large scale scenes
- Use meta learning or few-shot learning techniques to generalize to new scenes
- Method to accurately model transparent or reflective materials.
- Try hybrid approaches by combining NeRF with meshes and Point clouds
- Neural scene editing

Real Life Applications

- Creating Immersive VR Experience (Apple Vision Pro and Meta Quest)
- AR applications
- Scene understanding for industrial robots
- Virtual Real estate property tours
- 3D reconstruction from medical scans
- Virtual Try on applications
- Film Editing



Relevant to Coursework:

Simulate realistic environments for self driving car training. (Synthetic LiDAR data creation, HD map construction, sensor fusion, photorealistic rendering)

Conclusion

- NeRF addresses the limitations of prior MLP-based scene representation methods.
- NeRF achieves state-of-the-art (SOTA) results in novel view synthesis.
- Representing scenes as continuous 5D neural radiance fields is highly effective.
- NeRF outperforms methods using discretized voxel representations.
- Hierarchical sampling improves rendering efficiency.
- Further research is needed to optimize and render NeRFs more efficiently.

Demo : <https://youtu.be/JuH79E8rdKc?si=qXpL2bCvPGMCJJm6>

“

Thank You!!

Q&A ?

—

TEMPLATE NOTES

This template is part of the [NYU Templates collection](#).
Refer to our [Usage Guidelines](#) for help topics and quick
tips on how to use this template.

Download the [Grackle Slides](#) add-on to automatically run
accessibility checks on all aspects of your document and
get advice on how to make things better.

DUST3R : Geometric 3D Vision Made Easy

Denis Mbey Akola

Introduction

3D Dense Reconstruction is technique used to generate high-resolution 3D models from multiple 2D images or depth data.

Applications: Robotics, AR/VR, Cultural Heritage Preservation



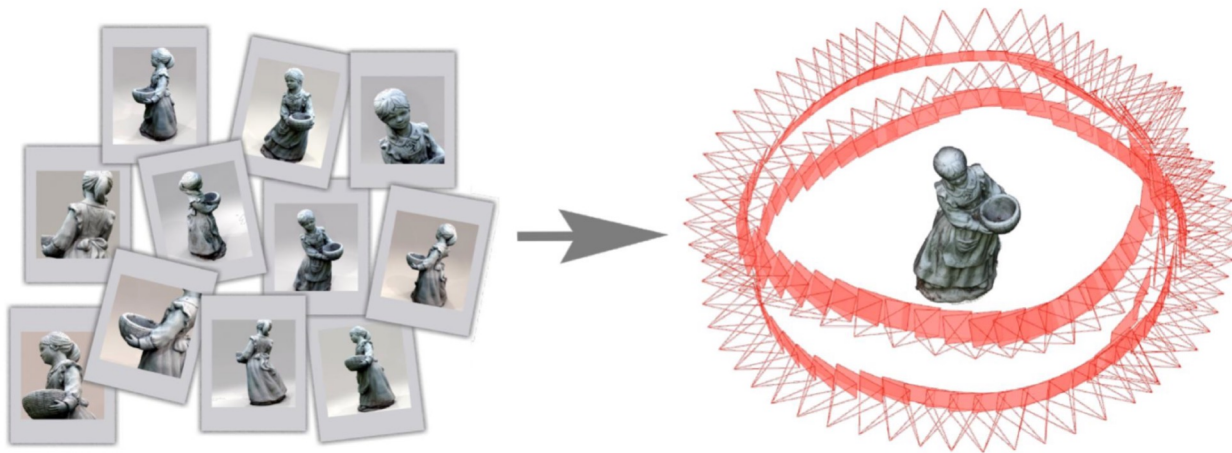
Input photos



Dense reconstruction

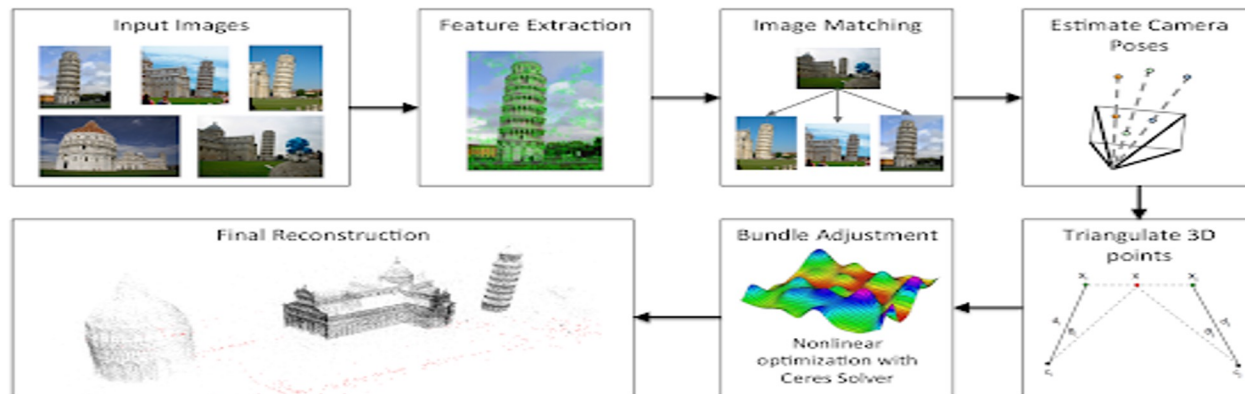
Conventional Methods for 3D Reconstruction

Multi-View Stereo (MVS): Requires set of posed and calibrated images of a scene is to reconstruct its dense 3D representation.



Conventional Methods for 3D Reconstruction

Structure From Motion (SfM) : Recover the camera poses and calibration matrix from a set of images taken from different viewpoints.

