Robust Depth Regularization in Gaussian Splatting

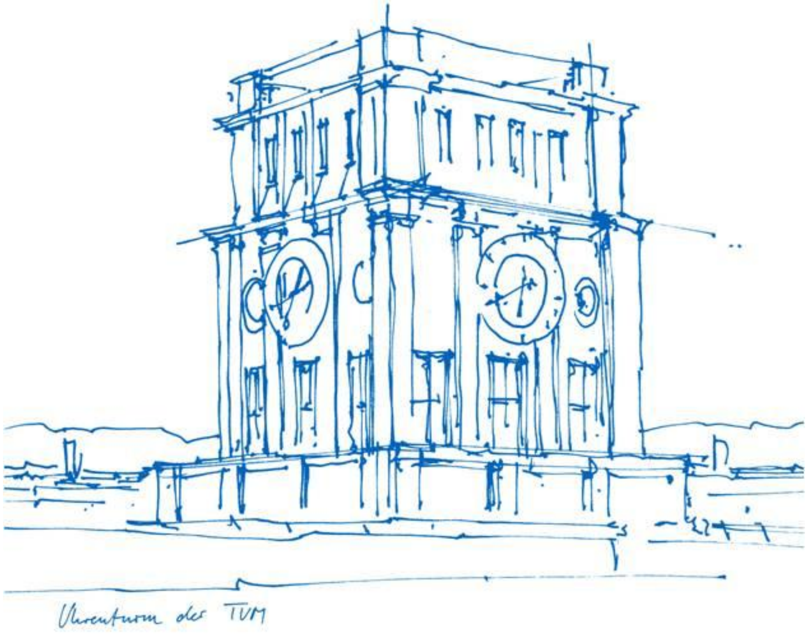
Pierre Reboud

Technichal University of Munich

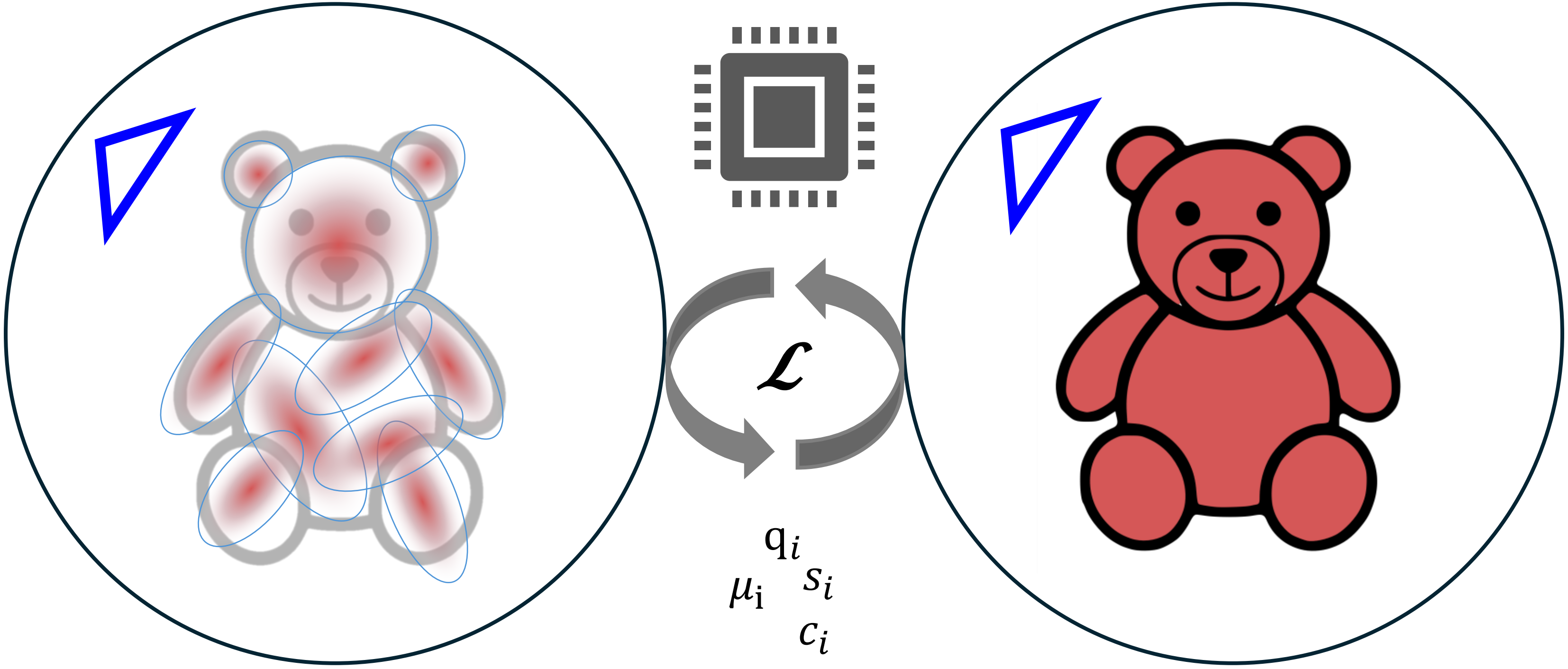
TUM School of Engineering

Computer Vision Group

Munich, 16th of July 2025



Gaussian Splatting - Introduction

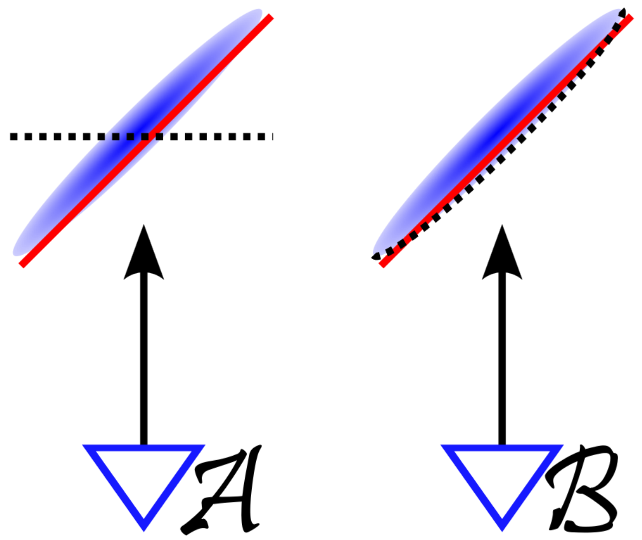


