UHASSELT
KNOWLEDGE IN ACTION

Maastricht University

Faculteit Wetenschappen
School voor Informatietechnologie
master in de informatica

Masterthesis

Deap learning in beeld: van videobeelden naar 3D model

Samed Aliustaoglu
Scriptie ingediend tot het behalen van de graad van master in de informatica

PROMOTOR :
Prof. dr. Nick MICHIELS

BEGELEIDER :
De heer Bram VANHERLE

2022
2023

De transnationale Universiteit Limburg is een uniek samenwerkingsverband van twee universiteiten in twee landen: de Universiteit Hasselt en Maastricht University.

UHASSELT
KNOWLEDGE IN ACTION

www.uhasselt.be
Universiteit Hasselt
Campus Hasselt:
Martelarenlaan 42 | 3500 Hasselt
Campus Diepenbeek:
Agoralaan Gebouw D | 3590 Diepenbeek

Maastricht University

Faculteit Wetenschappen
School voor Informatietechnologie

master in de informatica

Masterthesis

Deap learning in beeld: van videobeelden naar 3D model

Samed Aliustaoglu

Scriptie ingediend tot het behalen van de graad van master in de informatica

PROMOTOR :

Prof. dr. Nick MICHIELS

BEGELEIDER :

De heer Bram VANHERLE

Universiteit Hasselt

Masterproef voorgedragen tot het behalen van de graad van master in de informatica

Deep Learning in Beeld: Van Videobeelden naar 3D Model

Auteur:

Samed Aliustaoglu

Promotor:

Prof. Dr. Nick Michiels

Co-promotor:

De heer Bram Vanherle

Begeleider(s):

Dr. Lode Jorissen
Dr. Jeroen Put

Academiejaar 2022-2023

Dankwoord

Het voltooien van deze thesis vormt de kers op de taart van een intensieve en verrijkende periode in mijn academische carrière aan de Universiteit van Hasselt. Het was een reis vol uitdagingen, en het afronden van dit masterthesis zou onmogelijk zijn geweest zonder de steun en begeleiding van vele betrokkenen.

Allereerst wil ik mijn begeleider, Bram Vanherle, en promotor, Nick Michiels, oprecht bedanken voor zijn onvoorwaardelijke steun, geduld en expertise. Zijn feedback en waardevolle inzichten hebben mijn onderzoek niet alleen naar een hoger niveau getild, maar gaven me ook richting wanneer ik het gevoel had de weg kwijt te zijn. Een speciale dank gaat uit naar Nick Michiels voor het organiseren van de maandelijkse feedbackmomenten met andere studenten. Deze sessies gaven mij een duwtje in de rug en stelden me in staat om directe feedback van peers te ontvangen. Bovendien ben ik het departement van Visual Computing & A.I. zeer dankbaar voor het ter beschikking stellen van een GPU toen de mijne het begaf.

Ik wil ook de Universiteit van Hasselt en het Departement Informatica bedanken voor de kans om aan zo'n boeiend onderwerp te werken en mijn kennis in dit veld te verdiepen.

Mijn familie, vrienden en collega's hebben een speciaal plekje in mijn hart voor alle liefde, steun en vooral hun geduld. Hun onvoorwaardelijke steun tijdens dit uitdagende jaar is niet onopgemerkt gebleven. Ze boden niet alleen morele steun, maar zorgden ook voor de nodige afleiding wanneer de druk te hoog opliep.

Tot slot wil ik iedereen die, direct of indirect, heeft bijgedragen aan het volbrengen van deze thesis bedanken. Dankzij jullie heb ik deze belangrijke mijlpaal in mijn leven kunnen realiseren. Het was een geweldig avontuur.

Samenvatting

Er is een innovatieve techniek in de wereld van computer graphics geïntroduceerd, die bekend staat als de Neural Radiance Field, of kortweg NeRF. Deze techniek heeft belofte getoond in het creëren van hoogwaardige renders. Dit proefschrift is gewijd aan een diepgaande analyse van de mechanismen en principes achter NeRF. Een centraal aspect van dit onderzoek is de vraag of het mogelijk is om een driedimensionaal model te extraheren op basis van de output van NeRF, in het bijzonder wanneer de primaire input een video is - een vorm van data die traditioneel niet direct wordt geassocieerd met NeRF-inputs.

Om dit ambitieuze doel te bereiken, zullen diverse disciplines binnen de informatica en verschillende technologieën moeten worden geïntegreerd. Dit omvat het proces van het winnen van afbeeldingen uit videobeelden, het segmenteren van deze afbeeldingen en vervolgens het construeren van een nauwkeurig driedimensionaal model op basis van deze segmenten.

Gedurende dit proefschrift wordt het volledige proces gedetailleerd beschreven. Elke stap, zowel vanuit een technisch als praktisch perspectief, wordt grondig onderzocht en toegelicht. Hierdoor krijgt de lezer een helder en diepgaand inzicht in de potentie en de uitdagingen van het integreren van NeRF-technologie met videobronnen.

Inhoudsopgave

Hoofdstuk 1

Inleiding

3D-modellering is steeds invloedrijker geworden in sectoren zoals gaming en film, mede door de opkomst van VR/AR-technologieën. Ondanks dat het maken van 3D-modellen tijdsintensief en computationeel veeleisend is, bieden technologieën zoals machine learning kansen voor optimalisatie. Deze thesis onderzoekt de mogelijkheid om vanuit een video een 3D-model van een voorgrondobject te genereren met als doel 3D-modellering toegankelijker te maken. Hierbij worden diverse technieken gebruikt, waaronder NeRF's radiance field en 'salient object detection'. In deze digitale tijd worden we constant geconfronteerd met afbeeldingen en videos. De uitdaging is dan ook om deze beelden om te zetten naar 3D voor een meer meeslepende ervaring en het uitvoeren van geavanceerde simulaties.

De voorgestelde pijplijn omvat verschillende opeenvolgende stappen:

1. **Extractie van Afbeeldingen uit Video's**: Het primaire bronmateriaal wordt verwerkt om individuele frames of afbeeldingen te isoleren. Dit zorgt voor een gestandaardiseerde basis voor verdere analyse.

2. **Bepaling van Camera Positie en Rotatie met COLMAP**: Voor het accuraat genereren van een 3D model is het cruciaal om te weten vanuit welk perspectief elk beeld is genomen. COLMAP wordt ingezet om automatisch de positie en rotatie van de camera voor elk beeld te berekenen.

3. **Segmentatie van Afbeeldingen**: Om de kwaliteit van het uiteindelijke 3D model te optimaliseren, is het van belang om onnodige achtergrondinformatie te verwijderen. Door afbeeldingen te segmenteren, kunnen we de voorgrond objecten identificeren en isoleren.

4. **Genereren van een Radiance Field met NeRF**: NeRF (Neural Radiance Fields) is een techniek die een continu 3D scène model creëert. Het maakt gebruik van deep learning om een radiance field te genereren vanuit de gesegmenteerde 2D-afbeeldingen.

5. **3D Model Generatie**: Vanuit het gecreëerde radiance field wordt vervolgens het uiteindelijke 3D model gegenereerd. Dit biedt een gedetailleerde en ruimtelijke representatie van het object, klaar voor verdere toepassingen en analyses.

De waarde van deze aanpak wordt gemeten aan de hand van het uiteindelijke 3D-mesh van het voorgrondobject in de video. De kwaliteit van dit mesh zal grondig geëvalueerd worden. Afhankelijk van de resultaten kan deze aanpak invloed hebben op de toekomst van 3D-modellering en de toegankelijkheid ervan vergroten.

Door deze geïntegreerde benadering te volgen, streven we ernaar een robuuste en effectieve methode te leveren voor het converteren van conventionele video-opnamen naar gedetailleerde 3D modellen, met talrijke implicaties en toepassingen in uiteenlopende industrieën.

De opbouw van deze thesis is als volgt. In hoofdstuk 2 wordt een revisie van essentiële concepten gepresenteerd die noodzakelijk zijn voor het begrijpen van dit proefschrift. Hoofdstuk 3 behandelt het segmenteren van beeldmateriaal. Hoofdstuk 4 biedt een gedetailleerde beschrijving van NeRF, een techniek die een revolutie teweeg heeft gebracht in de wereld van computer vision. Vervolgens wordt in hoofdstuk 5 uitgelegd hoe een mesh gereconstrueerd kan worden vanuit een radiance field, wat het voornaamste theoretische gedeelte van het proefschrift vormt. Hoofdstuk 6 behandelt de implementatie en de onderbouwing daarvan. Ten slotte worden in hoofdstuk 7 de behaalde resultaten besproken.

Hoofdstuk 2

Revisie van fundamentele concepten

Informatici introduceren voortdurend nieuwe technieken, elk met hun eigen benaming. Gezien de veelheid aan termen kan een opfrisser op zijn plaats zijn. Deze sectie biedt een beknopt overzicht van de cruciale concepten die essentieel zijn voor het begrip van dit proefschrift. Degenen die minder bekend zijn met Machine Learning (ML) en computergraphics zullen deze sectie bijzonder nuttig vinden. Bent u al vertrouwd met de besproken onderwerpen? Dan kunt u dit hoofdstuk gerust overslaan.

2.1 Convolutionele lagen

Convolutionele lagen vormen de kern van Convolutional Neural Networks (CNN's) [1], die primair zijn ontworpen voor beeldgegevens. Hun primaire functie? Automatisch en adaptief leren van ruimtelijke kenmerken (=features) uit invoerafbeeldingen.

Kernpunten:

1. **Convolutiebewerking**: Door gebruik te maken van een filter (of kernel) van formaat $K \times K$, glijdt dit over de invoerafbeelding (of convolueert) en produceert een featuremap.

   Voor elke positie van het kernel:

   $$\mathrm{y}(i,j) = \sum_{u=0}^{K-1} \sum_{v=0}^{K-1} \mathrm{afb}(i+u, j+v) \times \mathrm{kernel}(u,v)$$

2. **Stride en padding**:

   - **Stride**: Het aantal pixels waarmee het filter over de afbeelding beweegt. Bijvoorbeeld, een stride van 1 betekent een beweging van één pixel per keer.
   - **Padding**: Het toevoegen van extra 'nul'-waarden rond een afbeelding. Dit helpt om de afmetingen van de afbeelding na de convolutie te behouden.

3. **Activatiefunctie**: Na de convolutiebewerking wordt vaak de ReLU (Rectified Linear Unit) functie toegepast:

   $$\mathrm{ReLU}(x) = \max(0, x)$$

Uitvoergrootte na convolutie

Gegeven een invoergrootte $W_1 \times H_1$, kernelsize K, padding P, en stride S, de grootte van de uitvoer is:

$$W_2 = \frac{W_1 - K + 2P}{S} + 1$$
$$H_2 = \frac{H_1 - K + 2P}{S} + 1$$

Een convolutionele laag in CNN's bevat vaak meerdere filters, die diverse features uit beelden halen. Deze lagen worden gecombineerd met andere soorten lagen, zoals poolinglagen, die de grootte van de feature map verminderen om de berekeningen efficiënter te maken.

Terwijl dit een basisoverzicht is, hebben geavanceerdere CNN-modellen vaak aanvullende lagen en technieken.

2.2 Transposed convolutie

Transposed convoluties, vaak aangeduid als "deconvoluties" [2], worden vaak gebruikt in CNN-architecturen voor up-sampling en het verhogen van de resolutie van featuremaps. Ze zijn essentieel in bepaalde toepassingen zoals beeldsegmentatie en generatieve modellen zoals Generative Adversarial Networks (GAN's) [3].

Kernpunten

1. **Transposed convolutiebewerking:** Het basisidee is om een input featuremap te nemen en de resolutie ervan te verhogen. Dit wordt gedaan door een filter (of kernel) van formaat $K \times K$ te gebruiken, dat over de featuremap beweegt en een opgeschaalde uitvoer produceert. De wiskundige formule is vergelijkbaar met die van de standaardconvolutie (Zie 2.1), maar het proces wordt omgekeerd.
2. **Stride en padding bij transposed convolutie:**
   - **Stride:** Het concept van stride blijft hetzelfde, maar in plaats van het verkleinen van de output, verhoogt een stride in transposed convolutie de dimensie van de output.
   - **Padding:** Padding bij transposed convoluties kan worden gebruikt om de precieze uitvoerafmetingen te regelen, vergelijkbaar met normale convoluties. Het kan echter enigszins contraintuïtief zijn, omdat het verwijderen van waarden kan leiden tot een grotere uitvoer, in tegenstelling tot normale convoluties.
3. **Activatiefunctie:** Net als bij normale convoluties wordt na de transposed convolutiebewerking vaak een activatiefunctie zoals ReLU toegepast:

   $$\text{ReLU}(x) = \max(0, x)$$

   #### Uitvoergrootte na transposed convolutie:

   Gegeven een invoergrootte $W_1 \times H_1$, kernelsize K, padding P, en stride S, de grootte van de uitvoer is:

   $$W_2 = (W_1 - 1) \times S + K - 2P$$
   $$H_2 = (H_1 - 1) \times S + K - 2P$$

Een transposed convolutielaag in CNN's bevat vaak meerdere filters die de informatie uit de ingangsfeaturemaps opschalen. Ze worden vaak gecombineerd met andere soorten lagen, zoals normale convolutionele lagen en poolinglagen.

Dit is een basisoverzicht van transposed convoluties. Net als bij reguliere CNN's kunnen geavanceerdere modellen die transposed convoluties gebruiken, extra lagen en technieken bevatten.

2.3 Pooling lagen

Pooling lagen zijn een integraal onderdeel CNN's en zijn bedoeld om de grootte van de featuremaps te verminderen, terwijl de belangrijkste informatie behouden blijft. Dit helpt om de rekencomplexiteit te verminderen en overfitting te voorkomen.

Soorten pooling:

- **Pooling operaties**:
  - **Max Pooling**: Kiest de maximale waarde binnen de kernel.
  - **Min Pooling**: Selecteert de minimale waarde binnen de kernel.
  - **Average/Mean Pooling**: Berekenen het gemiddelde van de waarden in een kernel.