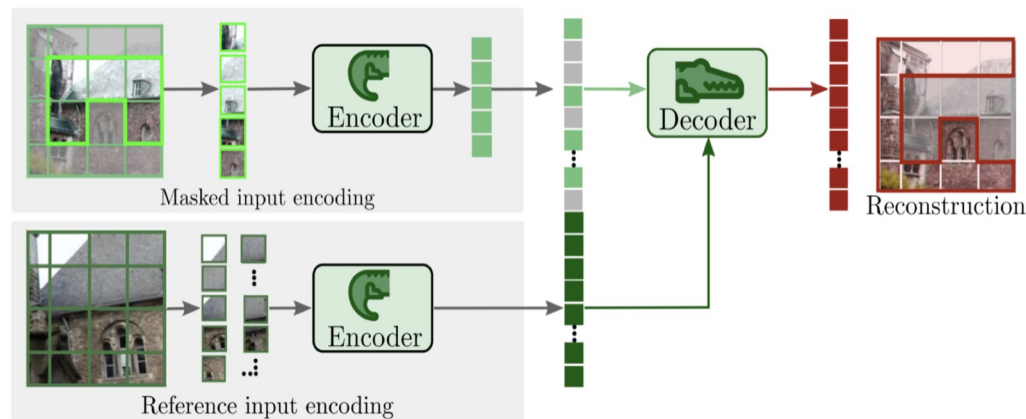


**Can we do MVS in the wild in a
more an elegant way without using
SfM?**

DUST3R : End-to-End 3D Reconstruction

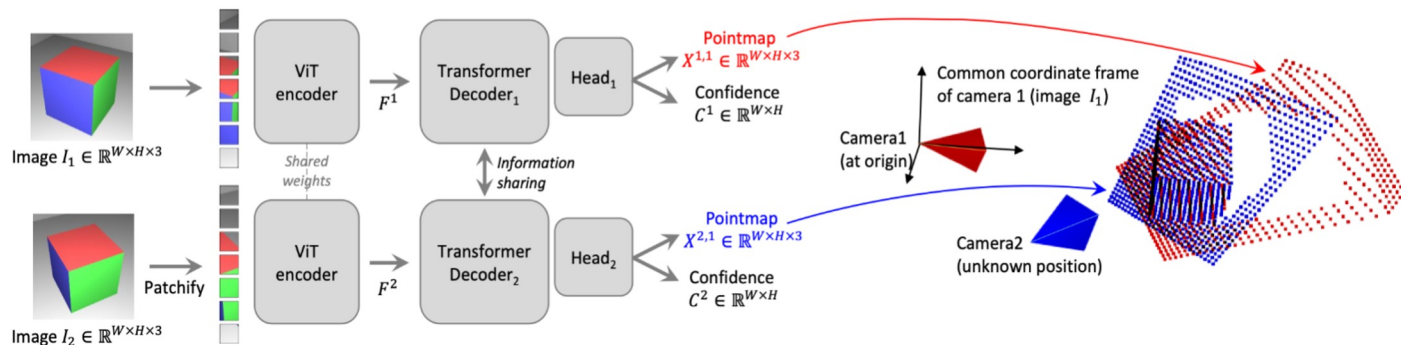
DUST3R is a holistic 3D reconstruction pipeline for uncalibrated and unposed images and unifies **monocular & binocular** 3D reconstruction.

DUST3R is inspired by the **CroCo model**; a cross-view completion pre-training method.



DUST3R : Architecture

Key innovations: Pointmap representation for MVS, optimizing global pointmap alignment



DUST3R: Geometric 3D Vision Made Easy – Wang et al., **CVPR 2024**

DUST3R: Pointmap Representation

Pointmap:

- **A dense 2D field of 3D points:** $X \in \mathbb{R}^{W \times H \times 3}$
- **One-to-one mapping** between image pixels and 3D scene points: $I_{ij} \leftrightarrow X_{ij}$

Camera and Scene:

- **Camera Ray** hits a single 3D point (ignoring translucent surfaces).
- **Pointmap from Depth Map:** $X_{ij} = K^{-1} \cdot [i \cdot D_{ij}, j \cdot D_{ij}, D_{ij}]^T$
- **Coordinate Transform:** $X_{n,m} = P_m \cdot P_n^{-1} \cdot h(X_n)$ where P_m, P_n are world-to-camera poses for images n and m .
- **Homogeneous mapping:** $h:(x,y,z) \rightarrow (x,y,z,1)$

DUSt3R : Datasets and Model Training

Datasets	Type	N Pairs
Habitat [103]	Indoor / Synthetic	1000k
CO3Dv2 [93]	Object-centric	941k
ScanNet++ [165]	Indoor / Real	224k
ArkitScenes [25]	Indoor / Real	2040k
Static Thing 3D [68]	Object / Synthetic	337k
MegaDepth [55]	Outdoor / Real	1761k
BlendedMVS [161]	Outdoor / Synthetic	1062k
Waymo [121]	Outdoor / Real	1100k

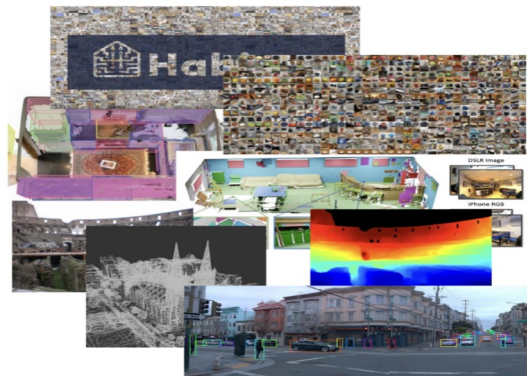


Image pairs are obtained using off-the-shelf retrieval and point matching algorithms, while **ground-truth pointmaps** are derived from camera intrinsics, poses, and depth maps.

DUST3R : Training Objective

Regression Loss: The regression loss is defined as the Euclidean distance

$$\ell_{\text{regr}}(v, i) = \left\| \frac{1}{z} X_i^{v,1} - \frac{1}{\bar{z}} \bar{X}_i^{v,1} \right\|$$

with $z = \text{norm}(X^{1,1}, X^{2,1})$ and $\bar{z} = \text{norm}(\bar{X}^{1,1}, \bar{X}^{2,1})$

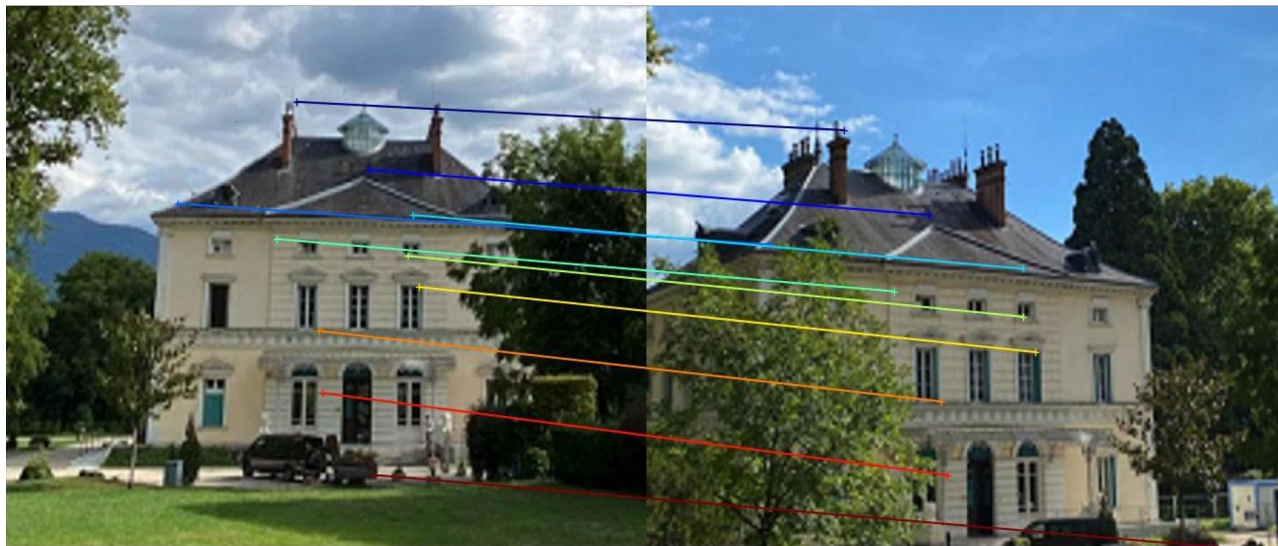
to handle the scale ambiguity between prediction and ground-truth

Confidence Aware Loss: An extension of the regression loss to handle the ill-defined 3D points (e.g. sky)

$$\mathcal{L}_{\text{conf}} = \sum_{v \in \{1,2\}} \sum_{i \in \mathcal{D}^v} C_i^{v,1} \ell_{\text{regr}}(v, i) - \alpha \log C_i^{v,1} \quad \text{where} \quad C_i^{v,1} = 1 + \exp \widetilde{C}_i^{v,1} > 1$$

DUSt3R Downstream tasks

DUST3R : Point Matching



Achieved by mutual nearest neighbor (MNN) search in the 3D pointmap space.

$$\mathcal{M}_{1,2} = \{(i, j) \mid i = \text{NN}_1^{1,2}(j) \text{ and } j = \text{NN}_1^{2,1}(i)\}$$
$$\text{with } \text{NN}_k^{n,m}(i) = \arg \min_{j \in \{0, \dots, WH\}} \|X_j^{n,k} - X_i^{m,k}\|.$$

DUST3R : Recovering Intrinsic

The pointmap is expressed in the first image coordinate frame (extrinsic as identical matrix), and we assume that the principal point is approximately centered. We only need to estimate the focal lengths by minimize:

$$f_1^* = \arg \min_{f_1} \sum_{i=0}^W \sum_{j=0}^H C_{i,j}^{1,1} \left\| (i', j') - f_1 \frac{(X_{i,j,0}^{1,1}, X_{i,j,1}^{1,1})}{X_{i,j,2}^{1,1}} \right\|$$

Method	Habitat	BlendedMVS	CO3D
Monocular	4.13° / 98.3%	3.40° / 99.4%	1.88° / 97.8%
Binocular	2.09° / 95.2%	2.61° / 98.4%	1.62° / 97.7%

Left: Average absolute error of field-of-view (FoV) estimates.

