IFFNeRF: Initialisation Free and Fast 6DoF pose estimation from a single image and a NeRF model

Matteo Bortolon[1,2,3], Theodore Tsesmelis[1], Stuart James[1,4], Fabio Poiesi[2] and Alessio Del Bue[1]

Abstract— We introduce IFFNeRF to estimate the six degrees-of-freedom (6DoF) camera pose of a given image, building on the Neural Radiance Fields (NeRF) formulation. IFFNeRF is specifically designed to operate in real-time and eliminates the need for an initial pose guess that is proximate to the sought solution. IFFNeRF utilizes the Metropolis-Hasting algorithm to sample surface points from within the NeRF model. From these sampled points, we cast rays and deduce the color for each ray through pixel-level view synthesis. The camera pose can then be estimated as the solution to a Least Squares problem by selecting correspondences between the query image and the resulting bundle. We facilitate this process through a learned attention mechanism, bridging the query image embedding with the embedding of parameterized rays, thereby matching rays pertinent to the image. Through synthetic and real evaluation settings, we show that our method can improve the angular and translation error accuracy by 80.1% and 67.3%, respectively, compared to iNeRF while performing at 34fps on consumer hardware and not requiring the initial pose guess. Project page: **https://mbortolon97.github.io/iffnerf/**

Fig. 1: From (*i*) a given image with an unknown pose and (*ii*) a NeRF model, we recover the pose by first (*iii*) sampling surface points using Metropolis-Hasting algorithm and (*iv*) casting rays from them in isocell distribution. We then (*iv/v*) correlate rays with the image to identify relevant rays using attention and (*vii*) recover the unknown 6DoF camera pose.

I. INTRODUCTION

Pose estimation is a cornerstone of robotic perception, that can also find application to autonomous vehicles and augmented reality. The accuracy of this estimated pose is key to several downstream tasks, *e.g.* in robotics, it can influence the capability to manipulate objects and navigate within an environment [1]–[4]. Recent advancements in neural and differentiable rendering, such as Neural Radiance Fields (NeRF) [5], have unveiled innovative and efficient ways to model real-world scenes. NeRF-based models can train generative models that capture the structure and appearance of both objects and scenes. They do that by leveraging images and their associated poses, eliminating the need for human supervision [5] and unlocking novel view synthesis (NVS) in complex scenarios. However, utilizing these models for subsequent camera re-localization in six degrees-of-freedom (6DoF) presents significant challenges. They typi-

cally demand an initial pose guess and rely on time-intensive optimization processes [6], [7].

Another benefit of NeRF-based approaches is that they can encode a broader range of appearance variations and eliminate the need for any mesh reconstruction [8]. iNeRF [6] pioneered 6DoF pose estimation using NeRF through an analysis-by-synthesis method. Because NeRF can render an image of a novel view based on a specified pose, the problem can be inverted to deduce the pose from an uncalibrated image. This optimization process entails minimizing the photometric error between the supplied image and the ones rendered from potential poses, iterating until an error minimum is achieved.

To address the limitations of previous works, we introduce IFFNeRF, an initialization-free method designed to estimate the 6DoF pose of a target image with respect to a NeRF model in real-time (Fig. 1). Our approach employs surface points sampled using the Metropolis-Hastings (M-H) algorithm [9] from within the volumetric NeRF model. From these sampled points, we cast a series of rays that collectively span a wide range of potential views. Each ray is defined by its origin, direction, and an associated surface color estimate, which is derived from pixel-level view synthesis. To narrow down the vast array of potential rays, we employ a learned attention map between the query image embedding and the embedding of each cast ray [10], enabling us to identify a subset of rays that are relevant to the image. This set of operations enables us to compute the target image pose in a closed-form way by solving a Least Squares problem.

