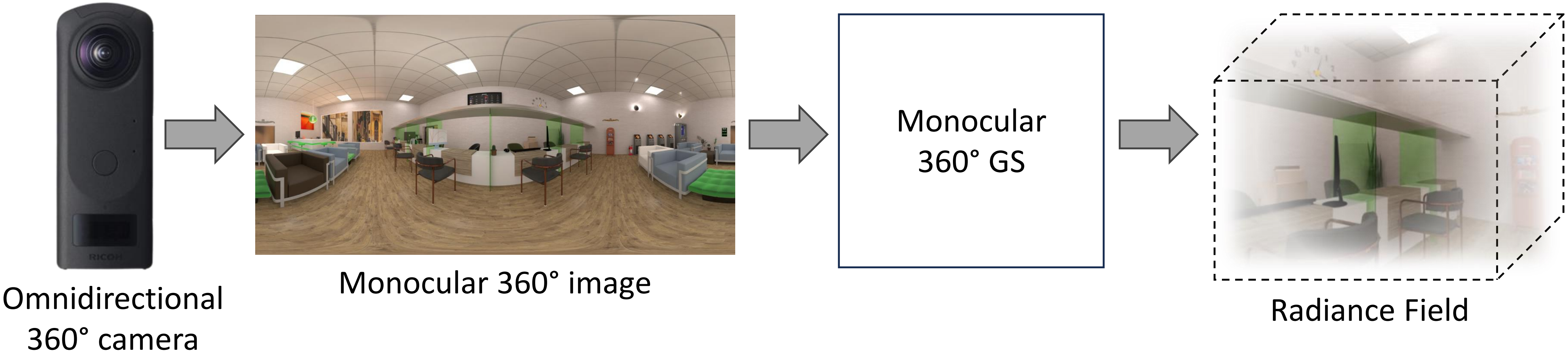
Monocular 360° GS (Guided Research Project)

Patrick Noras

Motivation



Omnidirectional 360° camera

Monocular 360° image

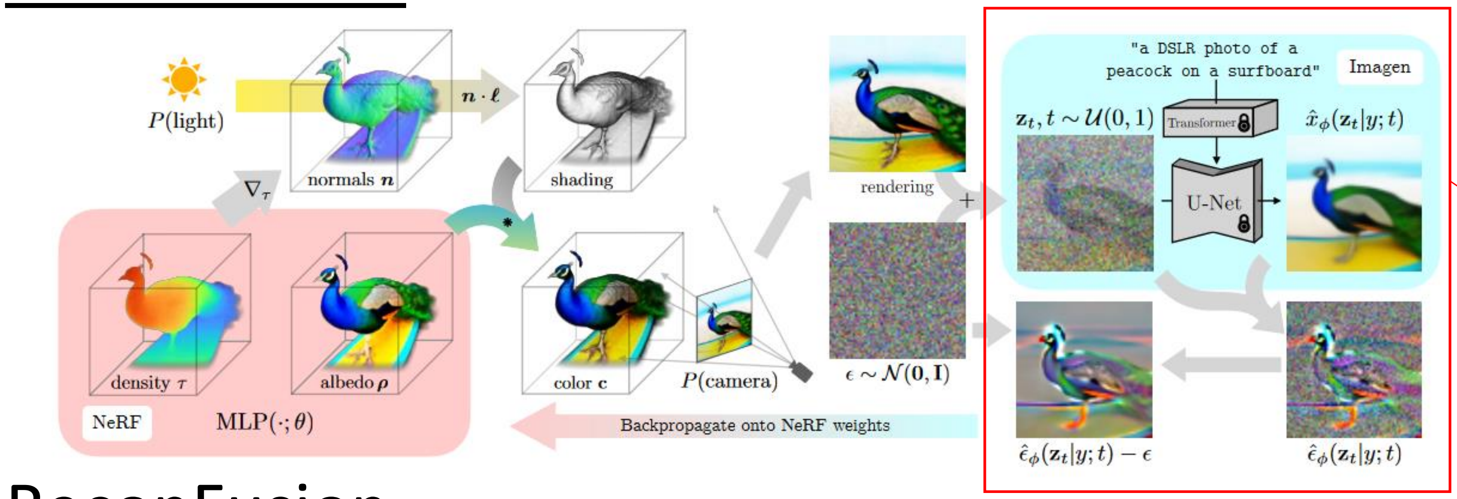
Monocular 360° GS

Radiance Field

- Input: Single (or few) omnidirectional 360° image(s)
- Problem formulation aligns with sparse input 3DGS/NeRF

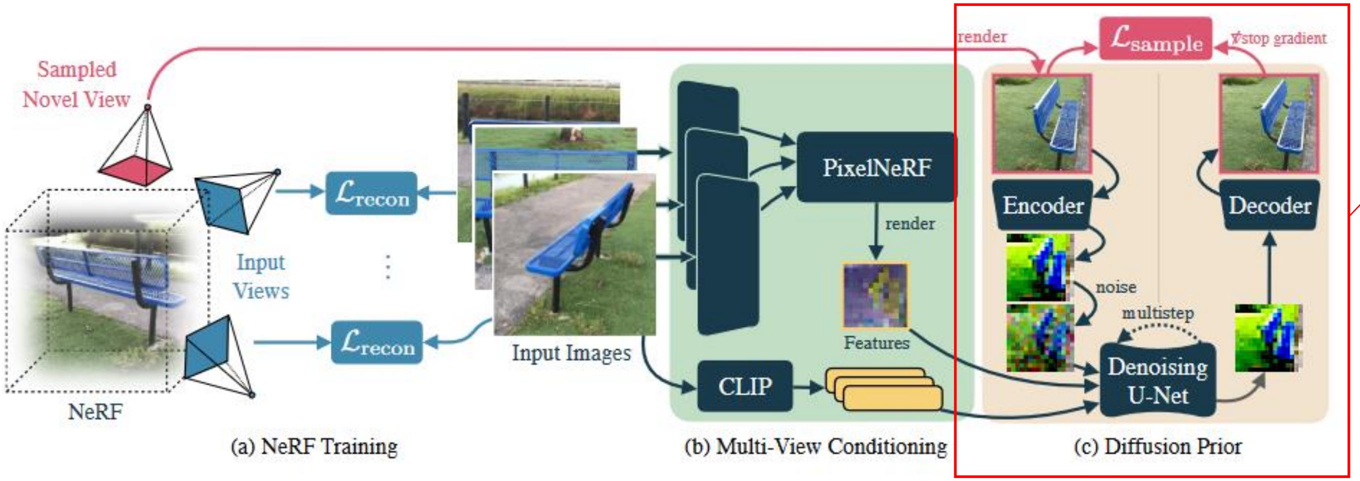
Related Work

- DreamFusion



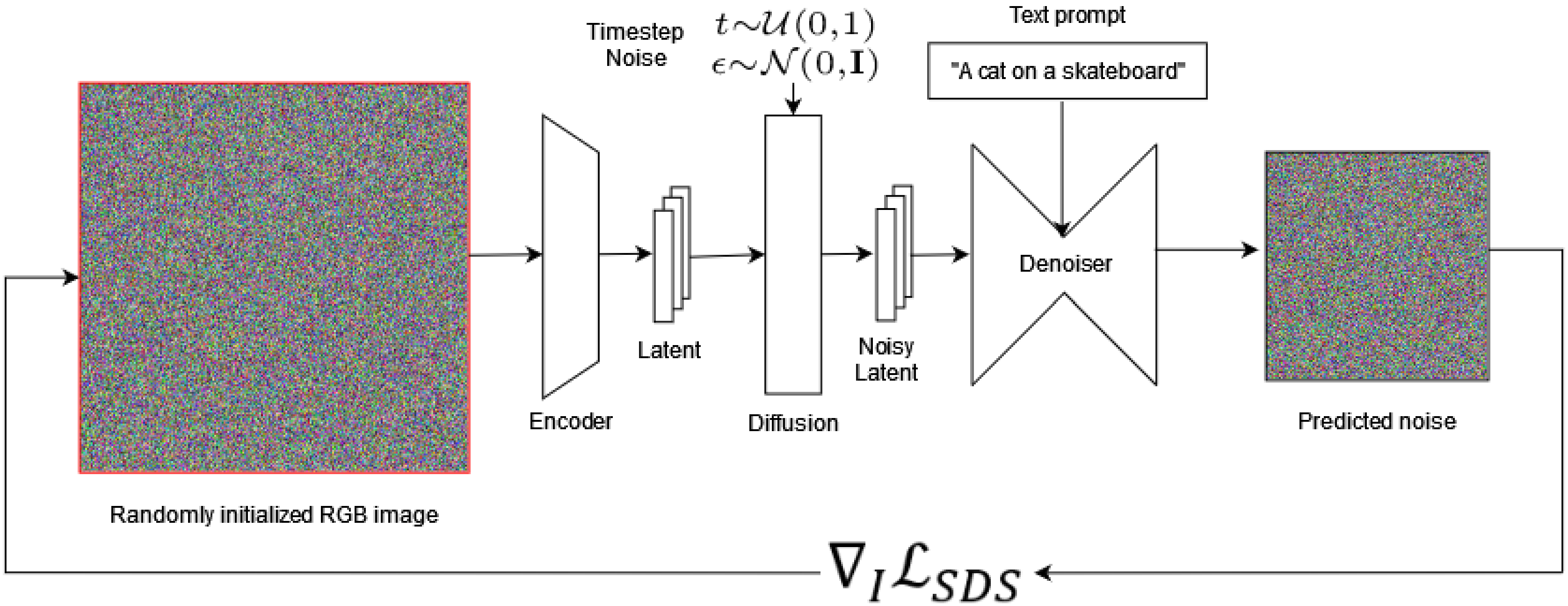
P(light)
$n \cdot \ell$
normals n
shading
∇_τ
density τ
albedo ρ
NeRF
MLP$(\cdot;\theta)$
color c
P(camera)
rendering
$\epsilon \sim \mathcal{N}(0, I)$
+
"a DSLR photo of a peacock on a surfboard"
Imagen
$z_t, t \sim \mathcal{U}(0,1)$
Transformer
U-Net
$\hat{x}_\phi(z_t|y;t)$
$\hat{\epsilon}_\phi(z_t|y;t) - \epsilon$
$\hat{\epsilon}_\phi(z_t|y;t)$
Backpropagate onto NeRF weights

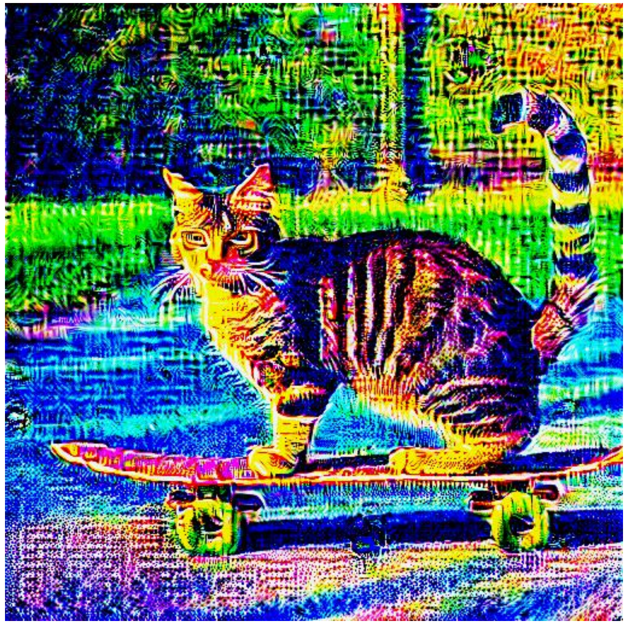
- ReconFusion



Guidance with diffusion models

Diffusion Model Guidance

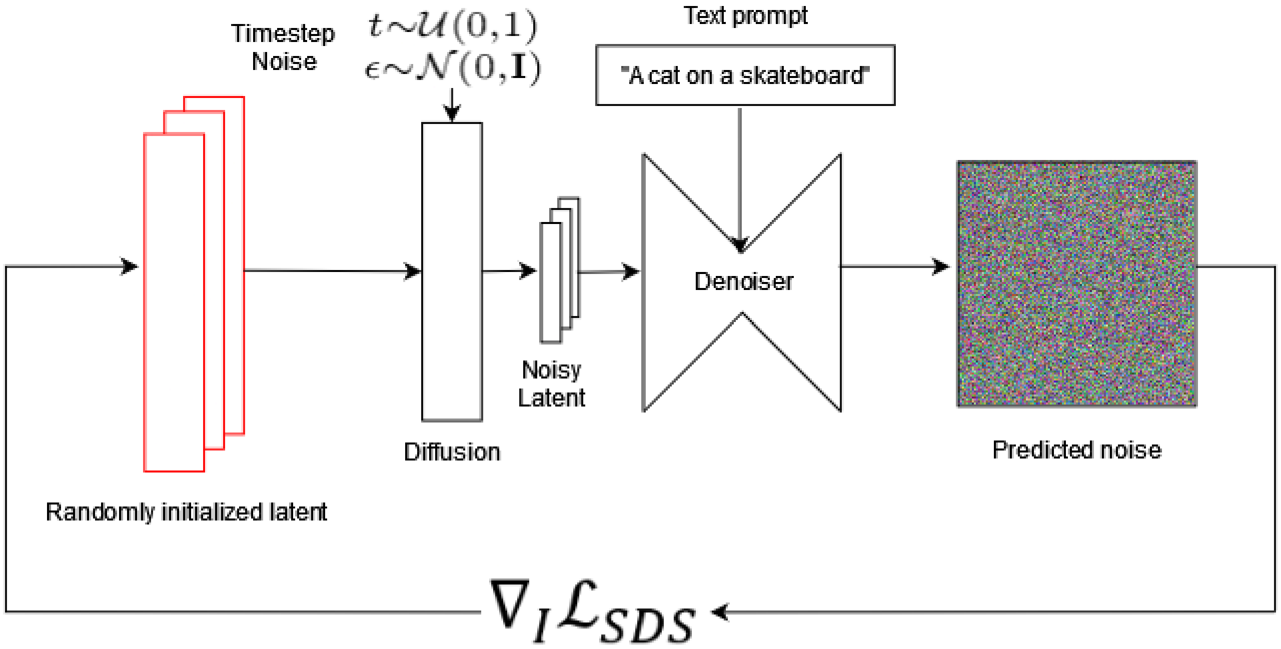

Timestep
Noise
$t \sim \mathcal{U}(0,1)$
$\epsilon \sim \mathcal{N}(0,\mathbf{I})$
Text prompt
"A cat on a skateboard"
Randomly initialized RGB image
Encoder
Latent
Diffusion
Noisy Latent
Denoiser
Predicted noise
$\nabla_I \mathcal{L}_{SDS}$

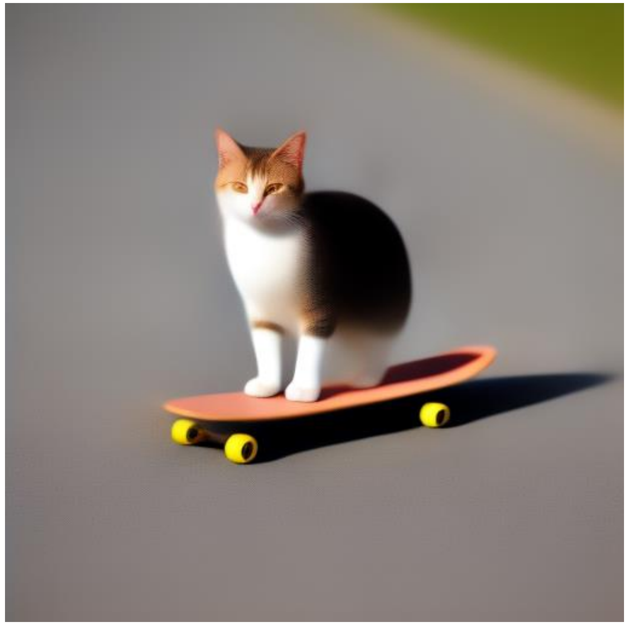


Results after 1000 iterations

- Optimize over randomly initialized RGB image directly
- Diffusion guidance with SDS loss conditioned on text prompt
- Issue: High saturation

Diffusion Model Guidance

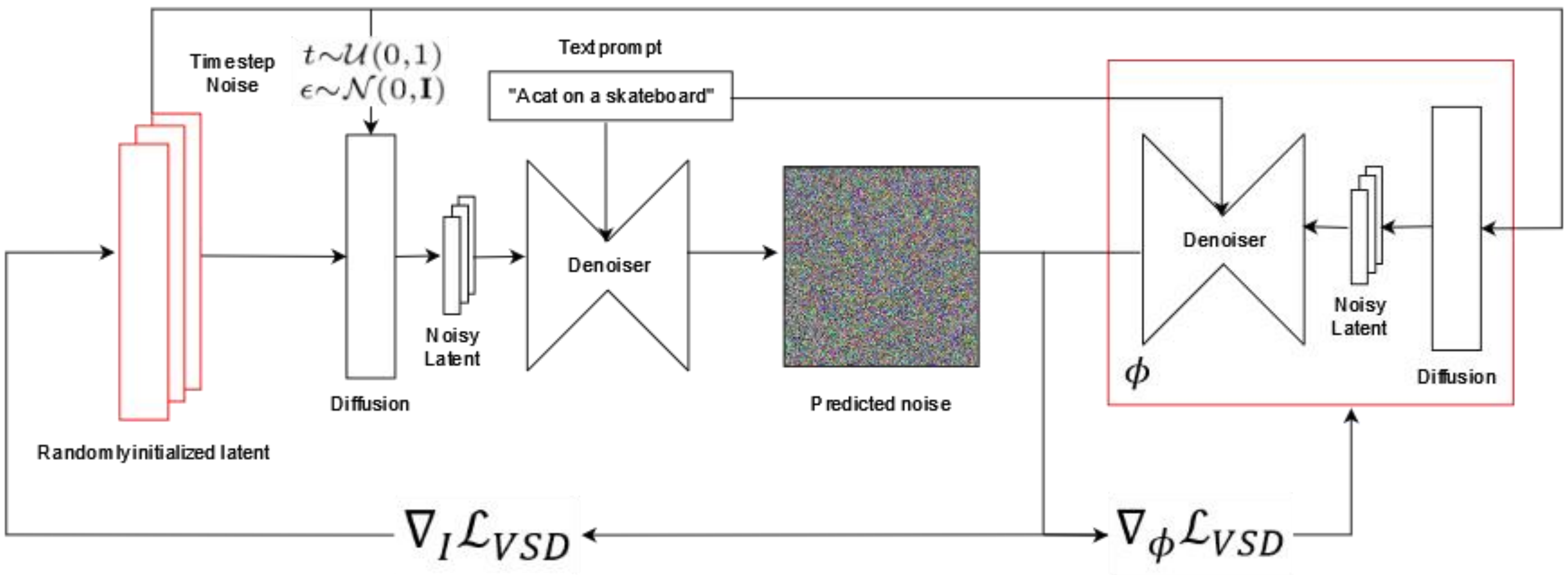

Timestep
Noise
$t \sim \mathcal{U}(0,1)$
$\epsilon \sim \mathcal{N}(0,\mathbf{I})$
Text prompt
"A cat on a skateboard"
Denoiser
Noisy Latent
Diffusion
Predicted noise
Randomly initialized latent
$\nabla_I \mathcal{L}_{SDS}$