Depth Formulation - Challenges

Challenges of current SOTA depth priors:

- Inability to represent locally curved surfaces.
- Inability to represent slanted surfaces with respect to a camera.

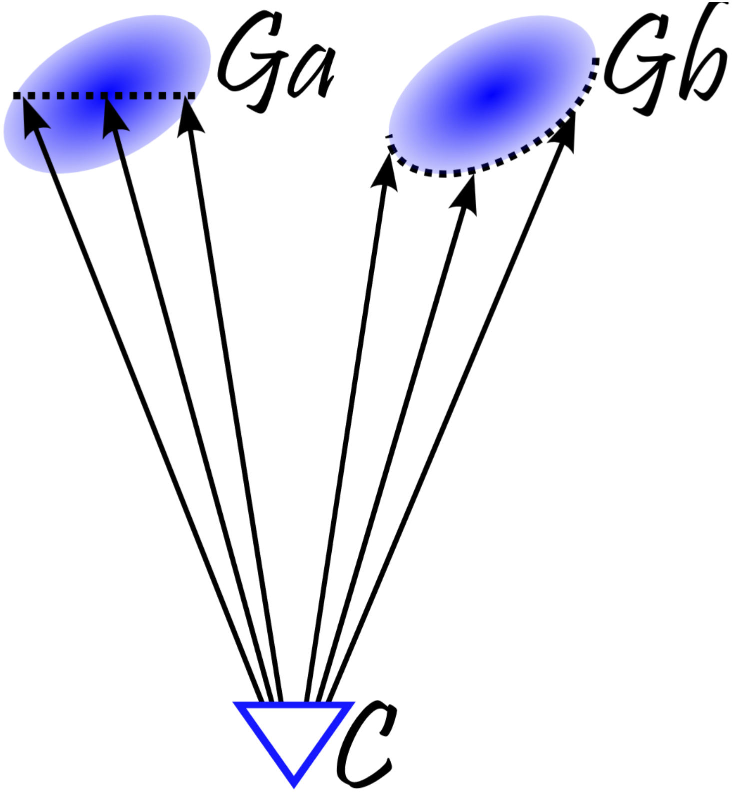
Use the Gaussian‘s curvature to increase its depth field expressiveness.