[1]M. Bortolon, T. Tsesmelis, S. James, A. Del Bue are with Pattern Analysis and Computer Vision (PAVIS), Fondazione Istituto Italiano di Tecnologia (IIT), Genoa, Italy alessio.delbue@iit.it
[2]M. Bortolon and F. Poiesi are with Technologies of Vision (TeV), Fondazione Bruno Kessler (FBK), Trento, Italy mbortolon@fbk.eu, poiesi@fbk.eu
[3]M. Bortolon is with University of Trento, Trento, Italy
[4]S. James is with the Department of Computer Science, Durham University, UK stuart.a.james@durham.ac.uk
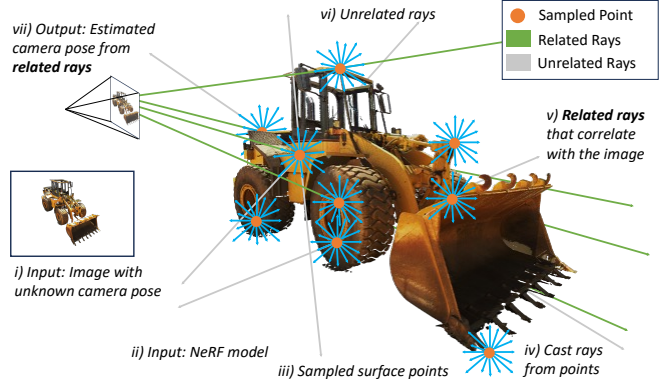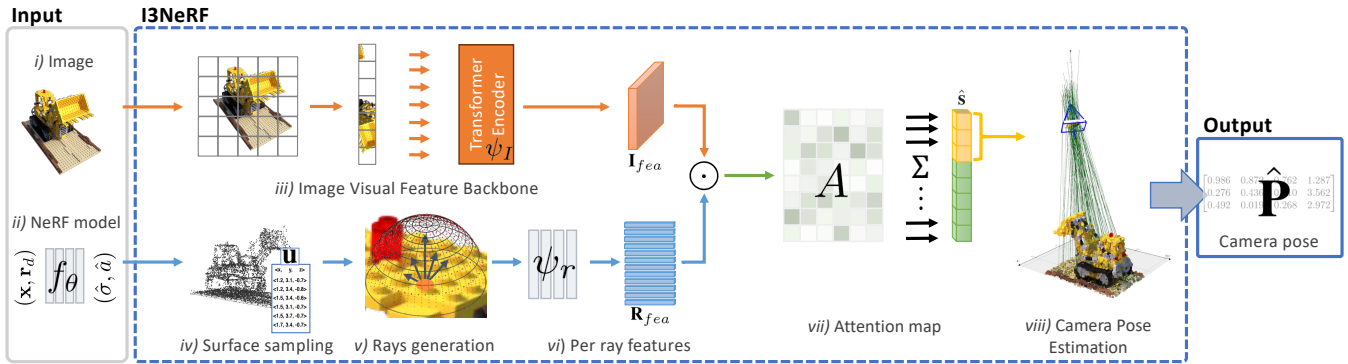
Fig. 2: Our IFFNeRF takes as input an *(i)* image, and *(ii)* a NeRF model. The model ψ_I encodes the image *(iii)* using a visual backbone. As for the NeRF model, we sample surface points *(iv)* using Metropolis-Hastings to identify candidates on the surface **u**, which can act as locations to project rays. We uniformly project rays *(v)* from the center isocells from the points, and estimate the ray corresponding parameters, color, and normal, using the NeRF model. We then embed the ray representation ψ_r *(vi)* and *(vii)* learn attention A between the ray embedding $\mathbf{I}_{fea}$ and $\mathbf{R}_{fea}$ to rank rays in relation to the image. We select the top-N rays and estimate the camera location *(viii)* using least squares, resulting in a 6DoF pose $\hat{\mathbf{P}}$ for the image.

We evaluate IFFNeRF on both synthetic and real objects, benchmarking it against iNeRF. Our experiments indicate that IFFNeRF consistently outperforms its competitor, in particular when we search the camera pose in the wild (either no initial pose is given or is randomly initialized). We further conduct an ablation study, scrutinizing different setups such as the attention map, backbone configurations, and various pose initializations. Lastly, in contrast to iNeRF, we illustrate that IFFNeRF operates in real-time and is more memory-efficient. In summary, our contributions are:

- Introducing a novel real-time method for estimating the 6DoF pose of a query image using a NeRF model;
- Eliminating the need for an initial camera pose, a requirement in iNeRF and similar methodologies;
- Proposing a new attention mechanism that proficiently evaluates the embeddings of cast rays and image pixels.

II. RELATED WORK

The groundbreaking NeRF work demonstrated the potential of neural scene representations for creating novel photorealistic views [5]. Since then, the research community have significantly pushed the boundaries of neural radiance fields to improve Novel View Synthesis (NVS) accuracy and rendering/training performance, and apply NeRF to other tasks, including 6D pose estimation [6].

Camera 6DoF pose estimation poses a critical perception challenge that can be tackled with deep learning or non-data driven methods [11], [12]. Among deep learning approaches, NeRF-based methods have recently gained relevant traction. iNeRF [6] iteratively tries to aligns a query image and a rendered image by optimizing camera pose based on photometric error. This demonstrates accurate pose estimation in controlled environments like synthetic or static indoor scenes. However, this accuracy comes with some shortcomings, including high inference time, inefficient pose updates, and dependence on accurate initial poses making it hard for iNeRF to perform outside of such controlled environments.

To address some of these issues, recent advancements have introduced parallel optimization using Monte-Carlo

sampling [7]. In this case, instead of a single candidate pose, they generate and try to optimize multiple poses at the same time with different initial poses, and then prune and regenerate them based on an optimization trend. This strategy improves convergence but does not eliminate the need for an initial camera pose while still manifesting high inference timings. Similarly, Loc-NeRF employs Monte-Carlo and particle filtering to localize a robot within an environment using a 3D NeRF map [13]. While this approach does not require prior pose information, it relies on multiple images and prior knowledge of their inter-image movement instead of a single image.

Alternative approaches, like CROSSFIRE [14], incorporate learned descriptors into the NeRF model. CROSSFIRE utilizes a joint training process of a visual backbone and the NeRF model to align 3D and 2D. During testing, CROSS-FIRE adopts an analysis-by-synthesis approach, minimizing feature discrepancies rather than color differences. However, both CROSSFIRE and parallel iNeRF need accurate initial camera positions to optimize for the actual camera pose.

While our method relies on a visual backbone to process the 2D image into features as CROSSFIRE, it does not follow an analysis-by-synthesis approach of a candidate image. Instead we propose an approach that only synthesizes a small volume of the 3D model, allowing us to address some of the aforementioned shortcomings.