Results after 1000 iterations

- Optimize over randomly initialized latent
- Diffusion guidance with SDS loss conditioned on text prompt
- Issues: Overly smoothed, missing details

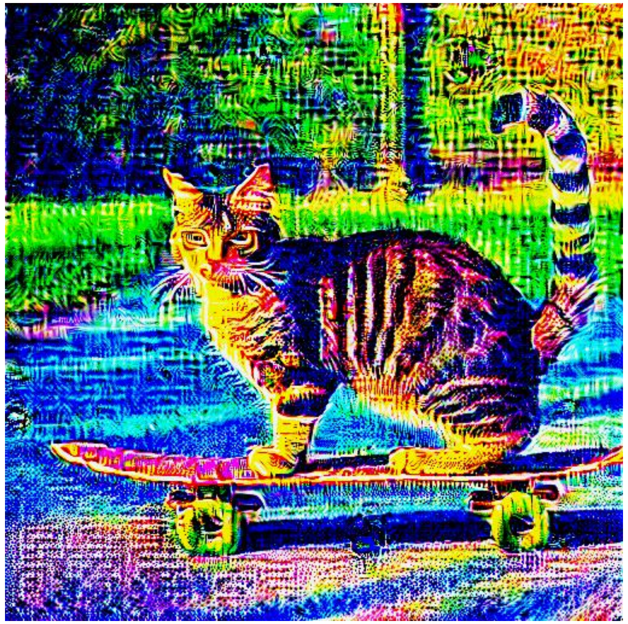
Diffusion Model Guidance


Timestep
Noise
$t \sim \mathcal{U}(0,1)$
$\epsilon \sim \mathcal{N}(0,\mathbf{I})$
Text prompt
"A cat on a skateboard"
Denoiser
Noisy Latent
Diffusion
Predicted noise
Randomly initialized latent
Denoiser
Noisy Latent
Diffusion
ϕ
$\nabla_I \mathcal{L}_{VSD}$
$\nabla_\phi \mathcal{L}_{VSD}$

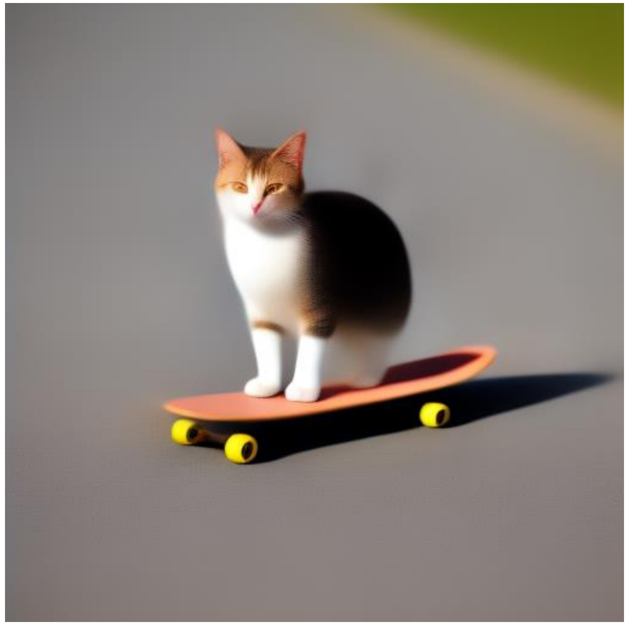
Results after 250 iterations

- ProlificDreamer introduces VSD loss
- Optimize over randomly initialized latent
- Diffusion guidance with VSD loss conditioned on text prompt

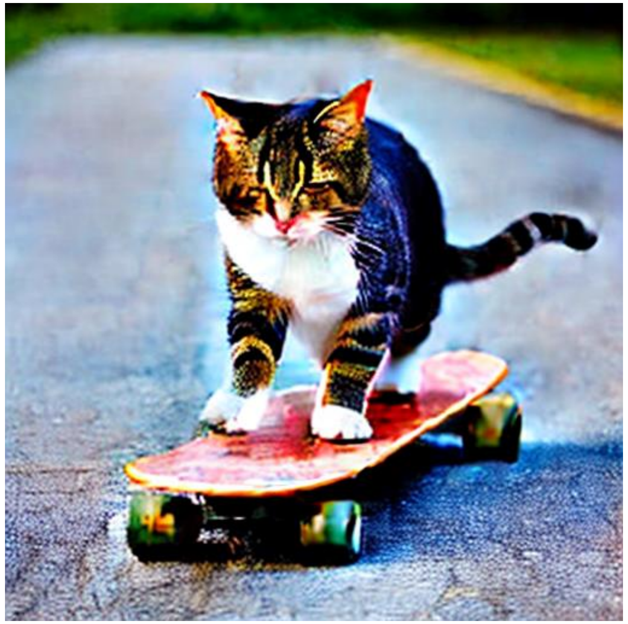
SDS vs VSD



SDS on RGB
1000 iterations

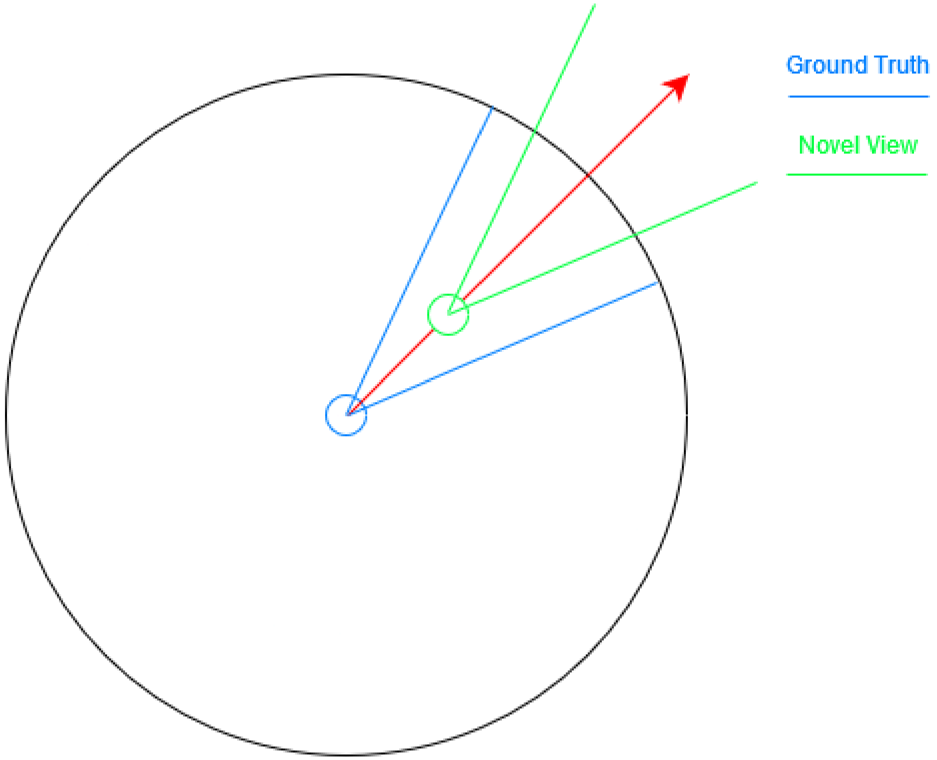


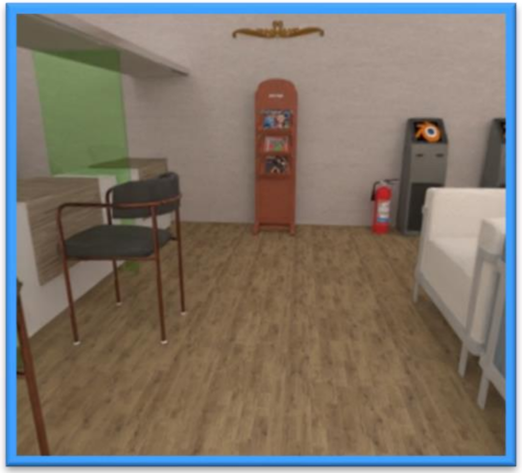
SDS on latent
1000 iterations

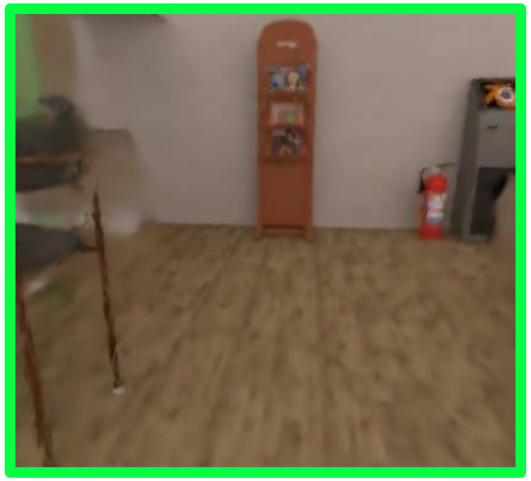


VSD on latent
250 iterations

Monocular 360° GS



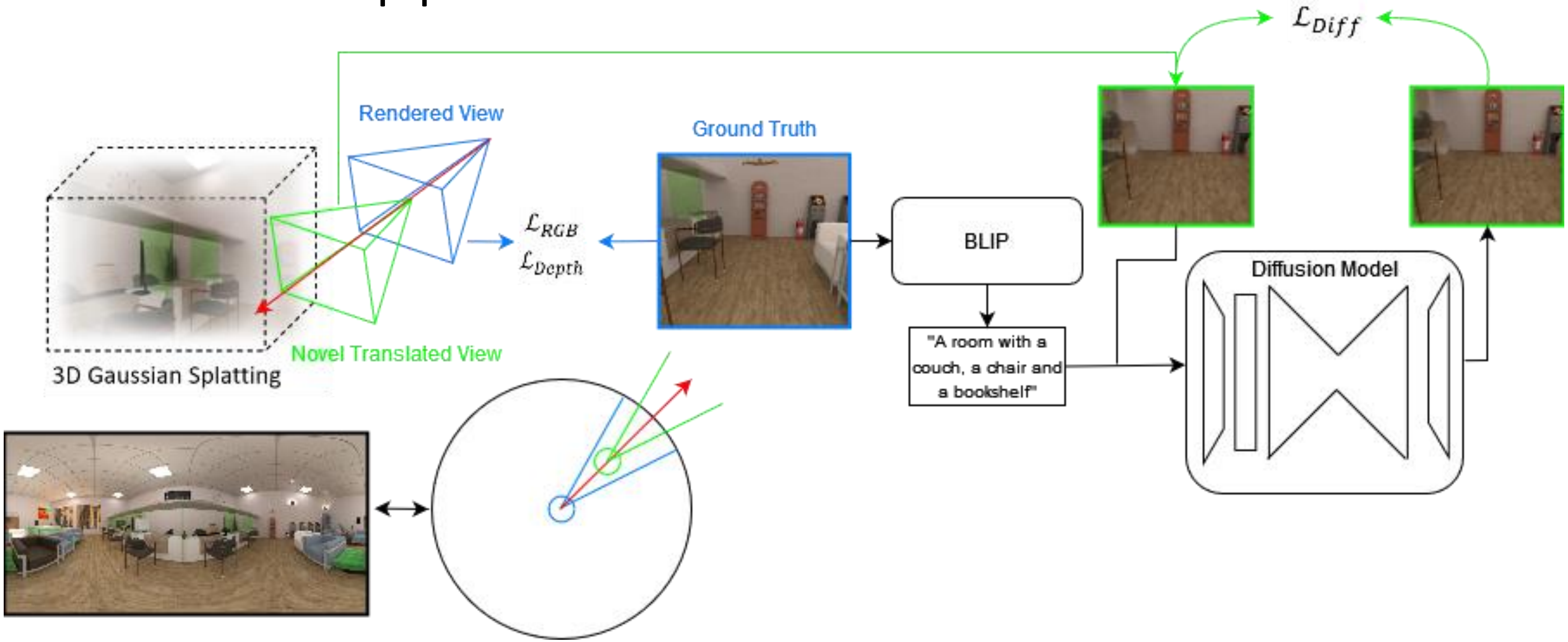




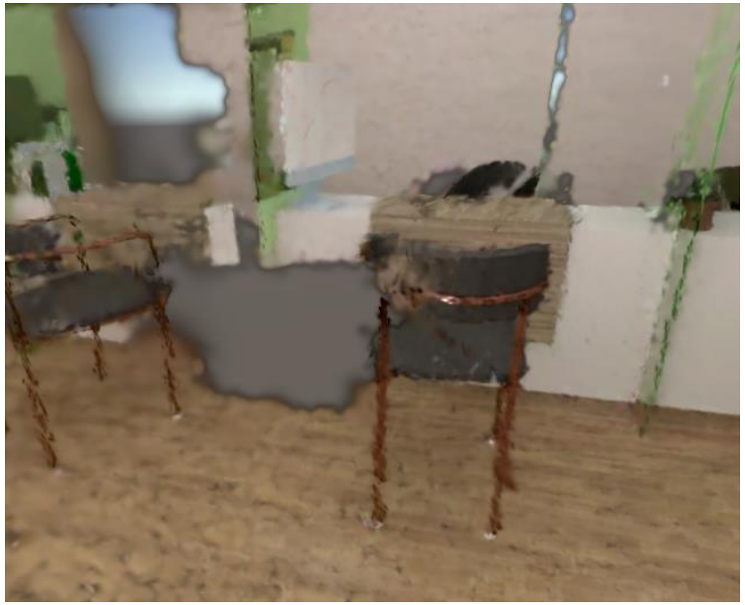
- Sampling of multiple ground truth images for standard 3DGS RGB and Depth loss
- Novel views created by translating virtual camera in viewing direction for diffusion (SDS/VSD) loss
- Use ground truth image for image captioning model for additional text conditioning

Monocular 360° GS

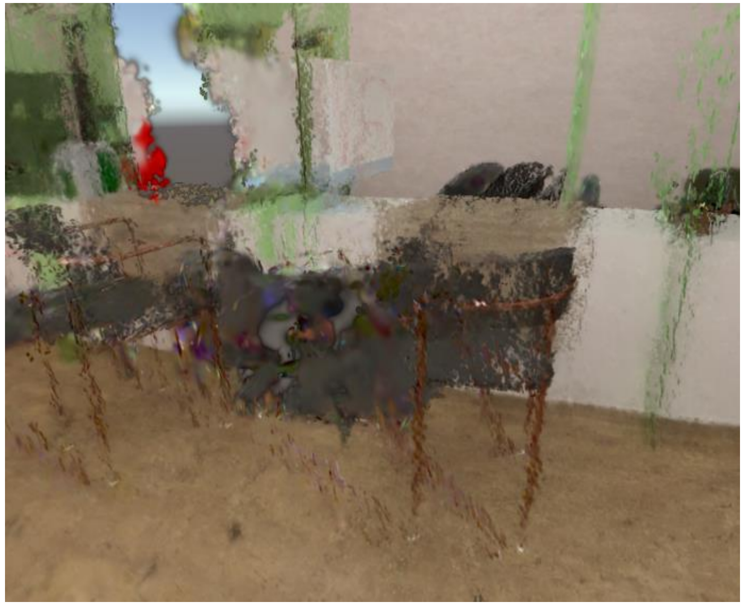
- First iteration of pipeline:



Monocular 360° GS: Standard 3DGS



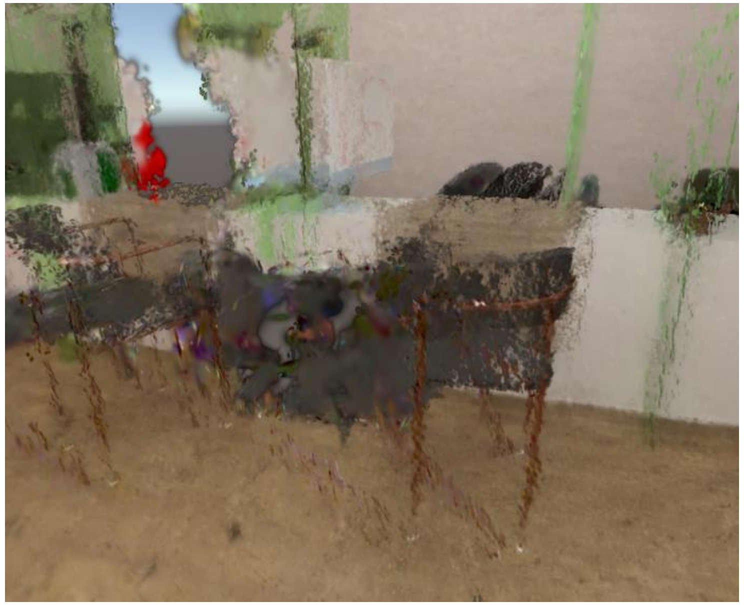
1000 iterations



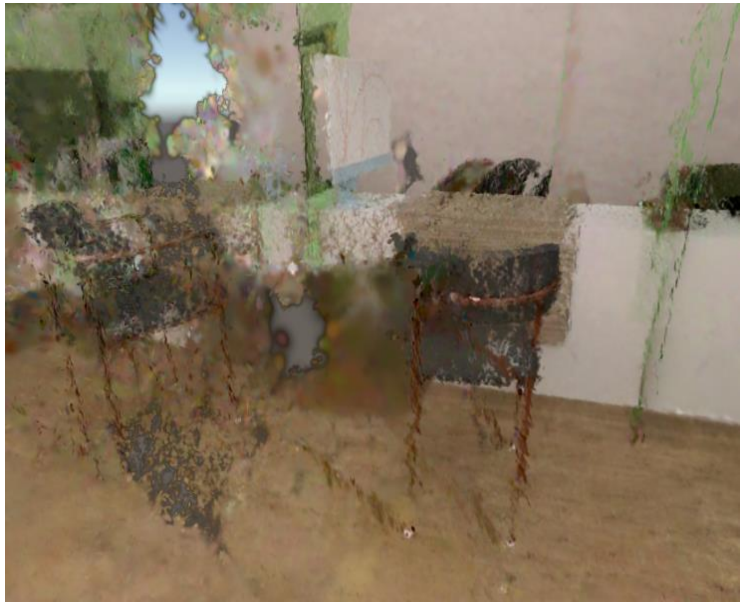
7000 iterations

- Training on ground truth data only: Smearing effect
- Objects which occlude holes, are essentially splatted over missing regions, due to the lack of view coverage in those areas

