

Vorlesung 11: Neural Radiance Fields

BA-INF 153: Einführung in Deep Learning für Visual Computing

Prof. Dr. Reinhard Klein

Nils Wandel

Informatik II, Universität Bonn

10.07.2024

Wiederholung: Transformer

letzte Woche

- ① Transformer
- ② Image GPT
- ③ Vision Transformer
- ④ SWIN Transformer
- ⑤ CLIP

Inhalt

Heute

- ① Neural Radiance Fields (NeRF)
- ② Instant NGP (Neural Graphics Primitives)
- ③ Präsentation des Transfer Center enaCom

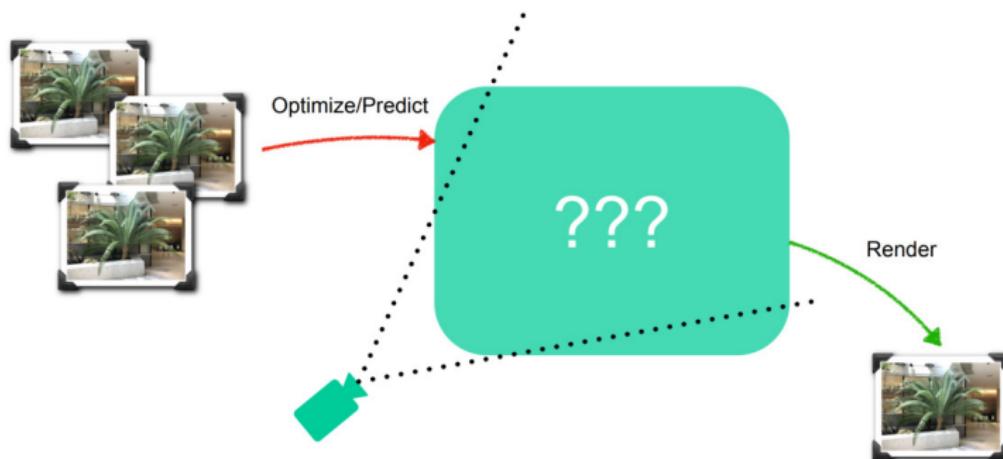
Abschnitt 1

Neural Radiance Fields (NeRF)

Wie können wir komplexe 3D Szenen repräsentieren?

Ziel

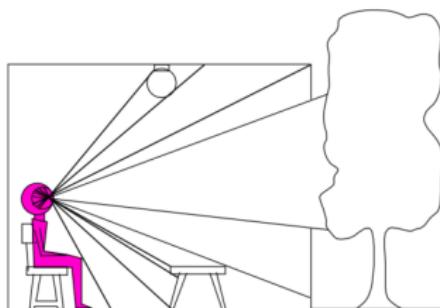
Zu gegebenen Bildern einer 3D Szene möchten wir neue Bilder / Blickrichtungen synthetisieren (Novel-View Synthesis).



⇒ Eine Szenen-Präsentation muss aus den Eingabebildern aufgebaut und gerendert werden.

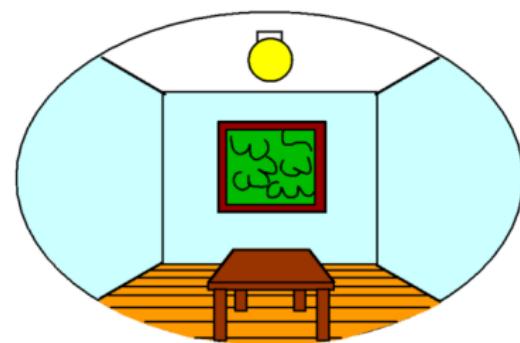
Was sehen wir?

3D world



Point of observation

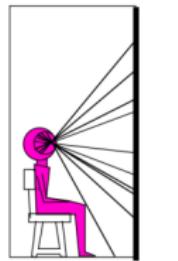
2D image



Lichtstrahlen treffen aus verschiedenen Richtungen aus der 3D Szene auf unsere Augen.

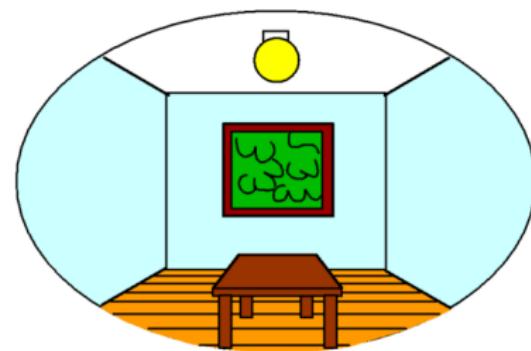
Was sehen wir?

3D world



Painted backdrop

2D image



Um eine 3D Szene auf einem 2D Bildschirm zu rendern, projizieren wir die Sehstrahlen auf die Bildebene des Bildschirms.