Depth Formulation - Derivation

Ellipsoidal iso-surface definition:

$$S_k = \{x \in \mathbb{R}^3 | ||x - \mu||_{\Sigma^{-1}} = k\}$$



Depth Formulation - Ellipsoidal Iso-Surface

Defining the ellipsoidal iso-surface as an extension of Yu et al 2024

Where:

- v is the pixel aligned bearing vector
- $\mu - o$ is the camera coordinate mean vector
- Σ is the Gaussian‘s covariance matrix

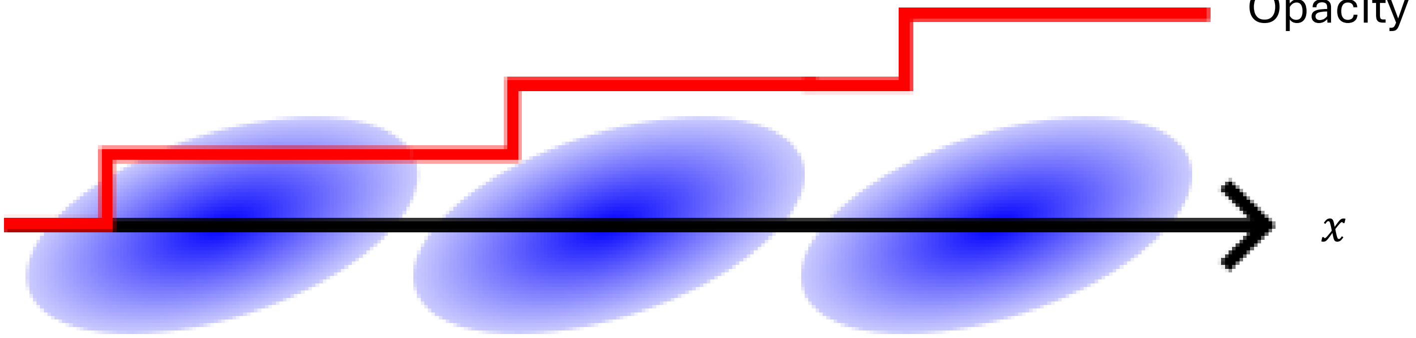
$$\alpha = \|v\|^2_{\Sigma^{-1}}$$

$$\beta = \langle v, (o - \mu) \rangle_{\Sigma^{-1}}$$

$$\gamma = \|o - \mu\|^2_{\Sigma^{-1}}$$

$$t^\star(v) = \frac{\beta - \sqrt{\beta^2 - \alpha(\gamma - k^2)}}{\alpha}$$

Depth Formulation - Alpha-Compositing Depth



$$t^{\star}_{\text{alpha\_composed}}(v) = \sum_{i=0}^{N} \frac{\alpha_i \prod_{j=0}^{i}(1-\alpha_j)}{1-\alpha_i} t^{\star}_i(v)$$

Depth Formulation - Ellipsoidal Iso-Surface Gradients

Steps:

1. Decompose the depth into subterms $\alpha, \beta, d1, d2$.
2. Recursively apply the chain rule.

Pitfalls for computing gradients on H:

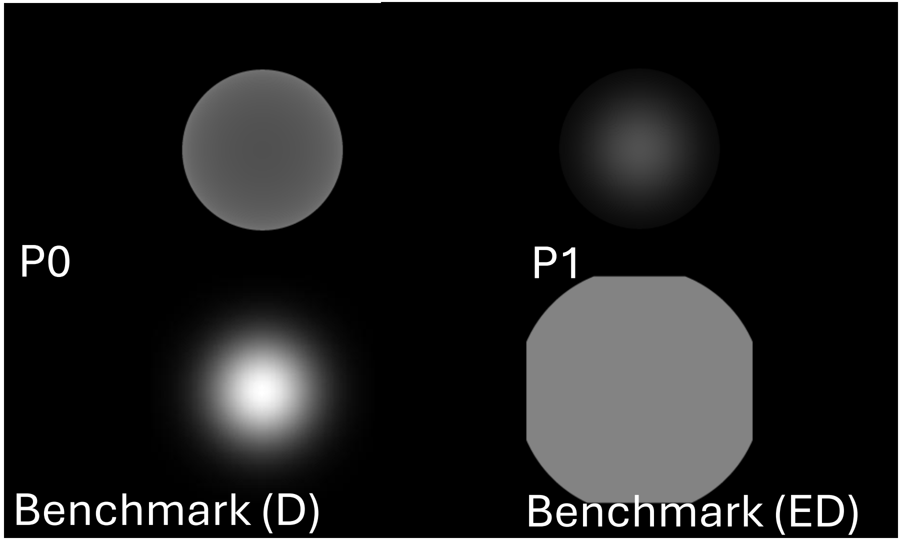
1. Consider H‘s double cover property of SO(3).
2. Use the 4D Euclidean dot product criterion to resolve the incumbent/target rotation‘s representation ambiguity.