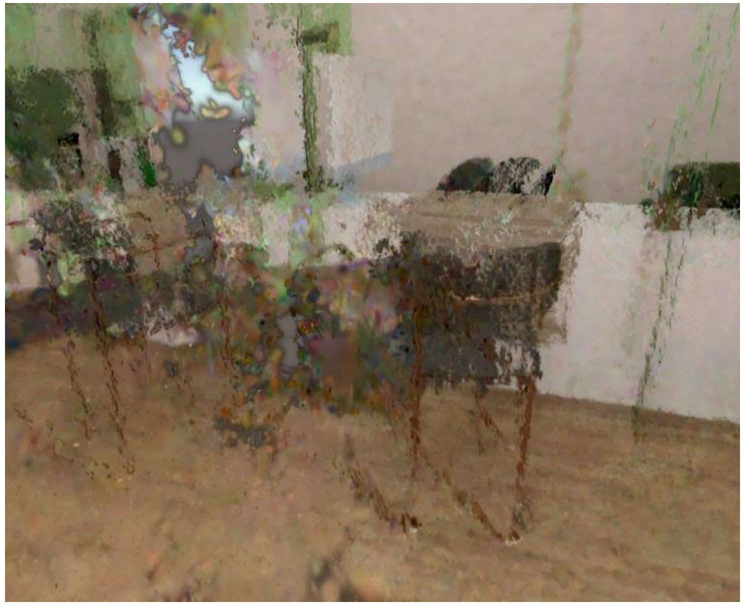
Diffusion and Text conditioning



No diffusion



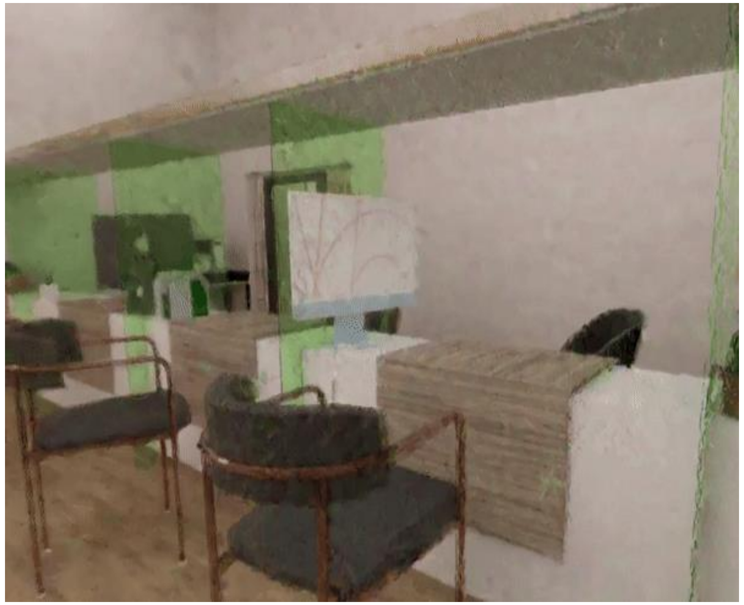
SDS with no text conditioning



SDS with text conditioning

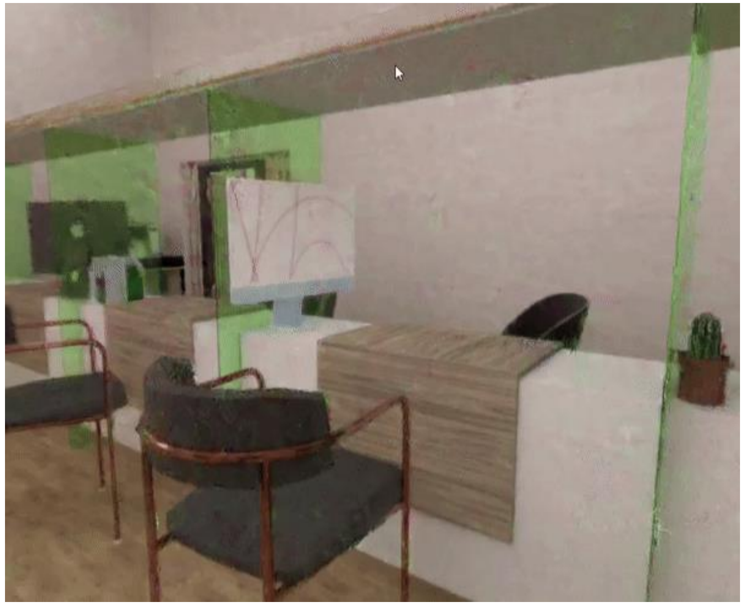
- Training with diffusion guidance (7000 iterations each)
- Less smearing effect with diffusion guidance but more noise

Difference in losses



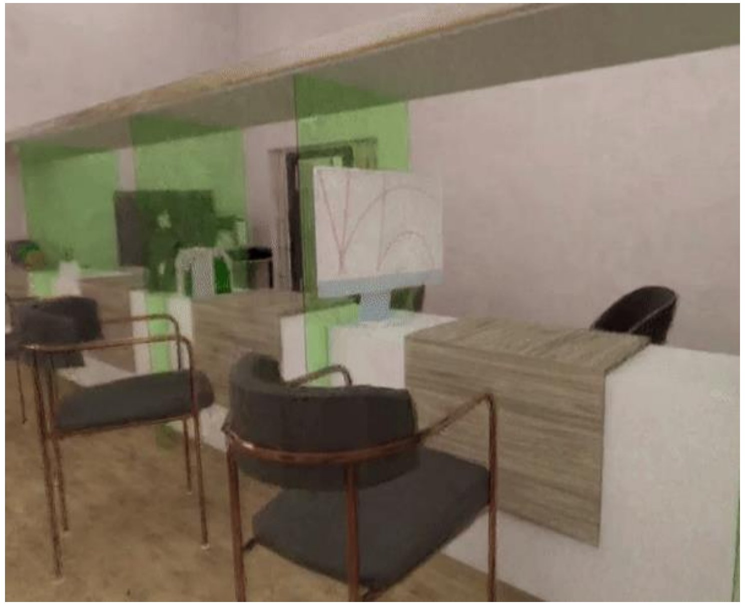
SDS loss

$\mathcal{L}_{\text{Diff}}(\phi, \mathbf{x}) = \mathbb{E}_{t \sim \mathcal{U}(0,1), \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \left[w(t) \| \epsilon_\phi(\alpha_t \mathbf{x} + \sigma_t \epsilon; t) - \epsilon \|_2^2 \right]$



VSD loss

$\mathcal{L}_{\text{VSD}}(\theta) \triangleq \mathbb{E}_{t,\epsilon,c} \;\; \omega(t) \left(\epsilon_{\text{pretrain}}(\boldsymbol{x}_t, t, y^c) - \epsilon_\phi(\boldsymbol{x}_t, t, c, y) \right)$

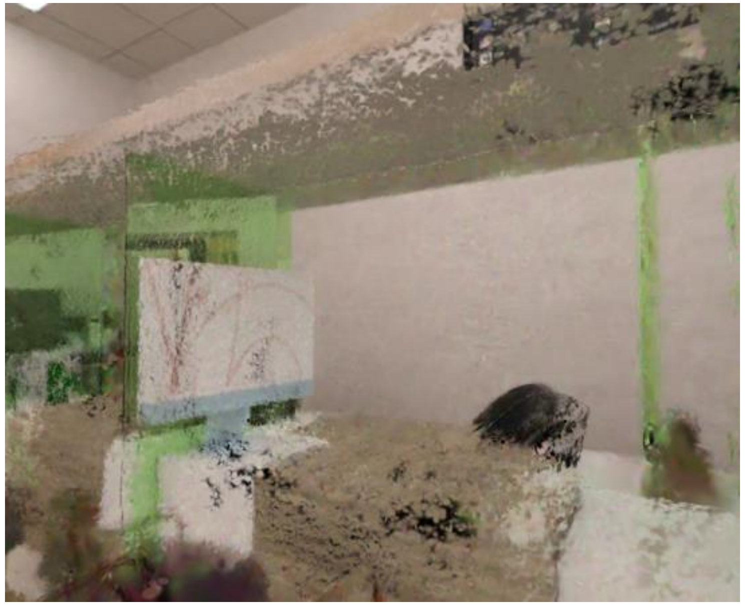


Multi-step denoising loss

$\mathcal{L} = \mathbb{E}_{\boldsymbol{\pi},\epsilon,t} \left[w_t || f_\theta(\boldsymbol{\pi}) - \hat{\boldsymbol{x}}_{0,\mathcal{T}} ||^2 + \mathcal{L}_{Perp}(f_\theta(\boldsymbol{\pi}), \hat{\boldsymbol{x}}_{0,\mathcal{T}}) \right]$

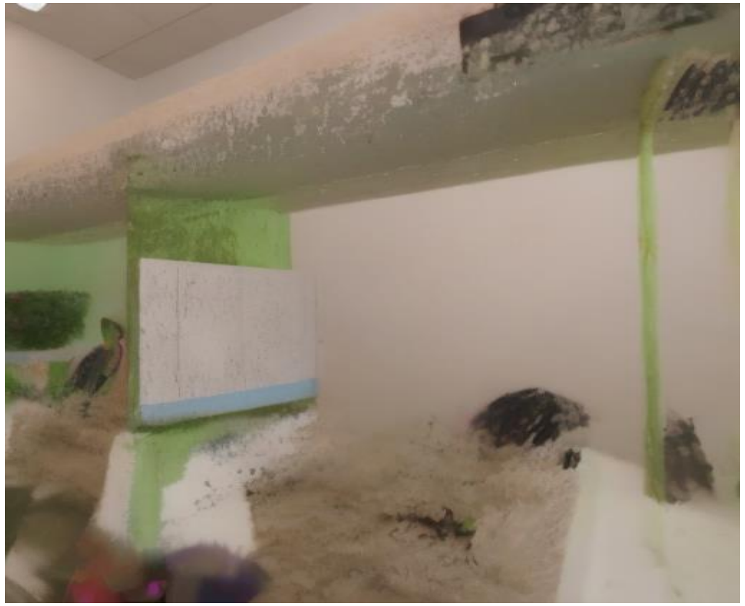
- Forward translation with random scaling
- Text conditioning
- Multi-step: t uniformly sampled and linear weighting decay

Multi-step denoising



Before diffusion

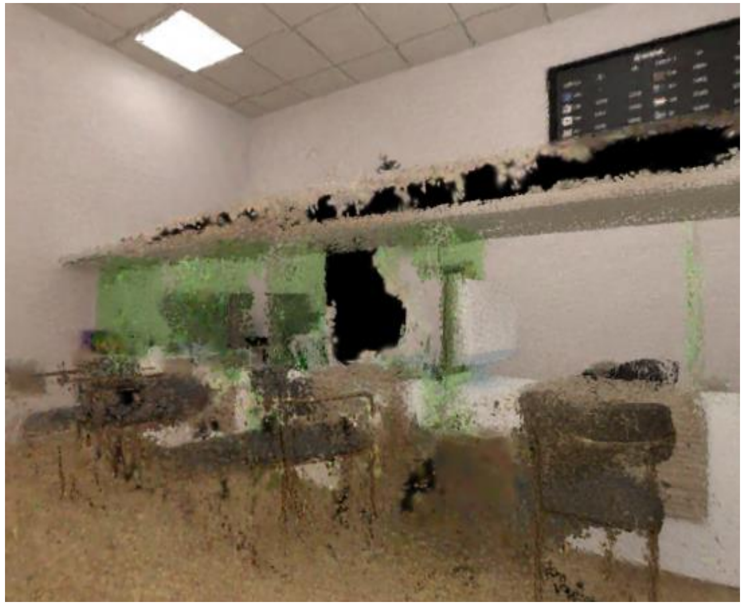
Multi-step denoising



After denoising

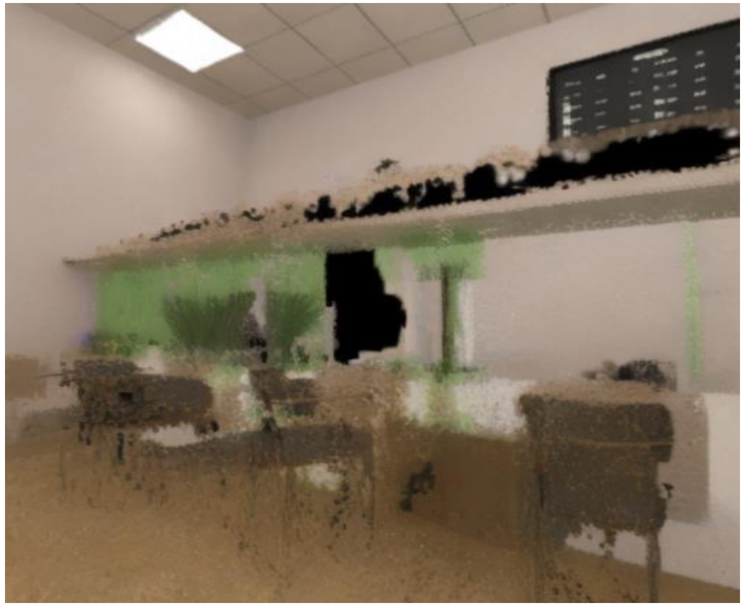
- ReconFusion and SparseFusion denoise input image for k uniformly sampled timesteps t, instead of denoising in one step as in SDS
- Loss calculated in pixel space in addition with perceptual loss

Multi-step denoising: Open Issues



Before diffusion

Multi-step denoising

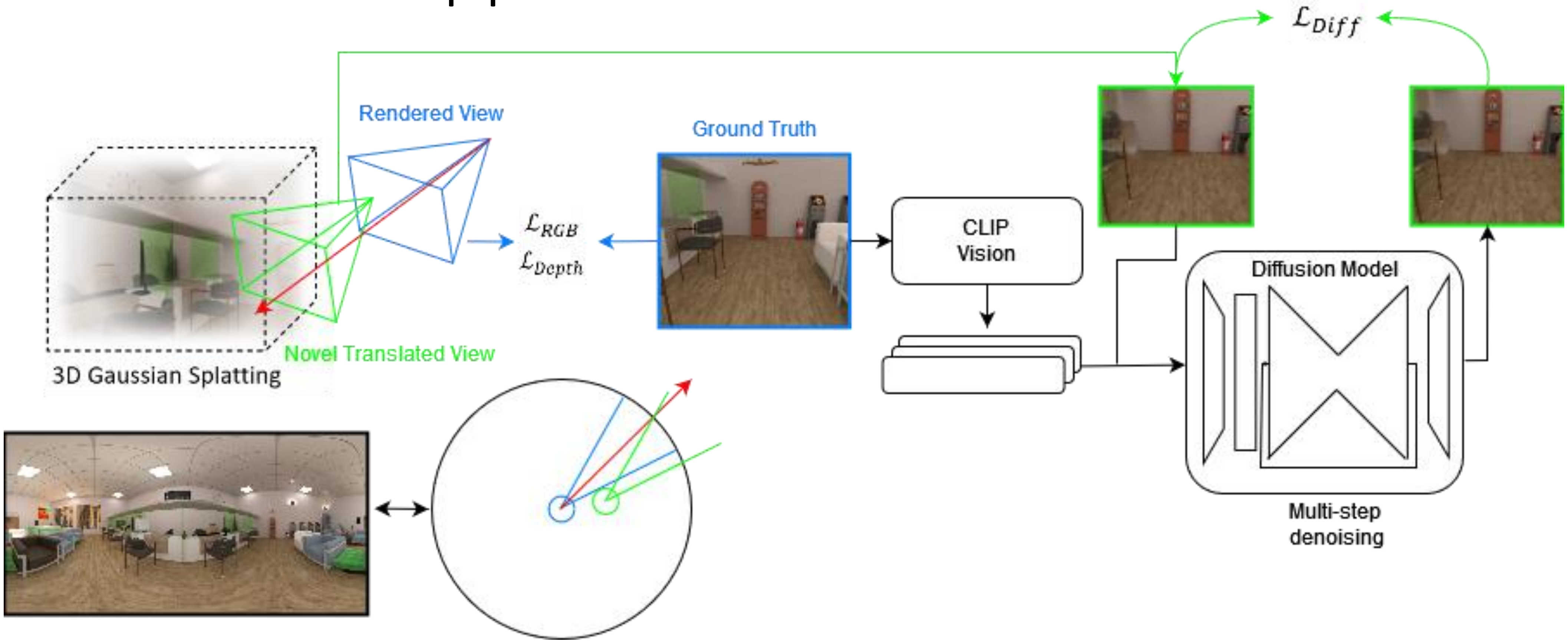


After denoising

- Diffusion process can’t fix large holes in the scene
- If added noise is large, model hallucinates more

Monocular 360° GS

- Second iteration of pipeline:



Multi-step denoising: Different parameters

t uniform with linear decay (0-1)
Weight loss decay (0.1 – 1 - 0.1)

t uniform with linear decay (1-0)
Weight loss decay (1 - 0.1)

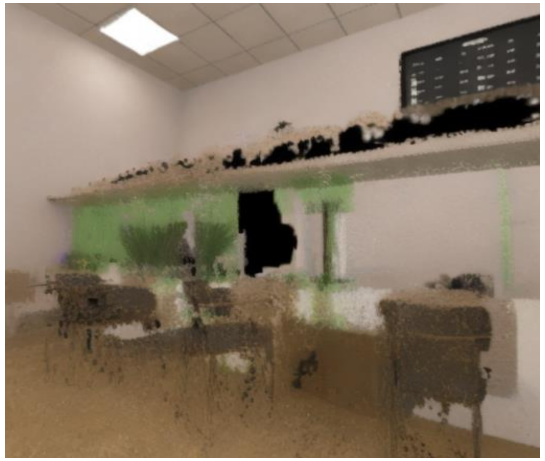
t uniform with linear decay (1-0)
Weight loss decay (1 - 0.1)
Translation in all directions

- Large noise at the end of training not as good as smaller noise at the end

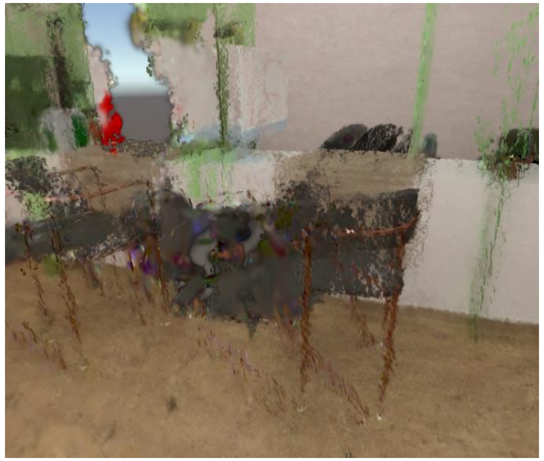
Monocular 360° GS: First Review

- Encountered problems with 3DGS approach:
  1. Low coverage regions introduce holes in 3D representation
     - Diffusion can’t fix areas with large holes
  2. Model overfits on ground truth data
     - Smearing effect