Right: Average 2D reprojection accuracy (%) at the threshold of 1% of image diagonal.

DUSt3R : Visual Localization

Given a **query image** and **retrieved database image**, the task can be achieved by :

1. First build the **pixel correspondences** from point matching, which in turn yields **2D-3D correspondences**. The camera pose is solved by the **PnP-RANSAC** with the estimated intrinsic.
1. Estimate the relative pose by point matching, convert the pose to world coordinate by scaling (scale factor obtain from the predicted pointmap and ground truth pointmap of the database image)

More information about PNP and RANSAC: https://docs.opencv.org/4.x/d5/d1f/calib3d_solvePnP.html

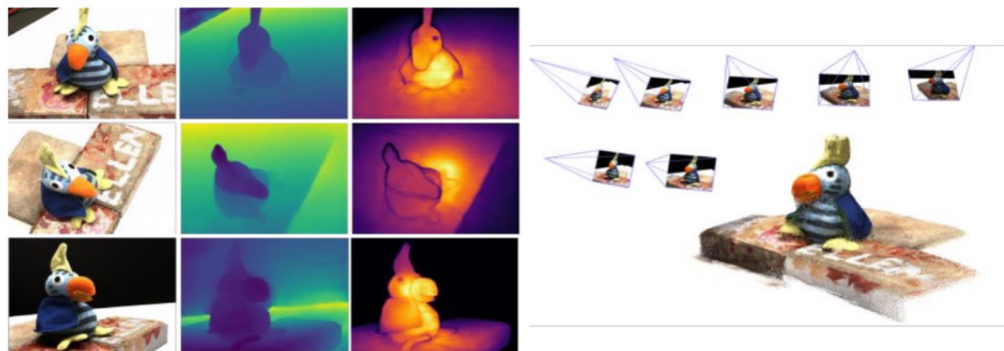
DUST3R : Visual Localization

Methods	7Scenes (Indoor) [113]							Cambridge (Outdoor) [48]					
	Chess	Fire	Heads	Office	Pumpkin	Kitchen	Stairs	S. Facade	O. Hospital	K. College	St.Mary's	G. Court	
FM	AS [102]	4/1.96	3/1.53	2/1.45	9/3.61	8/3.10	7/3.37	3/2.22	4/0.21	20/0.36	13/0.22	8/0.25	24/0.13
	HLoc [100]	2/0.79	2/0.87	2/0.92	3/0.91	5/1.12	4/1.25	6/1.62	4/0.2	15/0.3	12/0.20	7/0.21	11/0.16
E2E	DSAC* [11]	2/1.10	2/1.24	1/1.82	3/1.15	4/1.34	4/1.68	3/1.16	5/0.3	15/0.3	15/0.3	13/0.4	49/0.3
	HSCNet [54]	2/0.7	2/0.9	1/0.9	3/0.8	4/1.0	4/1.2	3/0.8	6/0.3	19/0.3	18/0.3	9/0.3	28/0.2
	PixLoc [101]	2/0/80	2/0.73	1/0.82	3/0.82	4/1.21	3/1.20	5/1.30	5/0.23	16/0.32	14/0.24	10/0.34	30/0.14
	SC-wLS [151]	3/0.76	5/1.09	3/1.92	6/0.86	8/1.27	9/1.43	12/2.80	11/0.7	42/1.7	14/0.6	39/1.3	164/0.9
	NeuMaps [124]	2/0.81	3/1.11	2/1.17	3/0.98	4/1.11	4/1.33	4/1.12	6/0.25	19/0.36	14/0.19	17/0.53	6/ 0.10
	DUST3R 224-NoCroCo	5/1.76	6/2.02	3/1.75	5/1.54	9/2.35	6/1.82	34/7.81	24/1.33	79/1.17	69/1.15	46/1.51	143/1.32
	DUST3R 224	3/0.96	3/1.02	1/1.00	4/1.04	5/1.26	4/1.36	21/4.08	9/0.38	26/0.46	20/0.32	11/0.38	36/0.24
	DUST3R 512	3/0.97	3/0.95	2/1.37	3/1.01	4/1.14	4/1.34	11/2.84	6/0.26	17/0.33	11/0.20	7/0.24	38/0.16



DUSt3R : Multi-View Pose Estimation

How? Obtained with relative pose estimation; Extract the pairwise camera poses from global alignment.



DUST3R : Global Optimization for Pointmaps

Goal: Align pointmaps $\{\chi_n \in \mathbb{R}^{W \times H \times 3}\}$ for all cameras $n=1, \dots, N$

Key Process:

- Predict pairwise pointmaps $\mathbf{X}^{n,n}$ and $\mathbf{X}^{m,n}$ and their associated confidence maps $\mathbf{C}^{n,n}$, $\mathbf{C}^{m,n}$ for each image pair $e=(n,m)$
- Use pairwise pose $P_e \in \mathbb{R}^{3 \times 4}$ and scaling σ_e for alignment.

Optimization

$$\chi^* = \arg \min_{\chi, P, \sigma} \sum_{e \in \mathcal{E}} \sum_{v \in e} \sum_{i=1}^{HW} C_i^{v,e} \|\chi_i^v - \sigma_e P_e X_i^{v,e}\|$$

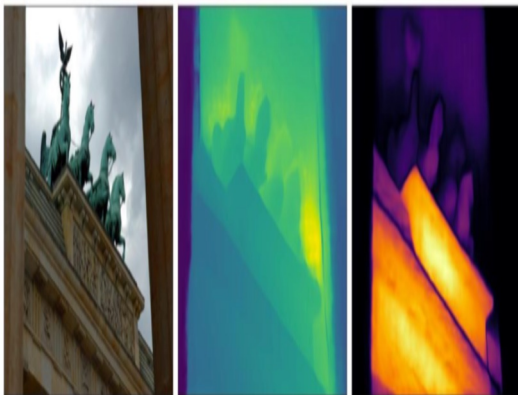
DUS_t3R : Multi-View Pose Estimation Results

Methods	N Frames	Co3Dv2 [94]			RealEstate10K [186]
		RRA@15	RTA@15	mAA(30)	mAA(30)
COLMAP+SPSG	3	~22	~14	~15	~23
PixSfM	3	~18	~8	~10	~17
Relpose	3	~56	-	-	-
PoseDiffusion	3	~75	~75	~61	-(~77)
DUS_t3R 512	3	95.3	88.3	77.5	69.5
COLMAP+SPSG	5	~21	~17	~17	~34
PixSfM	5	~21	~16	~15	~30
Relpose	5	~56	-	-	-
PoseDiffusion	5	~77	~76	~63	-(~78)
DUS_t3R 512	5	95.5	86.7	76.5	67.4
COLMAP+SPSG	10	31.6	27.3	25.3	45.2
PixSfM	10	33.7	32.9	30.1	49.4
Relpose	10	57.1	-	-	-
PoseDiffusion	10	80.5	79.8	66.5	48.0 (~80)
DUS_t3R 512	10	96.2	86.8	76.7	67.7