Plenoptische Funktion

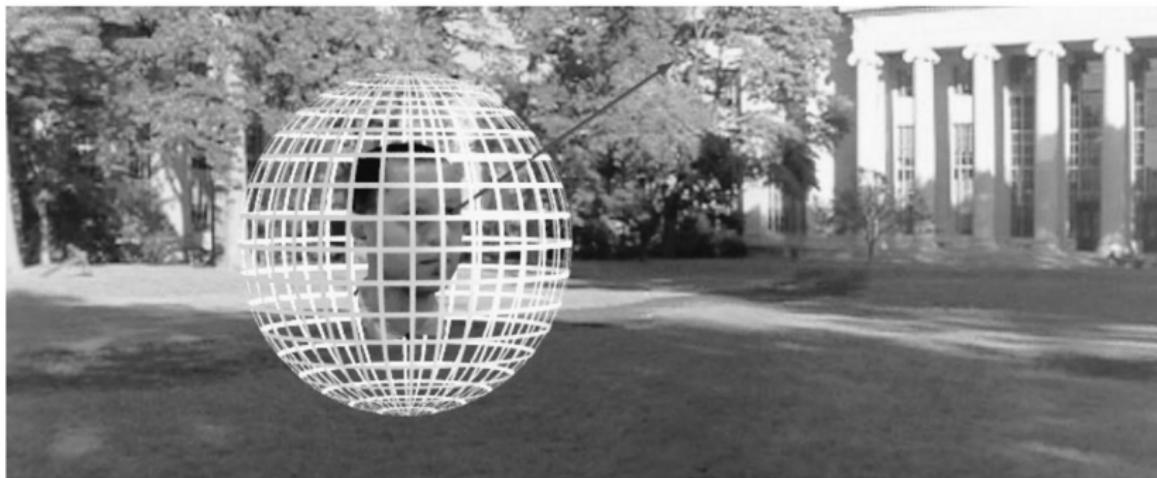
Die Plenoptische Funktion $P : (\theta, \phi, \lambda, t, V_x, V_y, V_z) \rightarrow \mathbb{R}$ ist ein allgemeines Konzept, mit der alles Sichtbare beschrieben werden kann.



Figure by Leonard McMillan

Im folgenden betrachten wir die Parameter von P etwas genauer.

Plenoptische Funktion - Blickrichtung



$$P(\theta, \phi)$$

θ und ϕ stellen die Blickrichtung dar. $P(\theta, \phi)$ gibt somit Helligkeitswerte für jede beliebige Blickrichtung zurück.

Plenoptische Funktion - Farben



$$P(\theta, \phi, \lambda)$$

Um Farben darstellen zu können, sollte P auch von der Wellenlänge des Lichts λ abhängen.

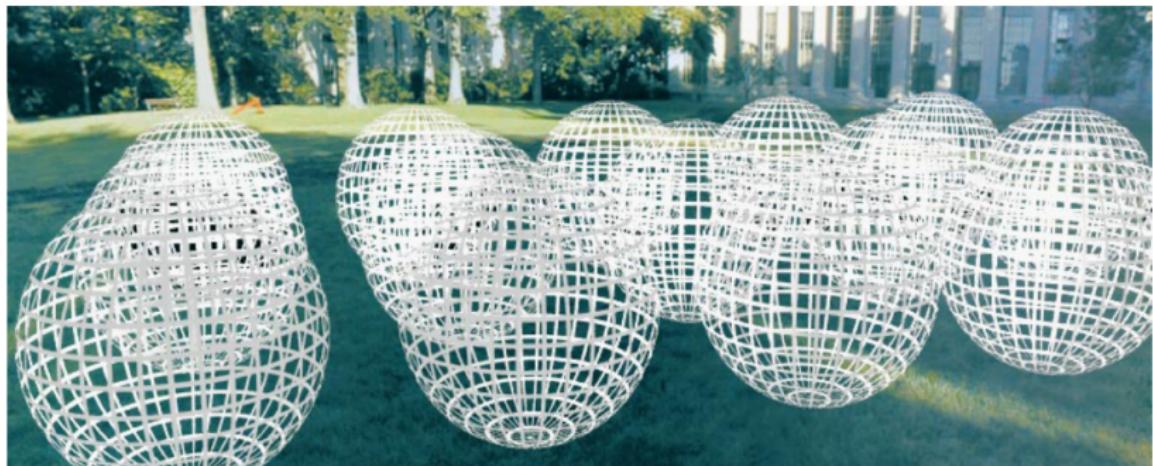
Plenoptische Funktion - Filme



$$P(\theta, \phi, \lambda, t)$$

Wenn sich die Szene über die Zeit ändert (z.B. in Filmen), dann muss P auch von der Zeit t abhängen.