1

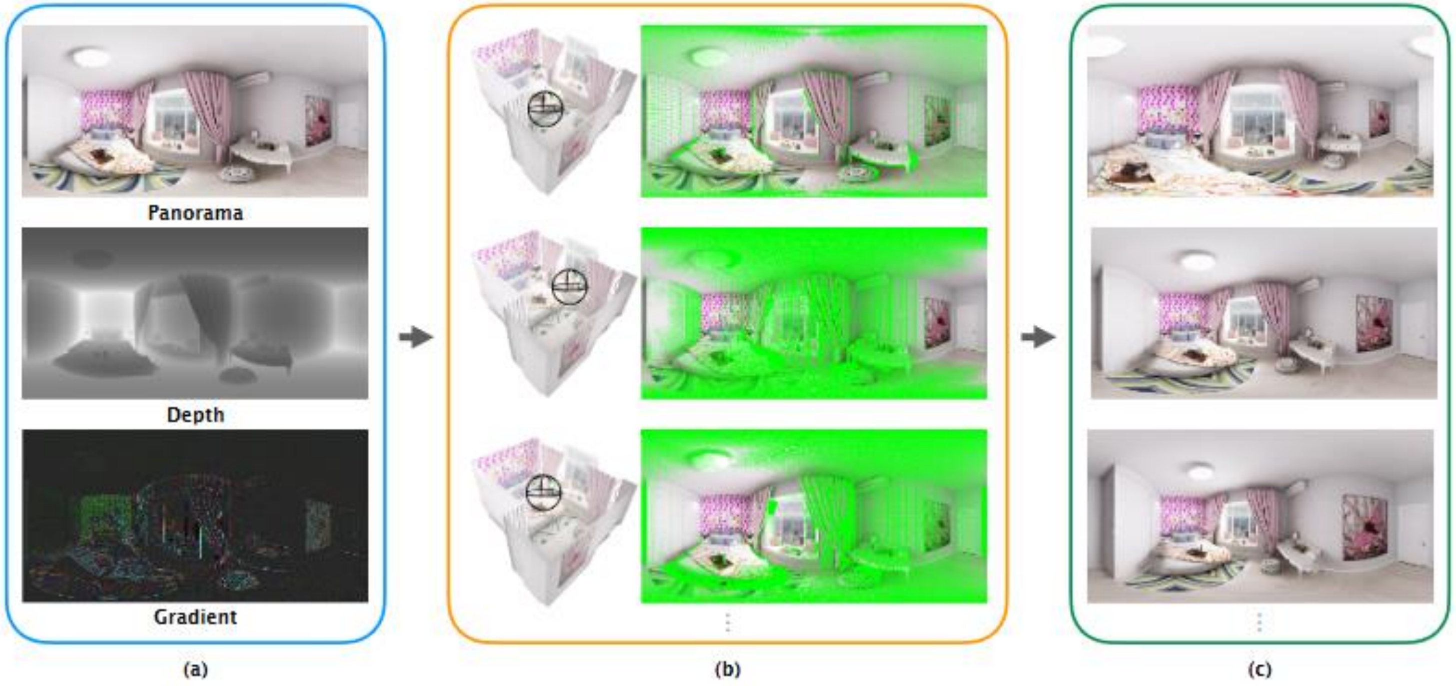


2



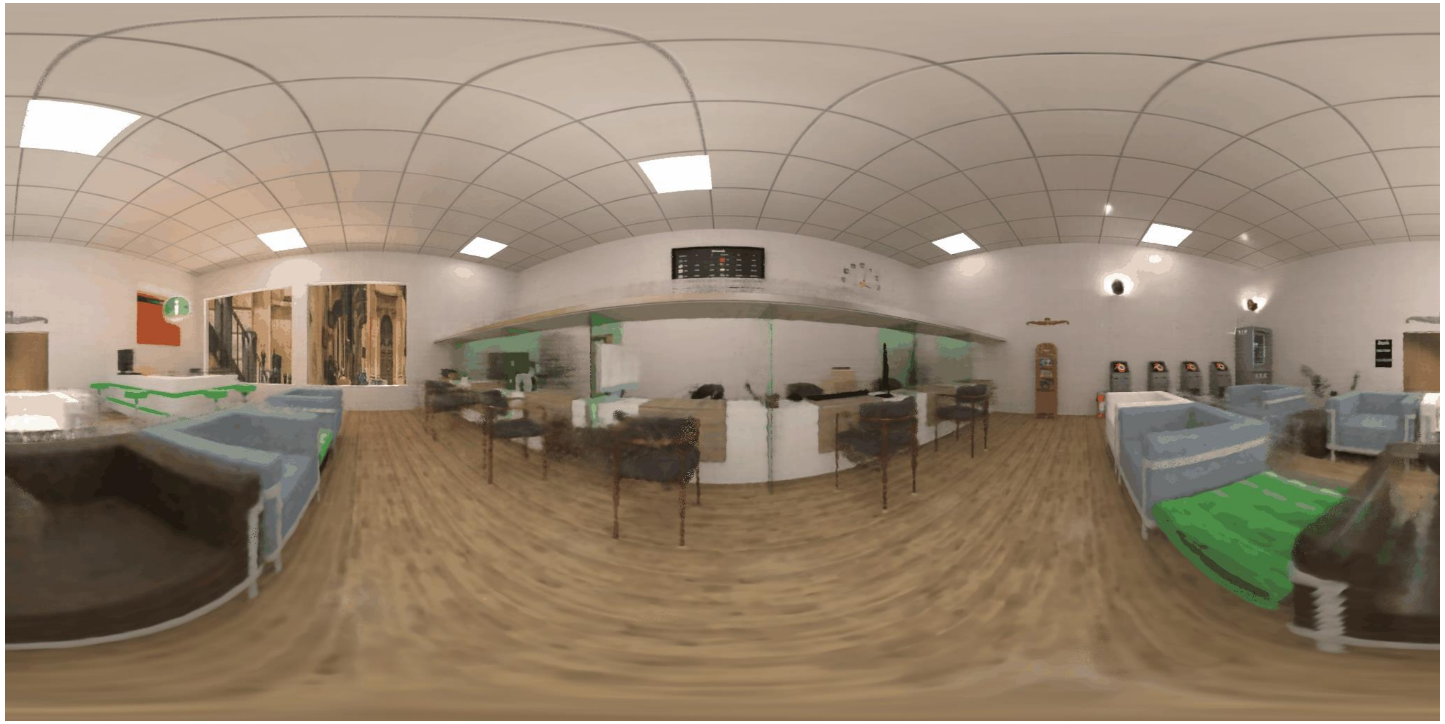
- New approach: Test if NeRF encounters similar problems

OmniNeRF



- OmniNeRF for novel view synthesis on single equirectangular image
- Synthesizes new panoramic views by projecting pixels to novel views
- Makes use of MLP pixel-based property for incomplete appearance

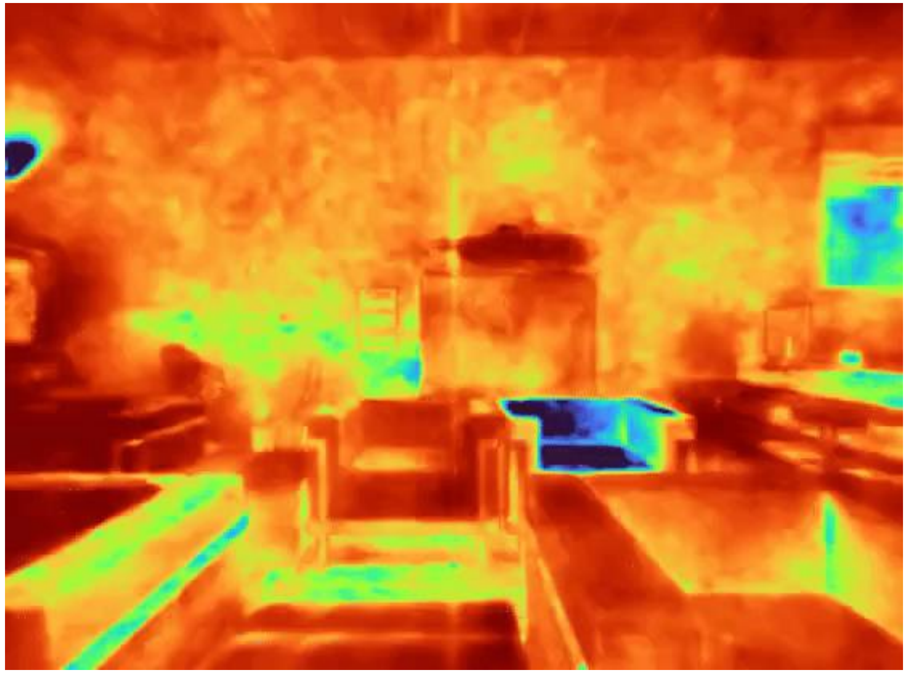
OmniNeRF: Results



- Similar issues as with 3DGS, OmniNeRF fills low coverage regions with objects occluding them
- Some areas remain transparent

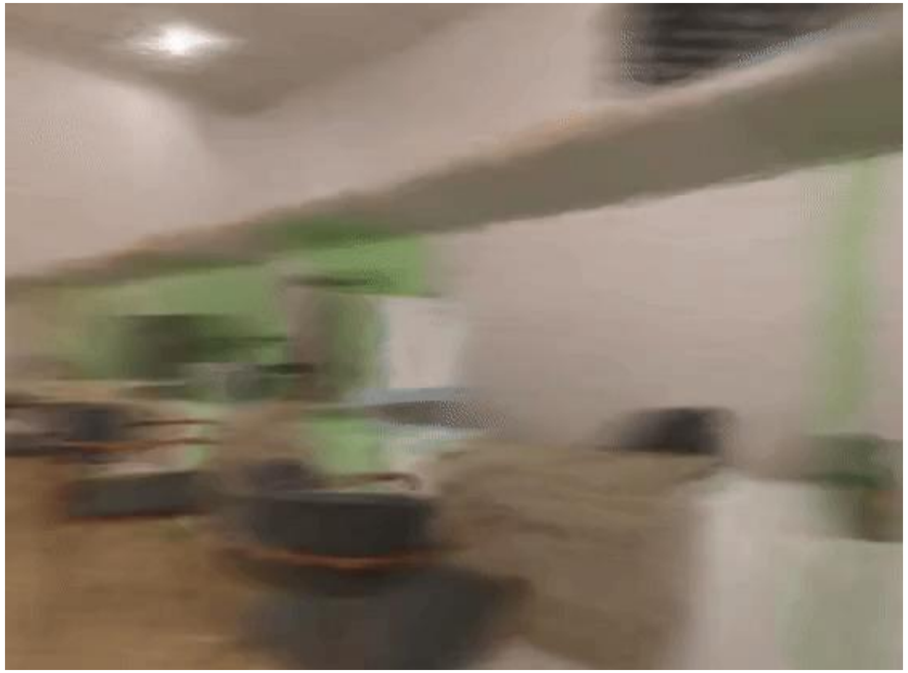
Monocular 360° Zip-NeRF

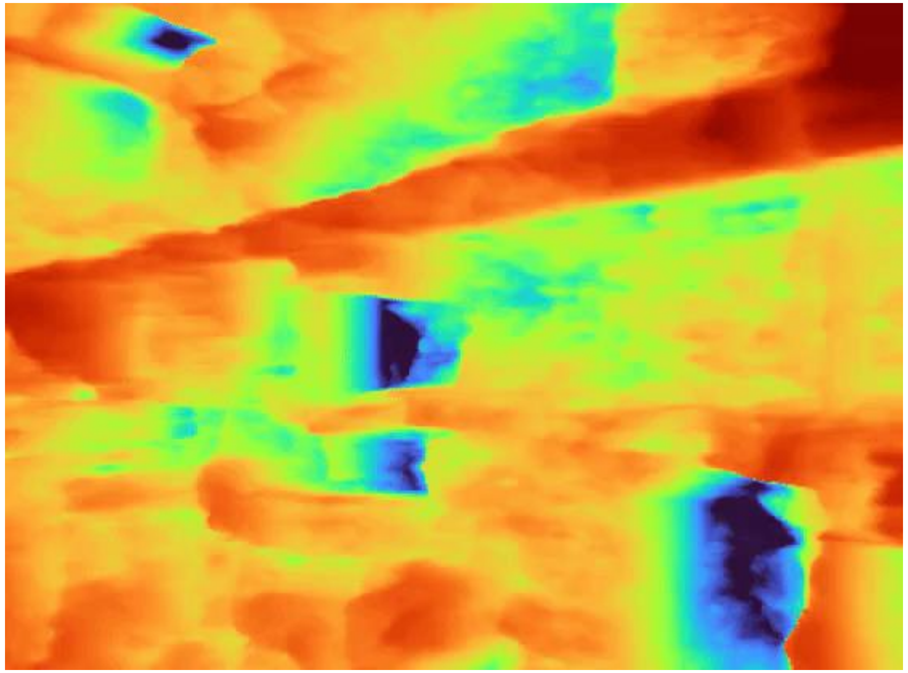
- Underlying 3D representation backbone: Zip-NeRF
  - Combination of iNGP and mip-NeRF 360
  - Utilized by ReconFusion
- Training on ground truth data only (20,000 iterations):



Monocular 360° Zip-NeRF

- Training on ground truth data only (20,000 iterations):

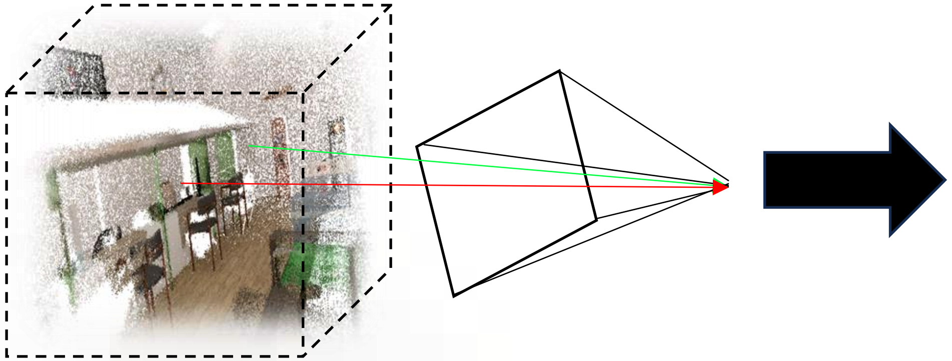




- Issues: Depth incorrect, Model has no idea how the scene looks like outside camera center

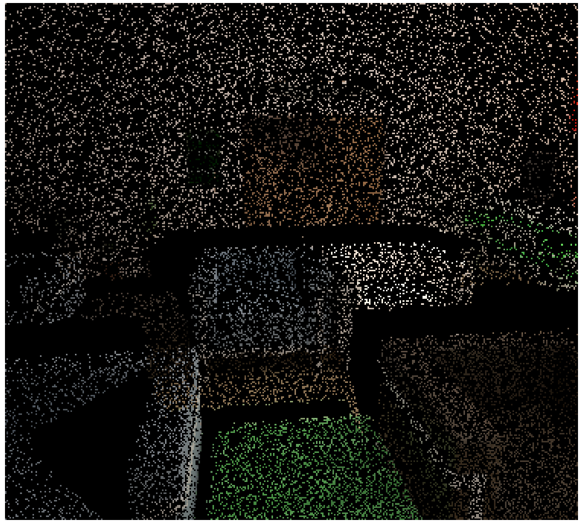
Partial Loss

- Motivation: Give Zip-NeRF more spatial awareness
- Reproject 3D points back to 2D for novel camera pose



Initial Point Cloud

Novel Camera

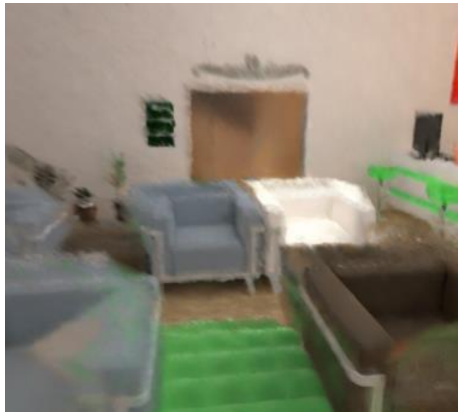


Partial Image +
Mask

Ground Truth View

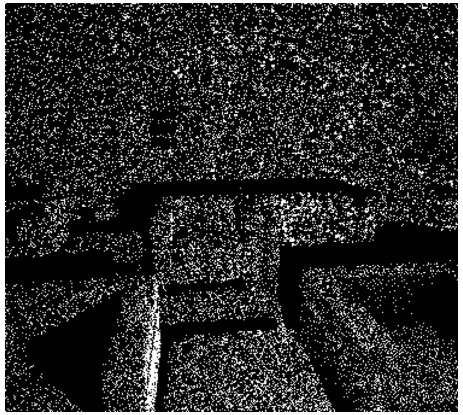
Partial Loss

- Mask rendered view from Zip-NeRF
- Color loss with novel masked rendered view and novel partial view



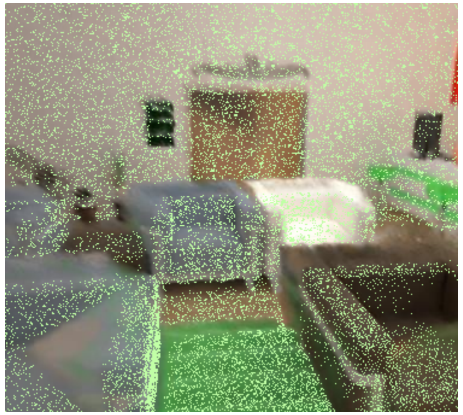
Novel Rendered View

+



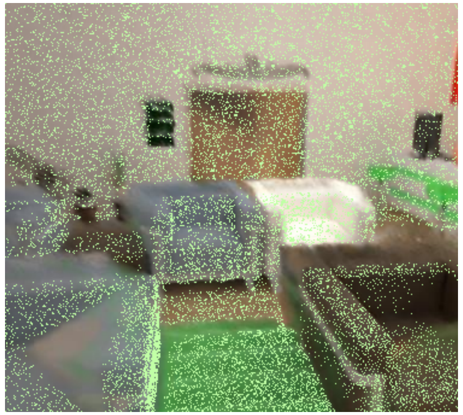
Novel View Mask

=

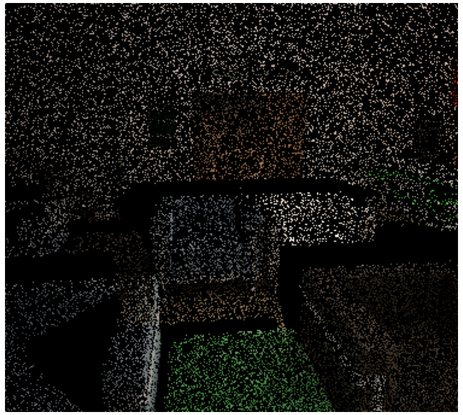


Novel Masked Rendered View

Partial Color Loss = (

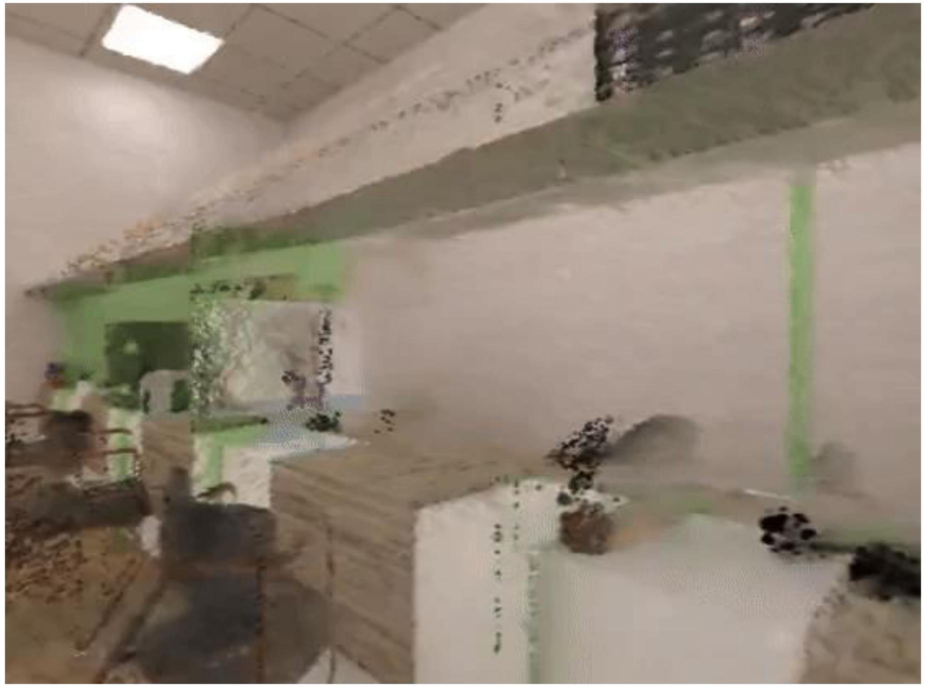


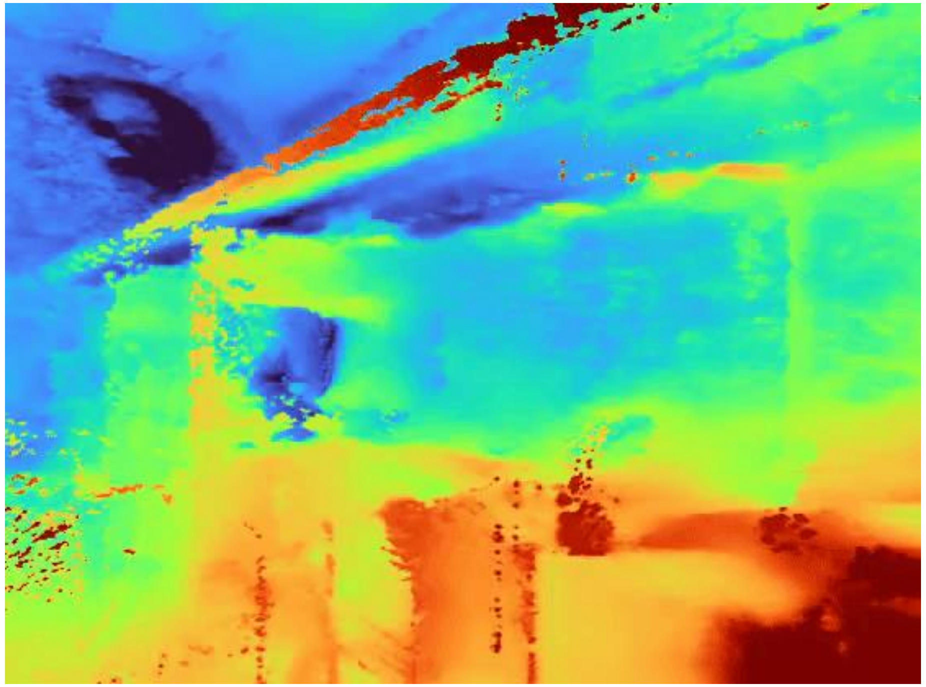
,



)