$$
\begin{aligned}
J_{t^\star,\mu} =& J_{t^\star,\alpha}(J_{\alpha,d_1} J_{d_1,\mu} + J_{\alpha,d2} J_{d_2,\mu}) \\
&+ J_{t^\star,\beta}(J_{\beta,d_1} J_{d_1,\mu} + J_{\beta,d2} J_{d_2,\mu}) \\
&+ J_{t^\star,\gamma}(J_{\gamma,d_1} J_{d_1,\mu} + J_{\gamma,d2} J_{d_2,\mu}) \\
J_{t^\star,s} =& J_{t^\star,\alpha}(J_{\alpha,d_1} J_{d_1,s} + J_{\alpha,d2} J_{d_2,s}) \\
&+ J_{t^\star,\beta}(J_{\beta,d_1} J_{d_1,s} + J_{\beta,d2} J_{d_2,s}) \\
&+ J_{t^\star,\gamma}(J_{\gamma,d_1} J_{d_1,s} + J_{\gamma,d2} J_{d_2,s}) \\
J_{t^\star,q} =& J_{t^\star,\alpha}(J_{\alpha,d_1} J_{d_1,q} + J_{\alpha,d2} J_{d_2,q}) \\
&+ J_{t^\star,\beta}(J_{\beta,d_1} J_{d_1,q} + J_{\beta,d2} J_{d_2,q}) \\
&+ J_{t^\star,\gamma}(J_{\gamma,d_1} J_{d_1,q} + J_{\gamma,d2} J_{d_2,q})
\end{aligned}
$$

Depth Formulation - Comparison

Possible depth formulations:

1. RGB+D: $k = 0$ + depth scale normalization
2. RGB+ED: $k = 0$ + alpha composition
3. P0: $k \geq 0$
4. P1: $k \geq 0$ + alpha composition
5. Further: $k \propto$ opacity



Training Losses

Depth regularization:

1. Gather the sparse SfM points.
2. Project them onto the dense estimated disparity map.
3. Retrieve the error terms and back-propagate the loss function.

$$\mathcal{L} := \lambda_1 \mathcal{L}_{L_1} + \lambda_2 \mathcal{L}_{\text{depth}} + \lambda_3 \mathcal{L}_{\text{SSIM}} + \lambda_4 \mathcal{L}_{\text{oflow}}$$

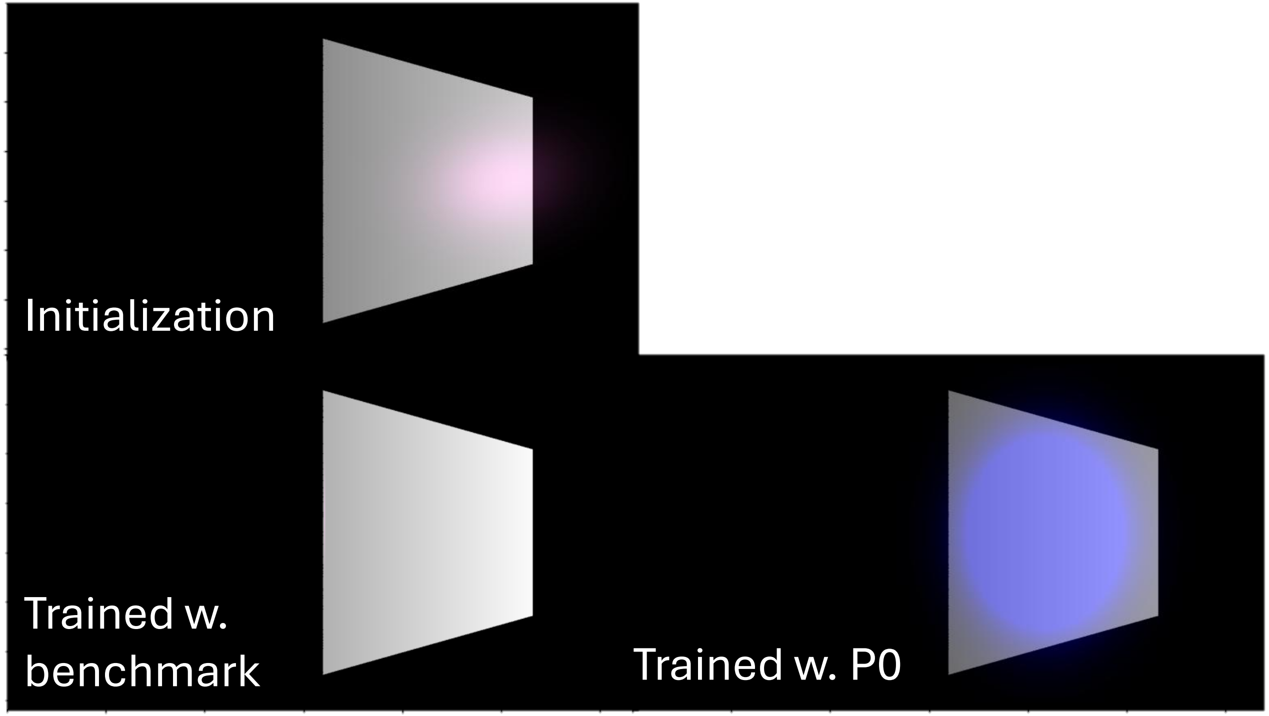
Choose a factor ω in line with the scene‘s scale, where D_i is the i^{th} image‘s depth map.