III. PRELIMINARIES

IFFNeRF exploits NeRF formulation [5] whose objective is to synthesize novel views of a scene by optimizing a volumetric function f_θ from a given set of input images $\{\mathbf{I}\}_{i=0}^{N-1}$ and corresponding camera poses $\{\mathbf{P}\}_{i=0}^{N-1} \in \mathbb{R}^{3\times4}$. f_θ is parameterized through the weights of a Multi-Layer Perceptron (MLP). The input to f_θ is defined as $(\hat{\sigma}, \hat{\mathbf{a}}) \leftarrow f_\theta(\mathbf{x}, \mathbf{r}_d)$, where $\mathbf{x}$ is a 3D point, $\mathbf{r}_d$ is a viewing direction, $\hat{\sigma}$ is a density, and $\hat{\mathbf{a}}$ is a color. A ray marching method generates a set of rays starting from the camera's optical center and going through the scene. Rays are lines with a single point of origin $\mathbf{r}_o$ that extends infinitely in one

direction $\mathbf{r}_d$. We define a generic ray $\mathbf{r}(t) = \mathbf{r}_o + t\mathbf{r}_d$ where $t \in \mathbb{Z}$ is the position along ray. The locations are defined between two clipping distances near (t_n) and far (t_f). To make volumetric rendering computationally tractable, a finite set of 3D points is sampled along each ray $\{t_0, t_1, ..., t_{\Gamma-1}\}$, where Γ is the number of sampled locations. NeRF employs a volumetric rendering function $\hat{\mathbf{c}}(\mathbf{r})$ to convert a ray into a color:

$$\hat{\mathbf{c}}(\mathbf{r}, \Gamma) = \sum_{i=0}^{\Gamma-1} s(i) \left(1 - e^{-\hat{\sigma}(\mathbf{r}(t_i))\delta_i}\right) \hat{\mathbf{a}}(\mathbf{r}(t_i)), \quad (1)$$

where $\delta_i = t_{i+1} - t_i$ is the distance between adjacent sampled 3D points, and $s(i)$ is the inverse of the volume density that is accumulated up to the ith spatial location, which in turn is computed as

$$s(i) = e^{-\sum_{j=0}^{i-1} \hat{\sigma}(\mathbf{r}(t_j))\delta_j}, \quad (2)$$

where $(1 - e^{-\hat{\sigma}(\mathbf{r}(t_j))\delta_i})$ is a density-based weight component: the higher the density value $\hat{\sigma}$ of a point, the larger the contribution on the final rendered color. f_θ is therefore trained on a photometric loss minimizing $\mathcal{L} = \sum_r ||\mathbf{c} - \hat{\mathbf{c}}||_2^2$, where $\mathbf{c}$ is the true color of the pixel. In the standard NeRF formulation, the model does not predict any volume points and their normals on the object's surface. To obtain both the 3D points $(\mathbf{x})$ and their corresponding normals $(\hat{\mathbf{n}}(\mathbf{x}))$, we adopt the Ref-NeRF approach [15].

In more recent architectures [16], [17], the MLP is replaced by fast specialized data structures, these include hashmaps [17] and grids [16]. We take advantage of these speed improvements in our NeRF model by adopting the TensoRF work as our baseline [16].

IV. OUR APPROACH

IFFNeRF aims to predict the camera pose $\hat{\mathbf{P}} \in \mathbb{R}^{3 \times 4}$ given an observed image $\mathbf{I}$ and a pre-computed NeRF model f_θ (as define in Sec. III). We firstly apply a Metropolis-Hastings algorithm to sample G surface points within the scene volume (Sec. IV-A), then we cast V rays $\tilde{\mathbf{r}}$ from an isocell at each surface point (Sec. IV-B), thus obtaining $\mathbf{R}$ containing $G \times V$ rays. We then learn an attention map A between embeddings of the image $\psi_I(\mathbf{I})$ and generated rays $\psi_r(\mathbf{R})$ (Sec. IV-C). Based on the information contained in the attention map, we select a subset of candidate rays that are likely to fall within the image. Finally, to recover $\hat{\mathbf{P}}$ at test time, we optimize using Least Squares over the selected rays (Sec. IV-D).

A. Surface point sampling

We begin by extracting from the 3D points $\mathbf{x}$ a number G of points $\mathbf{u}$ on the surface of the object. To achieve this, we employ the Metropolis-Hastings (M-H) algorithm. M-H works by sampling from a distribution $\mathcal{Q}$, in our case this distribution is the NeRF density output $\mathcal{Q} = \hat{\sigma}$ [9]. We bootstrap the M-H algorithm by uniformly sampling points inside the bounding box that encloses the object's surface:

$$\mathbf{u}_0 = \boldsymbol{\beta} \otimes (\mathbf{B}_{max} - \mathbf{B}_{min}) + \mathbb{1}_G \otimes \mathbf{B}_{min}, \quad (3)$$

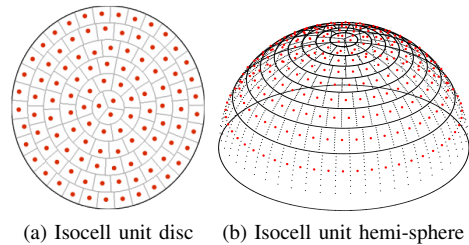(a) Isocell unit disc   (b) Isocell unit hemi-sphere

Fig. 3: Illustration of the Isocell ray generation method over a circular domain of a unit disk and unit sphere. The generated points indicate the ray positions within the equally spaced circle cells (we will denote them as "cell centres" for simplicity).

where $\mathbf{u}_0 \in \mathbb{R}^{G \times 3}$ are the sampled points at iteration 0, $\boldsymbol{\beta} = [\beta_0, \beta_1, ..., \beta_{G-1}]^T$ with each $\beta \sim \mathcal{U}[0, 1]$, $\mathbb{1}_G$ is a column vector of ones of length G, $\otimes$ is the Kronecker product, and $\mathbf{B}_{max}, \mathbf{B}_{min} \in \mathbb{R}^{1 \times 3}$ are the maximum and minimum 3D coordinates of the scene bounding box, respectively.