Partial Loss: Results

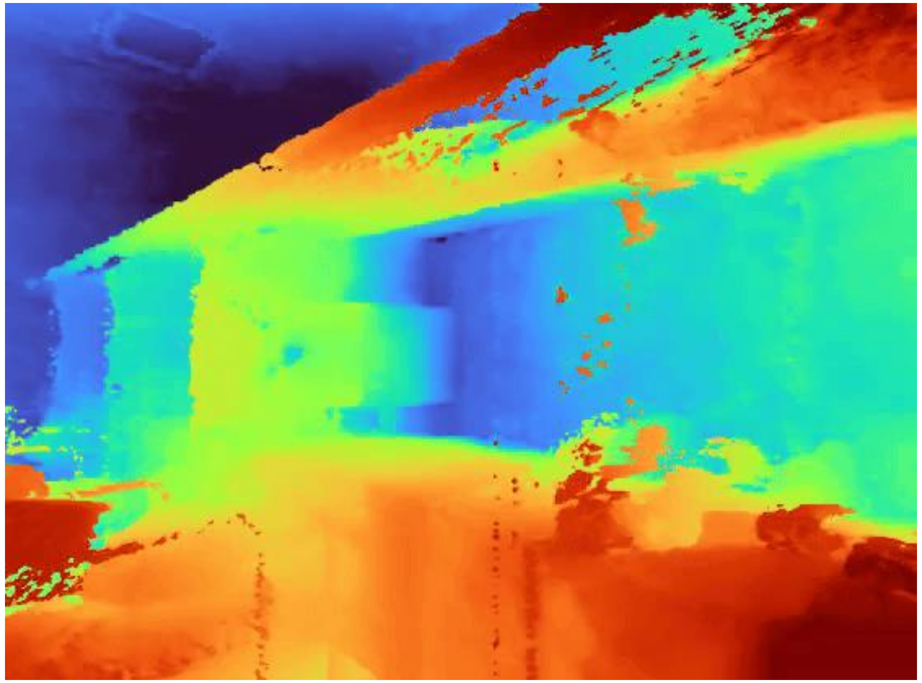




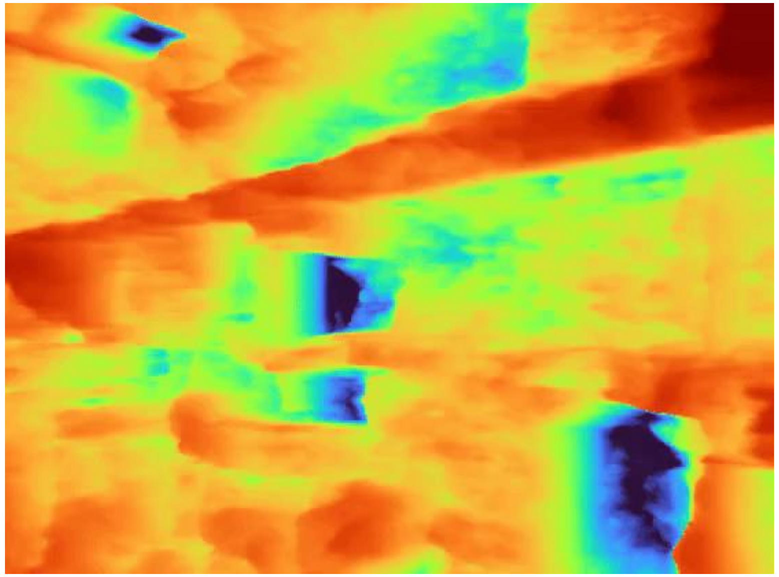
- Zip-NeRF has better spacial understanding of the scene now
- Issues: Smearing effect, Depth is broken

Depth Loss

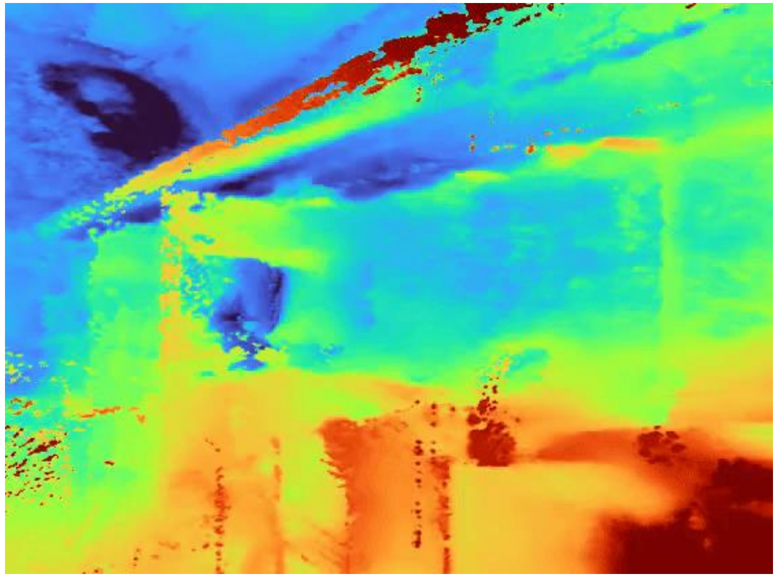
- Motivation: Leverage existing ground truth depth
- Integration of standard L1 loss between rendered and ground truth depth



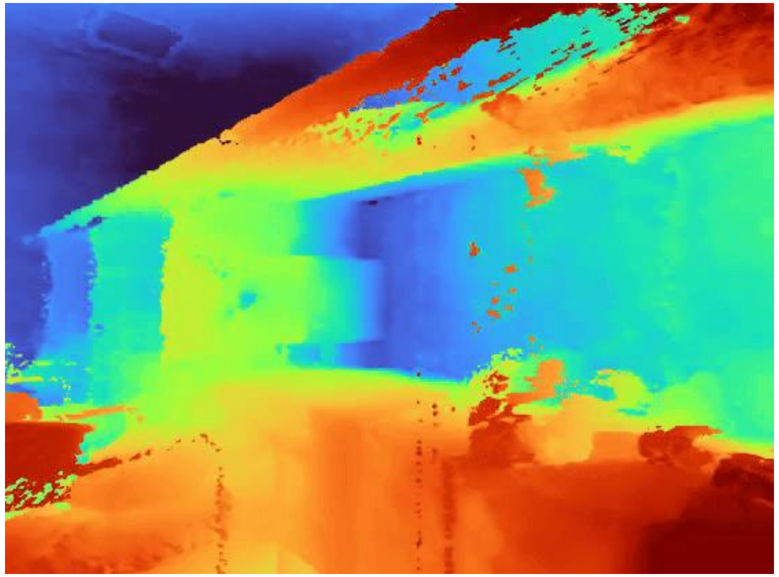
Depth Comparison



Ground Truth Views only



Ground Truth Views +
Partial Loss

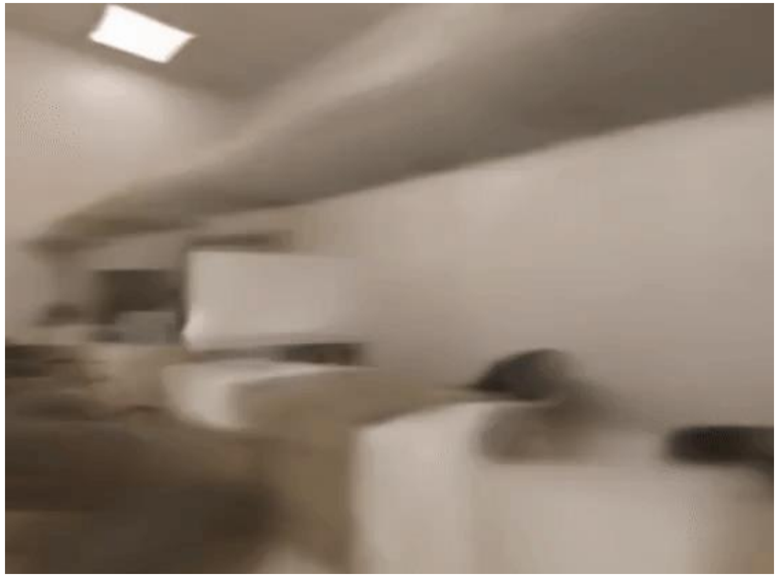


Ground Truth Views +
Partial Loss +
Depth Loss

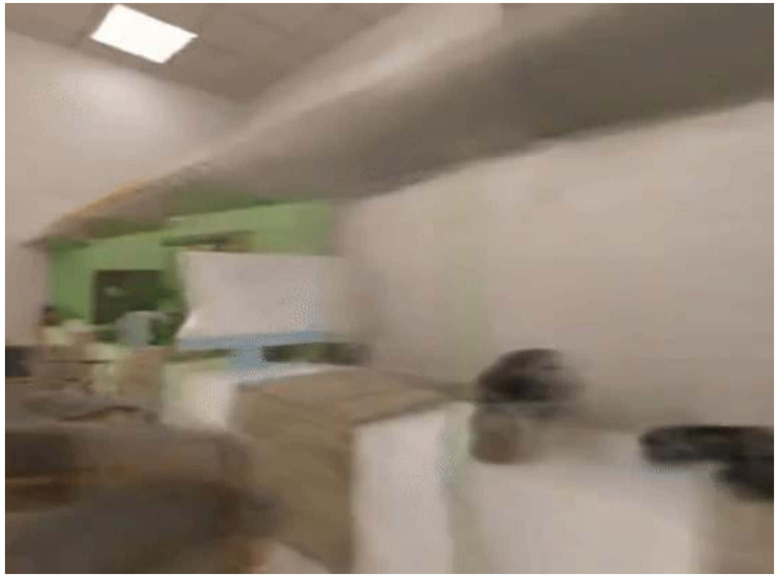
- Improvement in depth map quality

Diffusion Loss

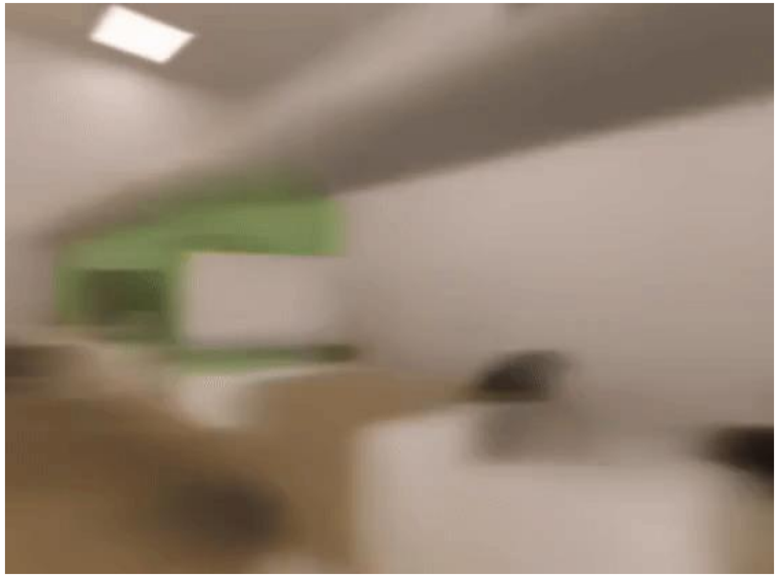
- Motivation: Reconstruct low coverage regions
- Applying SDS, VSD and Multi-step denoising loss



SDS Loss



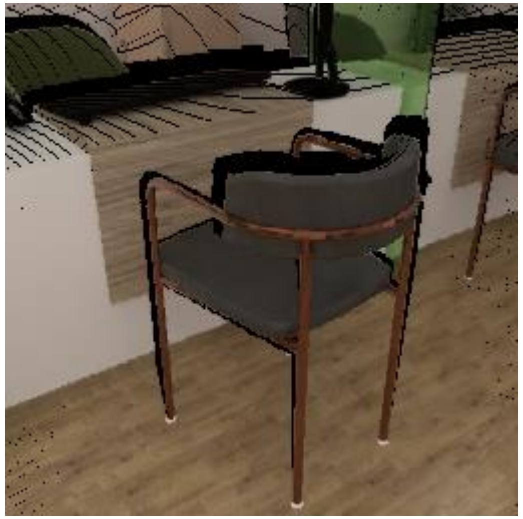
VSD Loss

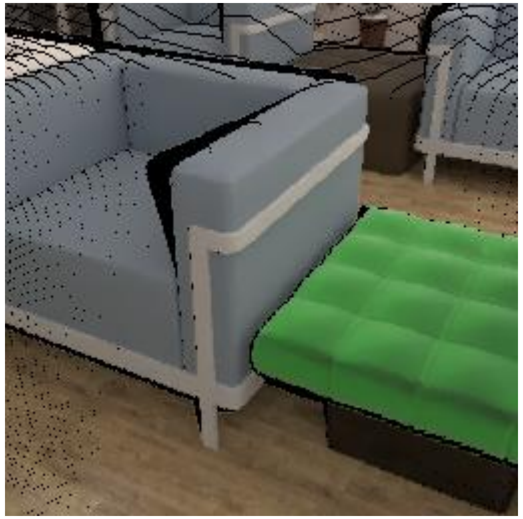


Multi-step denoising Loss

- Zip-NeRF Loss + Partial Loss + Depth Loss + Diffusion Loss

Revision: Low Coverage Regions

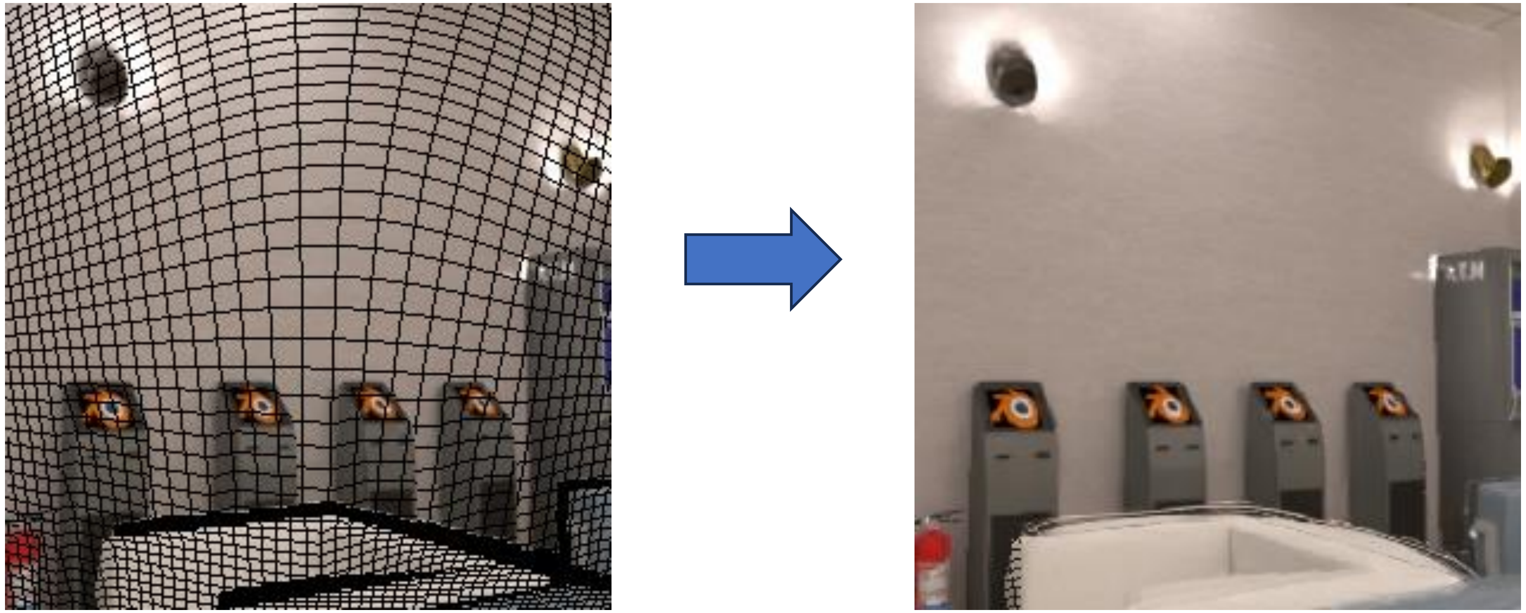




- Diffusion loss (SDS, VSD and Multi-step denoising) could not fix occluded regions, it only smoothed out noise
- Depth/Partial loss only helps with spacial awarness for know points