Werking:

Een pooling laag werkt door een klein kleiner (bijv. 2×2 of 3×3) over de featuremap te schuiven en voor elk kernel de gekozen pooling operatie (max, min, avg, etc.) uit te voeren. De kernel wordt meestal met een bepaalde stapgrootte (stride) verplaatst, en deze stapgrootte bepaalt met hoeveel de dimensies van de featuremappen worden gereduceerd.

Bijvoorbeeld: Voor max-pooling met een kernel van grootte $k \times k$, een stride s, en een originele featuremap van afmetingen $W_1 \times H_1$, zijn de afmetingen van de gereduceerde featuremap:

$$W_2 = \left\lfloor \frac{W_1 - k}{s} + 1 \right\rfloor$$
$$H_2 = \left\lfloor \frac{H_1 - k}{s} + 1 \right\rfloor$$

2.4 Batch normalization

Introductie: Batch Normalization, vaak afgekort als BatchNorm, is een techniek geïntroduceerd door Sergey Ioffe en Christian Szegedy [4]. Het primaire doel van BatchNorm is het verbeteren van de trainingssnelheid en stabiliteit van diepe neurale netwerken door het verminderen van het intern covariateshift probleem.

Werking: Tijdens de training worden de inputs van elke laag genormaliseerd op basis van de gemiddelde en variantie van de huidige mini-batch. Hierdoor hebben de waarden een gemiddelde van nul en een variantie van één.

Wiskundig wordt dit als volgt uitgedrukt:

$$\mu_B = \frac{1}{m} \sum_{i=1}^{m} x_i$$
$$\sigma_B^2 = \frac{1}{m} \sum_{i=1}^{m} (x_i - \mu_B)^2$$
$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$$
$$y_i = \gamma \hat{x}_i + \beta$$

Waar:

- μ_B het gemiddelde is van de mini-batch.
- σ_B^2 de variantie is van de mini-batch.

- $\hat{x}_i$ de genormaliseerde input is.
- γ en β schaal- en verschuivings variable die tijdens de training worden geleerd.
- ϵ kleine constante om deling door nul te voorkomen.

Voordelen:

- Versnelt de training door snellere convergentie.
- Maakt het mogelijk om hogere leersnelheden te gebruiken.
- Vermindert de noodzaak voor andere reguliere technieken zoals dropout.
- Kan lichte regulering bieden en het risico op overfitting verminderen.

2.5 world2camera

Om een punt van wereldcoördinaten naar cameracoördinaten te transformeren, wordt gebruik gemaakt van een transformatiematrix T:

$$T = \begin{bmatrix} R & \vec{t} \\ 0 & 1 \end{bmatrix}$$

waarbij:

- R de rotatiematrix is die de oriëntatie van de camera in de wereld beschrijft.
- $\vec{t}$ de translatievector is die de positie van de camera in de wereld beschrijft.

De transformatie van een punt P_w in wereldcoördinaten naar P_c in cameracoördinaten is:

$$P_c = T \times P_w$$

Voor de transformatie van punten van cameracoördinaten naar wereldcoördinaten wordt de inverse van de transformatiematrix T gebruikt:

$$P_w = T^{-1} \times P_c$$

2.6 Metrics

PSNR

PSNR [5] is een technische maat die vaak wordt gebruikt in de beeldverwerking. Het doel ervan is het meten van de kwaliteit van gereconstrueerde of gecomprimeerde beelden in vergelijking met het originele beeld. PSNR representeert de verhouding tussen het maximale mogelijke vermogen van een signaal en het vermogen van de verstoring die het signaal aantast. Deze ratio wordt meestal uitgedrukt in decibel (dB) en wordt berekend met de formule:

$$\text{PSNR} = 10 \times \log_{10} \left(\frac{\text{MAX}^2}{\text{MSE}} \right)$$

waarbij MAX de maximale pixelwaarde van het beeld is (voor een 8-bit grijsschaalafbeelding is MAX = 255) en MSE staat voor “Mean Squared Error”, het gemiddelde van de gekwadrateerde verschillen tussen het originele en het gereconstrueerde beeld. Een hogere PSNR waarde geeft doorgaans aan dat de kwaliteit van de reconstructie beter is, terwijl een lagere PSNR op grotere verschillen tussen het originele en het gereconstrueerde beeld wijst. Ondanks dat PSNR een objectieve maatstaf biedt, kan het soms niet overeenkomen met de subjectieve beoordeling van beeldkwaliteit door het menselijk oog. Hierdoor wordt het in bepaalde toepassingen aangevuld of vervangen door andere metrieken, zoals de Structural Similarity Index (SSIM). PSNR is een populaire maatstaf in domeinen zoals beeldcompressie, beeldreconstructie en videocodering.

Dice score

De Dice score [6, 7], soms aangeduid als de Dice-coëfficiënt of de Sørensen-Dice coëfficiënt, is een statistisch instrument dat de overeenkomst tussen twee sets beoordeelt. In beeldsegmentatietaken wordt deze vaak ingezet om de mate van overlap tussen voorspelde segmentaties en de ground truth te kwantificeren.

Beschouw twee binaire maskers: A (onze voorspelling) en B (ground truth). De Dice score voor deze maskers wordt als volgt gedefinieerd:

$$\text{Dice Score} = \frac{2 \times |A \cap B|}{|A| + |B|}$$

Hier staat $|A \cap B|$ voor het aantal elementen dat zowel in A als in B voorkomt, oftewel de doorsnede van de twee sets. De termen $|A|$ en $|B|$ vertegenwoordigen respectievelijk het aantal elementen in A en B.

In de wereld van beeldsegmentatie vertegenwoordigen A en B binaire maskers van gesegmenteerde regio's. Een waarde van 1 in een masker duidt op een gesegmenteerde regio, terwijl een 0 het tegenovergestelde aangeeft. Het aantal pixels waarvoor zowel de voorspelling als de werkelijke waarde gelijk is aan 1, geeft het aantal correct voorspelde pixels aan.

Samenvattend geeft de Dice score inzicht in de overeenkomst tussen twee binaire segmentaties. Een hogere score duidt op een grotere overeenkomst tussen de voorspelling en de werkelijke waarde.

De Structural Similarity Index

De Structural Similarity Index (SSIM) [8] is een geavanceerde metriek die in de beeldverwerking wordt gebruikt om de perceptuele kwaliteit van een gereconstrueerd of gecomprimeerd beeld ten opzichte van een origineel beeld te beoordelen. In tegenstelling tot traditionele metrieken zoals PSNR, die voornamelijk gericht zijn op pixel-niveau verschillen, probeert SSIM de veranderingen in structurele informatie, lichtintensiteit en textuur te kwantificeren, wat meer overeenkomt met menselijke perceptie.

De SSIM-index tussen twee beelden x en y wordt gedefinieerd als:

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}$$

waarbij μ_x en μ_y de gemiddelden van de beelden x en y zijn, σ_x^2 en σ_y^2 hun varianties zijn, σ_{xy} de covariantie tussen de beelden is, en c_1 en c_2 kleine constanten zijn om stabiliteit te garanderen wanneer de noemer dicht bij nul is.

Een SSIM-waarde van 1 duidt op perfecte gelijkenis tussen de twee beelden, terwijl een waarde van 0 volledige dis-similariteit aangeeft. De SSIM-index biedt een beter inzicht in beeldkwaliteit vanuit een menselijk perceptueel perspectief en wordt daarom vaak gebruikt in beeldverwerkingstaken zoals beeldcompressie, beeldreconstructie en beeldevaluatie.

Chamfer distance/score

De Chamfer distance [9] is een metriek die in computer vision en geometrische modellering wordt gebruikt om de gelijkenis tussen twee puntwolken te kwantificeren. Deze metriek heeft zijn oorsprong in het meten van verschillen tussen vormen en structuren.

De Chamfer distance tussen twee puntwolken, A en B, wordt berekend door voor elk punt in A de kortste afstand naar enig punt in B te vinden en vice versa. Hierdoor kunnen de verschillen tussen de twee sets punten worden bepaald, wat inzicht geeft in hoe nauwkeurig bijvoorbeeld een gereconstrueerd 3D-model een doel benadert.

De wiskundige definitie van de Chamfer distance is als volgt:

$$CD(A,B) = \frac{1}{|A|}\sum_{a\in A}\min_{b\in B}\|a-b\|_2 + \frac{1}{|B|}\sum_{b\in B}\min_{a\in A}\|a-b\|_2$$

Hierin staat $\|\cdot\|_2$ voor de euclidische afstand, die de "gewone" afstand tussen twee punten in een Euclidische ruimte representeert. Voor elk punt a in A wordt de dichtstbijzijnde buur b in B gezocht en hun afstand wordt toegevoegd aan de som. Dit proces wordt ook omgekeerd uitgevoerd, waarbij voor elk punt in B de dichtstbijzijnde buur in A wordt gezocht.

Deze som wordt genormaliseerd door de grootte van de respectievelijke puntwolken, wat resulteert in een gemiddelde afstand. In de context van 3D-modellering en reconstructie biedt de Chamfer distance een manier om de nauwkeurigheid van een gereconstrueerd model ten opzichte van een referentiemodel te beoordelen. Een kleinere Chamfer distance geeft aan dat de twee puntwolken dichter bij elkaar liggen, wat duidt op een betere gelijkenis tussen het gereconstrueerde model en het referentiemodel.

Hoofdstuk 3

Image segmentation

Het aantal visuele gegevens groeit exponentieel en beeldverwerkingstechnieken worden steeds meer toegepast in diverse sectoren, van medische beeldverwerking tot zelfrijdende auto's en virtual reality. Image segmentation speelt hierbij een cruciale rol, waarbij het beeld wordt opgedeeld in relevante onderdelen, zoals deelgebieden, objecten en texturen, om waardevolle informatie te extraheren. Door image segmentation kunnen machine learning technieken sneller en nauwkeuriger informatie uit beelden halen.

Image segmentation vindt toepassing in verschillende sectoren:

1. Medische beeldverwerking[10, 11]: Binnen de medische sector wordt machine learning gebruikt voor kankerdetectie en andere anomaliteiten. Het segmenteren van beelden stelt algoritmes in staat om subtiele afwijkingen te detecteren die voor het menselijk oog moeilijk zichtbaar zijn. Door gebruik te maken van computer vision kunnen artsen vroegtijdig kankercellen en andere medische problemen identificeren, waardoor de diagnose en behandeling worden verbeterd.

2. Autonome voertuigen[12]: Moderne voertuigen fungeren als grote computers op wielen en moeten in real-time hun omgeving begrijpen. Image segmentation helpt bij het identificeren en begrijpen van andere voertuigen, voetgangers, fietsers, verkeersborden, enz. Dit stelt autonome voertuigen in staat om geïnformeerde beslissingen te nemen en veilig en efficiënt te navigeren.

3. Augmented Reality (AR)[13]: In AR wordt segmentatie gebruikt om de voorgrond van de achtergrond te scheiden, waardoor virtuele objecten kunnen worden toegevoegd aan de echte wereld. Deze technologie maakt het mogelijk om een verrijkte ervaring te creëren door virtuele elementen te combineren met de fysieke omgeving.

Naast de genoemde voorbeelden wordt image segmentation ook toegepast in andere sectoren [14].

Image segmentation kan op verschillende manieren worden uitgevoerd, zowel met traditionele algoritmen als met machine learning-technieken. De laatste jaren heeft machine learning, zoals het U-Net model (sectie 3.2), terrein gewonnen ten opzichte van traditionele algoritmen voor image segmentation. Desondanks blijft het een uitdagend probleem om op te lossen. Een van de uitdagingen is het beoordelen van de prestaties van segmentatiemethoden, omdat er talloze manieren zijn om een afbeelding te segmenteren, bijvoorbeeld op basis van kleur, textuur, objecten of diepte. Voor computers zijn het allemaal slechts getallen per pixel, wat de taak complex maakt. Daarnaast is het resultaat van image segmentation vaak subjectief en afhankelijk van het beoogde doel en de specifieke toepassing.

Dit hoofdstuk zal een diepgaande verkenning bieden van relevante literatuur binnen het vakgebied van beeldsegmentatie. Daarna zal een uitgebreide analyse worden uitgevoerd van een

geavanceerde deep learning-architectuur die momenteel als state-of-the-art kan worden beschouwd.

3.1 Gerelateerde werken

U-Net is een veelgebruikte architectuur voor semantische segmentatie in de medische beeldvorming en andere domeinen. Deze architectuur heeft de karakteristieke vorm van een "U", waarbij het neergaande pad, oftewel de encoder, het beeld comprimeert tot een compacte representatie. Vervolgens decodeert het opwaartse pad, de decoder, deze representatie naar een segmentatiemasker. Voordat U-Net zijn intrede deed, werden Fully Convolutional Networks[15] (FCN's) geïntroduceerd die voor het eerst het idee voorstelden om CNN's eind-tot-eind te gebruiken voor pixelwijs voorspellen. Wat U-Net onderscheidt van FCN's, is het opwaartse pad dat zorgt voor de combinatie van low-level features met high-level features, wat resulteert in een betere lokaliseringsprecisie. Voor de opkomst van neurale netwerken werden semantische segmentatie taken vaak benaderd met traditionele beeldverwerkingsmethoden, zoals thresholding, region growing en watershed transform. Er zijn ook andere prominente netwerkarchitecturen zoals Mask R-CNN [16], wat voornamelijk bedoeld is voor instantie-segmentatie, DeepLab[17] dat gebruikmaakt van dilated convoluties en spatial pyramid pooling, en PSPNet[18] dat contextuele informatie uit verschillende schalen verzamelt. Een ander netwerk, SegNet[19], heeft ook een encoder-decoder structuur vergelijkbaar met U-Net, maar onthoudt alleen de max-pooling indices uit de encoder voor een betere positionele nauwkeurigheid. Een cruciaal kenmerk dat U-Net onderscheidt van veel van zijn tegenhangers, zijn de Skip-connectie tussen de encoder en decoder. Deze helpen verloren ruimtelijke informatie te herintroduceren, wat leidt tot nauwkeurigere segmentatiemasks. De skip-connectie, geïnspireerd door de ResNet-architectuur, verdient een afzonderlijke bespreking in deze literatuuroverzicht.