The M-H algorithm moves the point g, with $g = 1 \dots G$, only if $\mathcal{Q}(\mathbf{u}_{i+1,g}) \geq q$, where q is the acceptance ratio. If the condition is satisfied, $\mathbf{u}_{i+1,g}$ is a newly sampled random position. We define q as dependent from the Cumulative Distribution Function (CDF) Pr of the previous iteration:

$$q = Pr[Q_{i-1} < Q_{i-1,g}] \geq 0.6, \quad (4)$$

that indicates q as the 60th percentile, of the density of the selected points at the previous iteration. We define the points that are sampled at the last iteration of M-H as $\mathbf{u}$. These points are the ray origin, $\tilde{\mathbf{r}}_o$, for the following ray generation step, where we indicate with $\tilde{\mathbf{r}}$ the generated ray.

B. Rays generation from sampled surface points

From our sampled points $\mathbf{u}$, we cast a fixed number of rays in the direction oriented by the normal direction $\hat{\mathbf{n}}(\mathbf{u}_g)$, thus avoiding to cast rays toward the unobserved internal volume of the object. The normals $\hat{\mathbf{n}}(\mathbf{u})$ are obtained by querying the NeRF model. We adopt a deterministic method termed as Isocell [18], [19], which, compared to other commonly used rays sampling approaches, *e.g.* Monte-Carlo [20], achieves higher precision with fewer rays [21], [22]. The uniform ray distribution of an isocell partitions the surface of a unit sphere into equal area cells, where the cells center forms a uniform distribution on the unit sphere (Fig. 3). The distribution of an isocell provides the direction of the generated rays, identified as $\tilde{\mathbf{r}}_d$. Therefore, we cast V rays for each point $\mathbf{u}_g$, as shown in Fig. 4. We set $V = 27$ as we empirically found that it works well in practice.

The generated rays represent a collection of potential hypotheses, meaning that a subset of them will intersect the target image $\mathbf{I}$. In total, we have $N = V \times G$ generated rays $\tilde{\mathbf{r}}$ across the sampled points. Each generated ray have also a color $\tilde{\mathbf{r}}_c$, that it is computed through the same pixel-level approach of NeRF (Eq. 1). Note that, the application of the volumetric rendering function of Eq. 1 produces a single pixel for each ray. Moreover, if we assume an object is made of opaque surfaces, we can make our algorithm
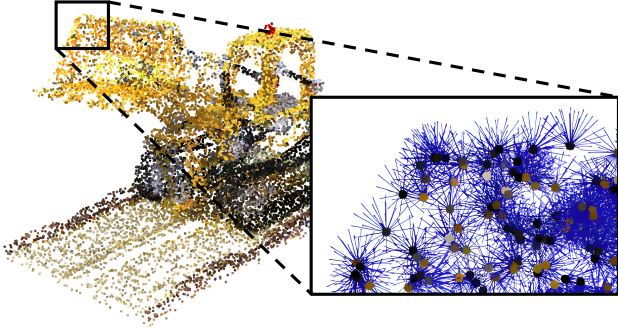
Fig. 4: Example of generated rays using our approach. The zoomed region highlights the ray cast operation (in this case 27 rays per isocell).

computationally more efficient by sampling only a few points located near the ray's origin (points that that are further away from the origin does not influence the rendered color).

In summary, the generated rays are defined by their origin $\tilde{\mathbf{r}}_o$, direction $\tilde{\mathbf{r}}_d$, and rendered color $\tilde{\mathbf{r}}_c$. To simplify the notation, we group and term these three components as $\tilde{\mathbf{r}}$. We can now leverage all the ray features to associate the rays with the image pixels.

C. Matching by attenuating cast rays to image

From all the cast rays $\tilde{\mathbf{r}}$, we identify a subset of them that maximally correlates with the given image $\mathbf{I}$ in order to recover the image pose ($\hat{\mathbf{P}}$). To avoid a time-consuming brute force search, we learn to score rays that are related to the image pixels. We achieve this by embedding ray parameters and the image into a higher-dimensional feature space.

Specifically, to enable our approach to discriminate rays that share similar appearance and position, we utilize a multi-layer perceptron (MLP) denoted as $\mathbf{R}_{fea} = \psi(\tilde{\mathbf{r}})$, where $\mathbf{R}_{fea} \in \mathbb{R}^{N \times C}$, and C denotes the number of channels. This MLP processes each ray independently, making the procedure insensitive to changes regardless how the rays are ordered. We project the MLP input into a higher-dimensional space using NeRF positional encoding [5]. This positional encoding expands subtle differences in the input, enabling the network to better distinguish similar rays [23].

To represent $\mathbf{I}$, we use DINOv2 [24] feature extractor, resulting in a set of features $\mathbf{I}_{fea} \in \mathbb{R}^{WH \times C}$, where W and H are the width and height of the image backbone's output, respectively. Then, we define an attention module, $\mathbf{M} \in \mathbb{R}^{W \times H \times C}$ to allow our model to correlate ray features, $\mathbf{R}_{fea}$, and image features, $\mathbf{I}_{fea}$, by using the image feature as query and the ray features as key: $\mathbf{M} = A(\mathbf{R}_{fea}, \mathbf{I}_{fea})$.