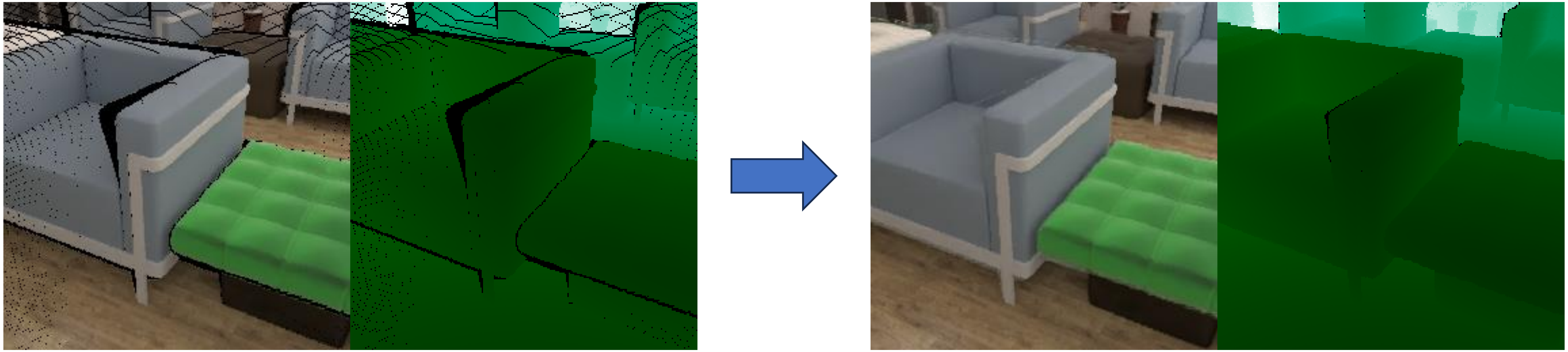
➡ Inpainting with Diffusion models

Inpainting



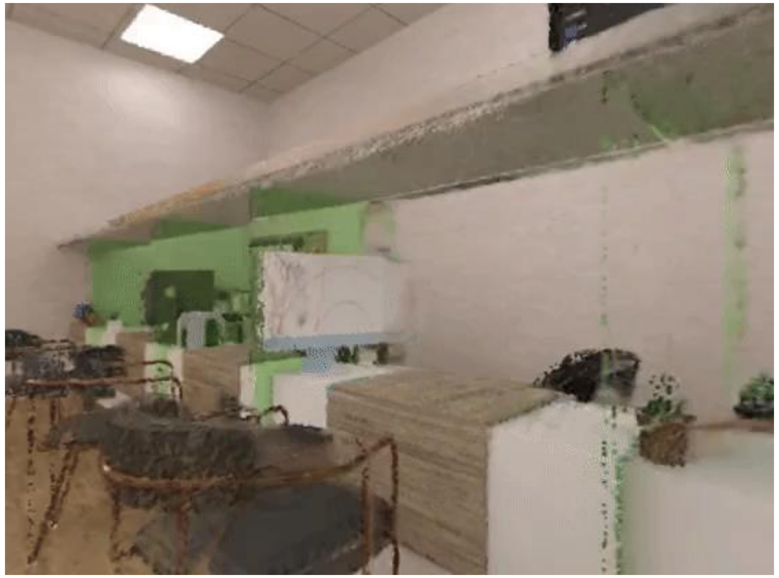
- RePaint: Starts from pure noise, image is denoised step-by-step
  - Inference relatively long (~30 seconds per image)
  - Only RGB inpainting

Inpainting

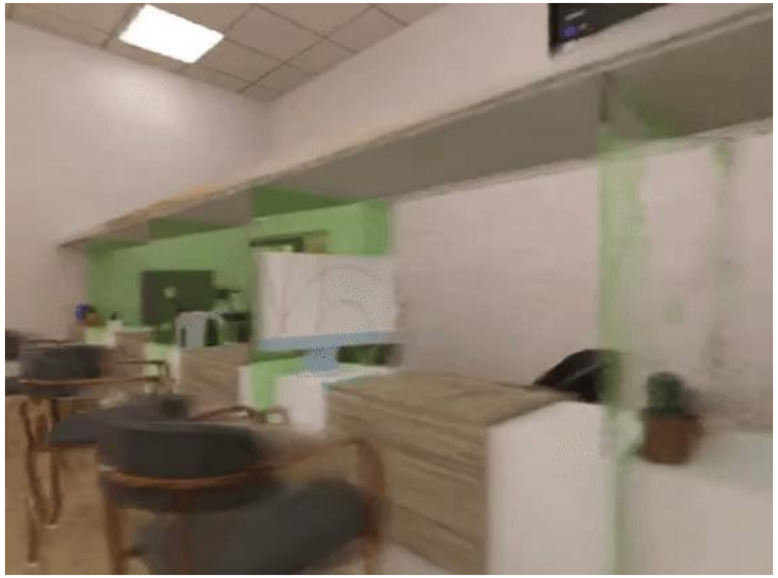


- RGBD^2: 3D scene reconstruction with RGBD diffusion inpainting for posed images
  - Make use of pretrained RGBD inpainting diffusion model
  - Inference faster then RePaint (~10 seconds per image)
  - Does depth map inpainting

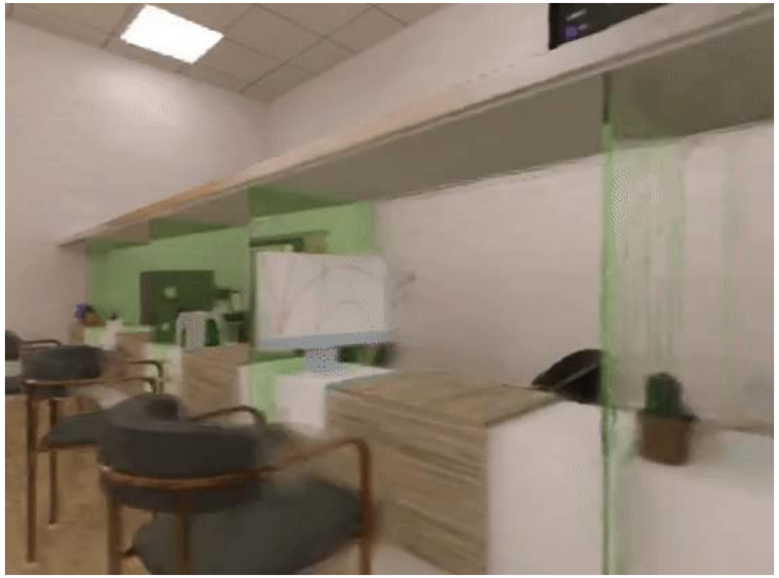
RePaint vs. RGBD²: RGB



Ground Truth Views +
Partial Loss +
GT Depth Loss (15.000)



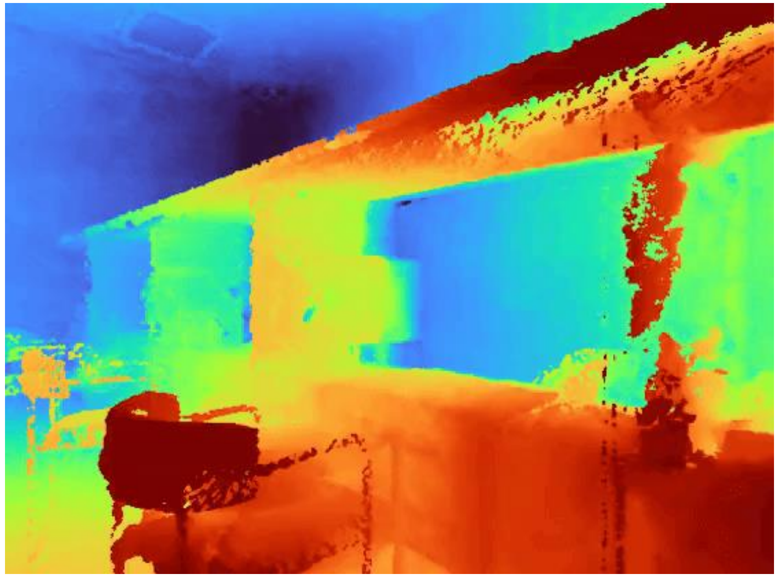
Ground Truth Views +
RePaint: RGB Inpainting +
Partial Depth Loss (5.000)



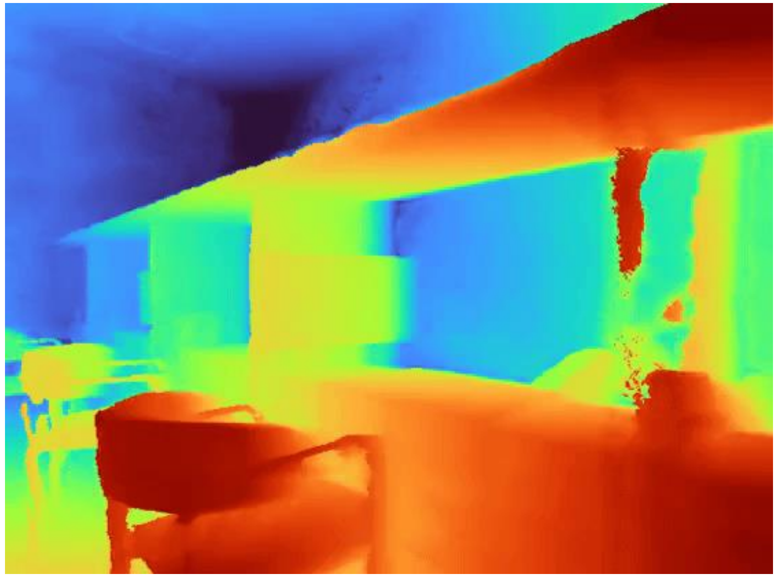
Ground Truth Views +
RGBD²: RGBD Inpainting (5.000)

- Both inpainting results show better quality for occluded regions
- RGBD² less blurry then RePaint inpainting

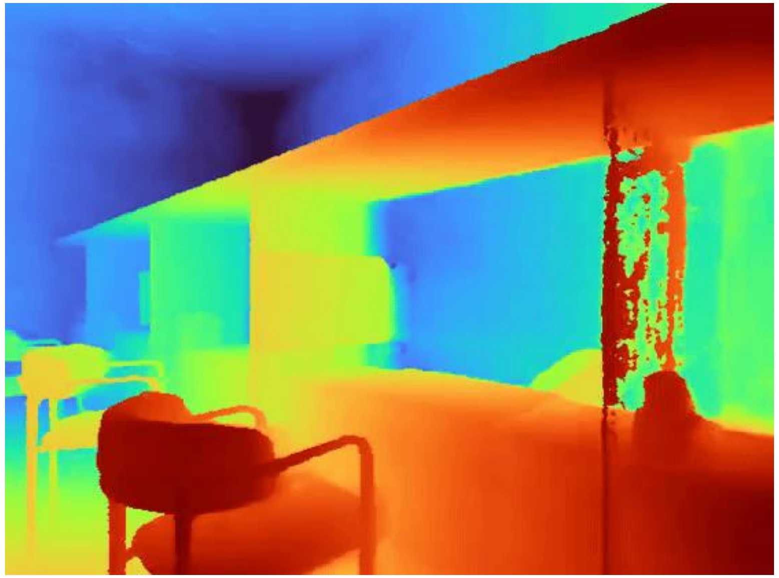
RePaint vs. RGBD²: Depth



Ground Truth Views +
Partial Loss +
GT Depth Loss (15.000)



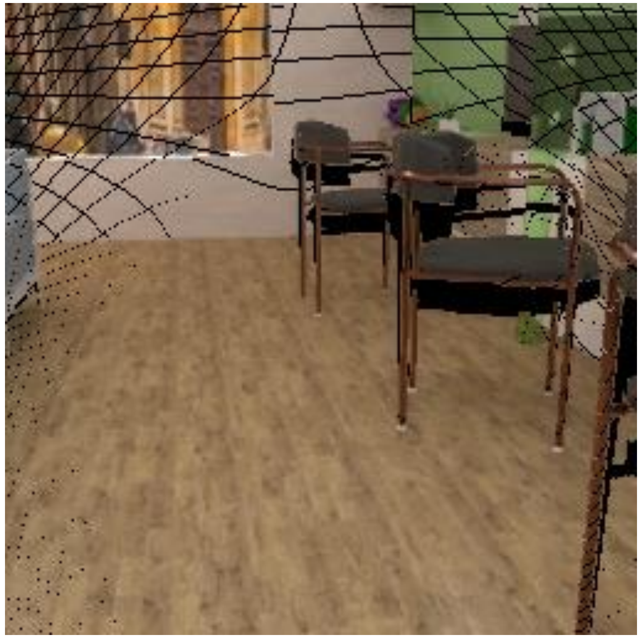
Ground Truth Views +
RePaint: RGB Inpainting +
Partial Depth Loss (5.000)

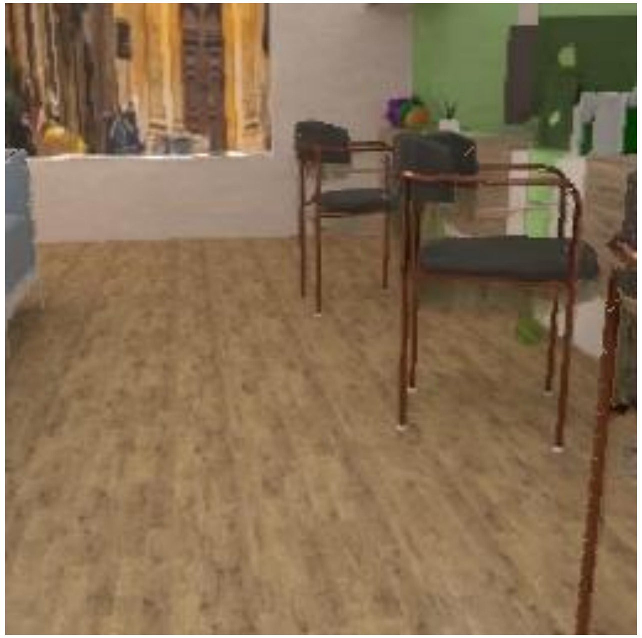


Ground Truth Views +
RGBD²: RGBD Inpainting (5.000)

- RePaint yields better results for depth with Partial Depth loss then only using ground truth depth
- RGBD² overall better then RePaint and previous approaches

Remaining Challenges: Inpainting

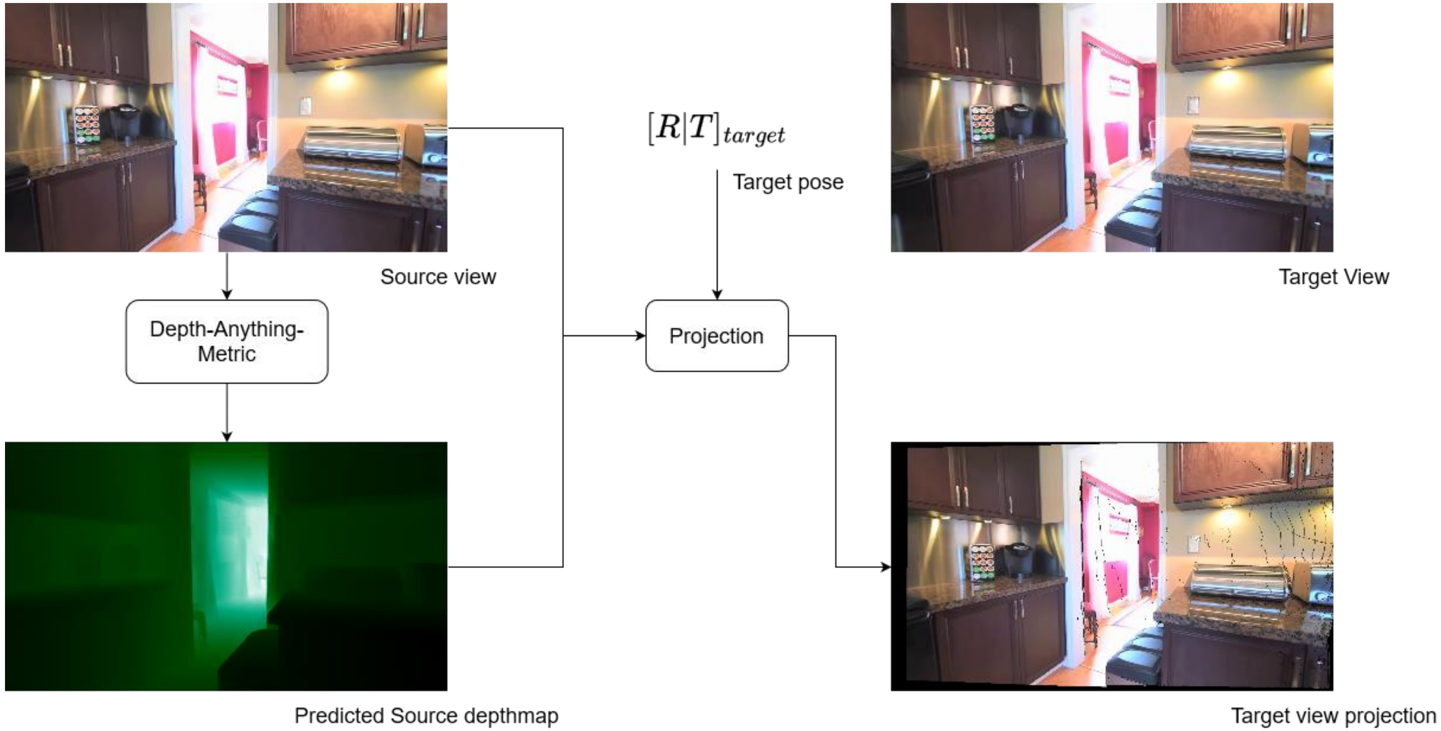




- Inpainting does not always give good results for our special case of partial images and hallucinates more from larger movements
  - Finetune RGBD² inpainting diffusion model on dataset like ours
  - Update point cloud with reprojection and newly added inpainting results for consistency

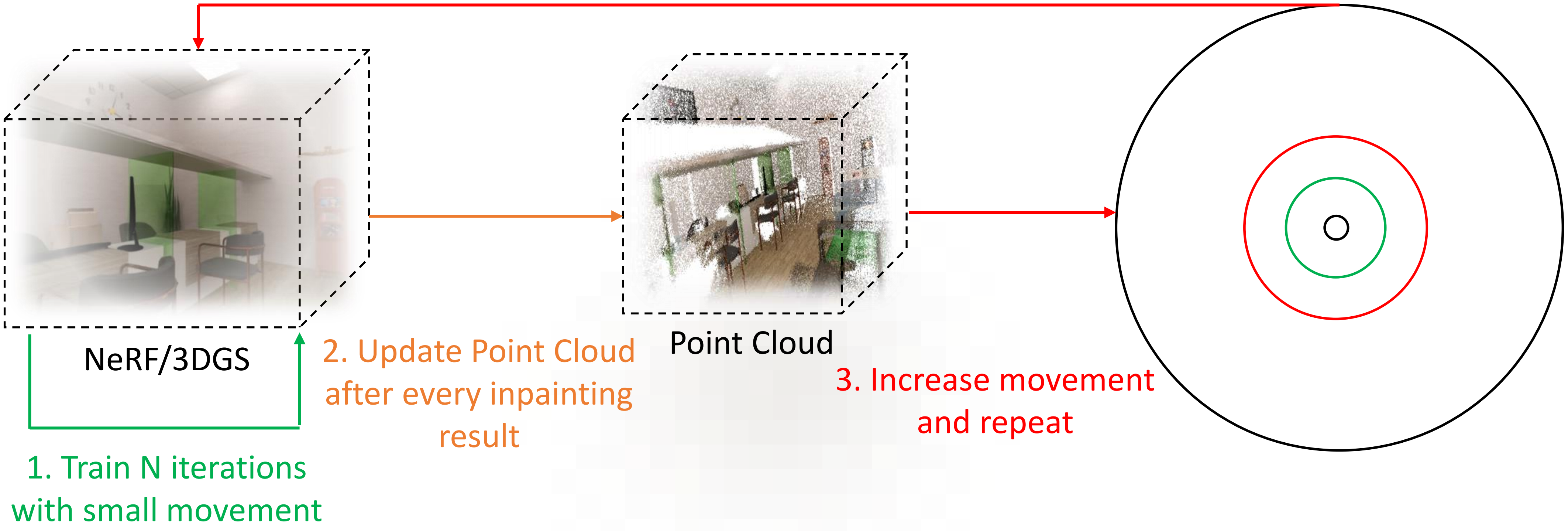
Remaining Challenges: Inpainting (Finetuning)