Plenoptische Funktion - Beobachtungsort

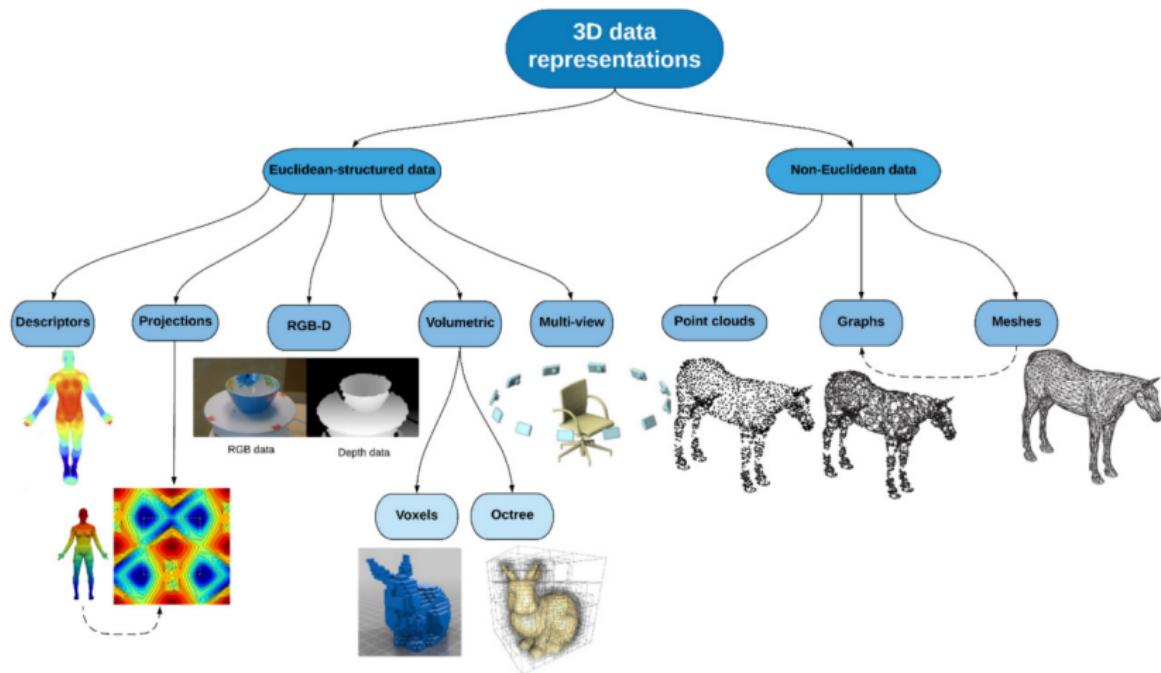


$$P(\theta, \phi, \lambda, t, V_x, V_y, V_z)$$

Des weiteren hängt P auch vom Beobachtungsort (V_x, V_y, V_z) ab. Dadurch können z.B. Tiefenschärfe, Stereo-Bilder oder holographische Bilder dargestellt werden.

3D Repräsentationen

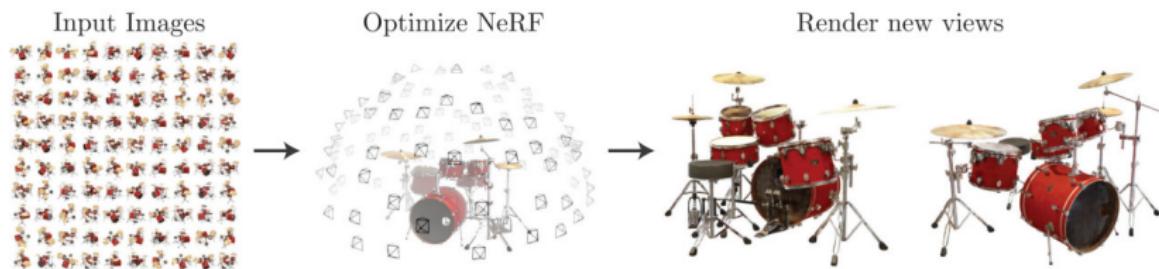
In der Computergrafik existieren zahlreiche Repräsentationen für 3D Daten.



... wir können dazu aber auch neuronale Netze verwenden.

Neural Radiance Field (NeRF)

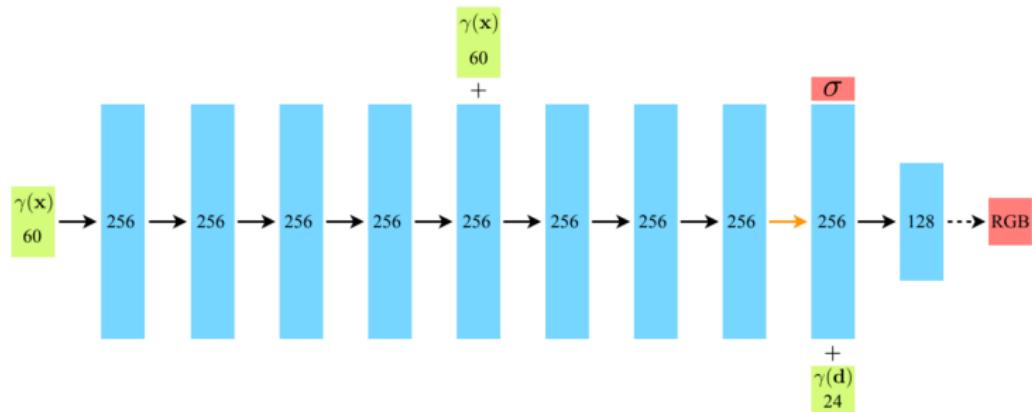
B. Mildenhall et Al: "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis" (2018)



Das Ziel von NeRFs ist, für gegebene Eingabebilder mit bekannten Positionen / Rotationen in 3D eine Szenenrepräsentation mit einem Neuronalen Netz zu lernen.