$$\mathcal{L}_{\text{depth}} := \sum_{i}^{N} \omega \left| \frac{1}{\hat{\mathcal{D}}_i} - \frac{1}{\mathcal{D}_i} \right|$$

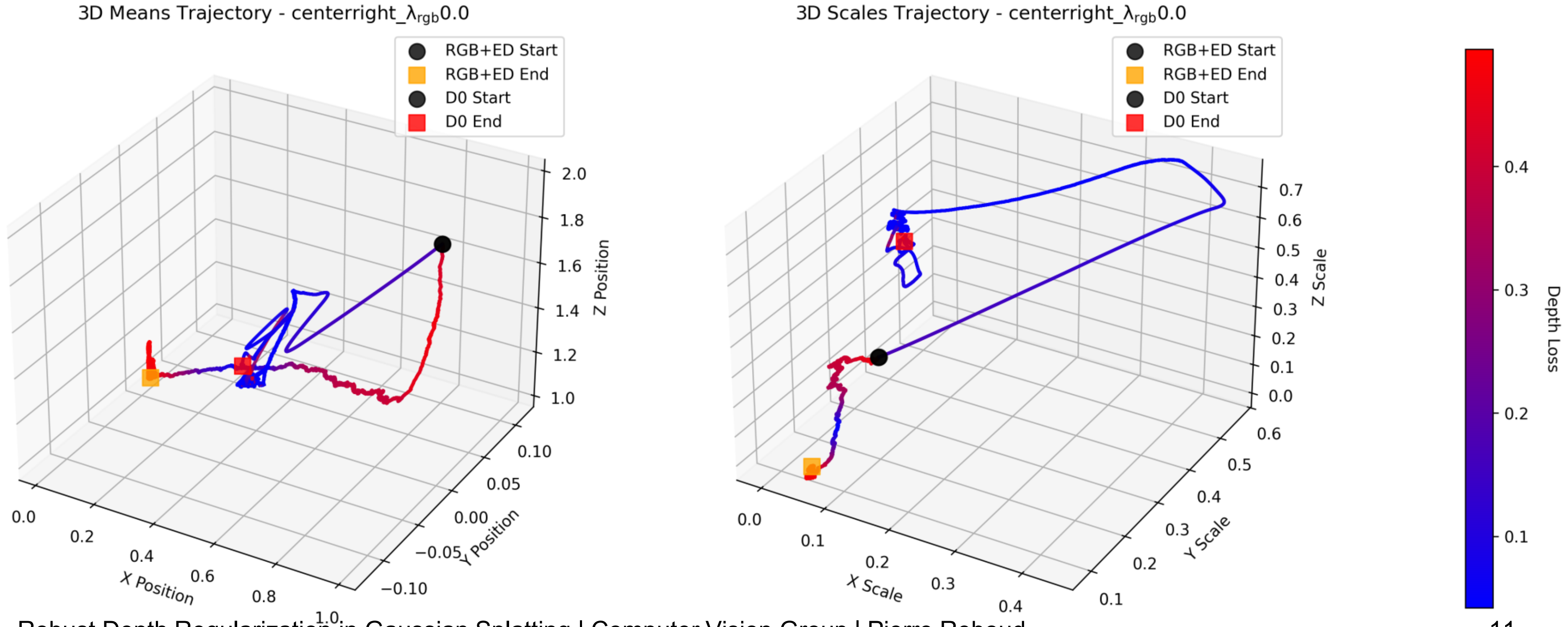
Slanted Square Scene - Setup

Initial setting:

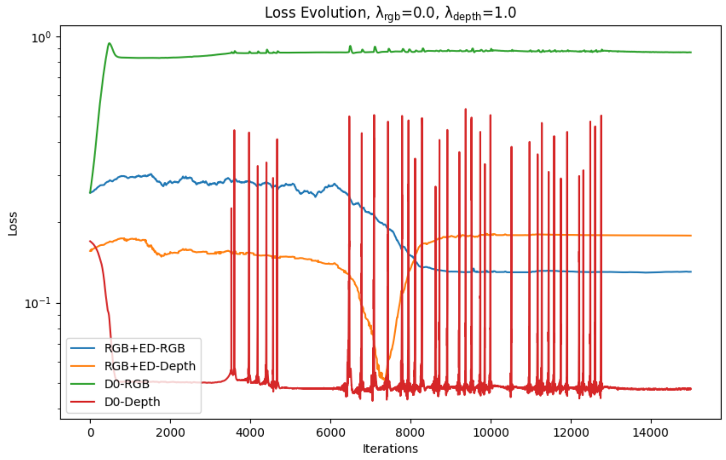
- A red Gaussian located at the right-hand side of a blue slanted square embedded in a 3D scene.
- The Gaussian is trained to approximate the square as accurately as possible.

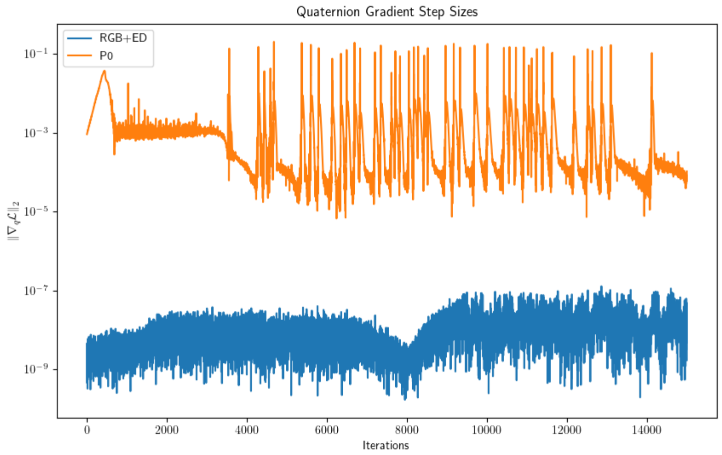


Slanted Square Scene – Parameter Optimization Trajectory