RRA : Relative Rotation Accuracy

RTA : Relative Translation Accuracy

DUS3R : Monocular Depth Estimation



Methods	Train	Outdoor				Indoor					
		DDAD[41]		KITTI [35]		BONN [80]		NYUD-v2 [115]		TUM [119]	
		Rel↓	$\delta_{1.25}$ ↑	Rel↓	$\delta_{1.25}$ ↑	Rel↓	$\delta_{1.25}$ ↑	Rel↓	$\delta_{1.25}$ ↑	Rel↓	$\delta_{1.25}$ ↑
DPT-BEiT[91]	D	10.70	84.63	9.45	89.27	-	-	5.40	96.54	10.45	89.68
NeWCRFs[174]	D	9.59	82.92	5.43	91.54	-	-	6.22	95.58	14.63	82.95
Monodepth2 [37]	SS	23.91	75.22	11.42	86.90	56.49	35.18	16.19	74.50	31.20	47.42
SC-SfM-Learners [6]	SS	16.92	77.28	11.83	86.61	21.11	71.40	13.79	79.57	22.29	64.30
SC-DepthV3 [121]	SS	14.20	81.27	11.79	86.39	12.58	88.92	12.34	84.80	16.28	79.67
MonoViT[182]	SS	-	-	09.92	90.01	-	-	-	-	-	-
RobustMIX [92]	T	-	-	18.25	76.95	-	-	11.77	90.45	15.65	86.59
SlowTv [117]	T	12.63	79.34	(6.84)	(56.17)	-	-	11.59	87.23	15.02	80.86
DUS3R 224-NoCroCo	T	19.63	70.03	20.10	71.21	14.44	86.00	14.51	81.06	22.14	66.26
DUS3R 224	T	16.32	77.58	16.97	77.89	11.05	89.95	10.28	88.92	17.61	75.44
DUS3R 512	T	13.88	81.17	10.74	86.60	8.08	93.56	6.50	94.09	14.17	79.89

By design, **depth prediction** is simply the **z** coordinate in the predicted 3D pointmap.

DUST3R : Multi-View Depth Estimation

Methods	Train	Outdoor				Indoor					
		DDAD[41]		KITTI [35]		BONN [80]		NYUD-v2 [115]		TUM [119]	
		Rel↓	$\delta_{1.25} \uparrow$	Rel↓	$\delta_{1.25} \uparrow$	Rel↓	$\delta_{1.25} \uparrow$	Rel↓	$\delta_{1.25} \uparrow$	Rel↓	$\delta_{1.25} \uparrow$
DPT-BEiT[91]	D	10.70	84.63	9.45	89.27	-	-	5.40	96.54	10.45	89.68
NeWCRFs[174]	D	9.59	82.92	5.43	91.54	-	-	6.22	95.58	14.63	82.95
Monodepth2 [37]	SS	23.91	75.22	11.42	86.90	56.49	35.18	16.19	74.50	31.20	47.42
SC-SfM-Learners [6]	SS	16.92	77.28	11.83	86.61	21.11	71.40	13.79	79.57	22.29	64.30
SC-DepthV3 [121]	SS	14.20	81.27	11.79	86.39	12.58	88.92	12.34	84.80	16.28	79.67
MonoViT[182]	SS	-	-	09.92	90.01	-	-	-	-	-	-
RobustMIX [92]	T	-	-	18.25	76.95	-	-	11.77	90.45	15.65	86.59
SlowTv [117]	T	12.63	79.34	(6.84)	(56.17)	-	-	11.59	87.23	15.02	80.86
DUST3R 224-NoCroCo	T	19.63	70.03	20.10	71.21	14.44	86.00	14.51	81.06	22.14	66.26
DUST3R 224	T	16.32	77.58	16.97	77.89	11.05	89.95	10.28	88.92	17.61	75.44
DUST3R 512	T	13.88	81.17	10.74	86.60	8.08	93.56	6.50	94.09	14.17	79.89

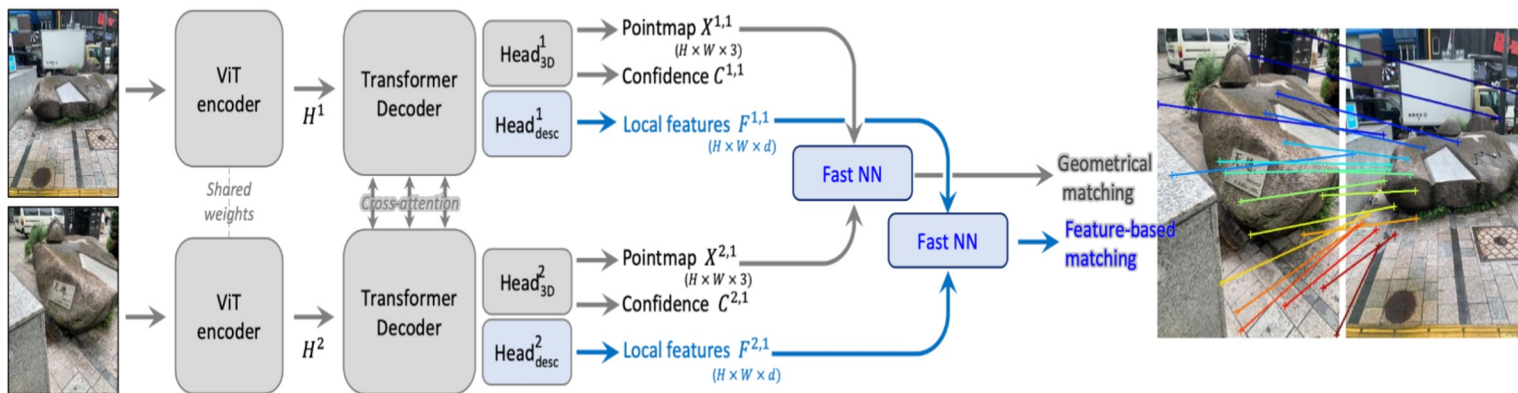
Rescale all predictions to align them together and aggregate all predictions via a **simple averaging weighted by the confidence**.

DUS_t3R : 3D Dense Reconstruction

	Methods	GT cams	Acc.↓	Comp.↓	Overall↓
(a)	Camp [12]	✓	0.835	0.554	0.695
	Furu [33]	✓	0.613	0.941	0.777
	Tola [134]	✓	0.342	1.190	0.766
	Gipuma [34]	✓	0.283	0.873	0.578
(b)	MVSNet [161]	✓	0.396	0.527	0.462
	CVP-MVSNet [158]	✓	0.296	0.406	0.351
	UCS-Net [18]	✓	0.338	0.349	0.344
	CER-MVS [65]	✓	0.359	0.305	0.332
	CIDER [157]	✓	0.417	0.437	0.427
	CasMVSNet [40]	✓	0.325	0.385	0.355
	PatchmatchNet [139]	✓	0.427	0.277	0.352
	GeoMVSNet [180]	✓	0.331	0.259	0.295
	DUS_t3R 512	×	2.677	0.805	1.741

**Can DUS3R be repurposed for robust
image matching?**

MASt3R : Grounding Image Matching in 3D



DUST3R with an additional head that regresses dense local feature maps, and train it with an InfoNCE loss

MASt3R : Training Objective

DUSt3R Loss:

$$\mathcal{L}_{\text{conf}} = \sum_{v \in \{1,2\}} \sum_{i \in \mathcal{D}^v} C_i^{v,1} \ell_{\text{regr}}(v, i) - \alpha \log C_i^{v,1}$$

Matching Loss (InfoNCE):

$$\mathcal{L}_{\text{match}} = - \sum_{(i,j) \in \hat{\mathcal{M}}} \log \frac{s_{\tau}(i, j)}{\sum_{k \in \mathcal{P}^1} s_{\tau}(k, j)} + \log \frac{s_{\tau}(i, j)}{\sum_{k \in \mathcal{P}^2} s_{\tau}(i, k)}$$