Neural Radiance Field (NeRF)

Idee: Verwende ein neuronales Netz (Neural Radiance Field NeRF), um eine Funktion $F_{\Theta} : (x, y, z, \theta, \phi) \rightarrow (R, G, B, \sigma)$ zu lernen, welche für jeden Punkt (x, y, z) in 3D einen "Dichtewert" σ ausgibt, sowie für jeden Punkt und Blickwinkel (x, y, z, θ, ϕ) einen Farbwert (R, G, B) .

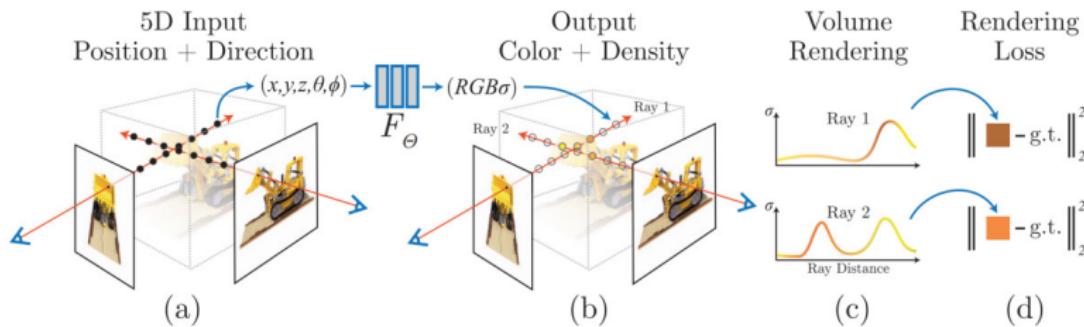


Anstatt (x, y, z, θ, ϕ) direkt als Eingabe zu verwenden, kann das Netzwerk mit Hilfe von Positional Encoding (ähnlich wie bei Transformern) besser feinere Details lernen:

$$\gamma(p) = (\sin(2^0 \pi p), \cos(2^0 \pi p), \dots, \sin(2^{L-1} \pi p), \cos(2^{L-1} \pi p))$$

Neural Radiance Field (NeRF)

Volume Rendering und Loss



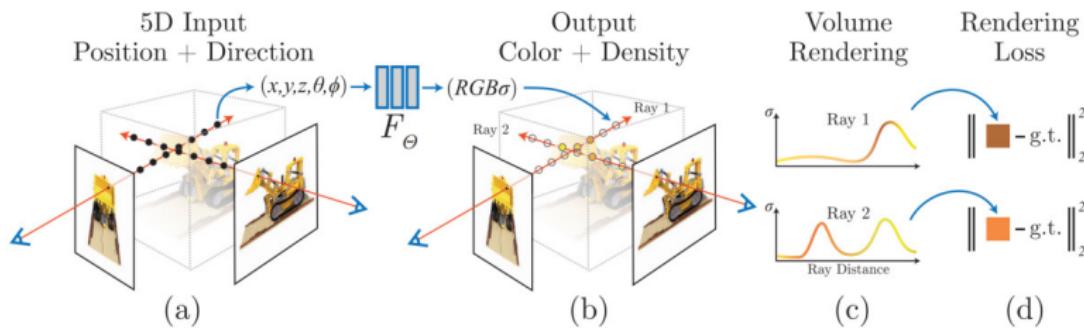
Um den Farbwert C eines Pixels im Bild zu bestimmen, schicken wir einen "Sehstrahl" (Ray) $r(t) = o + td$ in die 3D-Szene, wobei o dem Ursprung des Strahls (Auge) und d der Blickrichtung entspricht. Entlang dieses Strahls können wir nun die Farbwerte des NeRF aufintegrieren:

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

Hierbei werden die Farbwerte des NeRF mit der Dichte σ gewichtet und mit einem "Transmittance"-Faktor T multipliziert.

Neural Radiance Field (NeRF)

Volume Rendering und Loss



Um das Integral von $C(r)$ zu berechnen, samplen wir entlang des Strahls an diskrete Punkten t_i mit Abstand $\delta_i = t_{i+1} - t_i$:

$$\hat{C}(r) = \sum_{i=1}^N T_i (1 - \exp(-\sigma_i \delta_i)) c_i, \text{ wobei: } T_i = \exp\left(-\sum_{j=1}^{i-1} \sigma_j \delta_j\right)$$

Da dieser Volume Rendering Prozess differenzierbar ist, kann das NeRF mit Hilfe von Gradient Descent bezüglich eines Rendering-Losses (üblicherweise L_2 Distanz zu Ground Truth Bildern aus verschiedenen Blick-Richtungen) optimiert werden.