3.1.1 ResNet

ResNet, een afkorting van Residual Network, vertegenwoordigde een revolutionaire stap in de ontwikkeling van neurale netwerken werd geïntroduceerd door Kaiming He et al. [20]. Het onderscheidende kenmerk van ResNet is het gebruik van "skip" of "residual" verbindingen. Deze verbindingen laten activaties ongewijzigd door naar volgende lagen, waardoor het mogelijk is om eerdere lagen te överslaan". Hiermee biedt ResNet een oplossing voor het verdwijnende gradiëntprobleem, een bekend probleem bij het trainen van deep neurale netwerken.

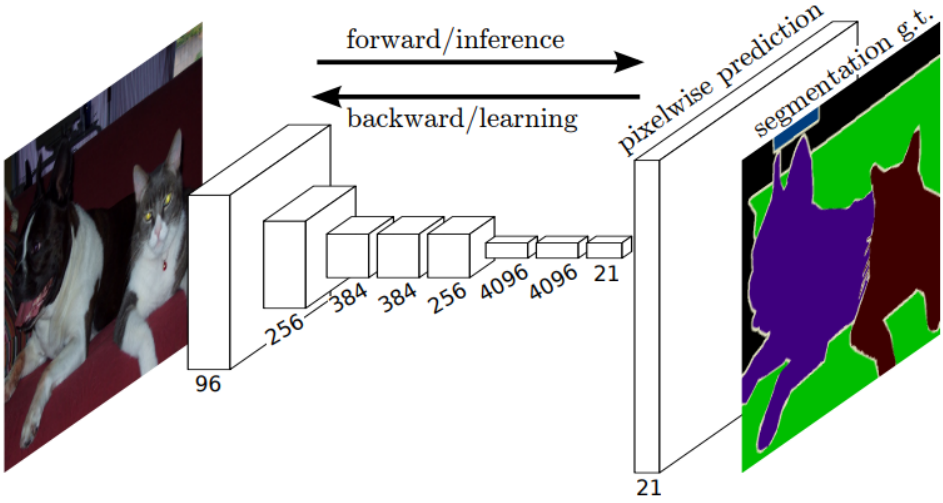
Met residueel leren kan het netwerk het verschil of residu tussen een gewenste functie $h(x)$ en de invoer x leren, uitgedrukt als $h(x) - x$. Deze benadering vereenvoudigt de training en bevordert een snellere convergentie. Daarnaast heeft het gebruik van deze "skip" of "residual" verbindingen het bijkomende voordeel dat er geen extra parameters of verhoogde computationele complexiteit aan het netwerk worden toegevoegd.

De impact van ResNet is verreikend. Niet alleen heeft het geholpen de prestaties van diepe netwerken te verbeteren, het heeft ook de deur geopend voor latere netwerkontwerpen. In het bijzonder hebben moderne segmentatiearchitecturen zoals UNet geprofiteerd van de principes die door ResNet zijn geïntroduceerd. Het gebruik van skip- of restverbindingen stelt deze modellen in staat om informatie effectief door het netwerk te leiden, waarbij zowel details als context beter behouden blijven.

3.2 U-Net

U-Net[11] is een state-of-the-art deep learning architectuur voor salient object detection [21] (SOD), oorspronkelijk ontwikkeld voor semantische segmentatie taken in medische beeldverwerking. Dit baanbrekende onderzoek werd sterk geïnspireerd door het paper "Fully Convolutional Networks for Semantic Segmentation"[15], figuur3.1 is het architectuur van FCN . Het introduceerde het concept van Fully Convolutional Networks (FCN) voor het uitvoeren van pixelgewijze

classificatie door middel van uitsluitend convolutie-lagen, zonder Fully conected layers. U-Net bouwde voort op deze ideeën en paste ze aan voor de specifieke behoeften van salient object detection in medische beeldverwerking.



Figuur 3.1: Architectuur Fully Convolutional Network [15]

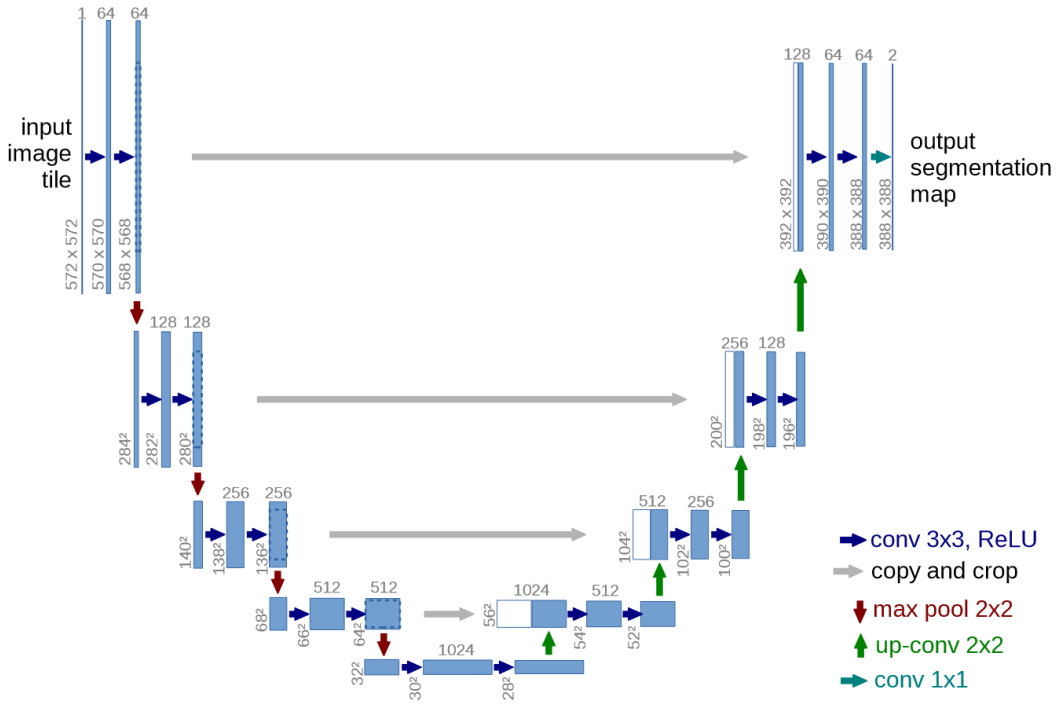
De U-vormige architectuur van U-Net (figuur3.2) werd geïnspireerd door de FCN's decoder-encoder structuur. De encoder, vergelijkbaar met het inzamelingsstadium van FCN (engels: Feature extraction - or downsampling stage), haalt kenmerken uit de inputgegevens. Aan de andere kant komt de decoder overeen met het uitbreidingsstadium van FCN. Een belangrijke aanpassing van U-Net was het toevoegen van skip-connections tussen de encoder en decoder. Deze skip-connections laten lage en hoge niveaus van kenmerkinformatie effectief combineren, waardoor het model gedetailleerde lokale informatie behoudt en tegelijkertijd een contextueel begrip van het geheel ontwikkelt. Hierdoor kon U-Net hoge prestaties bereiken, zelfs in situaties met kleine of ver uit elkaar gelegen objecten.

Een van de opvallende eigenschappen van U-Net is zijn vermogen om goed te presteren met een beperkt aantal trainingsvoorbeelden. Dit wordt mogelijk gemaakt door het gebruik van residual learning, waarbij de kenmerken worden geüpdatet door het verschil tussen de oorspronkelijke en de voorspelde kenmerken toe te voegen. Dit bevordert de convergentie van het model en vermindert het risico op het optreden van het verdwijnende gradiëntprobleem.

De flexibiliteit en leersnelheid van U-Net hebben het model razend populair gemaakt. In veel praktische toepassingen is het vaak moeilijk om een grote hoeveelheid gelabelde trainingsgegevens te verkrijgen. Traditionele deep learning modellen zouden kunnen lijden onder het gebrek aan trainingsdata, wat resulteert in onvoldoende prestaties. U-Net heeft echter de mogelijkheid om effectief te leren van slechts een beperkt aantal trainingsvoorbeelden, terwijl het toch hoge prestaties behoudt bij het uitvoeren van SOD-taken.

In 2022 werd U-Net opgevolgd door U2-Net[22], waarbij een belangrijke bijdrage werd geleverd door de introductie van ReSidual U-blocks (RSU) (figuur3.3). Deze RSU's zijn een aanpasbare variant van de traditionele U-structuur en bieden verbeterde representaties van de kenmerken, waardoor het model nog beter kan presteren in verschillende SOD-taken.

U-Net heeft zich bewezen als een krachtige en flexibele deep learning architectuur voor salient object detection, mede dankzij zijn inspiratie uit het FCN-paper van 2015 en de daaropvolgende innovaties. Zijn succes in het werken met een beperkt aantal trainingsgegevens en het vermogen om complexe patronen te leren maken het een waardevol hulpmiddel in het domein van computer vision en deep learning. Door een diepgaande analyse van de technische details van U-Net uit



Figuur 3.2: Visualisatie van de U-Net architectuur voorgesteld door Ronneberger et al. [11]

te voeren, kunnen we de werking en prestaties van het model beter begrijpen. De voortdurende ontwikkeling, zoals de introductie van U2-Net, zorgt ervoor dat U-Net een vooraanstaande rol blijft spelen in salient object detection en andere beeldverwerkings-taken.

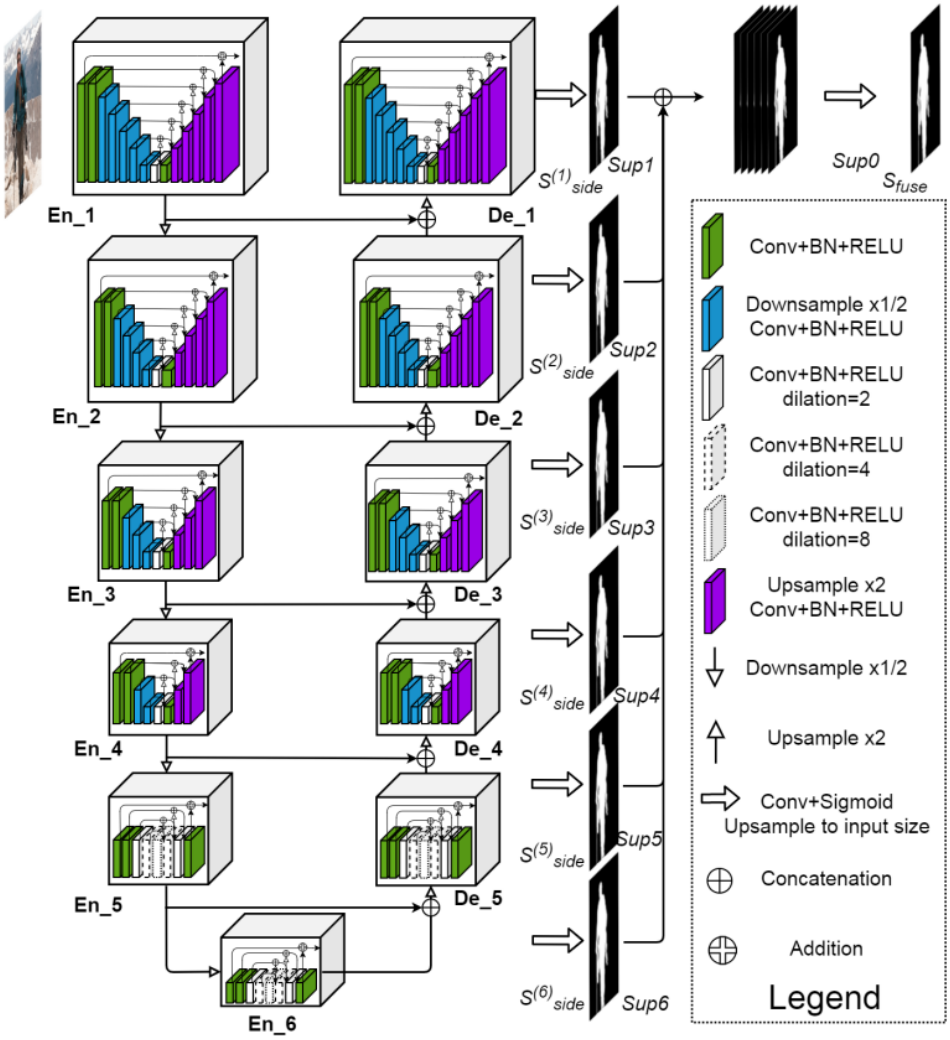
3.2.1 Architectuur

Figuur 3.2 toont de originele architectuur voorgesteld door Ronneberger et al. Deze architectuur is een doorontwikkeling van een eerder gerealiseerd netwerk genaamd "fully convolutional network"van Long et al. [15]. U-Net is specifiek ontwikkeld om met minder training data betere resultaten te behalen in image segmentation taken.

De naam van de architectuur is afgeleid van de U-vormige structuur. De input afbeelding komt aan het begin binnen en de segmentatie map wordt aan het einde als output geproduceerd. Het linker gedeelte van de architectuur, ook wel de encoder genoemd, voert een convolutie-operatie uit met een 3x3 kernel gevolgd door de ReLU activatie functie. De dubbele convolutie-operatie verdubbelt het aantal kanalen van de afbeelding. De output van deze dubbele convolutie-operatie wordt vervolgens gedownsampled met behulp van max-pooling met een 2x2 kernel. Op deze manier leert het netwerk lokale features terwijl de resolutie wordt verkleind voor verdere verwerking.