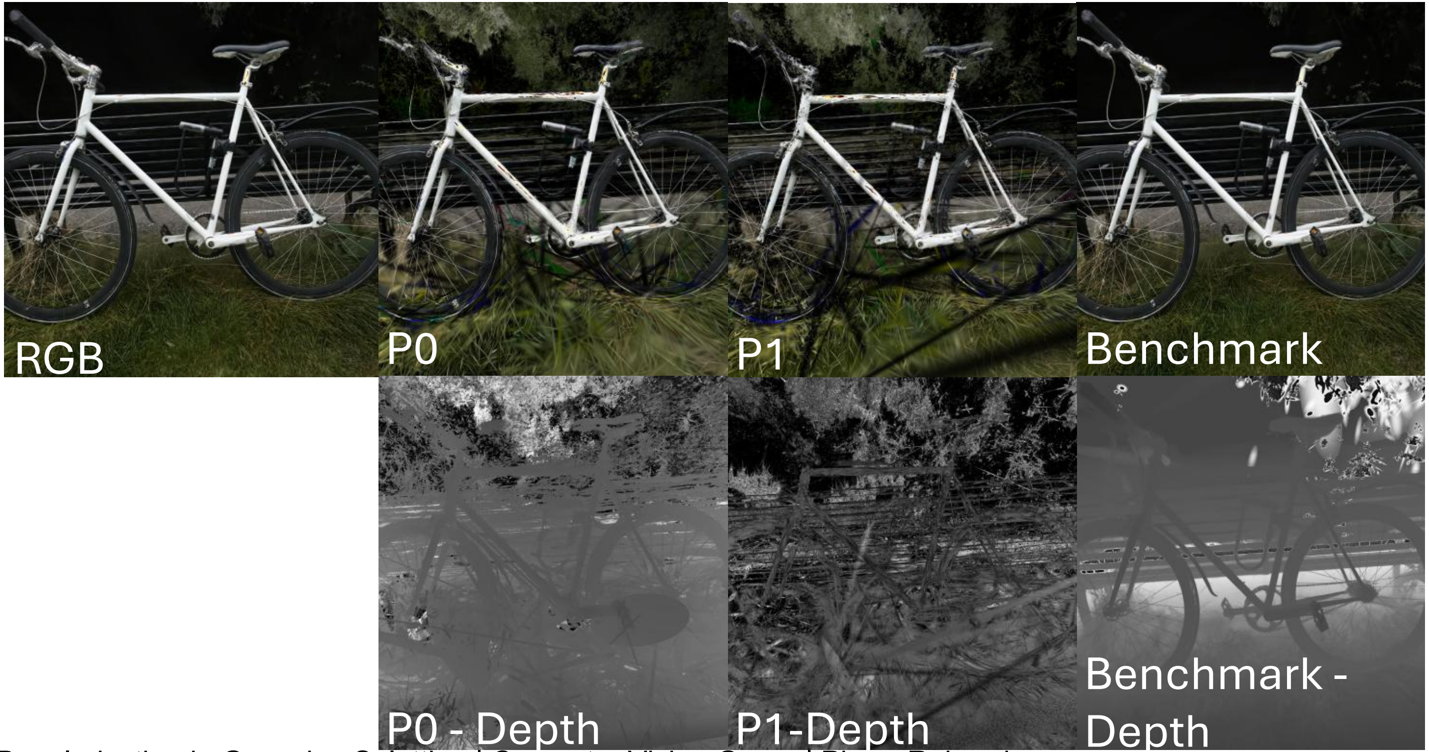


Slanted Square Scene – Loss Curves & Gradient Step Sizes



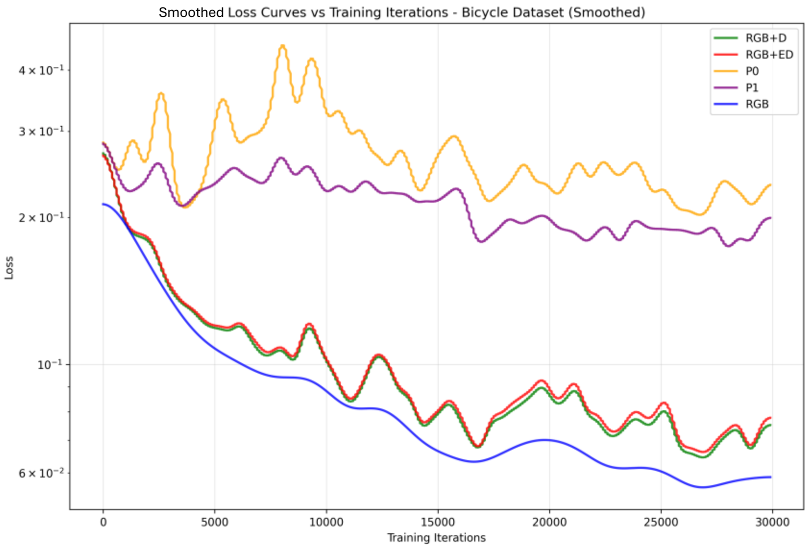


Mip-NeRF 360 - Bicycle Scene Renderings and Depth Maps



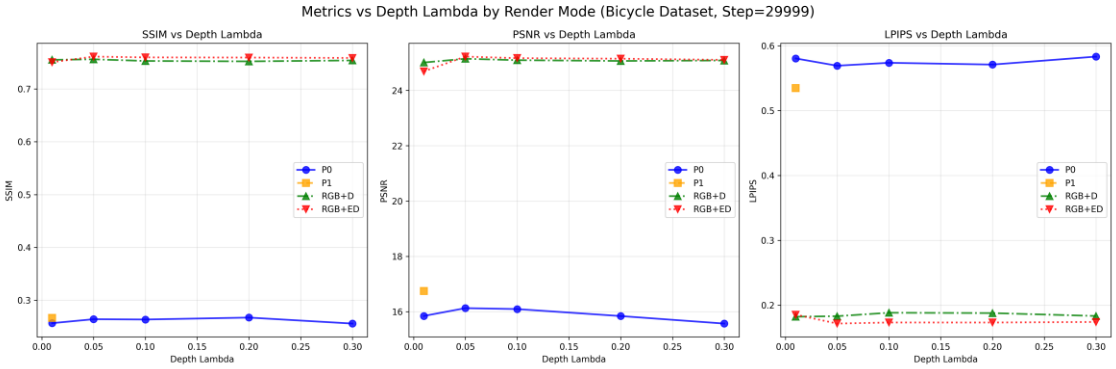
Mip-NeRF 360 - Training Metrics & Loss Curves