- Create synthetically similar looking partial images from datasets like RealEstate10K etc.
  - Create depth map with monocular depth estimation model to reproject from one view to another



Remaining Challenges: Inpainting (Extension)

- Train for N iterations with with small movements and update initial point cloud with newly added context for low coverage regions
  - Iteratively refine existing point cloud with inpainted novel views



NeRF/3DGS

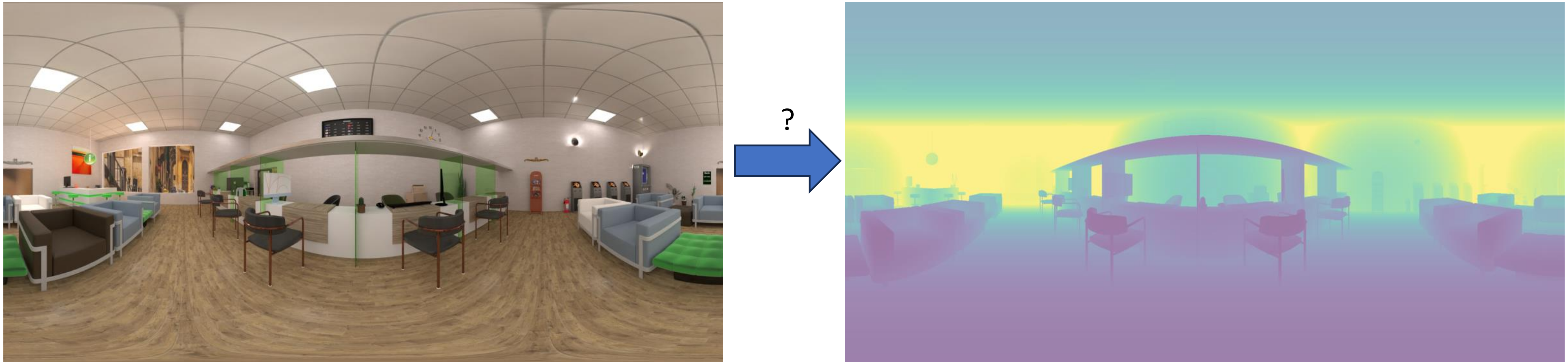
1. Train N iterations with small movement

2. Update Point Cloud after every inpainting result

Point Cloud

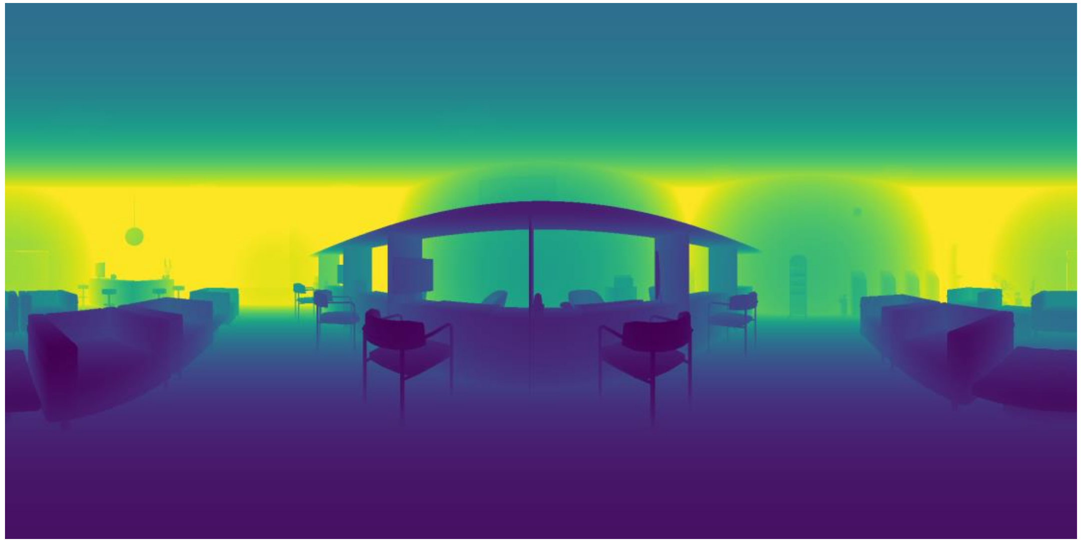
3. Increase movement and repeat

Remaining Challenges: Depth

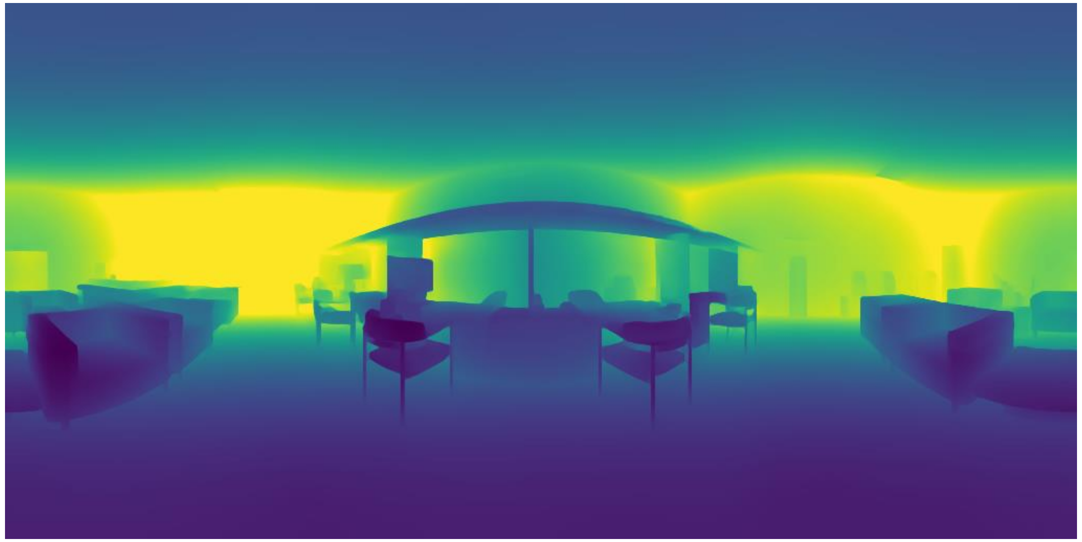


- So far we assumed a synthetic dataset where ground truth depth is given, but what if we only have RGB data available?
  - Estimate depth from single ERP image

Remaining Challenges: Depth (Estimation)



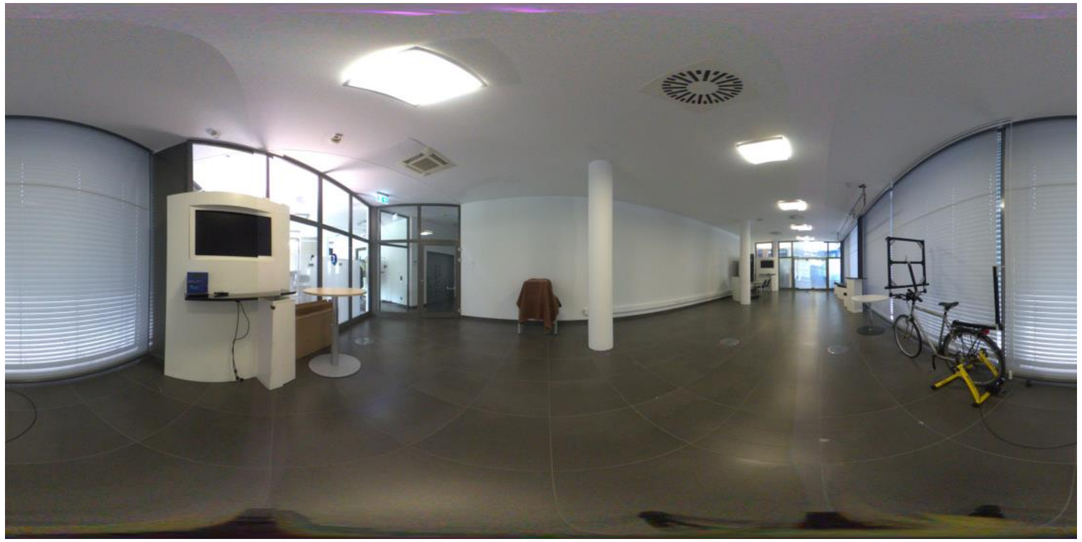
Ground Truth



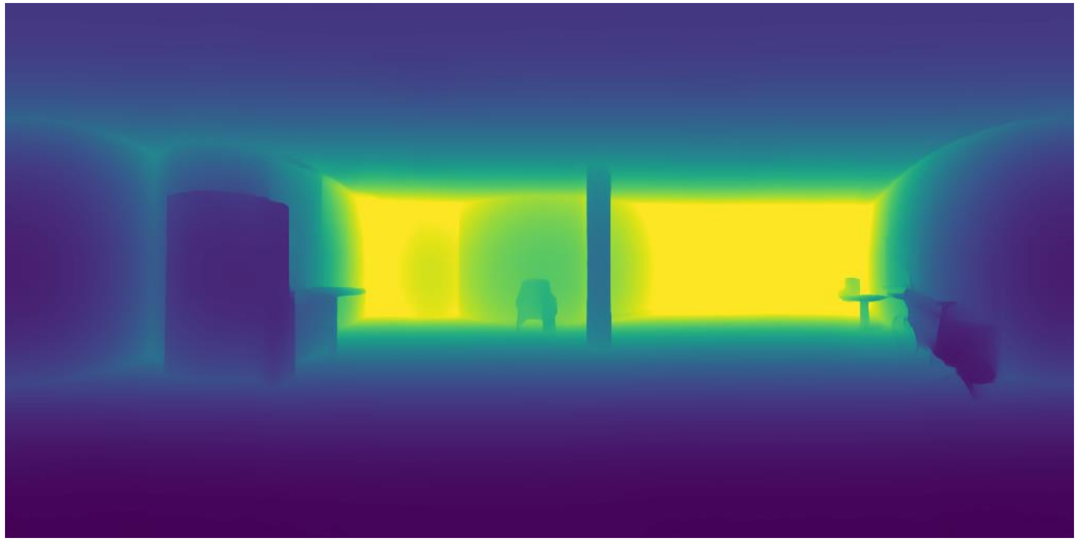
Depth-Anywhere (EGformer)

- 360 Monocular depth estimation via Depth-Anywhere (EGformer)
  - Good depth estimation important for initial information of scenery in training and for inpainting
  - Depth is only of shape 1024 x 512, half the resolution of the RGB image

Combining Solutions (Extension + Estimation)



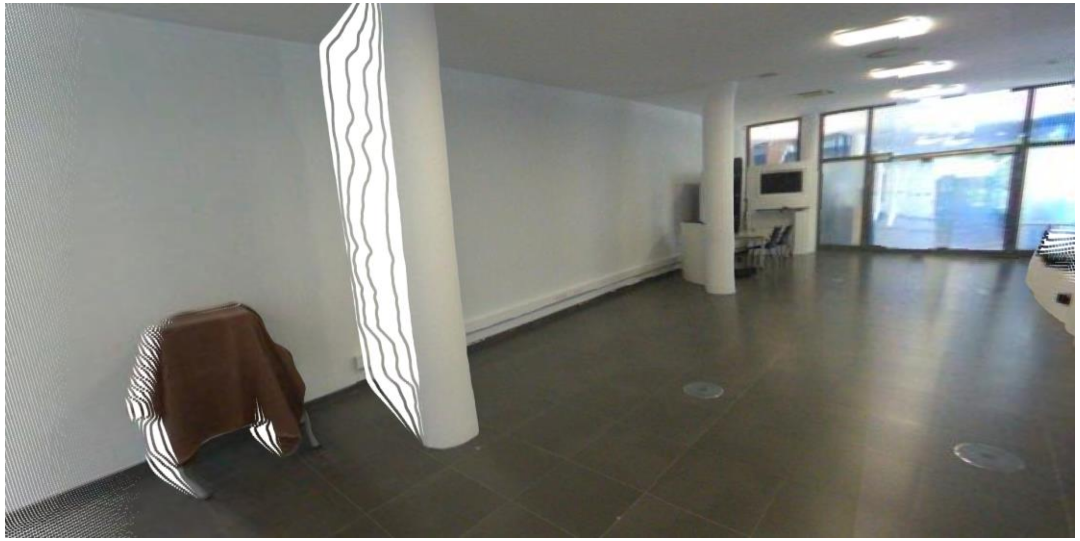
Non-synthetic RGB image



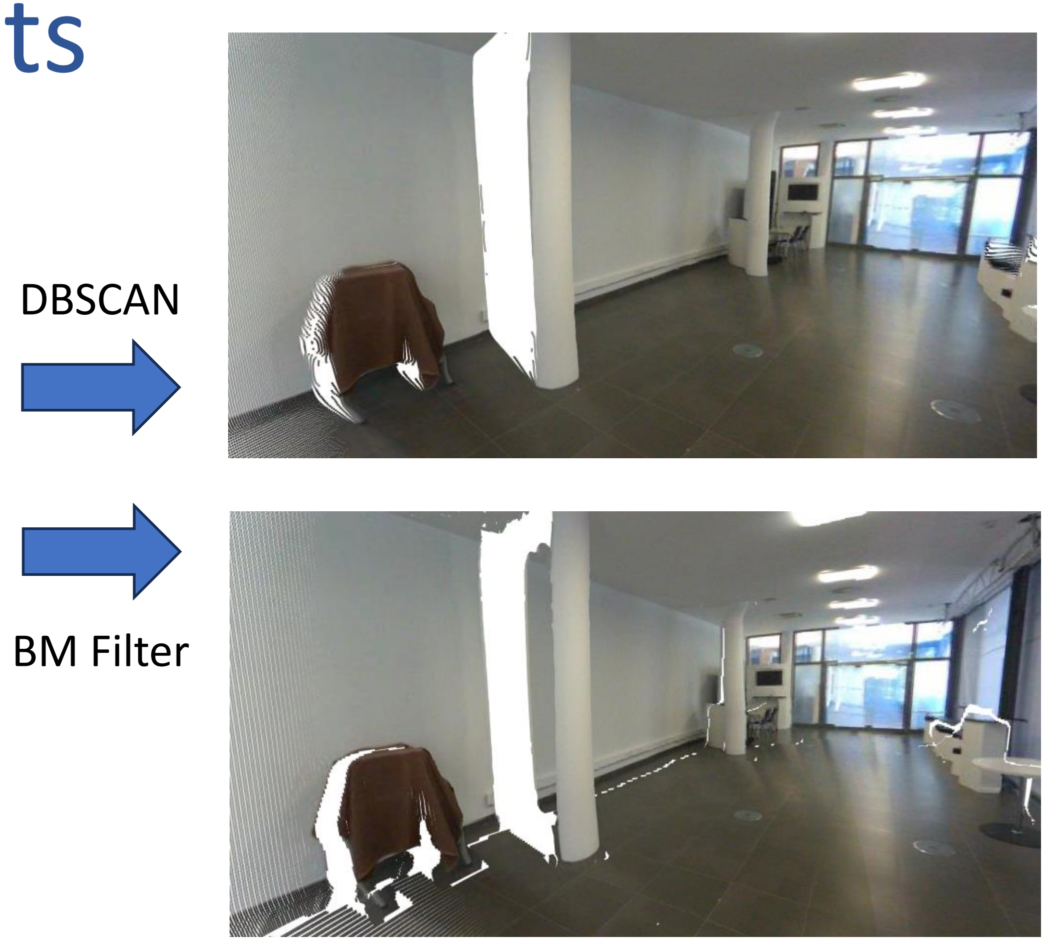
Estimated depth (upscaled)

- DFKI showroom as an application for a non-synthetic scenario with only a single ERP RGB image
  - Good depth estimation, with some exceptions (e.g. bicycle on the right)

Initial Point Cloud Artifacts



Initial Point Cloud

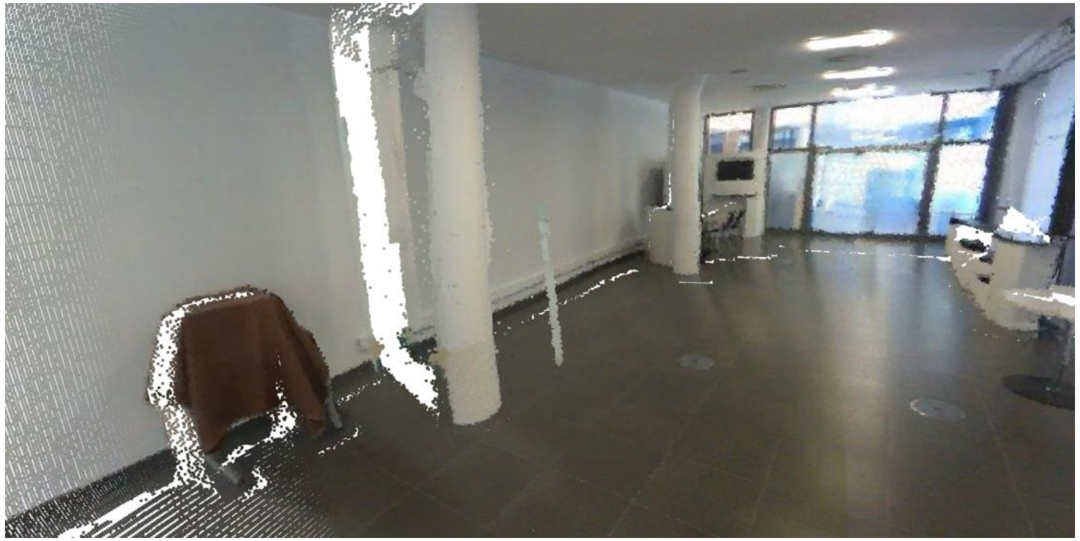


DBSCAN

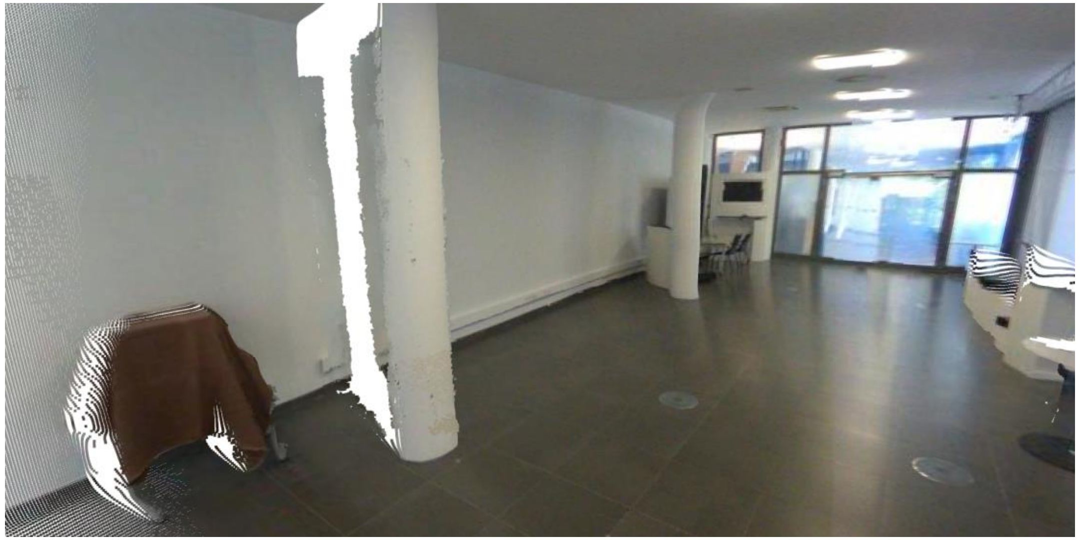
BM Filter

- Upscaling of depth map results in continous depth on foreground object edges
  - Cluster removal with DBSCAN in a radius around center
  - Bilateral median filter for introducing discontinuity in depth map