Het rechter gedeelte van de operatie, ook wel de decoder genoemd, is vergelijkbaar met één verschil. Hier wordt geüpsampled in plaats van gedownsampled, en dit gebeurt met behulp van een 2x2 convolutie-operatie, ook wel bekend als up-convolutie of transposed convolutie. Hierdoor wordt de resolutie verdubbeld, maar het aantal kanalen wordt gehalveerd. De output van het up-sampling gedeelte wordt vervolgens geconcateneerd met de output van het dubbele convolutie linker gedeelte. Deze concatenatie stelt het model in staat om zowel lokale als globale informatie te combineren voor een nauwkeurige segmentatie.

Opnieuw wordt een dubbele convolutie-operatie uitgevoerd in het rechter deel van de architec-



Figuur 3.3: Architectuur U2-Net met RSU in elke stage [22]

tuur om de resolutie terug te brengen naar het oorspronkelijke niveau. Deze keer wordt het aantal kanalen gehalveerd. Ten slotte wordt een 1x1 convolutie-operatie toegepast om tot de segmentatie map te komen. Deze operatie verandert de resolutie niet, maar reduceert alleen het aantal kanalen naar twee.

De intuïtie achter dit netwerk is dat de encoder leert om features te onderscheiden en te extraheren, terwijl de decoder de gedownsamplede features weer upsampled en combineert om nauwkeurige segmentatieresultaten te behalen. U-Net maakt op deze manier optimaal gebruik van zowel lokale als globale informatie, waardoor het model efficiënt en effectief werkt, zelfs met beperkte trainingsdata.

Hoofdstuk 4

Neuraal renderen

De laatste jaren heeft het gebied van computergraphics en computervisie een enorme transformatie doorgemaakt, mede dankzij de combinatie van deep learning en traditionele computationele methoden. Een interessante ontwikkeling hierin is de komst van Neuraal Renderen, wat potentieel de manier waarop we digitale beelden maken en ervaren, radicaal kan veranderen.

Binnen Neuraal Renderen valt vooral het concept van Neural Radiance Fields (NeRF) op. Met NeRF gebruiken we neurale netwerken om de band tussen 3D-coördinaten en de lichtuitstraling in een scene te begrijpen. Dit is een breuk met de oude manieren, die meer leunden op vaste vormen en lichtmodellen. Dankzij NeRF kunnen we verbazingwekkend realistische beelden maken, wat geweldig nieuws is voor zaken als virtual reality, augmented reality en digitale content creatie.

Dit hoofdstuk zoomt in op Neuraal Renderen, met een speciale focus op NeRF. We gaan de technische details verkennen, kijken naar hoe 3D-scenes en licht werken en duiken in de nieuwste optimalisatiemethoden. Door de beste onderzoeken in het veld te combineren, proberen we een compleet beeld te geven van de mogelijkheden en uitdagingen van Neuraal Renderen en zijn NeRF-onderdeel. Als je echt de ins en outs van NeRF wilt begrijpen, zit je hier goed.

Dit sectie biedt een diepgaande verkenning van de fundamenten en methodologieën van NeRF.

4.1 Gerelateerde werken

4.1.1 DeepSDF

DeepSDF[23] en Neural Radiance Fields (NeRF) zijn twee prominente methoden in het domein van neurale rendering en diep leren die de manier waarop we 3D-ruimte begrijpen en vertegenwoordigen, hebben gerevolutioneerd. Beide benaderingen bieden unieke perspectieven en vermijden traditionele representaties zoals meshes en voxelroosters ten gunste van continue representaties van 3D-objecten.

DeepSDF maakt gebruik van signed distance functions[24] (SDF's) in combinatie met diep leren. Een SDF kent aan elk punt in de 3D-ruimte een scalair waarde toe die de kortste afstand tot het oppervlak van een vorm aangeeft, met het teken van deze waarde die aangeeft of een punt zich binnen of buiten de vorm bevindt. DeepSDF gebruikt een neuraal netwerk om de SDF-waarde van een punt te voorspellen, waardoor het kan leren van datasets met vormen en hun bijbehorende SDF-waarden. Deze benadering biedt een continue, gedetailleerde en compacte representatie van 3D-vormen. Bovendien heeft DeepSDF generatieve capaciteiten, waarmee nieuwe en diverse vormen kunnen worden gecreëerd.

Aan de andere kant benadert NeRF 3D-rendering vanuit een andere invalshoek. Het maakt gebruik van diepe neurale netwerken om 3D-coördinaten en kijkrichtingen direct naar kleur en dichtheidswaarden te mappen, zonder een expliciete geometrische representatie te creëren. Hoewel NeRF uitblinkt in het vastleggen van ingewikkelde stralingsinformatie, vereist het mogelijk uitgebreide invoerbeelden voor training en biedt het niet direct een geometrische representatie.

Een specifieke toepassing van SDF-gebaseerde benaderingen is raymarching[25], waarbij stralen van een camera naar een scène worden gestuurd en gemarcheerd totdat ze een oppervlak kruisen. Door de differentieerbaarheid van raymarching kunnen gradiënten van een gerenderde afbeelding direct worden teruggevoerd naar de netwerkparameters, wat een scala aan toepassingen mogelijk maakt, van 3D-reconstructie tot vorminterpolatie.

In conclusie bieden zowel DeepSDF als NeRF innovatieve manieren om 3D-ruimte te benaderen en te representeren, elk met zijn eigen sterke punten en toepassingsgebieden.

4.1.2 Plenoxels

Plenoxels[26] zijn een manier om realistische 3D-beelden te maken van een reeks 2D-foto's die vanuit verschillende gezichtspunten zijn genomen. Ze doen dit door de 3D-ruimte te verdelen in kleine kubussen die voxels worden genoemd, en elke voxel een kleur en een doorzichtigheid toe te wijzen die afhangt van de kijkhoek. Op deze manier kunnen de voxels de complexe licht- en schaduweffecten van de scène vastleggen.

Om de optimale kleur en doorzichtigheid voor elke voxel te vinden, gebruiken plenoxels een wiskundige techniek die gradient descent wordt genoemd, die de parameters iteratief aanpast om de fout tussen het gerenderde beeld en de invoerfoto's te minimaliseren. Ze gebruiken ook een regularisatieterm die gladheid en spaarzaamheid in het voxelraster aanmoedigt, om ruis en artefacten te voorkomen.

Plenoxels zijn anders dan DeepSDF en NeRF die neurale netwerken gebruiken om de scène te modelleren. Neurale netwerken zijn krachtig maar traag en moeilijk te interpreteren, terwijl plenoxels eenvoudig maar snel en gemakkelijk te begrijpen zijn. Plenoxels kunnen een vergelijkbare of betere kwaliteit bereiken dan NeRFs in veel minder tijd en met minder resources.

4.2 Neural radiance field

4.2.1 Architectuur

Neural Radiance Field is een veelbelovende techniek in de computer vision gemeenschap die gebruik maakt van een multi-layer perceptron (MLP) om een 3D-scene te leren en te reconstrueren [27]. De architectuur van NeRF bestaat uit 9 lagen, elk met 256 neuronen, en een laatste laag met 128 neuronen. De input van het netwerk is de positional encoding van enkel de posities in de 3D-ruimte. Positional encoding is een techniek om 3D-posities te vertegenwoordigen als een vector van kenmerken, zodat het netwerk ruimtelijke informatie kan begrijpen. De eerste zeven lagen van het MLP gebruiken de ReLU (Rectified Linear Unit) activatiefunctie, terwijl de achtste laag de sigmoid-functie als activatie heeft. De sigmoid-functie wordt vaak gebruikt om waarden tussen 0 en 1 te normaliseren, wat nuttig is voor bepaalde schalingen of dichtheden.

Mildenhall et al. volgt de netwerkarchitectuur van DeepSDF [28] en voegt een residual connection toe. In de residual connection wordt de uitvoer van het vijfde laag geconcateneerd met de positional encoding van de posities. Deze concatenatie stelt het netwerk in staat om lokale en globale informatie te combineren en helpt om informatie van eerdere lagen te behouden, wat bijdraagt aan een beter leervermogen van het model.

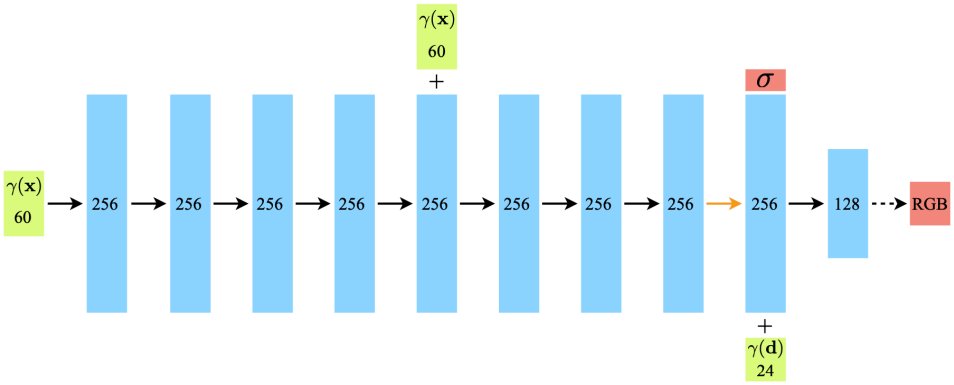
De voorlaatste laag heeft als output σ, dit is het output voor de densiteit. Dit betekent dat het model een waarde voorspelt die de dichtheid van de scène op een bepaald punt in de 3D-ruimte

aangeeft. Een hogere densiteit duidt op een hogere concentratie van materiaal op dat punt. Dit is een belangrijk kenmerk van NeRF, omdat het de mogelijkheid biedt om realistische 3D-scènes te reconstrueren met complexe materialen en lichteffecten.

Daarnaast bevat de achtste laag ook een feature vector met 256 neuronen, die wordt geconcateneerd met de positional encoding van de camera richting (θ, ϕ). Dit betekent dat de camera-informatie wordt gecombineerd met de kenmerken van het model, waardoor het model rekening kan houden met de kijkrichting van de camera bij het voorspellen van de RGB-waarde.

De laatste laag van het netwerk geeft als output de RGB-waarde, wat de kleur van een punt in de 3D-ruimte vertegenwoordigt. Door de combinatie van de densiteit en de RGB-waarde kan het NeRF-model een realistische weergave van een 3D-scene genereren, waarbij rekening wordt gehouden met de geometrie, materialen en belichting.

De architectuur van NeRF wordt gevisualiseerd in Figuur 4.1.



Figuur 4.1: Visualisatie van de architectuur van NeRF zoals voorgesteld door Mildenhall et al. [27]

De MLP-architectuur die hier is beschreven, staat bekend als de klassieke NeRF en heeft de weg geopend voor veel onderzoek op het gebied van neural radiance fields. Sinds 2020 is er een explosie van NeRF-onderzoek geweest en zijn verschillende architecturen voorgesteld om de prestaties en mogelijkheden van NeRF verder te verbeteren. Het vergelijken van de verschillende architecturen valt echter buiten het bestek van deze thesis. Niettemin blijft NeRF een veelbelovende benadering voor het genereren van realistische en gedetailleerde 3D-scènes uit beperkte sets van 2D-afbeeldingen.

De aankomende secties zullen zich verdiepen op de technische facetten die van significant belang zijn voor NeRF.

4.2.2 Sampling

NeRF maakt onderscheid tussen een 'coarse' en een 'fine' architectuur. In de coarse architectuur past het gestratificeerde sampling toe om punten te coderen. Ter contrast gebruikt het fine netwerk een hiërarchische sampling methode. Hierbij worden de gewichten van het coarse netwerk gebruikt om de gedetailleerde gebieden accurater te bemonsteren.

Stratified Sampling
Als je een lichtstraal volgt in een NeRF-scène, wil je 'samples' of punten langs die straal vastleggen. Dit helpt bij het bepalen van hun bijdrage aan de finale kleur en intensiteit van de straal, uitgedrukt als $\hat{C}_c(r)$ voor de coarse modus. Het willekeurig nemen van deze samples kan echter leiden tot inconsistenties of ruis, vooral als ze ongelijkmatig zijn verdeeld. Hier biedt stratified sampling een uitkomst.

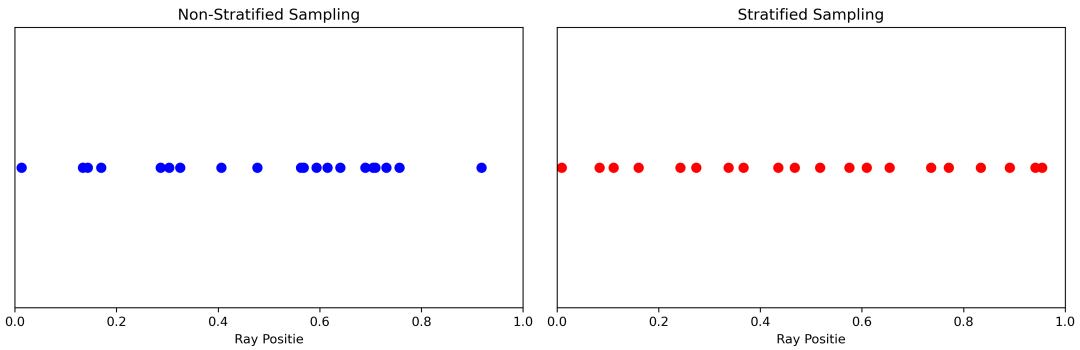
1. **Strata op de straal definiëren:** De straal wordt in gelijke segmenten verdeeld, ook wel strata genoemd.
2. **Bemonstering binnen elk stratum:** Uit elk stratum wordt één sample gehaald. Deze samples dragen bij aan het gewicht w_i voor de gehele straal.

Hiërarchische Sampling
NeRF's hiërarchische volume sampling techniek verhoogt de efficiëntie van het renderproces. Hier worden zowel een 'coarse' als een 'fine' netwerk toegepast.