We optimize the attention map by summing along the rows and converting it into a correlation score, $\hat{\mathbf{s}}$ per ray as:

$$\hat{\mathbf{s}} = \sum_{k=1}^{WH} \mathbf{M}_k. \tag{5}$$

The ground-truth score $\mathbf{s}$ is defined based on the distance between the camera and its projection onto the considered ray. We can compute the projection of the point on the line with: $t = max((\mathbf{p} - \tilde{\mathbf{r}}_o)\tilde{\mathbf{r}}_d, 0)$, where $\mathbf{p}$ is the camera position, $\tilde{\mathbf{r}}_o$ the generated ray and $\tilde{\mathbf{r}}_d$ the corresponding direction. Rays

are infinite only in one direction, so we restrict $t \in \mathbb{R}^+$ using the max operator.

Then we can compute the distance between the camera origin and its projection on the ray as follows $\mathbf{d} = \|(\tilde{\mathbf{r}}_o + t\tilde{\mathbf{r}}_d) - \mathbf{p}\|_2$. $\mathbf{d}$ is normalize in $[0, 1]$ with:

$$\mathbf{d}_{\tilde{r}} = 1 - tanh\left(\frac{\mathbf{d}}{\lambda}\right), \tag{6}$$

where λ is a regularization parameter that adjusts the score values range. This affects how the rays are correlated and ranked based on the image features.

Lastly, we normalize all the scores $\mathbf{s}$ as follows:

$$\mathbf{s} = \mathbf{d}_{\tilde{r}} \frac{HW}{\sum_{j=0}^{HW} \mathbf{d}_{\tilde{r}, j}}. \tag{7}$$

During training our model optimizes $\hat{\mathbf{s}}$ based on $\mathbf{s}$ by using the $L2$ norm loss function.

D. Test-time pose estimation

At test-time the predicted scores $\hat{\mathbf{s}}$, are used as a correlation metric to select the best N_{top} rays, which should be the most relevant rays to the image, thus facilitating pose estimation. Fig. 5 shows an example of the selected rays and the score distribution. Note that only a small set of rays is sufficient to estimate the camera pose. However, we empirically set $N_{top} = 100$ for robustness (Fig. 5a).

Because the camera pose estimation can be seen as the intersection point between the selected rays, we compute the ray intersection by searching for the solution of a weighted linear system of equations. Naturally, a set of 3D lines will not intersect at a single point, especially considering the noise introduced by discretizing the rays directions in the isocell. To overcome this, we compute the Least Squares solution that minimizes the sum of the squared perpendicular distances.

Formally, for the generated ray, $\tilde{\mathbf{r}}_j$ with $j = 1 \ldots N_{top}$, the error is given by the square of the distance from point $\mathbf{p}$ to its projection on $\tilde{\mathbf{r}}_j$:

$$\sum_{j=0}^{N_{top}-1} \left((\mathbf{p} - \tilde{\mathbf{r}}_{o,j})^T(\mathbf{p} - \tilde{\mathbf{r}}_{o,j}) - ((\mathbf{p} - \tilde{\mathbf{r}}_{o,j})^T\tilde{\mathbf{r}}_{d,j})^2\right). \tag{8}$$

An efficient way to compute the least square solution to Eq. 8 is to compute its derivative respect to $\mathbf{p}$, resulting in

$$\mathbf{p} = \sum_{j=0}^{N_{top}-1} \hat{\mathbf{s}}_j \left(\mathbb{I} - \tilde{\mathbf{r}}_{d,j}\tilde{\mathbf{r}}_{d,j}^T\right)\tilde{\mathbf{r}}_{o,j}, \tag{9}$$

where $\mathbb{I}$ is the identity matrix and $\hat{\mathbf{s}}_i$ the predicted ray scores.

V. EVALUATION

We compare IFFNeRF to the baseline method iNeRF, with same backbone as ours[1]. Using iNeRF's protocol, we compare results on Synthetic NeRF [5] and, the real-world dataset, Tanks & Temples [25]. For each dataset, we use the predefined training-test splits and evaluate iNeRF with two

---

[1]Parallel iNeRF and CROSSFIRE [7], [14] code and evaluation protocol were not available at the time of submission. Evaluating our method on their testing data was also impractical as testing samples in [7] are sampled randomly from larger dataset and the samples information is not available either.

(a) Synthetic NeRF

Method	Drums MAE	MTE	Chair MAE	MTE	Lego MAE	MTE	Ship MAE	MTE	Ficus MAE	MTE	Mic MAE	MTE	Hot Dog MAE	MTE	Materials MAE	MTE	Mean MAE	MTE	Time (s)
iNeRF (initialization by [6])	17.4	0.430	15.4	0.441	12.7	0.293	9.8	0.344	18.1	0.485	17.6	0.501	3.8	0.061	15.1	0.375	13,7	0.366	6.1
iNeRF (Random initialization)	74.2	1.493	100.5	2.571	93.0	1.868	89.1	1.535	93.9	1.985	99.3	2.520	86.5	1.707	84.3	1.734	90.1	1.927	6.1
Ours (w. DINOv2 pretrained)	**22.5**	1.210	_13.2_	1.106	**11.8**	1.009	50.5	1.318	_19.4_	1.417	**10.7**	1.046	**10.5**	1.164	33.1	1.252	_21.5_	1.190	**0.029**
Ours (w. DINOv2 fine-tuned)	28.3	**0.916**	**7.1**	_0.429_	17.6	_0.547_	_25.7_	_0.600_	18.8	_0.818_	14.8	_0.576_	15.6	_0.650_	**15.4**	_0.493_	17.9	_0.629_	**0.029**
Ours (w. DINOv2 fine-tuned) w/iNeRF	_26.2_	_0.921_	25.7	**0.351**	18.4	**0.462**	23.1	**0.524**	27.8	**0.639**	_13.5_	**0.521**	_15.4_	**0.537**	23.4	**0.455**	21.7	**0.551**	6.2

