4. Method

3D reconstruction from 2D images is a challenging problem as there can be numerous 3D configurations that can result in the same set of 2D images. In other words, the problem is under-constrained and lacks a unique solution. This is because 2D images only provide incomplete information about the 3D scene, limited to photometric traits like texture, shape, and lighting conditions. As a result, reconstructing the complete 3D geometry of a scene from 2D images can be challenging, especially when dealing with complex scenes, occlusions, or noisy input data.

The main objective of this study is to enhance novel view reconstruction outcomes by incorporating complementary monocular signals, such as depth- and normal maps, along with the photometric supervision signal. This helps to constrain the spatial positioning and surface orientation of objects in the depicted scene. To this end, we present a simple loss formulation that is derived from statistical considerations that can be easily tuned to work with synthetic and real data.

4.1. Depth Supervision with Statistical Weight Bounds

We assume a scene comprised of non-transparent objects. We model the scene with a neural field, $F_\theta : (\mathbf{x}, \mathbf{d}) \rightarrow (\mathbf{c}, \sigma)$, that maps input coordinates $\mathbf{x}$ and viewing directions $\mathbf{d}$ to volume densities $\sigma(\mathbf{x})$ and colors $\mathbf{c}$. Into the scene, we project a camera ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$, and we model the depth D at which the ray terminates, as the expectation of the distribution $f(t) = \mathcal{T}(t) \cdot \sigma(t)$, i.e. $D = \mathbb{E}[f(t)]$, with $f(t)$ being the likelihood that the ray stops at precisely t.

If we model $f(t)$ as a Gaussian distribution $f(t) = \mathcal{N}(\mu, \epsilon^2)$[1], then the expected value $\mathbb{E}[\mathcal{T}(t) \cdot \sigma(t)]$ is exactly the mode, i.e. the distribution is centered at $\mu = D$. The corresponding CDF $F(t)$ is easily found (in most statistics textbooks and also implemented in PyTorch). This CDF is the opacity function $F(t) = (1 - \mathcal{T}(t))$ and with this in mind, can be expressed as the cumulative sum of ray weights $\mathbf{w}$ in the discrete case. With $(1 - \mathcal{T})' = \mathcal{T}\sigma$ in mind we find:

[1] Usually this Gaussian scale parameter is symbolically represented by σ, but in this case, ϵ is chosen to distinguish it from the differential opacity (density) $\sigma(\mathbf{x})$ used to model the scene.

$$
\begin{aligned}
F(0 \to t_b) &= \int_0^{t_b} (1 - \mathcal{T}(t))' \, dt \\
&= (\mathcal{T}(0) - \mathcal{T}(t_b)) \\
&= (1 - \mathcal{T}(0 \ \to t_b)) \quad \text{constant density} \\
&= \sum_{i=1}^{N_b} w_i
\end{aligned}
\tag{4.1}
$$

In conclusion: With this assumption that $f(t)$ can be modeled as a Gaussian, and by knowing D and ϵ, the weights along the ray through the neural field are fully defined by the cumulative distribution function of the underlying (Gaussian) probability density function.

With this knowledge, we set ϵ as a hyper-parameter in a Gaussian that slightly overestimates the scale of $\phi(\mu, \epsilon^2)$, which we are using to model $f(t)$, and assign $\Phi(\mu, \epsilon^2)$ to the corresponding continuous CDF. The ray interval is then divided into two regions for ray interval midpoints t_i. The near region is denoted $near : t_i < D$ and the far region denoted $far : t_i > D$. The loss for interval-midsections $t_i \in near$ is then a simple mean squared error evaluated on ray weights w_i that exceed the bound given by $\Phi(t_i \in near)$, and for interval-midsections $t_i \in far$ we penalize weights w_i which subceed $\Phi(t_i \in far)$.

Furthermore, if this assumption holds, then the empirical rule states that 99.97% of the opacity that a ray encounters traveling through the neural field should lie within a region of $\pm 3\epsilon_\Phi$, i.e. the probability that the ray has not been absorbed at a depth of $D + 3\epsilon_\Phi$ is under 0.03%. By setting ϵ to a small value, we can ensure concentrated and compact surface presentations in the neural field, but must be careful to not constrict the volume too much, as smaller ϵ_Φ leads to steeper gradients in the optimization, and further, we hypothesize that the expressiveness of neural fields stems partially from the volume rendering of expansive surfaces, rather than ones with δ-shaped densities.

For practical reasons it makes sense to divide the *near* region further, and here we take inspiration from priors introduced in [Rem+22], who themselves followed [CL96; Che+21; Mat+00], where we define $\{t_{empty} : t < D - 3\epsilon_\Phi\}$, and redefine $\{t_{near} : D - 3\epsilon_\Phi \ < t < D\}$. The Loss term $\mathcal{L}_{\mathcal{CDF}}$ for the statistical weight bounds for the optimization of parameters θ in $F_\theta : (\mathbf{x}, \mathbf{d}) \to (\mathbf{c}, \sigma)$ can then formally be written as:

$$
\mathcal{L}_{\mathcal{CDF}} = \lambda_{empty} \mathcal{L}_{empty} + \lambda_\Phi \mathcal{L}_\Phi \tag{4.2}
$$

This re-partition of intervals allows for individual tuning of λ_{empty} and λ_Φ. This allows setting $\lambda_{empty} >> \lambda_\Phi$ whereby densities in the *empty* region are strongly penalized and without creating training instabilities from exploding gradients of loss