Het proces start met het coarse netwerk, waarbij een set locaties, N_c, wordt gekozen via stratified sampling. Vervolgens worden deze locaties geëvalueerd. De uitkomst van dit netwerk, $\hat{C}_c(r)$, is een combinatie van alle bemonsterde kleuren langs de straal. Elke kleur wordt gewogen, met gewichten w_i, gebaseerd op het medium en de onderlinge sample-afstand.

Het fine netwerk, met als doel een hogere nauwkeurigheid, neemt vervolgens over. De eerder berekende gewichten, w_i, worden genormaliseerd tot $\hat{w}_i$. Deze herverdeelde gewichten helpen bij het selecteren van een tweede set locaties, N_f, via inverse transform sampling. Het fine netwerk evalueert daarna beide sets en geeft een gedetailleerdere kleurvoorspelling voor de straal, genaamd $\hat{C}_f(r)$.

Het mooie van deze aanpak is de gerichte aandacht voor gebieden waar visueel belangrijke informatie verwacht wordt. Hoewel het concept gelijkt op dat van importance sampling, streeft NeRF ernaar een gedetailleerde kaart van het gehele domein op te stellen in plaats van elk sample individueel te benaderen.



Figuur 4.2: Visuele weergave van sampling technieken op intuïtieve wijze

Figuur 4.2, waarin een artificieel gegenereerde sampling data wordt getoond, illustreert aan de rechterkant 'stratified sampling' en aan de linkerkant 'random sampling' over een lijn (of, binnen de context van deze thesis, een 'ray'). Uit de visualisatie blijkt dat de stratified sampling gelijkmatig over intervallen plaatsvindt. Toch is de afstand tussen twee opeenvolgende punten (ΔX) niet steeds gelijk, aangezien de sampling binnen een stratum willekeur bevat. Echter, deze willekeur heeft een beperkte invloed in vergelijking met random sampling over de volledige ray. Voor een duidelijk begrip van 'hierarchisch sampling' kan men de non-stratified sampling visualisatie raadplegen. Als een bepaald gewicht (σ) wordt toegekend, kan er op bepaalde posities intensiever worden gesampled. Dit soort resultaten is zichtbaar in de visualisatie van non-stratified sampling. De uitkomsten van zowel stratified als hierarchisch sampling voor een ray binnen nerf zijn te raadplegen in sectie 6.5.1.

4.2.3 Positional encoding

NeRF maakt gebruik van positional encoding [29] om de directies en posities van een scène om te zetten naar een hogere dimensie voordat ze als input aan het netwerk worden gegeven. De input bestaat uit een vector van 2 voor de directie en een vector van 3 voor de positie. Omdat

deze inputvectoren vrij klein zijn, kan dit leiden tot het verlies van details in de reconstructie van de 3D-scène. Om dit probleem aan te pakken, wordt positional encoding toegepast om het signaal in een hogere dimensie te vertalen.

Positional encoding is een techniek die veel wordt gebruikt in neurale netwerken die werken met sequentiële of ruimtelijke gegevens. Het vertaalt de ruimtelijke coördinaten naar een reeks van extra kenmerken die worden toegevoegd aan de originele input. Deze extra kenmerken bevatten informatie over de absolute positie van elk punt in de ruimte, waardoor het netwerk in staat is om beter te generaliseren en complexe ruimtelijke patronen te leren.

Voor NeRF wordt de dimensie van de positie verhoogd met een factor van 10 en de dimensie van de directie met een factor van 4. Dit betekent dat elk punt in de ruimte nu wordt gerepresenteerd door een veel grotere vector met extra kenmerken. Door de dimensie van de input op deze manier te vergroten, worden hogere frequenties in de data beter aangeleerd en wordt het renderen van nieuwe standpunten met meer details mogelijk.

Het gebruik van positional encoding helpt NeRF om meer complexe details in de 3D-scène vast te leggen en resulteert in een betere algemene reconstructie. Het is echter belangrijk op te merken dat dit ook leidt tot een toename in het aantal parameters van het netwerk en daarmee tot een verhoogde computationele complexiteit. Dit kan leiden tot langere trainingstijden en vereist meer rekenkracht tijdens het inference-proces. Toch blijft positional encoding een waardevol hulpmiddel om de capaciteiten van NeRF te verbeteren en de kwaliteit van de 3D-reconstructies te verhogen.

Naast het gebruik van positional encoding heeft NeRF ook veel aandacht gekregen voor efficiëntieverbeteringen. NVIDIA heeft bijvoorbeeld een nieuwe encoding-methode geïntroduceerd genaamd "hash-encoding"(Raadpleeg sectie 4.4 voor meer informatie). Met hash-encoding worden de input directies en posities omgezet naar compacte hash-codes, waardoor de architectuur van het netwerk aanzienlijk kan worden verkleind. Hierdoor wordt de trainingsduur verkort en wordt de computationele complexiteit verminderd, waardoor NeRF geschikt wordt voor real-time en interactieve toepassingen. Deze efficiëntieverbeteringen hebben belangrijke voordelen opgeleverd voor de prestaties en bruikbaarheid van NeRF in verschillende praktische toepassingen.

Verdieping

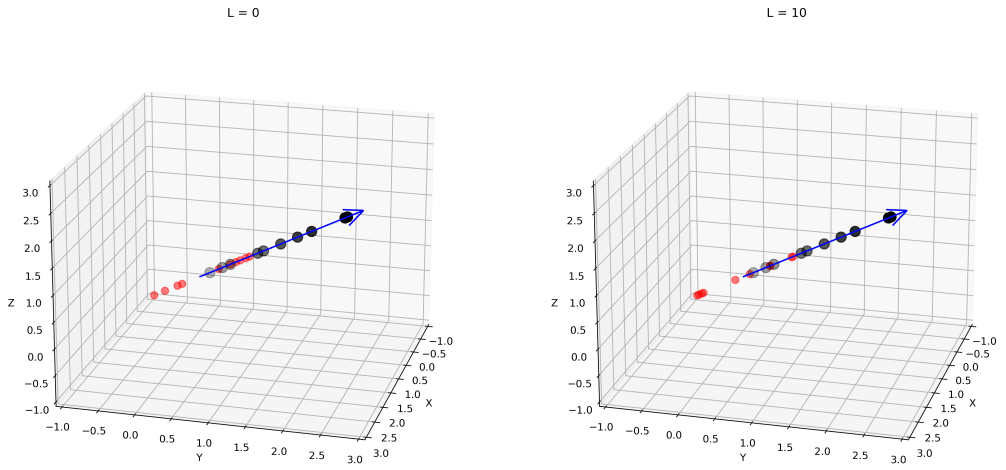
positional encoding is een cruciaal onderdeel van de NeRF-techniek. In NeRF-modellen, die diepe neurale netwerken gebruiken om 3D-scènes te renderen, is het belangrijk om niet alleen de kleur en dichtheid van een punt in de ruimte vast te leggen, maar ook de exacte positie. Aangezien neurale netwerken van nature geen gevoel van volgorde of positie hebben, helpt positional encoding om de 3D-coördinaten van een punt in de ruimte uniek te vertegenwoordigen.

Deze codering wordt bereikt door sinusfuncties van verschillende frequenties toe te passen op elke 3D-coördinaat. Voor een gegeven 3D-punt $p = (x, y, z)$ wordt de positional encoding uitgebreid naar:

$$PE(p) = \left[\sin(2^0\pi p), \cos(2^0\pi p), \sin(2^1\pi p), \cos(2^1\pi p), \ldots, \sin\left(2^{(L-1)}\pi p\right), \cos\left(2^{(L-1)}\pi p\right)\right]$$

Naarmate het coderingsniveau, weergegeven door L, toeneemt, neemt ook de frequentie van de sinusfuncties toe. Dit resulteert in meer oscillaties in de gecodeerde posities, wat helpt om fijne details in de ruimtelijke scène vast te leggen.

In de gegenereerde visualisaties (figuur 4.3) toont de blauwe lijn de originele straal in de 3D-ruimte. De zwarte stippen op deze lijn vertegenwoordigen specifieke punten langs de straal, terwijl de rode stippen hun positional encoding weergeven. Bij een lager niveau van L zijn de oscillaties rondom de punten minimaal, maar naarmate L toeneemt, zoals bij $L = 10$, worden de oscillaties veel dichter en frequenter, wat de mogelijkheid van het model om subtiele ruimtelijke variaties te detecteren verder verbetert.



Figuur 4.3: 3D-visualisatie van artificieel gegenereerde ray met positional encoding.

De essentie van positional encoding in de context van Neural Radiance Fields (NeRF) is een techniek die wordt gebruikt om het vermogen van neurale netwerken te verbeteren om details met een hoge frequentie in 3D-ruimte vast te leggen.

4.2.4 Volume rendering

In dit gedeelte wordt uitgelegd hoe een scène kan worden gerenderd met behulp van de klassieke volume rendering functie.

$$C(r) = \int_{t_n}^{t_f} T(t)\sigma(r(t))c(r(t), d)\, dt$$

met

$$T(t) = exp(-\int_{t_n}^{t} \sigma(r(s))\, ds$$

Volume rendering is een techniek die wordt gebruikt om een 3D-scène te projecteren op een 2D-beeldvlak, zoals het scherm van een computer. Deze techniek maakt gebruik van een integraalvergelijking om de kleur van een punt op een straal (ray) te berekenen.

De volume rendering functie $C(r)$ berekent de kleur van een object op basis van een ray. De ray wordt beschreven door een functie $r(t) = o + td$, waarbij o de oorsprong van de ray is en d de richting van de ray. De integratie in de volume rendering functie wordt toegepast over de lengte van de ray, van t_n tot t_f.

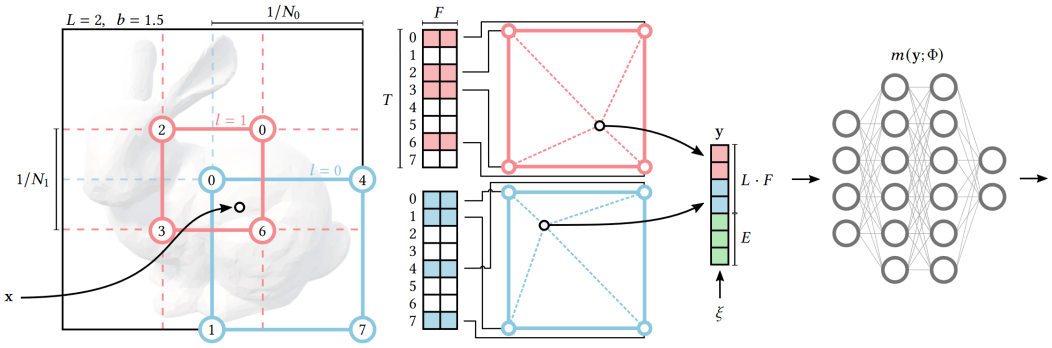
In de formule voor $C(r)$ wordt gebruik gemaakt van de extinctiecoëfficiënt $\sigma(r(t))$, die aangeeft hoeveel licht wordt geabsorbeerd of verstrooid terwijl het door het medium reist. De transmissiefunctie $T(t)$ vertegenwoordigt de hoeveelheid licht die wordt doorgelaten terwijl het door het medium reist, en wordt berekend door een exponentiële afname te integreren van t_n tot t.

Het resultaat van de integraalvergelijking $C(r)$ geeft de uiteindelijke kleur van het punt op de ray, en dit proces wordt herhaald voor elke pixel op het scherm om het complete beeld van de 3D-scène te renderen.

De volume rendering functie speelt een cruciale rol in het genereren van realistische beelden van 3D-scènes en wordt veel gebruikt in computer graphics en visualisatie om complexe materialen en lichteffecten te modelleren. Het renderen van scènes met volume rendering vereist geavanceerde wiskundige en computationele technieken om de integratie nauwkeurig en efficiënt uit te voeren.

4.3 Instant-NGP: Multiresolution Hash Encoding

Ontwikkeld door NVIDIA, is de Multiresolutie Hash Encoding een innovatieve benadering die is gepresenteerd in het paper "Instant Neural Graphics Primitives with a Multiresolution Hash Encoding"[30]. Deze methode heeft aanzienlijke interesse gewekt binnen de computer vision community, met name door de verbeteringen in memory efficiëntie en trainingssnelheid voor NeRF. De kern van de Multiresolutie Hash Encoding is het optimaliseren van het trainingsproces van neurale renderen door inputdata op een unieke manier te coderen.



Figuur 4.4: Pipeline encoderen van afbeelding [30].

Het start met het segmenteren van het domein in hiërarchische vierkante roosters, elk gedefinieerd in meerdere dimensies d. In deze benadering wordt een tweedimensionale context beschouwd. De illustratie toont enkel een deel van de groene en blauwe roosters voor een beter overzicht. Let op: de roosters segmenteren de gehele afbeelding. Daarnaast zijn er, afhankelijk van de dimensie, nog extra tegels aanwezig.

Elk rooster kent een resolutie die systematisch toeneemt ten opzichte van zijn voorganger. De factor van deze toename is variabel. De bepalende parameters zijn: L (aantal lagen), $N_{\min}$ (minimale resolutie) en $N_{\max}$ (maximale resolutie). Vervolgens wordt de groeifactor vastgesteld als:

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

Daarmee wordt de resolutie voor een specifiek roosterniveau ℓ bepaald:

$$N_\ell = \lfloor N_{\min} \times b^\ell \rfloor$$

Stap 1: Celselectie

Bij het overwegen van een specifiek niveau ℓ, wordt de invoercoördinaat x geschaald volgens de roosterresolutie van dat niveau. Voorafgaand aan het afronden, worden de grenzen van de cel bepaald door de schaling van x met N_ℓ. Formeel gezien, voor het naar beneden afronden:

$$\lfloor x_\ell \rfloor := \lfloor x \cdot N_\ell \rfloor$$

en voor het naar boven afronden:

$$\lceil x_\ell \rceil := \lceil x \cdot N_\ell \rceil$$

Vervolgens worden de coördinaten van de celhoeken geïdentificeerd op basis van deze geschaalde waarden.