Total Loss

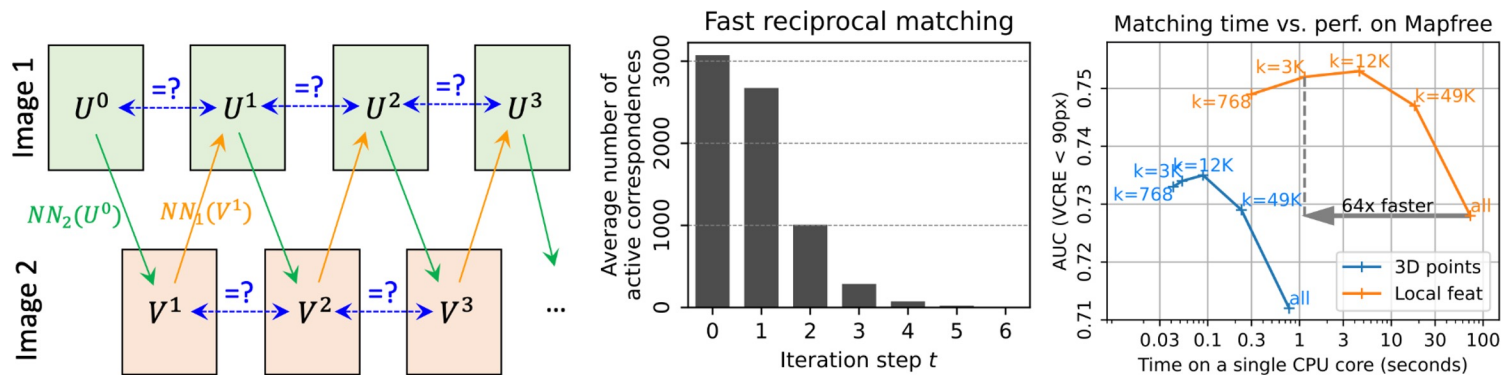
$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{conf}} + \beta \mathcal{L}_{\text{match}}$$

Ground truth matches are obtained by finding reciprocal correspondences between on the ground-truth 3D point maps $X^{\wedge 1,1} \leftrightarrow X^{\wedge 2,1}$

Oord, Aaron van den, Yazhe Li, and Oriol Vinyals. "Representation learning with contrastive predictive coding."



MASt3R : Fast Reciprocal Matching



Naive Reciprocal Matching : $O(W^2H^2)$ **Fast Reciprocal Matching:** $O(WH)$

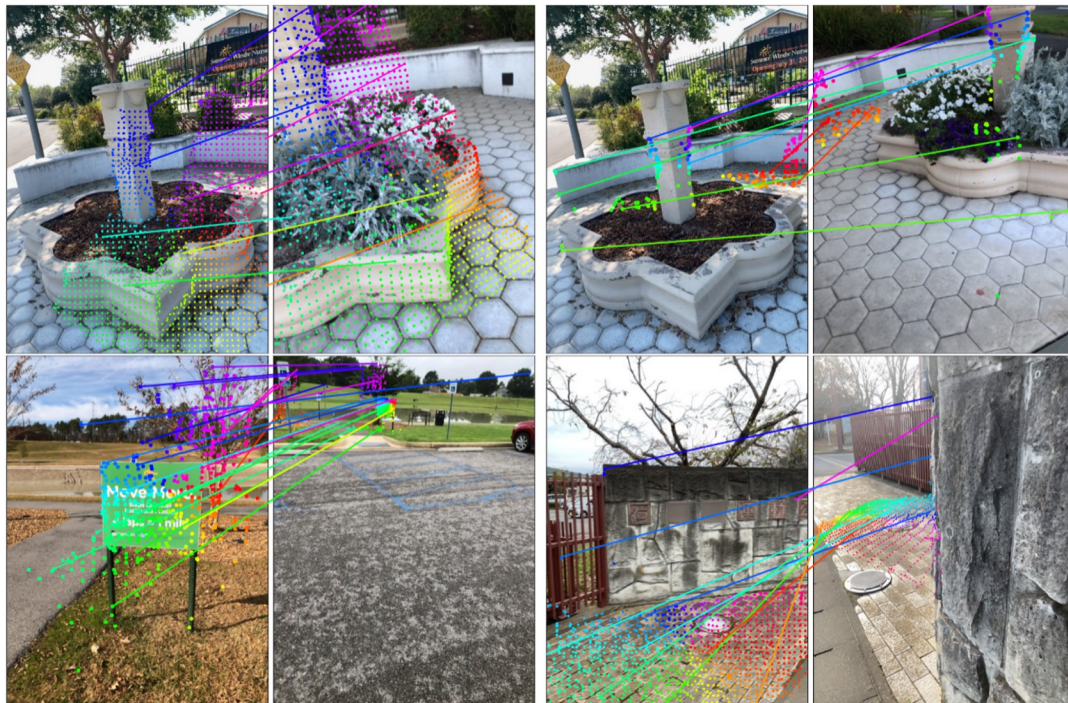
MASt3R Downstream Tasks

MASt3R : Image Matching

	depth	VCRE (<90px)			Pose Error			
		Reproj. ↓	Prec. ↑	AUC ↑	Med. Err. (m,°) ↓		Precision ↑	AUC ↑
RPR [5]	DPT	147.1 px	40.2%	0.402	1.68m	22.5°	6.0%	0.060
SIFT [52]	DPT	222.8 px	25.0%	0.504	2.93m	61.4°	10.3%	0.252
SP+SG [72]	DPT	160.3 px	36.1%	0.602	1.88m	25.4°	16.8%	0.346
LoFTR [82]	KBR	165.0 px	34.3%	0.634	2.23m	37.8°	11.0%	0.295
DUST3R [102]	DPT	116.0 px	50.3%	0.697	0.97m	7.1°	21.6%	0.394
MASt3R	DPT	104.0 px	54.2%	0.726	0.80m	2.2°	27.0%	0.456
MASt3R	(auto)	48.7 px	79.3%	0.933	0.36m	2.2°	54.7%	0.740
MASt3R (direct reg.)		53.2 px	79.1%	0.941	0.42m	3.1°	53.0%	0.777

Viewpoint-Condition Robustness Evaluation (VCRE) - Evaluates image matching robustness under varying viewpoints & conditions.

MASt3R : Image Matching



MASt3R: Relative Pose Estimation

Methods		Co3Dv2			RealEstate10K
		RRA@15	RTA@15	mAA(30)	mAA(30)
(a)	Colmap+SG [21,72]	36.1	27.3	25.3	45.2
	PixSfM [50]	33.7	32.9	30.1	49.4
	RelPose [115]	57.1	-	-	-
	PosReg [100]	53.2	49.1	45.0	-
	PoseDiff [100]	80.5	79.8	66.5	48.0
	RelPose++ [49]	(85.5)	-	-	-
	RayDiff [116]	(93.3)	-	-	-
	DUSt3R-GA [102]	96.2	86.8	76.7	67.7
(b)	DUSt3R [102]	94.3	88.4	77.2	61.2
	MASt3R	94.6	91.9	81.8	76.4

(a) Multi-view Methods (b) Pairwise methods

MASt3R : Multi-View 3D Reconstruction

	Methods	Acc.↓	Comp.↓	Overall↓
(c)	Camp [13]	0.835	0.554	0.695
	Furu [31]	0.613	0.941	0.777
	Tola [90]	0.342	1.190	0.766
	Gipuma [32]	0.283	0.873	0.578
(d)	MVSNet [110]	0.396	0.527	0.462
	CVP-MVSNet [109]	0.296	0.406	0.351
	UCS-Net [17]	0.338	0.349	0.344
	CER-MVS [55]	0.359	0.305	0.332
	CIDER [107]	0.417	0.437	0.427
	PatchmatchNet [99]	0.427	0.277	0.352
	GeoMVSNet [119]	0.331	0.259	0.295
(e)	DUS3R [102]	2.677	0.805	1.741
	MASt3R	0.403	0.344	0.374

(c) Hand crafted methods (b) Domain specific methods (e) Zero shot methods

Future Works

1. 3D reconstruction for dynamic scenes

Wang, Q., Zhang, Y., Holynski, A., Efros, A. A., & Kanazawa, A. (2025). *Continuous 3D Perception Model with Persistent State*. arXiv. <https://arxiv.org/abs/2501.12387>

1. Overcoming the computational inefficiency of DUST3R especially global optimization for multi-view scenarios where $N \gg 2$.