Neural Radiance Field (NeRF)

... warum der Betrachtungswinkel (θ, ϕ) wichtig ist:

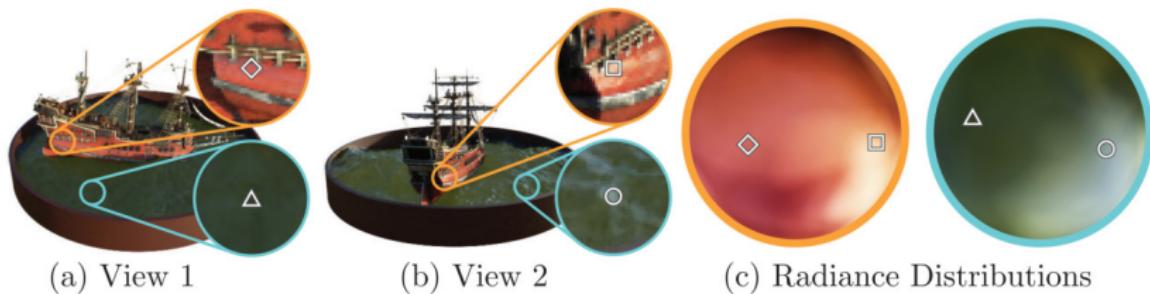


Figure: Insbesondere glänzende Oberflächen geben je nach Betrachtungswinkel unterschiedliche Farb- / Helligkeitswerte zurück.

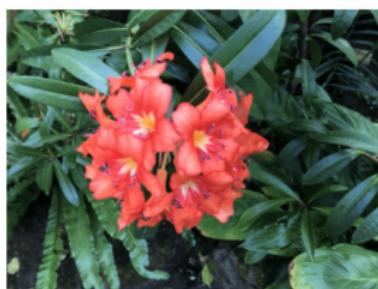
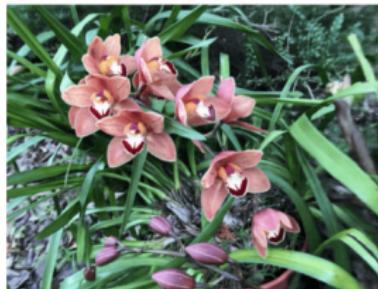
NeRF: Resultate

Mit neural Radiance Fields können auch komplexe Szenen-Repräsentationen gelernt werden:



NeRF: Resultate

Mit neural Radiance Fields können auch komplexe Szenen-Repräsentationen gelernt werden:



NeRF: Diskussion

Vorteile

- ① Sehr detailreiche Resultate und realistische novel view Synthese
- ② Auch glänzende Oberflächen können modelliert werden
- ③ Aus σ lässt sich 3D Geometrie schätzen

Nachteile

- ① Overfitting des neuronalen Netzes auf eine einzelne 3D Szene
- ② Training des neuronalen Netzes ist rechenintensiv. Dieses Problem wurde mittlerweile jedoch weitestgehend gelöst (siehe Instant-NGP)

Abschnitt 2

Instant Neural Graphics Primitives

Aus "Instant Neural Graphics Primitives with a Multiresolution Hash Encoding"
von Thomas Müller et al. (2022)

Instant NGP

Idee

Können wir NeRFs beschleunigen, indem wir (wie bei Transformern) das Positional Encoding lernen, anstatt sin und cos Kurven mit verschiedenen Frequenzen als Netzwerk-inputs zu verwenden?

- Ja, man könnte zum Beispiel lernbare Parameter in einem 3D Gitter für verschiedene Positionen anordnen.

Im Extremfall könnte auf diese Weise mit einem sehr hoch aufgelöstem Gitter für jeden Ort einen Parameter eingeführt werden, der z.B. die Dichte enthält - dann müsste das Netzwerk nur noch die Identitätsabbildung für σ lernen. Allerdings würde das sehr viel Speicher ($O(N^3)$) verbrauchen!

⇒ Kann dieses Problem intelligenter gelöst werden?

- Ja, mit Multiresolution Hash Encodings!