Stap 2: Coördinatie en Hashing

Elke hoekcoördinaat wordt vervolgens geconverteerd met behulp van een hashfunctie:

$$h(x) = x \oplus (\pi_i y)$$

Waarbij $\oplus$ verwijst naar de bitwise XOR-bewerking en π_i grote priemgetallen zijn. In deze tweedimensionale setting zijn slechts twee coëfficiënten noodzakelijk. Deze hashcoördinaten dienen als indices binnen een specifiek geformuleerde hashtabel. Deze tabel is gekoppeld aan elk roosterniveau en bevat twee kernparameters: T (aantal slots) en F (features per slot). De uiteindelijke plek binnen de tabel wordt bepaald door de modulo van T te nemen.

Stap 3: Verfijning door Interpolatie

Na het identificeren van de relevante hashtabelslots voor een bepaalde roostercel, worden de waarden verfijnd met lineaire interpolatie om een exacte waarde voor x vast te stellen. Hierbij wordt bilineaire interpolatie gebruikt voor een tweedimensionale context. Deze interpolatie-stap zorgt ervoor dat de codering continu blijft, wat essentieel is voor een gradiëntgebaseerde optimalisatie.

Stap 4: Integratie van Data

Op dit moment zijn voor x diverse vectoren van grootte F geformuleerd, elk corresponderend met een roosterniveau ℓ. Om deze data samen te brengen, worden de vectoren geïntegreerd in een enkele codering van grootte $L \times F$. Dit gecombineerde resultaat wordt vervolgens gevoed aan een compact neuraal netwerk met ReLU-activatiefunctie.

Hoofdstuk 5

Mesh reconstructie

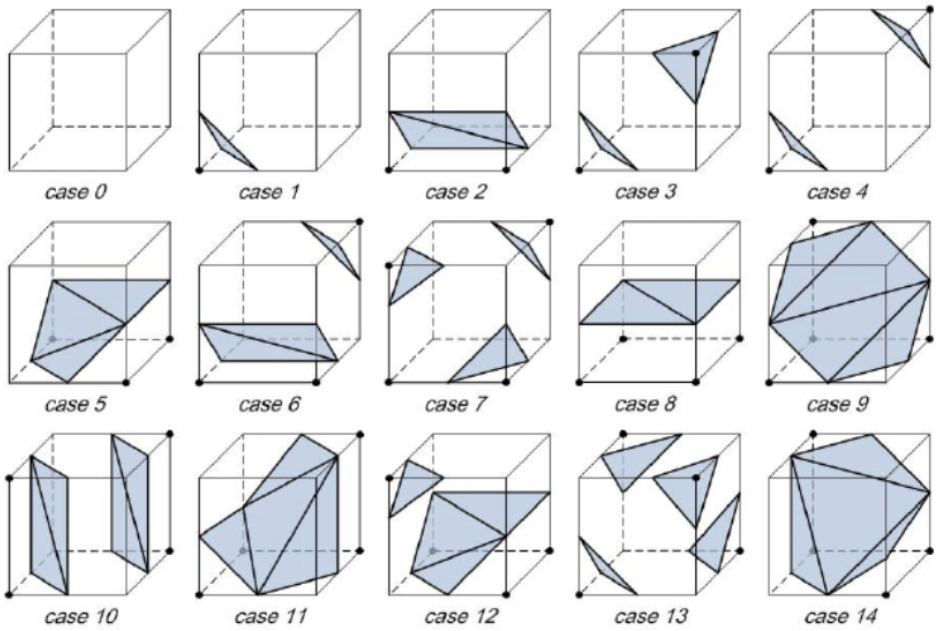
Deze sectie behandelt mesh reconstructie van neural radiance fields met behulp van het Marching Cubes algoritme. Dit algoritme werd voor het eerst geïntroduceerd om driehoeksmodellen te genereren van oppervlakken met constante dichtheid uit 3D medische gegevens [31]. Net als bij volume rendering, werkt dit direct op voxel data. Het Marching Cubes algoritme genereert een 3D model vanuit een voxel grid door een kubus door de voxel grid te laten marcheren, vandaar de naam Marching Cubes.

Om te bepalen of een datapunt in de voxelgrid in aanmerking komt, moet het voldoen aan een bepaalde drempelwaarde, aangeduid als σ. Bij neural radiance fields wordt de doorlaatbaarheid (dichtheid, aangeduid met f) gebruikt. Een oppervlak wordt gegenereerd op locaties waar de dichtheid de drempelwaarde σ kruist. Dus wanneer een voxelhoekpunt een waarde f heeft boven de drempelwaarde σ en een aangrenzend punt een waarde f onder σ, dan wordt er een oppervlaksegment gegenereerd dat die twee punten verbindt. Mathematisch gezien, als er twee punten A en B zijn met dichtheidswaarden $f(A)$ en $f(B)$, dan is de positie van het oppervlaksegment, P, gegeven door:

$$P = A + \frac{\sigma - f(A)}{f(B) - f(A)} \times (B - A)$$

Het originele Marching Cubes algoritme definieerde 15 unieke configuraties. Later werd dit uitgebreid tot 33 aparte cases [32]. Deze "cases"zijn te vinden in een lookup table en afhankelijk van de positie van de vertices zal het juiste oppervlak worden geconstrueerd.

Figuur 5.1 toont alle unieke cases afhankelijk van de posities van de vertices. De zwarte punten op de hoekpunten van de kubussen vertegenwoordigen datapunten uit de voxelgrid waarbij de dichtheid boven de drempelwaarde σ ligt. Het bepalen van het oppervlak gebeurt door middel van lineaire interpolatie van vertices op een edge, zoals hierboven wiskundig beschreven.



Figuur 5.1: De 14 unieke cases uit het oorspronkelijke artikel voor oppervlakte reconstructie [31].

Hoofdstuk 6

Implementatie details

6.1 Project structuur

Dit hoofdstuk behandelt de implementatiedetails . De komende secties bieden een uitgebreide uitleg van de implementatie en de bevindingen voor elk onderdeel. De volgende pipeline illustreert de stappen van video naar 3D-object:

De pipeline om vanuit afbeeldingen tot een 3D mesh te komen verloopt als volgt. Allereerst wordt U-Net getraind. U-Net is gekozen vanwege zijn efficiënte architectuur voor het segmenteren van netwerken met weinig trainingsdata. Bovendien presteert U-Net goed bij objecten 'in het wild'. Omdat we niet het gehele object segmenteren, maar enkel een object op de voorgrond, is eveneens voor dit netwerk gekozen. Na de training worden afbeeldingen geëxtraheerd uit een video, waarbij de focus ligt op het object dat gemodelleerd moet worden. U-Net werkt namelijk met afbeeldingen en niet met video's. Vervolgens kan het getrainde U-Net worden ingezet om een masker te creëren dat gebruikt wordt voor de segmentatie van een object. De gesegmenteerde afbeeldingen dienen vervolgens om de pose (oriëntatie en positie) van de camera te bepalen. Met de pose voor elke gesegmenteerde afbeelding kan de NeRF (of instant-ngp) worden getraind om een radiance field te genereren. Tenslotte wordt er via het marching cubes algoritme een mesh geëxtraheerd als laatste stap.

Deze pipeline combineert diverse technieken, van het vastleggen en bemonsteren van video's tot het trainen van neurale netwerken voor beeldsegmentatie en 3D-reconstructie. Het uiteindelijke resultaat omvat een gedetailleerde 3D-mesh van de scène, waarbij het prominente object als afzonderlijk en geïdentificeerd element wordt weergegeven. Dit vertegenwoordigt een boeiende toepassing van computer vision en machine learning, waarbij complexe 3D-modellen worden gecreëerd vanuit eenvoudige 2D-afbeeldingen.

De experimenten zijn uitgevoerd op een computersysteem met de onderstaande specificaties:

1. Besturingssysteem: Microsoft Windows 11 Pro, versie 10.0.22621, Build 22621.
2. Processor: AMD Ryzen 7 7700x.
3. Systeemgeheugen: 64 GB.
4. GPU: NVidia RTX 4080 met 16 GB geheugen.
5. Python-versie (via Conda): 3.10.10.
6. CUDA-versie: 10.7.
7. Pytorch-versie: 2.0.1 (stable)

6.2 U-Net

In sectie 3.2.1 is een uitgebreide beschrijving gegeven van de oorspronkelijke architectuur van U-Net. Ondanks deze gedetailleerde beschrijving is er besloten om enkele gematigde aanpassingen te maken, omdat U-Net intrinsiek flexibel is en ruimte biedt voor dergelijke wijzigingen.

Alvorens we de aanpassingen behandelen, is het belangrijk om te verklaren waarom voor U-Net is gekozen. Hoewel dit al kort is aangestipt in sectie 3.2, verdient het verdere toelichting. U-Net is een architectuur die niet alleen hoogwaardige prestaties levert en efficiënt omgaat met geheugen, maar ook geen last heeft van het verdwijnende gradiënten-probleem. Daarnaast is het een flexibele architectuur die snel traint. Bij grotere datasets kan het identificeren van objecten samen met hun bijbehorende maskers uitdagend zijn. Mocht U-Net niet aan de verwachtingen voldoen, dan bestaat er ook nog een opvolger: U2-Net.

De eerste significante aanpassing in de architectuur heeft betrekking op het gebruik van 'padded' convolutie-operaties binnen de dubbele convolutielaag. Dit specifieke ontwerpkeuze minimaliseert de computationele-overhead tijdens de preproccessing fase. In het oorspronkelijke onderzoek hebben de auteurs een aanpak toegepast waarbij extra pixels rond de randen van de oorspronkelijke gegevens werden gespiegeld. Dit werd gedaan om de effecten van unpadded convolutielagen te compenseren. Een andere belangrijke toevoeging is de implementatie van batchnormalisatie. U-Net en batchnormalisatie zijn in dezelfde tijdsperiode geïntroduceerd. Batchnormalisatie, zoals beschreven in [4], biedt meerdere voordelen, waaronder het versnellen van het trainingsproces van het neurale netwerk en het verminderen van de gevoeligheid voor hyperparameterafstemming. Bovendien draagt het bij aan het voorkomen van gradiëntexplosies, een bekend probleem in het trainen van diepe neurale netwerken.

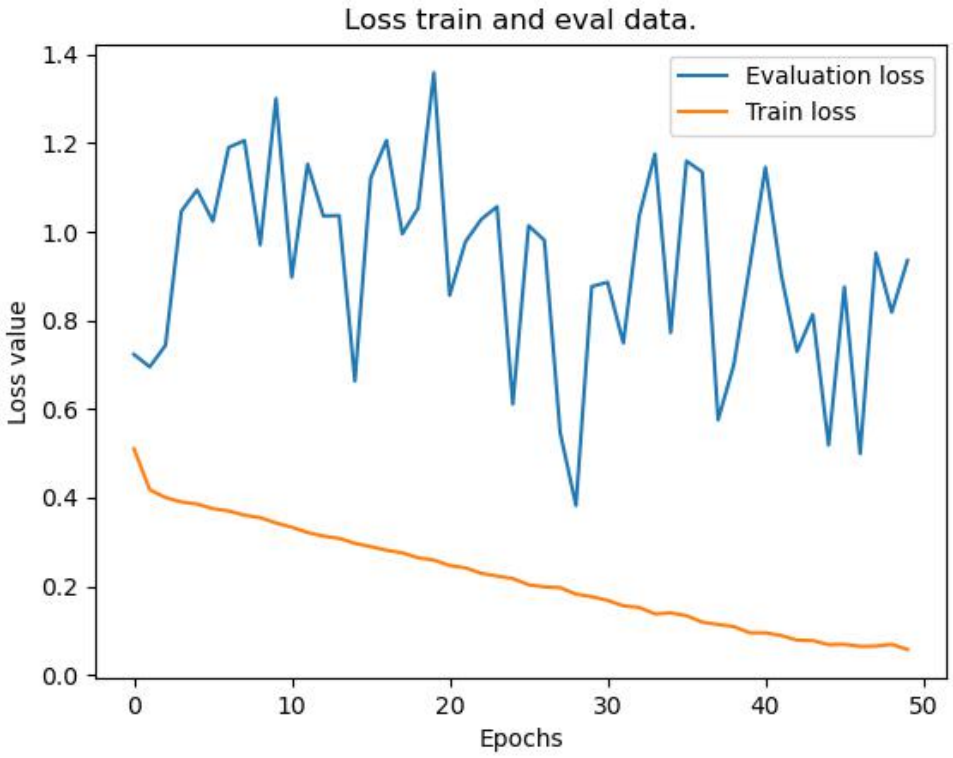
Voor het trainen van het U-Net model wordt gebruikgemaakt van de ECSSD-dataset [33]. Deze dataset bestaat uit een verzameling van 1000 afbeeldingen, elk vergezeld van hun respectievelijke masks. Ter evaluatie van het getrainde model wordt de CSSD-dataset [34] gebruikt. Deze dataset omvat 200 afbeeldingen, elk met bijbehorende masks.

Bij het evalueren van de prestaties van het model worden verschillende metrieken gehanteerd. Ten eerste wordt gebruikgemaakt van de BCEWithLogitsLoss, wat staat voor de "Binary Cross-Entropy Loss with Logits". Deze variant van de klassieke binary cross-entropy loss function is rekenkundig stabieler dan de traditionele BCELoss gevolgd door een sigmoid-activatie, zoals beschreven in [35]. Bovendien worden nauwkeurigheid en de Dice-score gebruikt als aanvullende evaluatiematen. De Dice-score, een veelgebruikte metriek in beeldsegmentatie, meet de overeenkomst tussen de voorspelde en werkelijke mask. Om de veranderingen in loss-waarde over de trainingsperiode te visualiseren, wordt Figuur 6.1 getoond. Hierbij worden de epochnummers op de X-as weergegeven en de corresponderende loss-waarde op de Y-as. Dit geeft inzicht in de evolutie van de netwerkprestaties tijdens het trainingsproces en evolutieproces.