Cabon, Y., Stoffl, L., Antsfeld, L., Csurka, G., Chidlovskii, B., Revaud, J., & Leroy, V. (2025). *MUST3R: Multi-view Network for Stereo 3D Reconstruction*. arXiv. <https://arxiv.org/abs/2503.01661>.

1. Leveraging MAST3R and DUST3R in other applications domains.

Murai, R., Dexheimer, E., & Davison, A. J. (2025). *MASt3R-SLAM: Real-time dense SLAM with 3D reconstruction priors*. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).

Conclusion & Takeaways

1. DUS₃R and MAS₃R provide good representations for various 3D vision tasks correspondence matching, depth estimation, dense 3D reconstruction.
1. 3D perception can also greatly benefit from large scale pre-training

Thank you for listening!

Any questions?

Zero-1-to-3: Zero-shot One Image to 3D Object

Ruoshi Liu¹ Rundi Wu¹ Basile Van Hoorick¹ Pavel Tokmakov² Sergey Zakharov² Carl Vondrick¹

¹ Columbia University ² Toyota Research Institute

Problems with 3D Generative Task

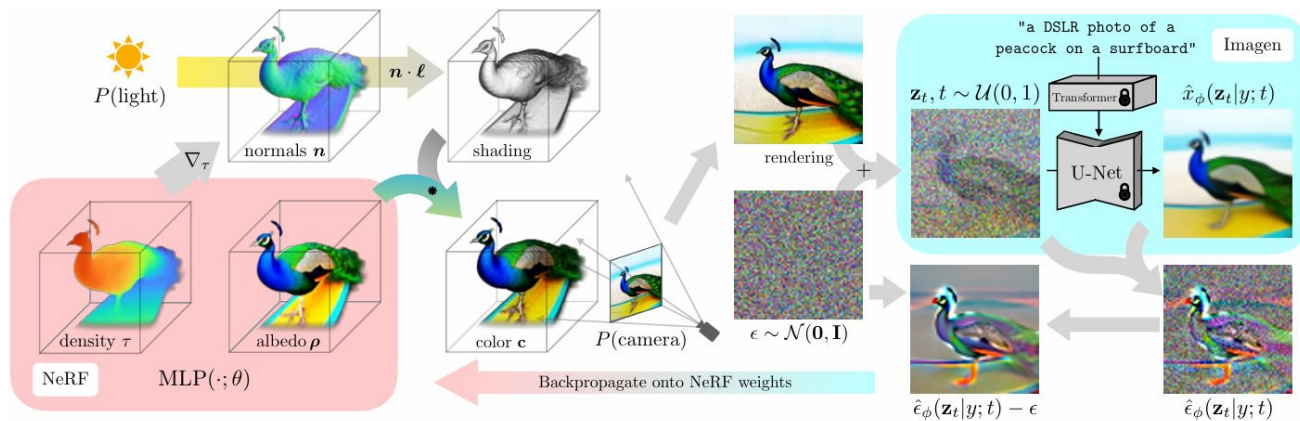
- The World is 3D
 - Humans perceive 3D concepts naturally.
 - We have a strong 3D prior from interactions with the 3D world.
- The majority of vision dataset is 2D
 - LAION: **5B**
 - Objectverse-XL: **10M**
- Model trained with 3D data show limited generalization
 - Data is expensive - difficult to scale
 - Rely on domain specific prior - poor generalization

Motivation

- 3D data has rich geometric prior - but the data is limited
- **Large scale** 2D visual data include different views of objects
- Does vision model trained on internet scale 2D data already learnt some 3D priors?

Related Work

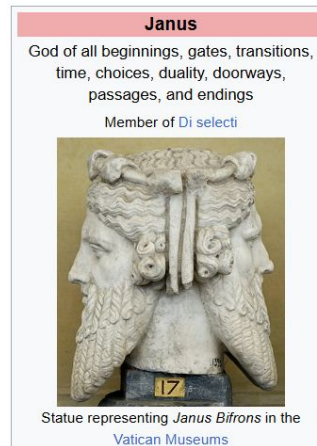
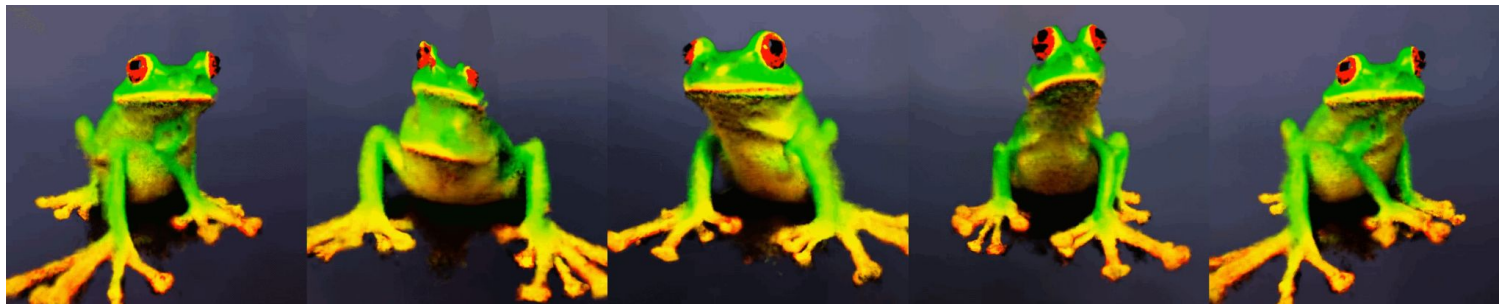
- Does vision model already learnt some 3D priors?



Related Work

3D Consistency from text conditioning:

- Text prompt {"overhead view," "front view," "side view," or "back view"}
is added based on the camera's position
- Not works too well...