(b) Tanks And Temples (real)

Method	Barn MAE	MTE	Caterpillar MAE	MTE	Family MAE	MTE	Ignatius MAE	MTE	Truck MAE	MTE	Mean MAE	MTE	Time (s)
iNeRF (initialization by [6])	26,5	0,208	42,9	0,166	42,8	0,794	31,4	0,723	31,6	0,370	35,0	0,452	6.1
iNeRF (Random initialization)	89.2	0.682	89.3	2.559	93.9	1.505	84.1	1.489	94.4	1.042	90.2	1.455	6.1
Ours (w. DINOv2 pretrained)	**22.3**	0.121	33.2	0.472	24.3	0.638	**15.6**	0.495	22.3	0.140	_23.5_	0.373	**0.029**
Ours (w. DINOv2 fine-tuned)	_26.9_	0.060	_30.4_	_0.427_	_22.1_	0.585	18.7	0.455	**20.7**	0.104	23.7	_0.326_	**0.029**
Ours (DINOv2 fine-tuned) w/iNeRF	_26.9_	**0.037**	**28.3**	**0.395**	**18.1**	**0.561**	19.5	**0.297**	_21.9_	**0.096**	22.9	**0.277**	6.2

TABLE I: Evaluation of 6DoF pose estimation on two datasets: (a) Synthetic NeRF [5] and (b) Tanks & Temples [25]. We report the results in terms of Mean Angular Error (MAE) and Mean Translation Error (MTE). The lower MAE and MTE, the better. Best-performing results are highlighted in **bold** and second-best results are underlined.
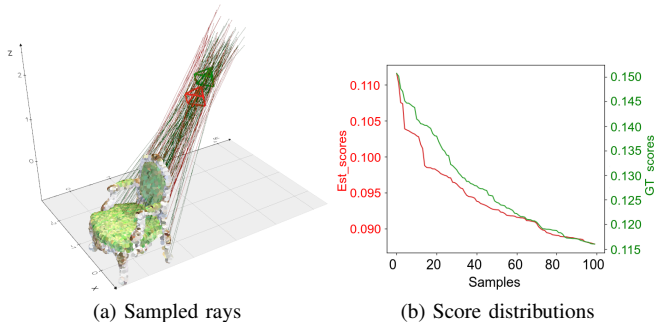

(a) Sampled rays   (b) Score distributions

Fig. 5: Example of the test-time pose estimation on the Chair object of Synthetic NeRF. (a) Top N (red) *vs.* top N ground-truth (green) rays. (b) Corresponding score distribution of top N and ground-truth rays.

initializations: *i)* the initialization proposed by iNeRF, where a given starting pose is between $[-40°, +40°]$ degrees and of $[-0.2, +0.2]$ units of rotational and translation error from the ground-truth target pose, respectively; *ii)* the initialization obtained by randomly selecting a pose from the ones available in the training set. Additionally, we conduct an ablation study on IFFNeRF's backbone configuration (pretrained *vs.* fine-tuned) and the use of IFFNeRF as an initialization for iNeRF. Evaluation metrics include mean angular (MAE) and translation (MTE) errors, and we measure the inference time in Tab. I.

Implementation Details: We implement IFFNeRF with PyTorch and train for 1.5K iterations ($\sim$45mins) on the training split on a standard workstation with a NVIDIA GeForce RTX 3090. We use the Adam optimizer with learning rate 10^{-3}. Hyper-parameters are set as $\lambda = 1$, $G = 5000$ and 800 M-H iterations.

A. Datasets

We use two datasets: *i) Synthetic NeRF* [5]: This dataset was released together with NeRF. Synthetic NeRF comprises eight scenes (Chair, Drums, Ficus, Lego, Materials, Ship, Mic, Hot Dog), includes both camera intrinsic and extrinsic parameters, provides train and test splits of 100 images (33%) and 200 images (66%), respectively. *ii) Tanks & Temples* [25]: This dataset was originally created to evaluate 3D reconstruction methods and embeds challenging scenarios as real-world objects with variations in size (small and large), acquired from a human-like viewpoints, at various distances from the object, and with different illumination conditions, such as shadows. As in [16], we evaluate on five scenes: Barn, Caterpillar, Family, Ignatius and Truck. We use the train and test splits as specified in [26], with the split depending on the object, having on average ≈ 247 training images (87%) and ≈ 35 testing images (12%). Like [7], we resize all the objects for both datasets to fit inside a unit box. Therefore, the translation error is relative to the object size and we define it in units.

B. Discussion on results

We report the quantitative results in Tab. I and the qualitative results in Fig. 6 for both datasets.

Effect of initialization on iNeRF: In both Synthetic NeRF (Tab. Ia) and Tanks and Temples (Tab. Ib), iNeRF performs best when initialized from poses near the known camera (following [6] evaluation). However, in more realistic settings with random initialization, the relative error is $21.2°$ degrees and 0.218 units worse in terms of angular and translation errors, respectively.