De accuracy metric vergelijkt, op een pixel-niveau, de overeenkomst tussen de ground truth en de predicted output die is gegenereerd door het model. Dit proces omvat een gedetailleerde vergelijking van elke individuele pixel in de predicted output van het model met de bijbehorende pixel in de ground truth-gegevens. Deze vergelijking wordt pixelgewijs uitgevoerd en stelt vast in hoeverre de modelvoorspelling overeenkomt met de werkelijke waarden op een pixel-per-pixel basis.

Om dit te visualiseren, wordt Figuur 6.2 gepresenteerd, waarbij de horizontale as (X-as) de verschillende epochnummers voorsteld, terwijl de verticale as (Y-as) de accuracy metric in procent toont. Deze grafiek biedt inzicht in hoe de nauwkeurigheid van het model zich ontwikkelt naarmate het gedurende de trainingsperiode vordert. Het meten van nauwkeurigheid op pixelniveau geeft ons inzicht in hoe goed het model individuele objecten en grenzen in beeldsegmentatie leert en voorspelt.

De Dice-score is een metriek die wordt gebruikt in beeldsegmentatie om objectgelijkenis op een objectieve manier te evalueren. Hierbij wordt de ground truth vergeleken met de uitvoer van



Figuur 6.1: BCEWithLogitsLoss per epoch voor training en evaluatie dataset.

het model. In Figuur 6.3 wordt een grafiek getoond die de verloop van de Dice-score per epoche weergeeft. De Dice-score varieert tussen 0 en 1, waarbij een score van 1 een volledige overlap aangeeft en een score van 0 duidt op geen enkele overlap.

De formule voor de Dice-score is:

$$Dice = 2 * (Intersectie)/(Predictie + GroundTruth)$$

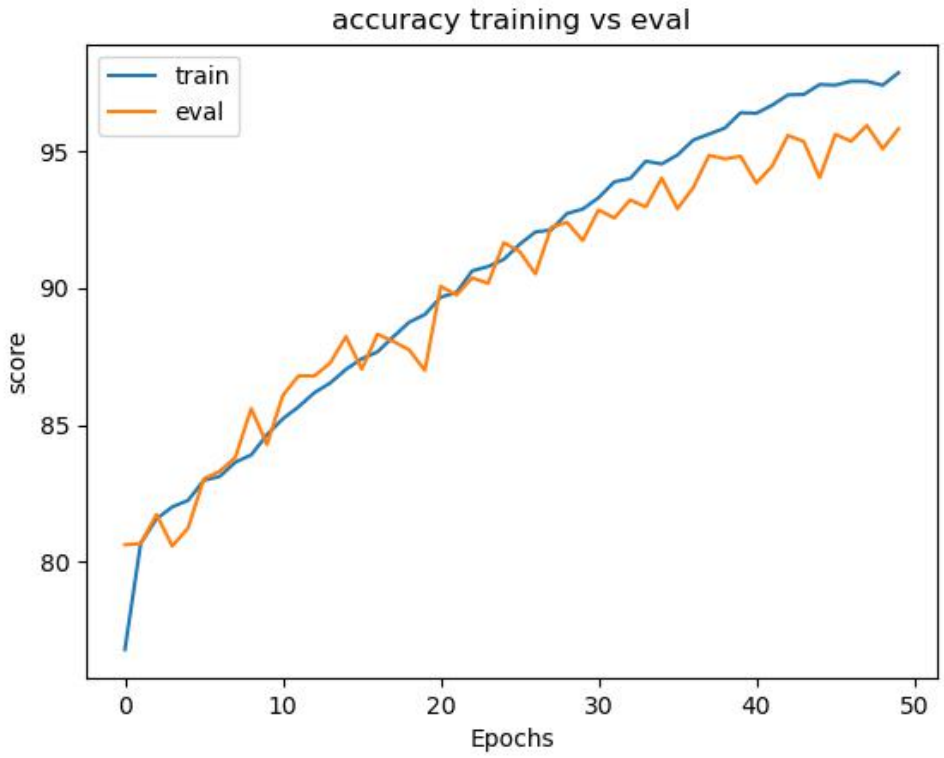
De "Intersectie"wordt berekend door elementgewijs de ground truth en prediction met elkaar te vermenigvuldigen, waarna een reduce-sum operatie wordt toegepast op de element-gewijs vermenigvuldigde arrays. De overige delen van de formule zijn eenvoudig af te leiden. Voor een uitgebreidere toelichting over de Dice score, zie sectie 2.6.

6.3 Segmenteren

Het hoofddoel van dit onderzoek is de transformatie van videodata naar een 3D-mesh representatie. Ondanks dat het U-Net model voornamelijk is ontworpen voor afbeeldingssegmentatie, biedt video, als reeks van afbeeldingen, interessante kansen voor het gebruik van U-Net in objectsegmentatie.

Elke video heeft een specifieke beeldsnelheid, vaak uitgedrukt in frames per seconde (FPS). Als voorbeeld: een 10-seconden durende video met een FPS van 30 leidt tot 300 frames, wat in feite 300 individuele afbeeldingen zijn. Deze afbeeldingen kunnen efficiënt uit de video worden geëxtraheerd met behulp van het *imageio-framework*[36].

Na training van het netwerk en het verkrijgen van de afbeeldingen (frames) uit een video, kunnen deze afbeeldingen worden gesegmenteerd met hun overeenkomstige image masks. Deze masks



Figuur 6.2: Accuracy model in %.

worden geproduceerd door het U-Net model tijdens de inferentie. Om alleen het gesegmenteerde object zichtbaar te maken, wordt een bitwise AND-operatie uitgevoerd op de originele afbeelding en het gegenereerde mask.

De werking van deze bitwise AND-operatie op afbeeldingen is als volgt: voor elk overeenkomstig pixel in de twee afbeeldingen (de originele afbeelding en het masker), als beide pixels een waarde van 1 hebben, dan is het resultaat ook 1, anders is het resultaat 0. Dit kan wiskundig worden uitgedrukt als:

$$R(x, y) = A(x, y) \wedge B(x, y) \tag{6.1}$$

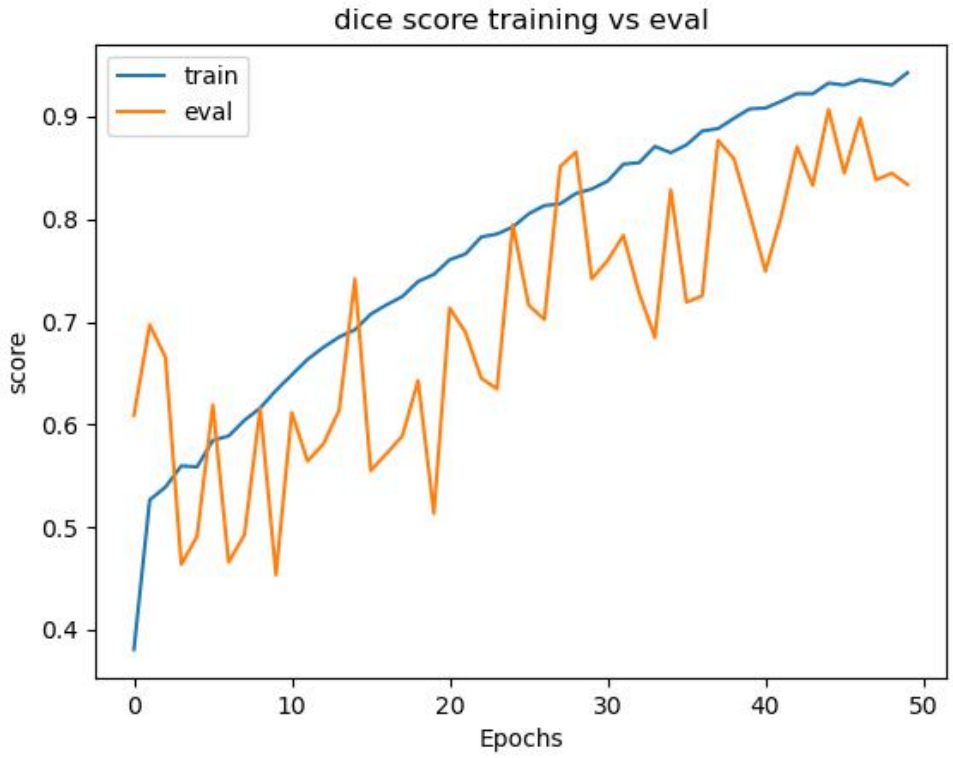
waarbij:

- R het resultaat van de AND-operatie is.
- A en B de twee te vergelijken afbeeldingen of maskers zijn.
- x, y de coördinaten voorstellen van een specifieke pixel in de afbeelding.
- $\wedge$ de AND-operatie symboliseert.

Het eindresultaat van deze operatie is een afbeelding waarop alleen het gewenste object zichtbaar is, terwijl de achtergrond weggehaald is. Dit proces staat ook bekend als 'Salient object detection'. Er is een illustratief voorbeeld van een dergelijk gesegmenteerd beeld dat in figuur 6.4 is weergegeven. Deze bewerking wordt op alle uit de video gehaalde frames toegepast.

6.4 Data voorbereiding voor NeRF met COLMAP

Deze sectie behandelt het proces van het voorbereiden van data voor NeRF. De input voor NeRF bestaat uit de cameraposities en -oriëntaties. De cameraposities en -oriëntaties worden



Figuur 6.3: Dice score

m.b.v. COLMAP bepaald. COLMAP [37, 38, 39] is een populaire open-source tool voor 3D-reconstructie van afbeeldingen. Het maakt gebruik van fotogrammetrische oplossingen, waaronder Structure from Motion[40] (SfM) en Multi-View Stereo[41] (MVS).

COLMAP begint met het extraheren en beschrijven van features voor elke afbeelding. Deze features zijn interessante punten die gemakkelijk herkenbaar en traceerbaar zijn over verschillende afbeeldingen, zoals hoeken of textuurgebieden. Op basis van de beschrijvingen van deze punten wordt elke afbeelding vergeleken met alle andere afbeeldingen om overeenkomsten te vinden. Dit leidt tot een lijst van feature-paren die vermoedelijk naar hetzelfde 3D-punt in de wereld verwijzen.

Vervolgens wordt voor elk paar afbeeldingen een geometrisch model geschat, zoals de fundamentele of essentiële matrix. Deze matrices coderen de epipolaire geometrie tussen de twee afbeeldingen en kunnen worden gebruikt om de relatieve rotatie en translatie tussen de twee camera's te bepalen. De fundamentele matrix F kan bijvoorbeeld worden uitgedrukt als:

$$F = [e'] \times P' P^{+} \tag{6.2}$$

waarbij $[e']$ de antisymmetrische matrix is afgeleid van het epipool e' en P en P' de projectiematrices zijn van twee verschillende cameraposities.

Na het schatten van de relatieve cameraoriëntaties en -posities voor afbeeldingsparen, wordt een globale optimalisatie uitgevoerd om alle cameraoriëntaties en -posities in een gemeenschappelijk coördinatensysteem te brengen.

De relatieve rotaties kunnen worden uitgedrukt in verschillende vormen, waaronder rotatiematrix, eulerhoeken of quaternions. Een quaternion q voor rotatie kan worden gegeven door:

$$q = (w, x, y, z) \tag{6.3}$$

Figuur 6.4: Originele afbeelding, segmentatie van U-Net-inferentie en de gesegmenteerde afbeelding

waarbij w het reële deel is en (x, y, z) het imaginaire deel. COLMAP kiest ervoor om quaternions te gebruiken omdat ze compact zijn en niet gevoelig zijn voor gimbal lock, een probleem dat zich voordoet bij eulerhoeken.

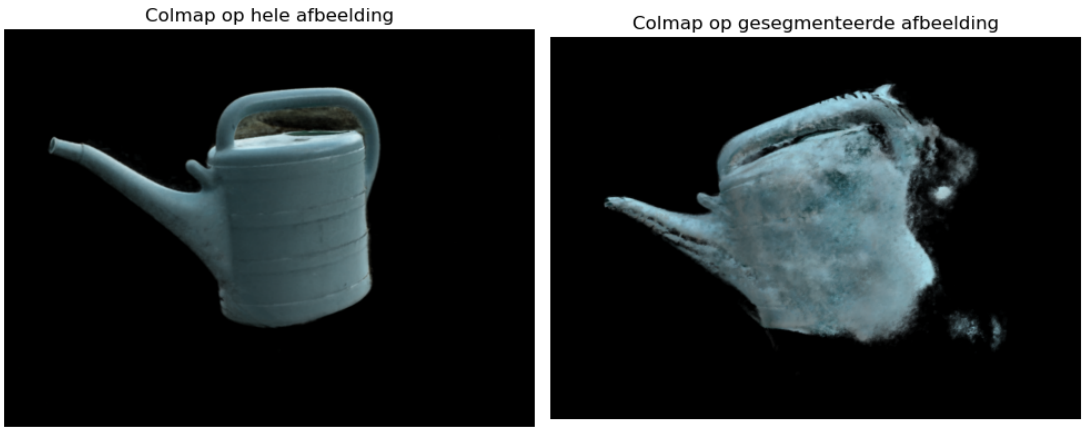
De translatiegegevens geven de positie van de camera in de 3D-ruimte weer. Met deze quaternion en translatiegegevens kan een transformatiematrix T worden opgesteld als:

$$T = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \tag{6.4}$$

waarbij R de rotatiematrix is, afgeleid van de quaternion, en t de translatievector vertegenwoordigt.

Tot slot worden de quaternion- en translatiegegevens, de output van COLMAP, omgezet in bruikbare transformatiematrices die de positie en oriëntatie van elke camera in een 3D-scène beschrijven. Dit wordt in een transform.json-bestand geschreven waarbij elk frame gekoppeld wordt aan zijn camerapositie en -oriëntatie.