Point Cloud Extension Artifacts



Point Cloud after 100 projections



Point Cloud after 100 projections
with cluster removal

- Due to some incorrect depth predictions from inpainting model, floaters are added to the point cloud
  - Cleanse to be added point cloud with DBSCAN
  - Or introduce stricter depth thresholds

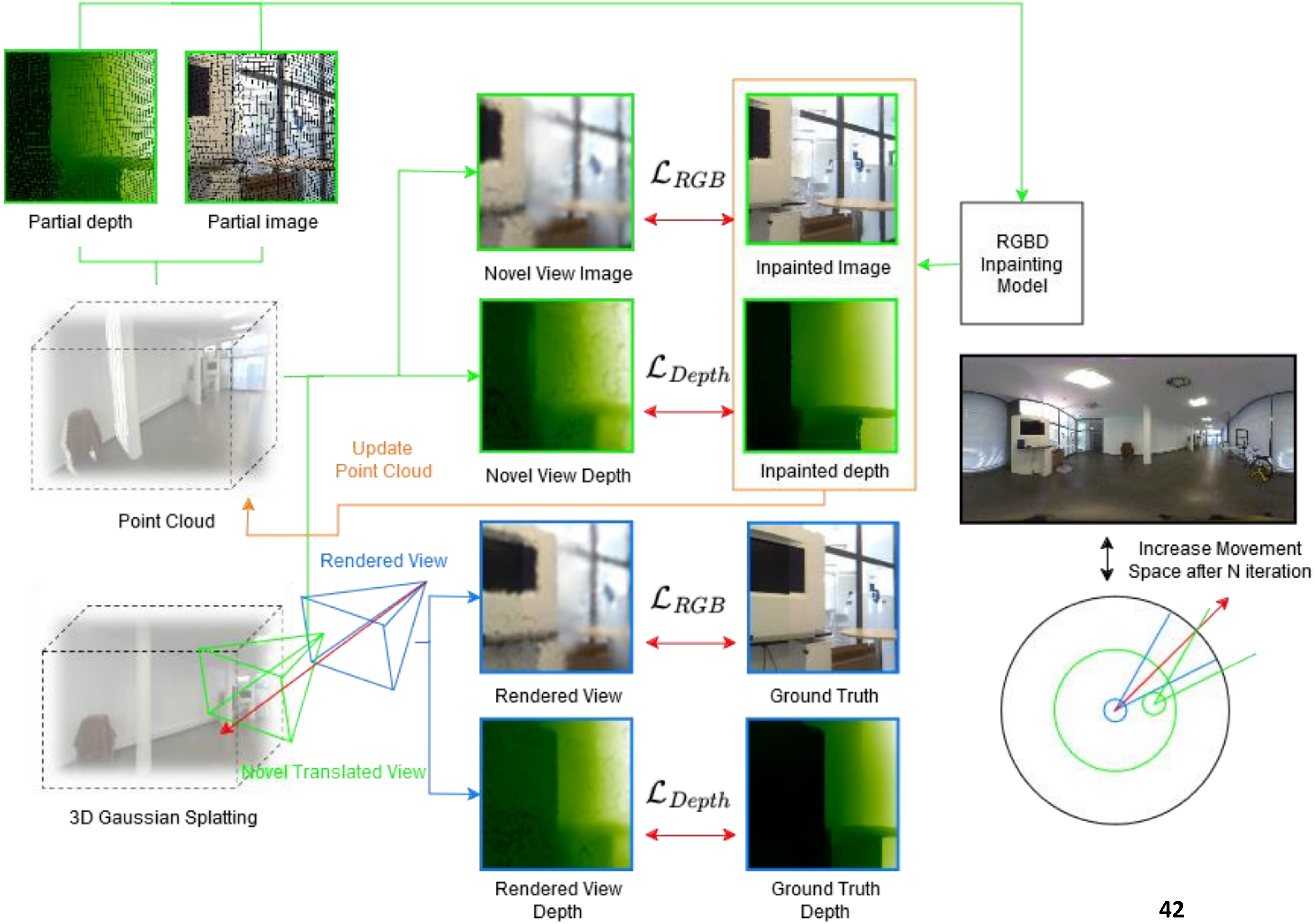
Early Inpainting

- Inpainting remains the biggest bottleneck when it comes to training speed
- Furthermore, after a certain amount of iterations, inpainting won‘t add much more context
  - Earlier inpainting results which have been added to the poincloud will at some point cover enough region

- Early Inpainting: Start off by inpainting much earlier on during training and then proceed with Partial Loss

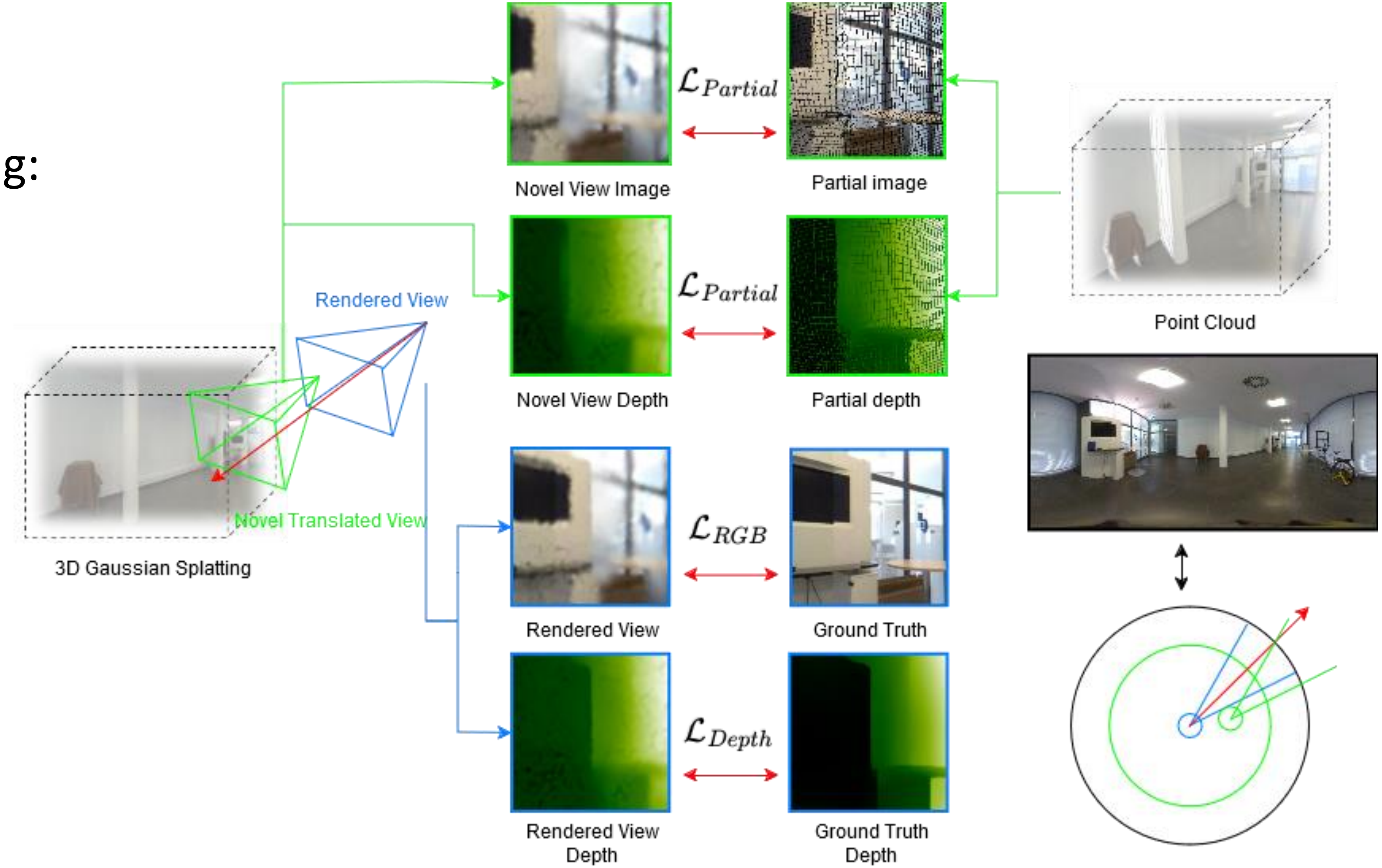
Monocular 360° GS

- Current approach:
  - During Early Inpainting:
    - Center View RGB + Depth Loss with ground truth
    - Novel View RGB + Depth Loss with inpainted RGBD
    - Update Point Cloud after every inpainting result
    - Increase movement space after every N iterations

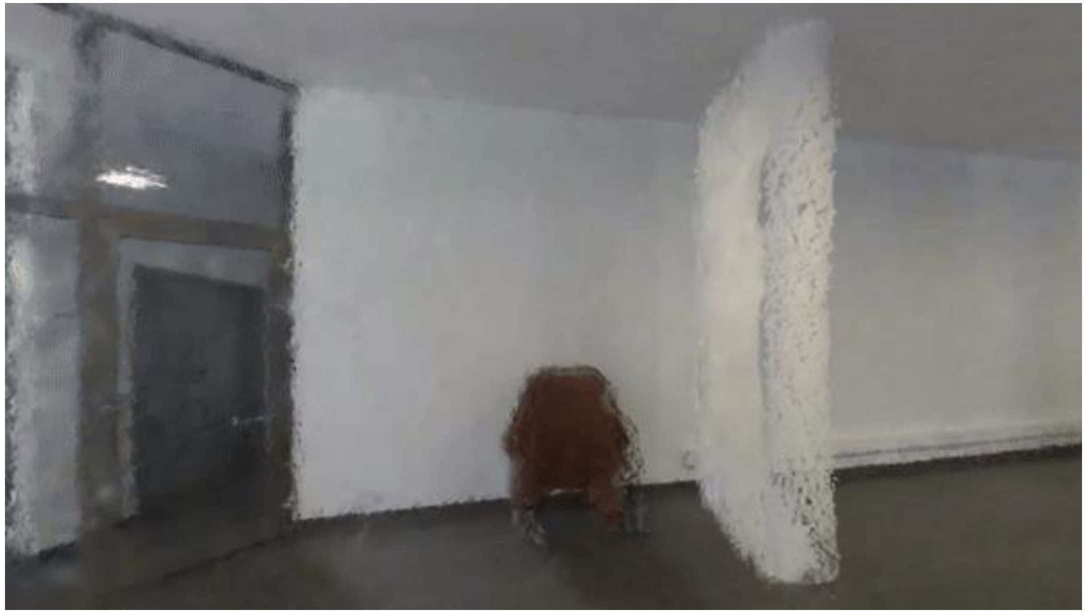


Monocular 360° GS

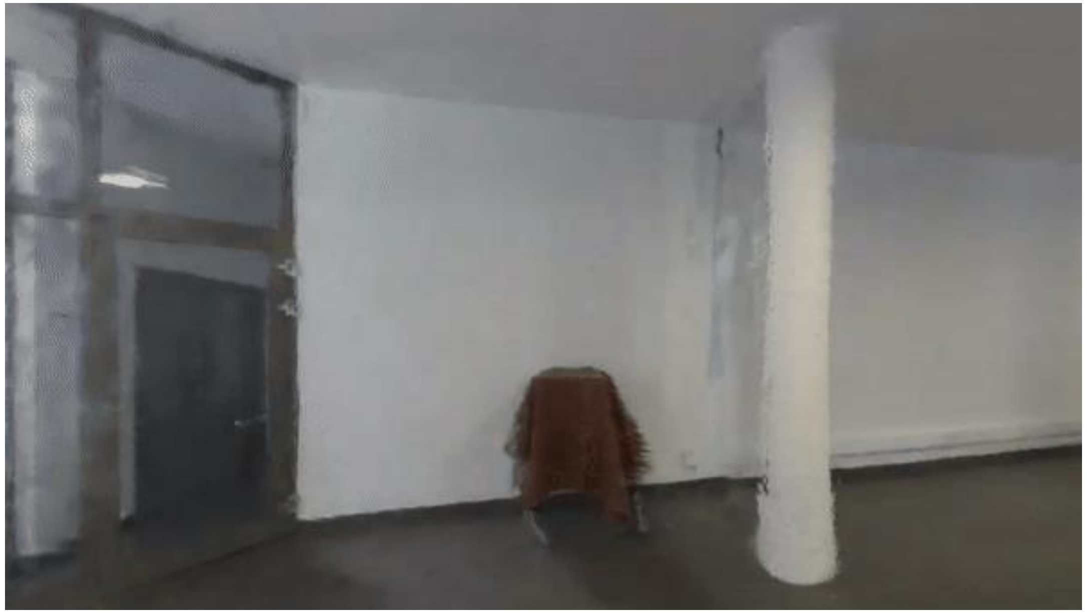
- Current approach:
  - Remainder of training:
    - Center View
RGB + Depth
Loss with ground truth
    - Novel View
Partial RGBD
Loss with partial images



Monocular 360° GS: Results



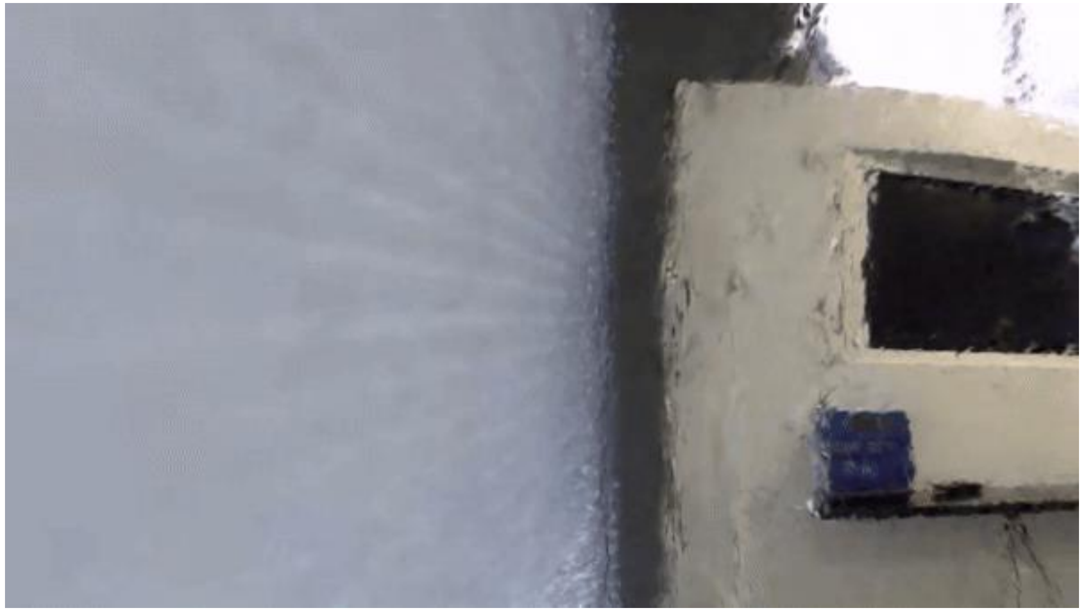
Standard 3DGS



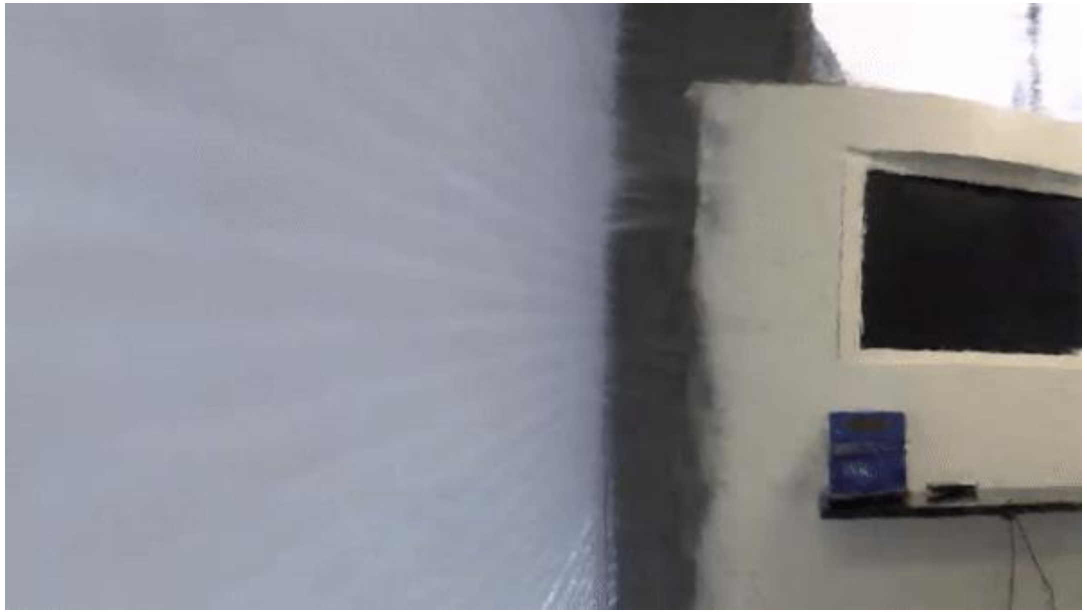
Monocular 360° GS

- Standard 3DGS: Problems with occluded and low coverage regions
  ⇨ Smearing Effect
- Monocular 360° GS: Model yields better results in occluded regions and overall quality

Monocular 360° GS: Results



Standard 3DGS



Monocular 360° GS

- Standard 3DGS: Problems with occluded and low coverage regions
  → Smearing Effect
- Monocular 360° GS: Model yields better results in occluded regions and overall quality

Monocular 360° GS: Second Review

- Encountered problems with current approach:
  1. Estimated ERP depth very important for initial training during early inpainting
     - Complex scenes generally harder for Monocular 360 depth estimators
  2. Novel view camera moves sometimes out of scenery or inside objects
  3. Inpainting model gets influenced a lot by edges of foreground objects
     - Occluded regions don’t have smooth transitions in color
  4. Model has problems with continuous appearances which are occluded by foreground objects
     - Sudden change of object appearance