Synthetic & real datasets: Tab. Ia shows that our IFFNeRF consistently outperforms iNeRF with random initialization reducing the average angular error by $72.2°$ degrees and the translation by 1.29 units, respectively, in the case of the fine-tuned backbone. The same significant performance improvement can be observed in Tanks & Temples (real). Tab. Ib shows a $66.5°$ and 1.129 improvement for rotation
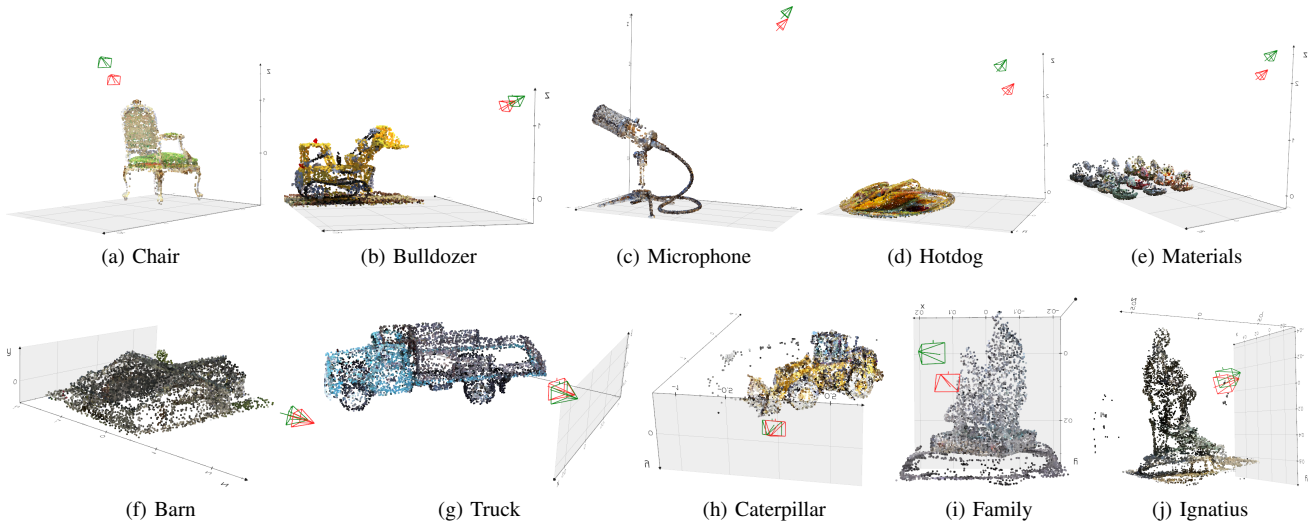
Fig. 6: Examples of camera pose estimation results of our approach on 10 different objects from both the synthetic NeRF (1st row) and the Tanks and Temples (2nd row) datasets. Green indicates the ground-truth camera and red indicates the estimated camera.

and translation errors, respectively. These results suggest that IFFNeRF can be resilient to images taken from a wide range of camera poses. Conversely, iNeRF with the initialization bounds specified in [6], on synthetic data, consistently achieves better results. However, this behavior of iNeRF is not confirmed with real data, whereas IFFNeRF clearly stands out with better performance while not requiring constrained initialization. Furthermore, applying iNeRF as a refinement step for IFFNeRF's solution leads to improved results, especially on real data.

Comparison of backbones and noise robustness: The effect of pretrained *vs.* fine-tuned backbones depends on the dataset. In Synthetic NeRF (Tab. Ia), we can observe a significant improvement with the fine-tuned backbone for both rotation and translation errors. However, in Tanks & Temples (Tab. Ib), we can observe that only the translation error slightly decreases by 0.047, while the rotation error increases by 0.2. Such small variation may be caused by the pretrained DINOv2 as it can already extract highly informative features from real-world data. Our architecture seems also to be robust to noise, without re-training in the NeRF-synthetic dataset, and despite introducing a 50% Gaussian noise to the image, the errors increased only by $14.01°$ and $0.201u$. Synthetic data might represent a domain shift because it is unlikely that they were used during training.

IFFNeRF as initialization to iNeRF: As the output pose of IFFNeRF provides a near estimate of the solution, we can also use this as an initialization for iNeRF optimization process. It can be seen that in Synthetic NeRF, in contrast to our best-performing result, only the mean translation error decreases by 0.078, whereas the angular error increases by 3.8. In Tanks and Temples, both translation and angular errors decrease by 0.049 and 0.6 respectively. However, it is interesting to observe that, in contrast to iNeRF with its original initialization [6], the combination of IFFNeRF and iNeRF performs relatively poorly on synthetic (by 15.2 and

0.399 higher angular and translation error, respectively), but significantly better on real-world data. This could be due to the fact that real data present shadows in the scenes. The camera and its handling system is not transparent, as in the synthetic data. These generated shadows seem to affect iNeRF, especially in the case of lateral translation.

Computational performance: In terms of inference time, IFFNeRF is significantly more efficient than iNeRF, achieving real-time performance at 34fps. In the case of IFFNeRF with iNeRF optimization, we revert to the running time of iNeRF; however, as noted in the case of real-world datasets (*i.e.* Tanks & Temples) this step further improves accuracy. In addition, it is worth mentioning that our method is also faster than the CROSSFIRE [14] and Parallel-iNeRF [7], where on the same hardware they report 200ms and 20s, respectively, compared to 29ms of our solution. A Limitation of IFFNeRF is to require training for each scene. Finally, in terms of memory, at test time IFFNeRF only uses 3294 MB of memory as opposed to the 4870 MB of iNeRF.

VI. CONCLUSIONS

We have proposed a ray sampling by attention method for estimating 6DoF camera poses given a single image and a NeRF model of the scene. Our experimental evaluation shows that IFFNeRF can achieve surprising results without the need for an initialization while being faster and requiring fewer memory resources. The method advantages stem from the ray generation strategy that can efficiently sample a wide range of camera poses hypothesis coupled with the efficiency of the attention module that maps rays to image pixel features. The proposed method achieves improved robustness over both synthetic and real-world datasets, while it can also be adopted for real-time applications in robotics and other fields. Future research will be focused on improving the accuracy between the registered model and the observed target image together with generalising the model to account for several scenes and objects.