depth loss	optical flow loss	render mode	SSIM	LPIPS	PSNR
False	False	RGB	**0.820719**	**0.147808**	27.625149
	True	P0	0.325942	0.644561	14.398811
		P1	0.321939	0.626977	14.721218
		RGB+D	0.813236	0.156236	27.045279
		RGB+ED	0.807660	0.164491	26.743068
True	False	P0	0.330259	0.630369	14.575312
		P1	0.347030	0.592160	15.034364
		RGB+D	0.820411	0.149342	**27.695976**
		RGB+ED	0.820637	0.148114	27.687360
	True	P0	0.330666	0.632180	14.462277
		P1	0.324824	0.581280	15.094447
		RGB+D	0.816056	0.154421	27.339291
		RGB+ED	0.811594	0.159039	27.065453

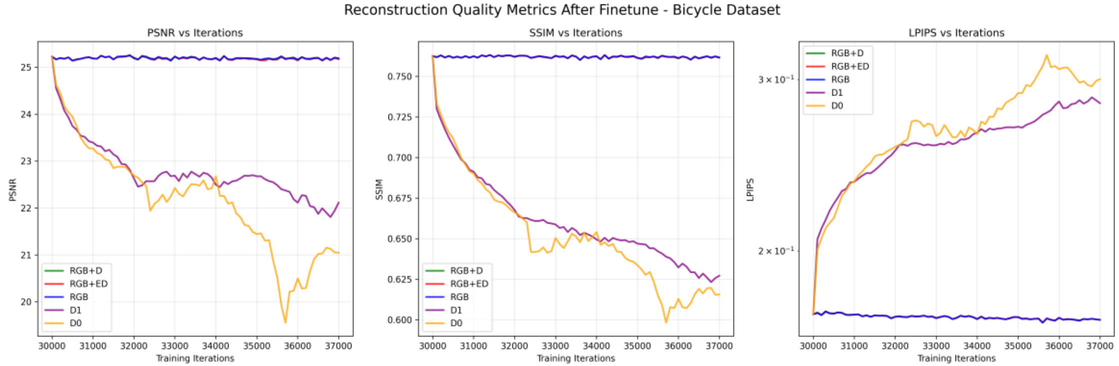


Mip-NeRF 360 - Further Evaluations

Ablations on the regularization strength differentiated by depth formulation.



Depth regularized fine-tuning on a scene fully trained without regularization.



Conclusion

Slanted square scene:

1. The ellipsoidal iso-surface is more expressive than planar depth priors in certain settings.
2. This formulation can be less prone to parameter collapse.
3. It is also more compute intensive and can lead to instabilities when using a projected SGD aproach.

Mip-NeRF 360:

1. Training with our depth regularization through sparse SfM supervision fails decisively.
2. Spurious floating Gaussians plague the reconstructed trained scene.

Future work:

1. Use monocular depth estimation to leverage dense depth map supervision.
2. Use Riemannian SGD algorithms to increase optimization trajectory stability and outcome reliability.

Appendix – Riemannian Adam (Bécigneul et al 2019)

1. Gradient in $\mathcal{T}_{q_k} S^3$: $$g_k = \nabla_q \mathcal{L}(q_k) - \langle \nabla_q \mathcal{L}(q_k), q_k \rangle q_k \quad (7.1)$$

2. Moment Updates: $$m_{k+1} = \beta_1 \tilde{m}_k + (1 - \beta_1) g_k \quad (7.2)$$
$$v_{k+1} = \beta_2 \tilde{v}_k + (1 - \beta_2) g_k \odot g_k \quad (7.3)$$

3. Bias Correction: $$m'_{k+1} = \frac{m_{k+1}}{1 - \beta_1^{k+1}}, \quad v'_{k+1} = \frac{v_{k+1}}{1 - \beta_2^{k+1}} \quad (7.4)$$

4. Tangent Update Vector: $$u_{k+1} = -r \frac{m'_{k+1}}{\sqrt{v'_{k+1}} + \epsilon} \quad (7.5)$$

5. Exponential Map Retraction Update: $$q_{k+1} = \exp|_{q_k}(u_{k+1}) = \cos(\|u_{k+1}\|) q_k + \sin(\|u_{k+1}\|) \frac{u_{k+1}}{\|u_{k+1}\|} \quad (7.6)$$

6. Parallel Transport of Moments: $$\tilde{m}_{k+1} = \mathrm{PT}_{q_k \to q_{k+1}}(m_{k+1}), \quad \tilde{v}_{k+1} = \mathrm{PT}_{q_k \to q_{k+1}}(v_{k+1}) \quad (7.7)$$