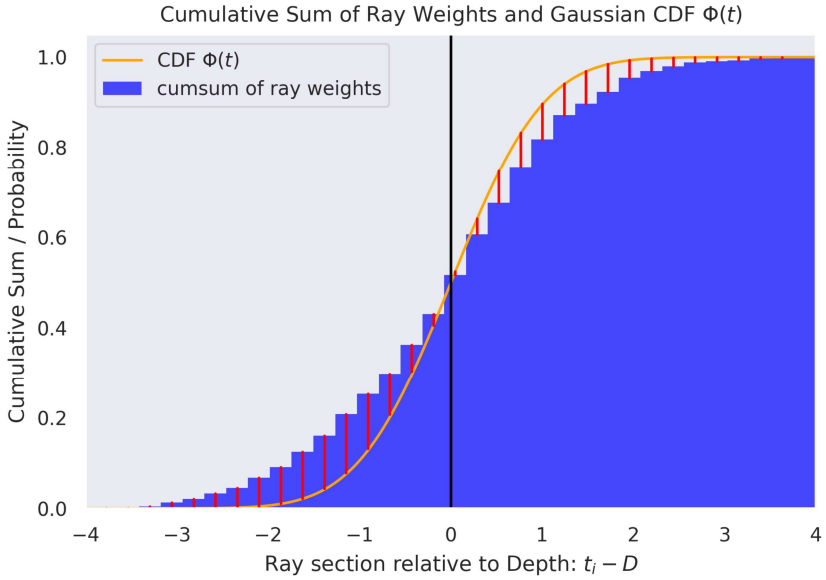

Figure 4.1.: Integrating over the likelihood $f(t)$, that a camera ray $\mathbf{r}(t)$ traversing the neural field F_θ along t terminates at exactly t, produces the Gaussian CDF $\Phi(t)$ if the densities along $f(t)$ are assumed to be normally distributed around D. The cumulative sum of ray weights can then be supervised by bounds given by $\Phi(t)$, where we penalize weights exceeding $\Phi(t)$ for $\{t_{near} : t - D < 0\}$, and for exceeding $\Phi(t)$ for $\{t_{far} : t - D > 0\}$.

incurred close to surfaces. With T_k as the number of indexes in each sum, the empty loss is defined as

$$\mathcal{L}_{empty} = \frac{1}{T_i} \sum_{t_i \in empty} w_i^2 \tag{4.3}$$

With $\mathcal{W}_i$ denoting the cumulative sum of weights w_i up to including i, the loss term for $\mathcal{L}_\Phi$ is then defined as follows:

$$\mathcal{L}_\Phi = \frac{1}{T_i} \sum_{t_i \in near} \max\{\mathcal{W}_i - \Phi(t_i), 0\}^2 + \frac{1}{T_j} \sum_{t_j \in far} \max\{\Phi(t_j) - \mathcal{W}_j, 0\}^2 \tag{4.4}$$

It should be noted that λ_Φ scales the loss contributed by each interval over the entire range linearly, while ϵ_Φ controls the slope of the Gaussian CDF in a non-linear way, and thereby the scale of how quickly deviations incur a penalty on the optimization.

As with most deep learning parameters, it is worth tuning ϵ_Φ to the application, as

setting it too low can lead to nonoptimal density distributions impacting performance, while setting it too high makes benefits from depth supervision weaker. Finally, far is set to being an open interval, as it encourages fully opaque surfaces in the vicinity of the depth measurement, but we found that the impact over limiting it to a multiple of ϵ_Φ was negligible.

4.2. Adaptation to Real Data

Real data generated through physical processes are never perfect, and as such, we introduce a slight modification of the method. To model measurement uncertainty, we introduce an offset β. Instead of evaluating the bounds with one Gaussian CDF centered at D, we evaluate for the bounds two separate Gaussian CDFs positioned at $D \pm 2\epsilon_\Phi$ as depicted in Figure 4.2, such that, within a tolerance of the estimated depth uncertainty, the model's predicted densities aren't erroneously constrained.

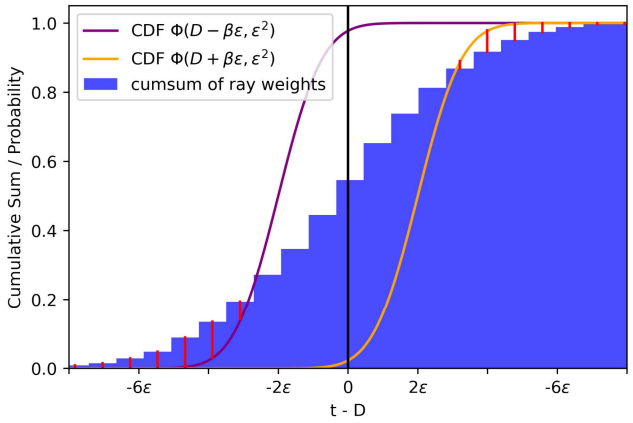

Figure 4.2.: To account for real-world sensor noise, and improper camera calibration, we adapt the loss presented in Figure 4.1 by introducing a parameter β, which acts as an X-Axis offset and describes the expected measurement error. We extend *near* and far regions by $\pm\beta\epsilon$ respectively. By supervising points in *near* with the upper bound given by $\Phi(D - \beta\epsilon, \epsilon^2)$, and points in far with the lower bound given by $\Phi(D + \beta\epsilon, \epsilon^2)$, the impact of measurement errors is effectively limited. In all real-world experiments we set $\beta = 2$ and for synthetic experiments, we set $\beta = 0$. Lower β generally leads to better results, if depth and camera data are well-calibrated and accurate.