Instant NGP - Vergleich von Positional Encodings

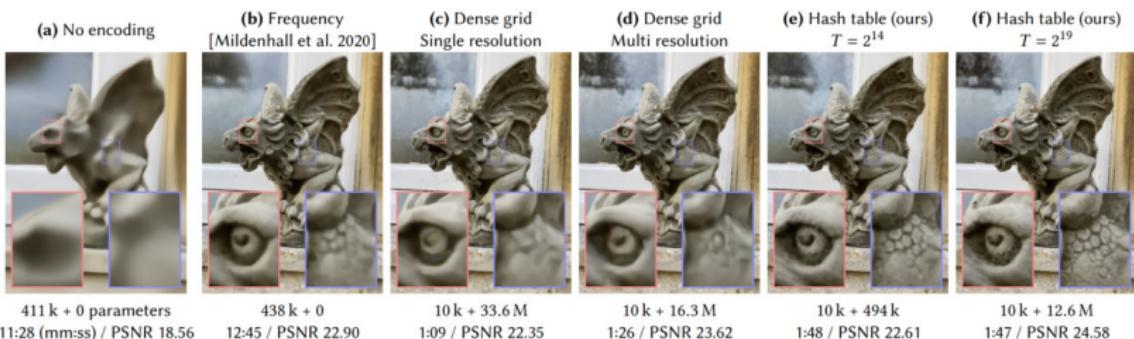


Fig. 2. A demonstration of the reconstruction quality of different encodings and parametric data structures for storing trainable feature embeddings. Each configuration was trained for 11 000 steps using our fast NeRF implementation (Section 5.4), varying only the input encoding and MLP size. The number of trainable parameters (MLP weights + encoding parameters), training time and reconstruction accuracy (PSNR) are shown below each image. Our encoding (e) with a similar total number of trainable parameters as the frequency encoding configuration (b) trains over 8x faster, due to the sparsity of updates to the parameters and smaller MLP. Increasing the number of parameters (f) further improves reconstruction accuracy without significantly increasing training time.

Figure: Vergleich von unterschiedlichen positional Encodings.

Instant NGP - Architektur

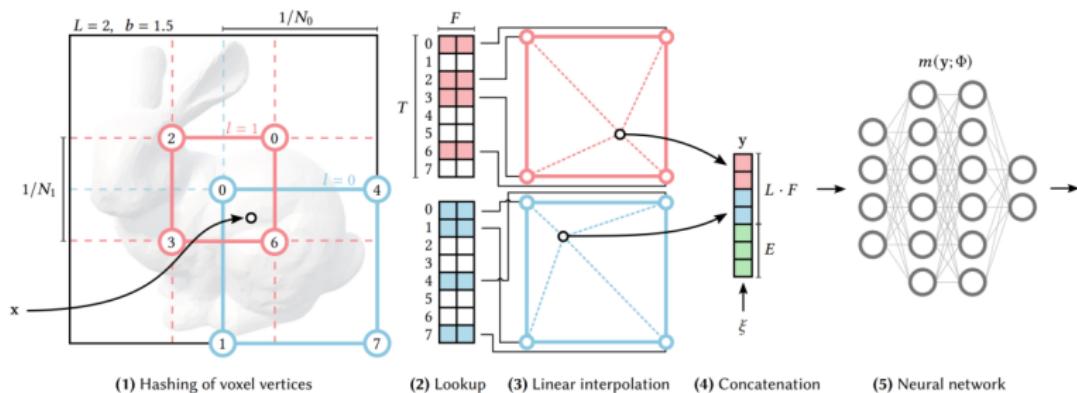


Fig. 3. Illustration of the multiresolution hash encoding in 2D. (1) for a given input coordinate x , we find the surrounding voxels at L resolution levels and assign indices to their corners by hashing their integer coordinates. (2) for all resulting corner indices, we look up the corresponding F -dimensional feature vectors from the hash tables θ_l and (3) linearly interpolate them according to the relative position of x within the respective l -th voxel. (4) we concatenate the result of each level, as well as auxiliary inputs $\xi \in \mathbb{R}^E$, producing the encoded MLP input $y \in \mathbb{R}^{LF+E}$, which (5) is evaluated last. To train the encoding, loss gradients are backpropagated through the MLP (5), the concatenation (4), the linear interpolation (3), and then accumulated in the looked-up feature vectors.

Figure: Architektur von Instant-NGP. Beispiel mit $L = 2$ Multi-resolution Layern (rot und blau markiert) und Hash-Tabellen der Größe $T = 8$ welche in jedem Eintrag einen $F = 2$ dimensionalen Feature Vektor enthalten. Im Gegensatz zu diskreten Tokens in Transformern benötigen NeRFs kontinuierliche positional encodings. Dies kann mit bilinearer Interpolation zwischen den Gitterpunkten erreicht werden. E entspricht zusätzlichen Hilfs-Inputs (z.B. der Blickrichtung). Schließlich berechnet ein (relativ kleines) MLP die Ausgabe (z.B. RGB-Farbwert und Dichte). [Quelle]

Instant NGP - Multi-Resolution

Bei L Multi-resolution Ebenen, wird die Auflösung N_l einer Ebene $l \in \{0, 1, \dots, L - 1\}$ gegeben durch:

$$N_l := \lfloor N_{\min} \cdot b^l \rfloor$$

Wobei b wie folgt definiert ist:

$$b := \exp \left(\frac{\ln(N_{\max}) - \ln(N_{\min})}{L - 1} \right)$$

Dies führt zu einer geometrischen Reihe von verschiedenen Layer-Auflösungen mit $N_0 = N_{\min}$ und $N_{L-1} = N_{\max}$. Typische Werte für b liegen zwischen $[1.26, 2]$ (im Gegensatz zu einem Octree kann b also auch kleinere Werte als 2 einnehmen). Für jede Auflösungs-Ebene wird eine separate Hash-Tabelle mit T lernbaren Encoding-Parametern $\in \mathbb{R}^F$ gespeichert.