REFERENCES

[1] P. Marion, P. Florence, L. Manuelli, and R. Tedrake, "Label fusion: A pipeline for generating ground truth labels for real rgbd data of cluttered scenes," in *ICRA*, 2018. 1

[2] L. Manuelli, W. Gao, P. R. Florence, and R. Tedrake, "kpam: Keypoint affordances for category-level robotic manipulation," in *ISRR*, 2019. 1

[3] Y. Xu, W. Wan, J. Zhang, H. Liu, Z. Shan, H. Shen, R. Wang, H. Geng, Y. Weng, J. Chen, *et al.*, "Unidexgrasp: Universal robotic dexterous grasping via learning diverse proposal generation and goal-conditioned policy," in *CVPR*, 2023. 1

[4] S. Rajeev, Q. Wan, K. Yau, K. Panetta, and S. Agaian, "Augmented reality-based vision-aid indoor navigation system in gps denied environment," in *Mobile Multimedia/Image Processing, Security, and Applications*, 2019. 1

[5] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," in *ECCV*, 2020. 1, 2, 4, 5

[6] L. Yen-Chen, P. Florence, J. T. Barron, A. Rodriguez, P. Isola, and T.-Y. Lin, "iNeRF: Inverting neural radiance fields for pose estimation," in *IROS*, 2021. 1, 2, 5, 6

[7] Y. Lin, T. Müller, J. Tremblay, B. Wen, S. Tyree, A. Evans, P. A. Vela, and S. Birchfield, "Parallel inversion of neural radiance fields for robust pose estimation," in *ICRA*, 2023. 1, 2, 4, 5, 6

[8] M. Bortolon, A. Del Bue, and F. Poiesi, "VM-NeRF: Tackling Sparsity in NeRF with View Morphing," in *ICIAP*, 2023. 1

[9] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *The journal of chemical physics*, vol. 21, no. 6, 1953. 1, 3

[10] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *NeurIPS*, 2017. 1

[11] Y. Zhu, M. Li, W. Yao, and C. Chen, "A review of 6d object pose estimation," in *ITAIC*, 2022. 2

[12] G. Marullo, L. Tanzi, P. Piazzolla, and E. Vezzetti, "6d object position estimation from 2d images: a literature review," *Multimedia Tools and Applications*, vol. 82, no. 16, 2023. 2

[13] D. Maggio, M. Abate, J. Shi, C. Mario, and L. Carlone, "Loc-nerf: Monte carlo localization using neural radiance fields," in *ICRA*, 2023. 2

[14] A. Moreau, N. Piasco, M. Bennehar, D. Tsishkou, B. Stanciulescu, and A. de La Fortelle, "Crossfire: Camera relocalization on self-supervised features from an implicit representation," in *ICCV*, 2023. 2, 4, 6

[15] D. Verbin, P. Hedman, B. Mildenhall, T. Zickler, J. T. Barron, and P. P. Srinivasan, "Ref-nerf: Structured view-dependent appearance for neural radiance fields," in *CVPR*, 2022. 3

[16] A. Chen, Z. Xu, A. Geiger, J. Yu, and H. Su, "Tensorf: Tensorial radiance fields," in *ECCV*, 2022. 3, 5

[17] T. Müller, A. Evans, C. Schied, and A. Keller, "Instant neural graphics primitives with a multiresolution hash encoding," *ACM Trans. Graph.*, vol. 41, no. 4, 2022. 3

[18] L. Masset, O. Brüls, and G. Kerschen, "Partition of the circle in cells of equal area and shape," Structural Dynamics Research Group, Aerospace and Mechanical Engineering Department, University of Liege, 'Institut de Mecanique et Genie Civil (B52/3), Tech. Rep., 2011. 3

[19] B. Beckers and P. Beckers, "Fast and accurate view factor generation," in *FICUP, An International Conference on Urban Physics*, 2016. 3

[20] T. Malley, "A shading method for computer generated images," *Master's thesis, Dept. of Computer Science, University of Utah*, 1988. 3

[21] L. Jacques, L. Masset, and G. Kerschen, "Direction and surface sampling in ray tracing for spacecraft radiative heat transfer," *Aerospace Science and Technology*, vol. 47, 2015. 3

[22] T. Tsesmelis, I. Hasan, M. Cristani, A. D. Bue, and F. Galasso, "Rgbd2lux: Dense light intensity estimation with an rgbd sensor," in *WACV*, 2018. 3

[23] M. Tancik, P. P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. T. Barron, and R. Ng, "Fourier features let networks learn high frequency functions in low dimensional domains," in *NeurIPS*, 2020. 4

[24] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, M. Assran, N. Ballas, W. Galuba, R. Howes, P.-Y. Huang, S.-W. Li, I. Misra, M. Rabbat, V. Sharma, G. Synnaeve, H. Xu, H. Jegou, J. Mairal, P. Labatut, A. Joulin, and P. Bojanowski, "Dinov2: Learning robust visual features without supervision," *arXiv:2304.07193*, 2023. 4

[25] A. Knapitsch, J. Park, Q.-Y. Zhou, and V. Koltun, "Tanks and temples: Benchmarking large-scale scene reconstruction," *ACM Transactions on Graphics*, vol. 36, no. 4, 2017. 4, 5

[26] L. Liu, J. Gu, K. Z. Lin, T.-S. Chua, and C. Theobalt, "Neural sparse voxel fields," in *NeurIPS*, 2020. 5