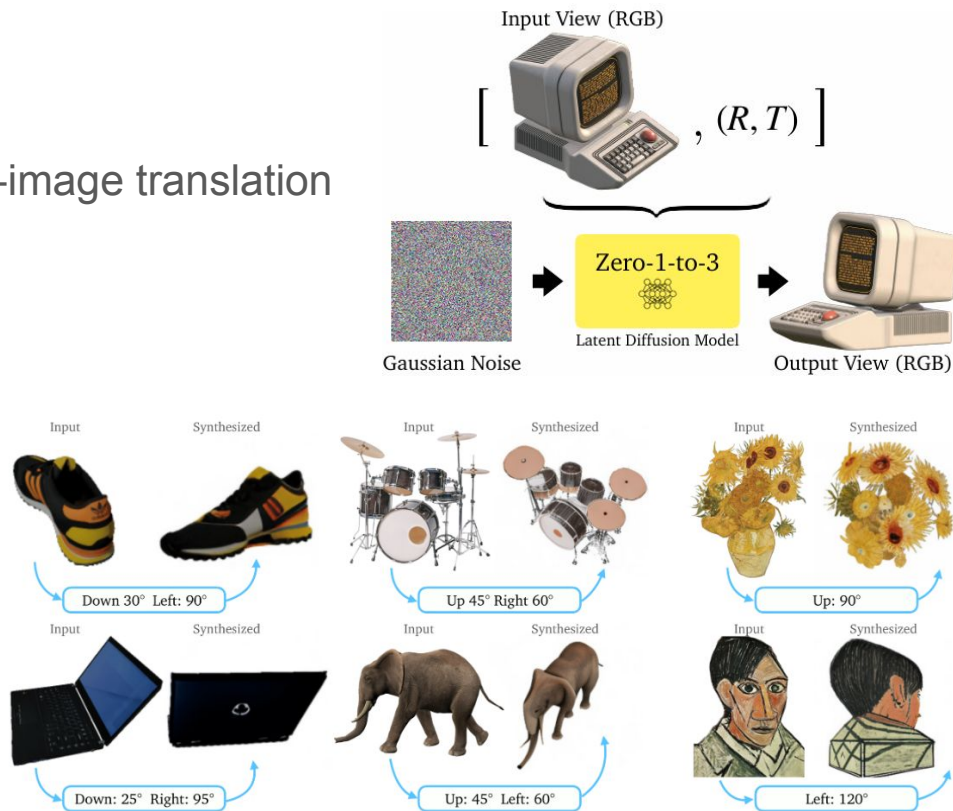
Zero123

- Task: Viewpoint-conditioned image-to-image translation

$$\hat{x}_{R,T} = f(x, R, T)$$

R: camera rotation

T: camera translation




Zero123

- Task: Viewpoint-conditioned image-to-image translation
- Challenge:
 - No explicit viewpoint information
 - Viewpoint bias



Figure 2: **Viewpoint bias in text-to-image models.** We show samples from both Dall-E-2 and Stable Diffusion v2 from the prompt “*a chair*”. Most samples show a chair in a forward-facing canonical pose.

Zero123

- Task: Viewpoint-conditioned image-to-image translation
 - Method:
 - Finetune **pretrained** diffusion model with **3D object** data to learn **controls over the camera**
 - Dataset: $\{(x, x_{(R,T)}, R, T)\}$
 - Objective: $\min_{\theta} \mathbb{E}_{z \sim \mathcal{E}(x), t, \epsilon \sim \mathcal{N}(0,1)} \|\epsilon - \epsilon_{\theta}(z_t, t, c(x, R, T))\|_2^2$.
- 



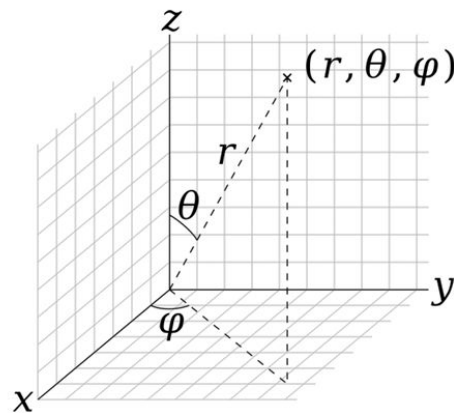
Zero123

$$\min_{\theta} \mathbb{E}_{z \sim \mathcal{E}(x), t, \epsilon \sim \mathcal{N}(0,1)} \|\epsilon - \epsilon_{\theta}(z_t, t, c(x, R, T))\|_2^2.$$

- Posed CLIP Conditioning
 - **CLIP** embeddings of original image + Relative camera movement
 - Used in U-Net's **cross-attention** layers.
- Input Image Conditioning
 - Comes from the LDM encoder.
 - Directly **concatenated** with U-Net inputs.

More Details

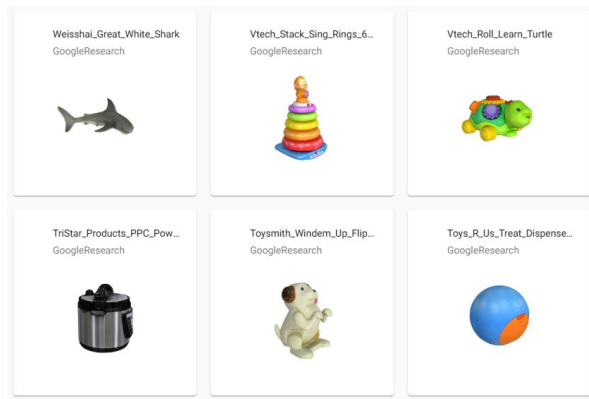
- Relative camera position represented by $[\theta, \sin(\theta), \cos(\varphi), r]$
- Object centric image only
 - ~800K Objectverse 3D models
 - Need remove background first for in the wild image



Results

- Task: Novel view synthesis
- Data: Google scanned objects, RTMV
- Baseline: Few-shot NeRF & other diffusion based method

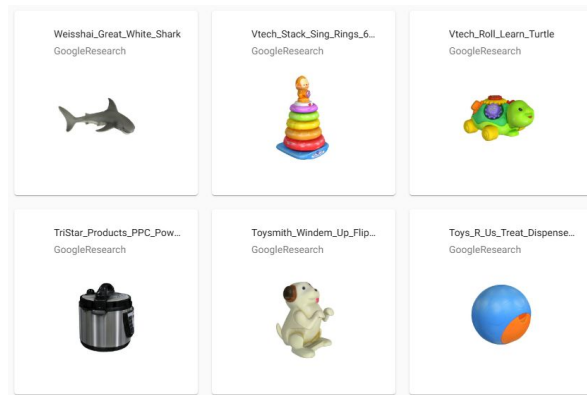
	DietNeRF [23]	Image Variation [1]	SJC-I [53]	Ours
PSNR \uparrow	<u>8.933</u>	5.914	6.573	18.378
SSIM \uparrow	<u>0.645</u>	0.540	0.552	0.877
LPIPS \downarrow	<u>0.412</u>	0.545	0.484	0.088
FID \downarrow	<u>12.919</u>	22.533	19.783	0.027



Results

- Task: 3D reconstruction
- Data: Google scanned objects, RTMV
- Baseline: Few-shot NeRF & other diffusion based method

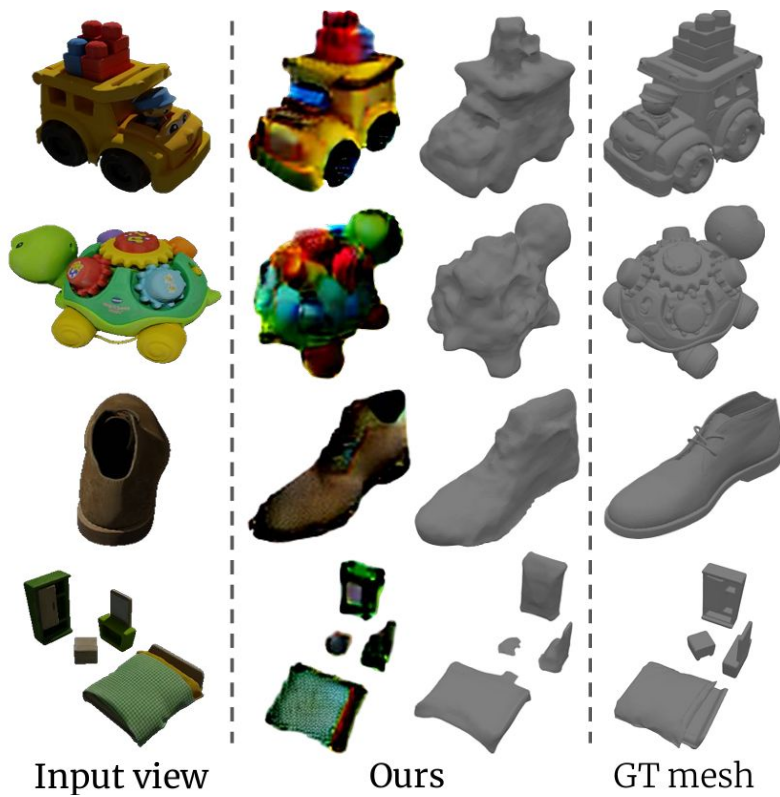
	MCC [59]	SJC-I [53]	Point-E [34]	Ours
CD ↓	0.1230	0.2245	<u>0.0804</u>	0.0717
IoU ↑	0.2343	0.1332	<u>0.2944</u>	0.5052