Table 1. Hash encoding parameters and their ranges in our results. Only the hash table size T and max. resolution N_{\max} need to be tuned to the task.

Parameter	Symbol	Value
Number of levels	L	16
Max. entries per level (hash table size)	T	2^{14} to 2^{24}
Number of feature dimensions per entry	F	2
Coarsest resolution	N_{\min}	16
Finest resolution	N_{\max}	512 to 524288

Instant NGP - Positional Encoding mit Hashing

Um die passenden Werte aus der Hash-Tabelle zu lesen, wird jeder Gitterkoordinate x eine eindeutige, pseudo-zufällige Adresse mit folgender Hash-funktion zugeordnet:

$$h(x) = \left(\bigoplus_{i=1}^d x_i \pi_i \right) \bmod T$$

wobei $d = 3$ der Anzahl Dimensionen entspricht, π_i großen Primzahlen entsprechen und \bigoplus einer bit-weise XOR-Operation entspricht. Schließlich wird durch eine Modulo-Operation mit der Tabellen-Größe T garantiert, dass die Adressen innerhalb der Hashtabelle bleiben.

Um das Encoding für eine beliebige, kontinuierliche Koordinate zu erhalten, werden die Encodings der benachbarten Gitterpunkte (tri-)linear interpoliert.
Diese Encodings werden für alle Auflösungsebenen berechnet und konkateniert.

Instant NGP - MLP Architektur für NeRFs

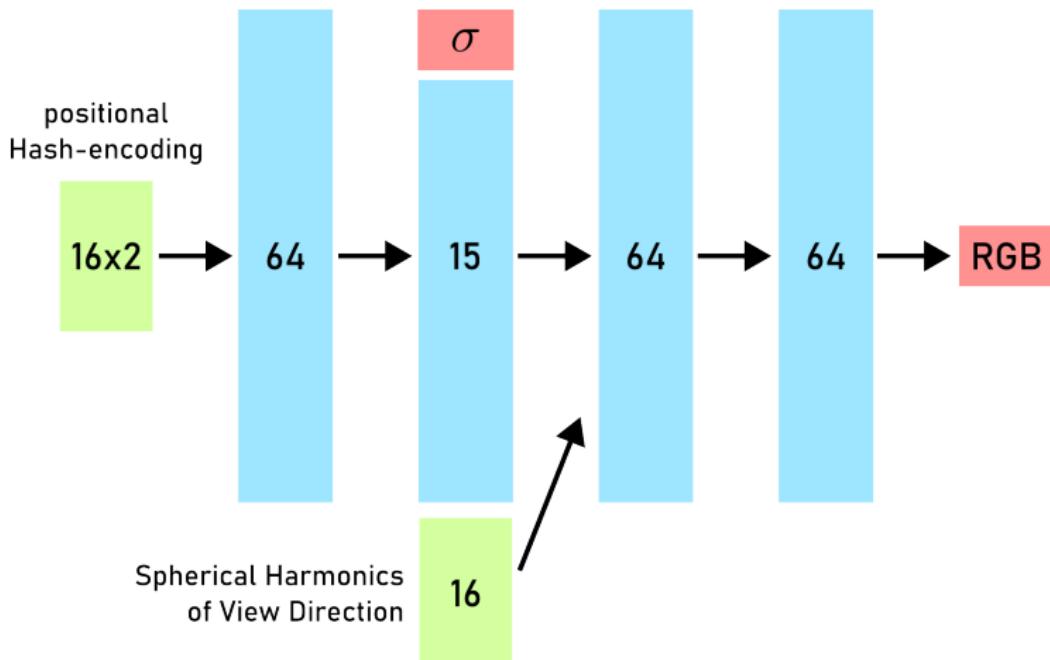


Figure: Architektur des multi-layered Perceptrons für Neural Radiance Fields in Instant NGP

Instant NGP - Blickrichtung

Zusätzlich werden weitere Hilfs-Inputs für die Blickrichtung konkateniert. Hierbei kommen die ersten 16 Basis-Funktionen der Spherical Harmonics zum Einsatz.

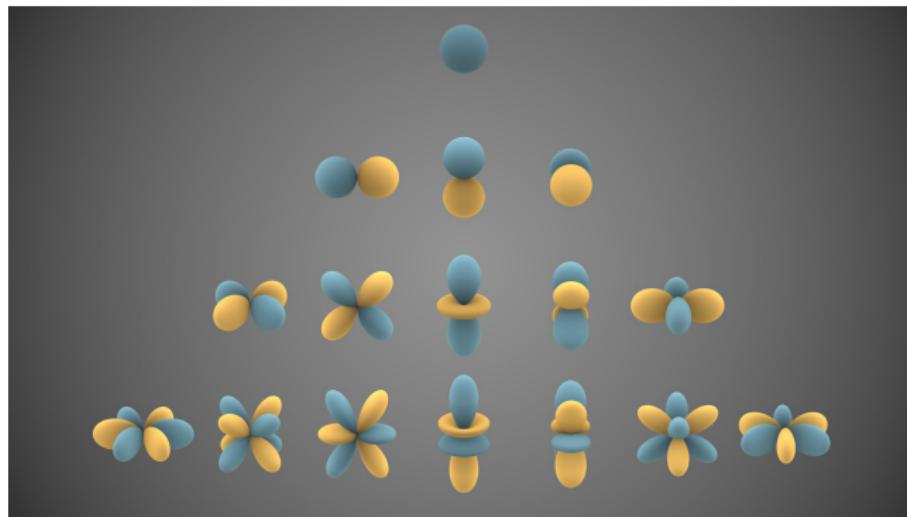


Figure: Visualisierung der ersten 16 spherical harmonics. [Quelle]

Instant NGP - Hash Kollisionen

Was passiert bei Hash-Kollisionen?

Das Netzwerk muss lernen, robust gegenüber Hash-kollisionen zu werden. Da die Kollisionen unabhängig auf den einzelnen Auflösungsebenen passieren, gibt es trotzdem für jede Koordinate ein eindeutiges positional Encoding.

Intuition: Große Bereiche im Bild haben keine Dichte. Hier kann das Netzwerk schon auf groben Auflösungen lernen, dass $\sigma = 0$ ist. Folglich haben die Hasheinträge auf den feineren Auflösungen in diesen Bereichen keinen Einfluss mehr auf die Netzwerkausgabe und nur sehr kleine Gradienten.

Diese Einträge auf den feineren Auflösungen können dafür dann mehr Einfluss auf die Netzwerkausgabe nehmen, wo viele Details gelernt werden müssen (z.B. auf der Oberfläche von Objekten).

Instant NGP - Resultate

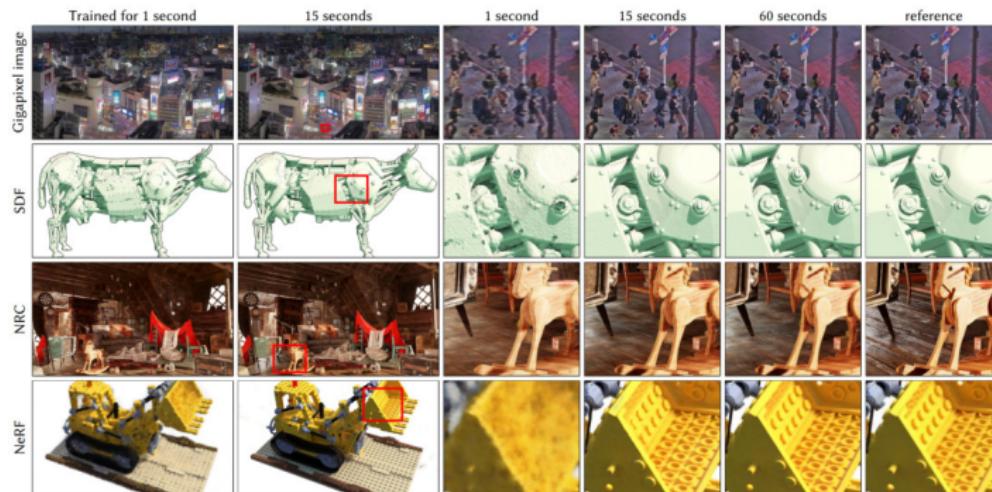


Fig. 1. We demonstrate instant training of neural graphics primitives on a single GPU for multiple tasks. In *Gigapixel image* we represent a gigapixel image by a neural network. *SDF* learns a signed distance function in 3D space whose zero level-set represents a 2D surface. Neural radiance caching (*NRC*) [Müller et al. 2021] employs a neural network that is trained in real-time to cache costly lighting calculations. Lastly, *NeRF* [Mildenhall et al. 2020] uses 2D images and their camera poses to reconstruct a volumetric radiance-and-density field that is visualized using ray marching. In all tasks, our encoding and its efficient implementation provide clear benefits: rapid training, high quality, and simplicity. Our encoding is task-agnostic: we use the same implementation and hyperparameters across all tasks and only vary the hash table size which trades off quality and performance. Tokyo gigapixel photograph ©Trevor Dobson (CC BY-NC-ND 2.0), Lego bulldozer 3D model ©Håvard Dalen (CC BY-NC 2.0)

Figure: Instant NGP erzeugt detaillierte Resultate bereits nach einer Trainingszeit von wenigen Sekunden. Außerdem lässt sich dasselbe Konzept auch zur Kompression von hochauflösten Bildern, für Signed Distance Functions (SDF) oder zum Neural radiance caching anwenden.