Results



Results



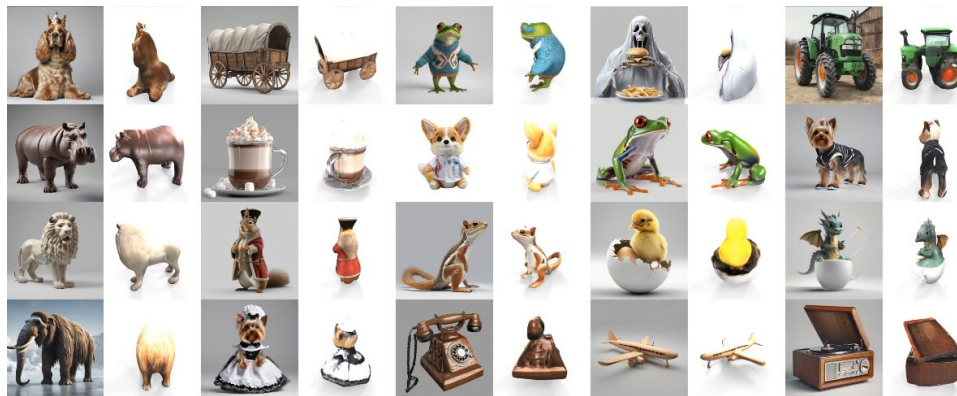
Summary

- Strength:

- Novel method to leverage pretrained diffusion model for zero-shot novel view synthesis
 - Showed large vision model have learned rich 3D priors
 - Strong result with great impact
-

- Weakness:

- **Object centric** image only



More on NVS

- Real world vision data is not object centric...
 - Camera won't always look at a object
 - Real world scenes have different scale
 - Backgrounds are also important

ZeroNVS: Zero-Shot 360-Degree View Synthesis from a Single Image

Kyle Sargent¹, Zizhang Li¹, Tanmay Shah², Charles Herrmann², Hong-Xing Yu¹,
Yunzhi Zhang¹, Eric Ryan Chan¹, Dmitry Lagun², Li Fei-Fei¹, Deqing Sun², Jiajun Wu¹

¹Stanford University, ²Google Research

- Key idea:
 - 3 DoF -> 6 DoF + 1 camera condition for non-object centric scene
 - Scale Normalization
 - SDS anchoring for background diversity

ZeroNVS

- Key idea:

- **3 DoF -> 6 DoF + 1 camera condition**
- Scale Normalization
- SDS anchoring

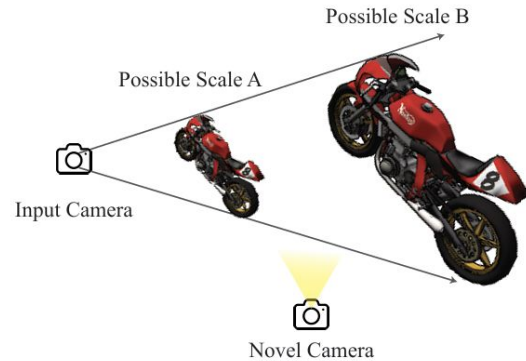
3 DoF rotation + 3 DoF translation + **FOV**

FOV is added for different cameras

ZeroNVS

- Key idea:

- 3 DoF \rightarrow 6 DoF + 1 camera condition
- **Scale Normalization**
- SDS anchoring



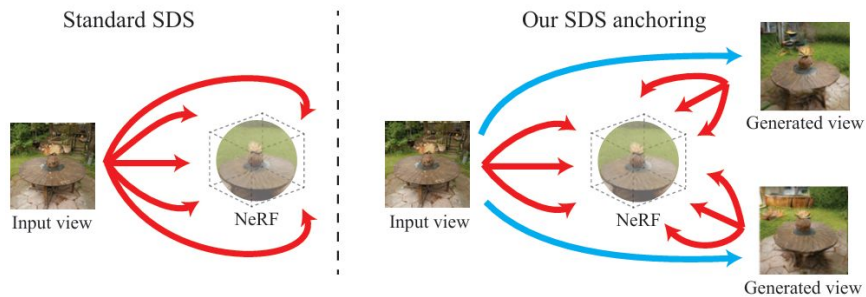
Scale Normalization based on:

- Camera
- Depth

ZeroNVS

- Key idea:

- 3 DoF -> 6 DoF + 1 camera condition
- Scale Normalization
- **SDS anchoring**



$$\mathcal{L}_{SDS}(\tilde{X}) = \left\| w(\sigma) \left(\epsilon_{\theta}(\tilde{X} + \epsilon, X_i, \mathbf{M}(D, f, E, i, j)) - \epsilon \right) \right\|_2^2$$

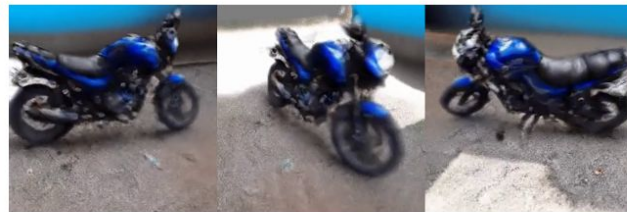
$$\mathcal{L}_{SDS \text{ anchoring}}(\tilde{X}) = \left\| w(\sigma) \left(\epsilon_{\theta}(\tilde{X} + \epsilon, X_{i_{\text{nearest}}}, \mathbf{M}(D, f, E, i, j)) - \epsilon \right) \right\|_2^2$$

CO3D



Input view

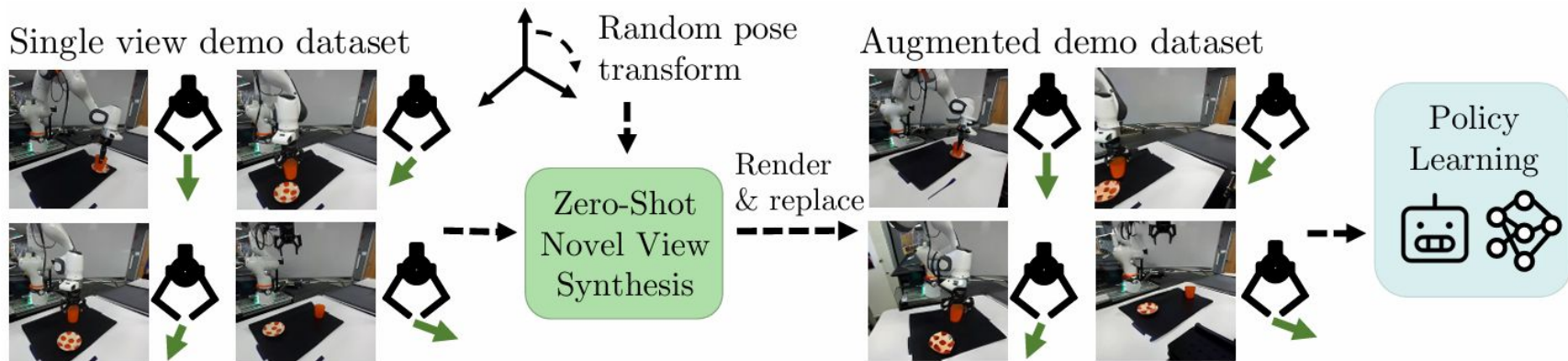
Novel views



Input view

Novel views

Applications in Robot Learning



Summary

- Strength:
 - Showed that 3D aware diffusion model + SDS can be applied to scene
 - Real world applications
- Weakness:
 - Sometime the scene could still be 3D inconsistent