Het gebruik van COLMAP om transformatiematrices te bepalen voor en na beeldsegmentatie is cruciaal. Wanneer de positie en oriëntatie vóór de segmentatie worden vastgesteld op de originele frames, duurt dit proces langer. Toch levert het opmerkelijke resultaten op. Het bepalen van de matrices voor gesegmenteerde beelden is daarentegen sneller, maar gaat wel ten koste van de kwaliteit van het eindresultaat. Figuur 6.5 illustreert het verschil in effect op het radiance field in beide situaties.



Figuur 6.5: Impact van pose-bepaling op het radiance field.

De reden achter dit effect is vrij eenvoudig. COLMAP kan, dankzij een groter aantal features, de camera matrix nauwkeuriger vaststellen. Een onnauwkeurige positie of oriëntatie kan echter

het netwerk hinderen bij het leren van een scène. Bij gesegmenteerde objecten kan COLMAP alleen features vinden op het object zelf en aan de randen ervan. Dit betekent niet alleen dat er minder features beschikbaar zijn voor het bepalen van de matrices, maar ook dat deze features zo sterk op elkaar lijken dat er fouten kunnen optreden bij het matchen ervan tijdens het oplossen van de epipolaire geometrie[42]. Voor de beste resultaten is het dus essentieel om deze waarden vast te stellen vóór het segmentatieproces.

6.5 NeRF

Na de transformatie van het prominente object in de video naar gesegmenteerde afbeeldingen, kan de training van NeRF beginnen. Dit is echter alleen mogelijk als de juiste cameramatrix is geëxtraheerd met het script dat besproken is in sectie 6.4. Het is cruciaal om te begrijpen dat de volgorde van deze stappen een significante invloed heeft op het uiteindelijke resultaat. Als Colmap wordt toegepast op de direct uit de video verkregen afbeeldingen, zonder eerdere segmentatie, zal NeRF niet correct convergeren naar een bruikbare radiance field. Daarom moet Colmap altijd worden uitgevoerd ná de segmentatie van de afbeeldingen.

Een triviaal maar belangrijk detail is dat als het object niet goed gesegmenteerd kan worden, dit een negatief effect zal hebben op het eindresultaat. Tijdens experimenten waarbij de segmentatie niet correct werd uitgevoerd en het object gaten vertoonde, leidde dit ook tot een mesh met gaten bij de extractie uit de radiance field.

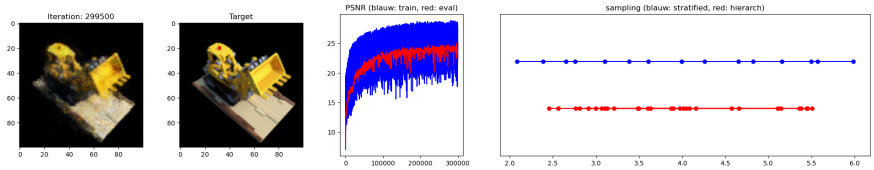
Met de juiste transform.json voor de gesegmenteerde afbeeldingen op zijn plaats, kan de training van start gaan. In eerste instantie werd de oorspronkelijke Vanilla NeRF-architectuur gebruikt om een radiance field te genereren. Hoewel dit effectief was voor data van lagere resolutie, zoals synthetische data, bleek het niet effectief voor data van hogere resolutie. Daarom is voor data van hogere resolutie de NeRF-versie van NVidia gebruikt. Meer details hierover worden besproken in de volgende sectie.

6.5.1 Vanilla NeRF

Technische aspecten van de Neural Radiance Field zijn al uitgebreid besproken in sectie 4.2. Deze sectie gaat dieper in op de bevindingen tijdens de experimentele fase. Het oorspronkelijke doel was om met Vanilla NeRF een mesh te genereren. De implementatie van Vanilla NeRF is echter in PyTorch gedaan. Autograd, een kerncomponent van PyTorch, wordt gebruikt voor gradient descent. Dit systeem slaat tussentijdse waarden op van tensorbewerkingen, waardoor een computationele grafiek ontstaat. Grotere netwerken hebben meer lagen en operaties, elk met hun geheugenvereisten. Omvangrijke invoergegevens of grote laagformaten verhogen het geheugengebruik. Bij terugpropagatie worden gradiënten stapsgewijs berekend met behulp van deze opgeslagen waarden. Het geheugen slaat ook de gradiënten voor elke parameter op. Bij training met grotere batches neemt het geheugengebruik toe. Daarom vergen grote netwerken of datasets veel geheugen, wat het nut van strategieën zoals gradiëntcontrolepunten of modelparallelisme onderstreept.

De Vanilla NeRF-architectuur is diepgaand en gebruikt twee netwerken: één voor 'coarse' en één voor 'fine' sampling. Bij 'synthetische' data met afmetingen van 100x100x3 zijn er geen problemen. Echter, bij grotere inputdata neemt ook de tussentijdse tensorwaarde toe, waardoor Vanilla NeRF veel geheugen gebruikt. Ondanks dat er slechts één afbeelding per forward pass werd gebruikt, was het geheugengebruik aanzienlijk. Daarom is er overgeschakeld naar Instant-NGP van NVLabs, dat een compactere architectuur en een andere vorm van encodering hanteert. Het primaire doel van dit proefschrift is niet het repliceren of verbeteren van een Neural Radiance Field, maar het genereren van een mesh uit zo'n veld. Daarom is er niet verder ingegaan op geavanceerde geheugenbeheerstrategieën in PyTorch.

Zoals eerder aangegeven, is er wel getraind op synthetische data. Figuur 6.6 toont de trainingsresultaten voor deze data.



Figuur 6.6: Training resultaat vanilla NeRF

De eerste afbeelding toont de gegenereerde output, terwijl de tweede de werkelijke (ground truth) weergave laat zien. Vervolgens wordt de PSNR (peak-signal-to-noise-ratio) gepresenteerd voor zowel de trainingsdata als de evaluatiedata. De evaluatiecurve vertoont een redelijk stabiele stijging. Daarentegen is er een aanzienlijke variatie te zien bij de trainingsdata. Dit kan verklaard worden doordat bepaalde inputparameters (de positie en kijkrichting van de camera) resulteren in minder nauwkeurige renderresultaten dan andere. Desondanks is er over het algemeen een opwaartse trend zichtbaar, wat suggereert dat het netwerk de scène succesvol leert modelleren. De laatste grafiek visualiseert de sampling intervallen voor zowel stratified als hierarchische sampling. Hieruit blijkt duidelijk dat stratified sampling zich concentreert op een bepaald stratum, terwijl hierarchische sampling meer gericht en gefocust samplet.

6.5.2 Instant-NGP

Instant-NGP, ontworpen door NVidia, is een geavanceerd NeRF netwerk. In tegenstelling tot de traditionele NeRF van Mildenhall et al., gebruikt deze versie slechts één netwerk en hanteert het een unieke encoding, genaamd Hash-encoding. Dankzij deze encoding kan het radiance field op een snellere en geheugenefficiënte manier worden aangeleerd. Voor meer details over deze encoding, zie sectie 4.3.

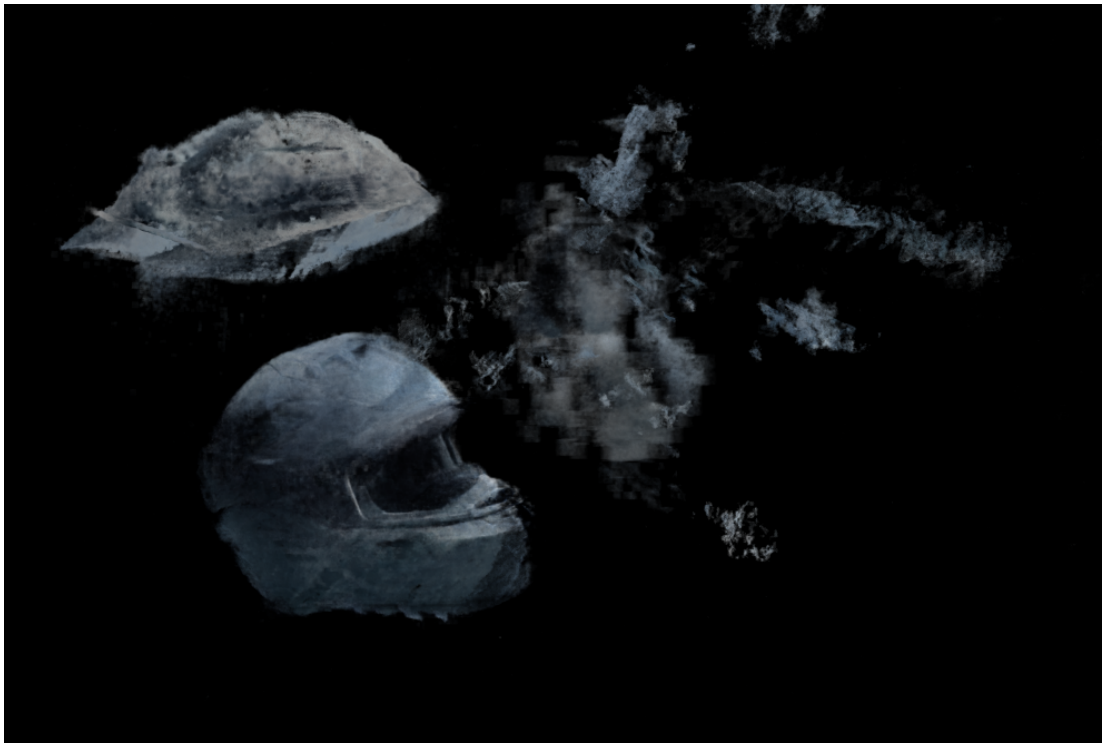
Zoals eerder opgemerkt in de introductie van sectie 6.5, had de originele NeRF problemen met geheugenefficiëntie. Daarom is het kant-en-klare Instant-NGP netwerk een geschikte keuze om een mesh te creëren en te extraheren uit afbeeldingen die zijn verkregen uit een video.

Zodra de gesegmenteerde afbeeldingen samen met hun cameramatrix beschikbaar zijn, kan het trainingsproces voor het radiance field starten. Een bijkomend voordeel van Instant-NGP is dat het in staat is om de radiance field in real-time te visualiseren. De radiance field, geproduceerd door Instant-NGP, is te zien in Figuur 6.7.

Opvallend is dat het gerenderde radiance field onvolkomenheden vertoont, zoals gaten in het eindresultaat. Dit is hoofdzakelijk te wijten aan de inadequate segmentatie door U-Net. Tijdens de segmentatie werden delen van het voorgrond-object weggelaten (gaten in object). De kwaliteit van de afbeeldingen heeft significant invloed op het eindresultaat. Een verbeterde segmentatieprocedure zou dit probleem kunnen verhelpen.

Er is een vooraf getraind U2-Net van GitHub ingezet om verbeterde segmentatieresultaten te bereiken [43]. Dit U-net was in staat objecten naadloos te segmenteren zonder gaten in de voorgrond-object. Echter, na het extraheren van de camera matrix en de training via GitHub, werden er artefacten waargenomen, zoals te zien is in figuur 6.8. De oorzaak van deze artefacten lag in de kwaliteit van de opgenomen video. Beelden die geblurd waren en afbeeldingen waren niet gelijkmatig verdeeld over verschillende invalshoeken. In het volgende hoofdstuk worden objecten met geoptimaliseerde resultaten getoond, waarbij tijdens de opnames rekening is gehouden met de genoemde beperkingen.

Figuur 6.7: Radiance field real-time gerendered met gaten (u-net)



Figuur 6.8: Radiance field real-time gerendered met gaten (rembg)

Hoofdstuk 7

Resultaat

De voorgaande sectie behandelde uitvoerig de implementatiedetails en de (kwantitatieve en kwalitatieve) resultaten van elk onderdeel in de pipeline, met uitleg over de gemaakte keuzes. Het primaire doel van dit proefschrift is het vormen van een 3D mesh aan de hand van radiance fields. In deze sectie ligt de focus op de resultaten, die uitgebreid worden besproken.

7.1 U-Net vs U2-net (rembg-package)

Figuur 6.4 toont het masker dat door mij is getraind. Op het eerste gezicht lijkt er niets mis. Echter, bij nader onderzoek zijn er gaten in het masker zichtbaar. Deze gaten zorgen voor een slecht radiance field, wat resulteert in een inferieure mesh. In dit geval zijn de gaten niet zeer groot, maar dat is niet altijd het geval voor alle geëxtraheerde frames of objecten. De keuze voor U-Net als segmentatie-tool werd gemaakt omdat, volgens auteurs, dit model snel kan leren van een beperkte dataset en objectief kwalitatieve resultaten kan leveren voor objecten in diverse omstandigheden. Gezien het doel om het creëren van 3D-objecten te democratiseren (met behulp van radiance fields), leek U-Net een gepaste keuze. Echter, de resultaten toonden het tegendeel aan. Daarom is er overgestapt naar de opvolger van U-Net, namelijk U2-Net, om te onderzoeken of betere segmentatie mogelijk is met een auto-encoder deep-learning model. Er is geen nieuw model vanaf de grond opgebouwd, maar in plaats daarvan is een bestaand model met voorgetrainde gewichten gebruikt [43]. Vervolgens zijn beide methoden vergeleken door de SSD-dataset [34] te segmenteren en de resultaten te contrasteren met de ground truth.

	U-Net	**U2-Net (Rembg)**
SSIM	0.6578932476489098	0.9386606667303793
RMSE	5.1803596544726185	2.195993513804987
Accuracy	0.6514971778117543	0.8550214386230386

Tabel 7.1: Metrics voor U-Net en U2-Net

U2-Net presteert beter op elke metric (tabel 7.1) voor het segmenteren van afbeeldingen vergeleken met zijn voorganger U-Net. Bovendien bevatten de gesegmenteerde objecten met U2-Net geen gaten in het voorgrondobject, wat de reden was om een voorgetraind model te gebruiken voor het segmentatieproces. Dit leidde ertoe dat NeRF betere resultaten produceerde.

7.2 Objectieve meting mesh

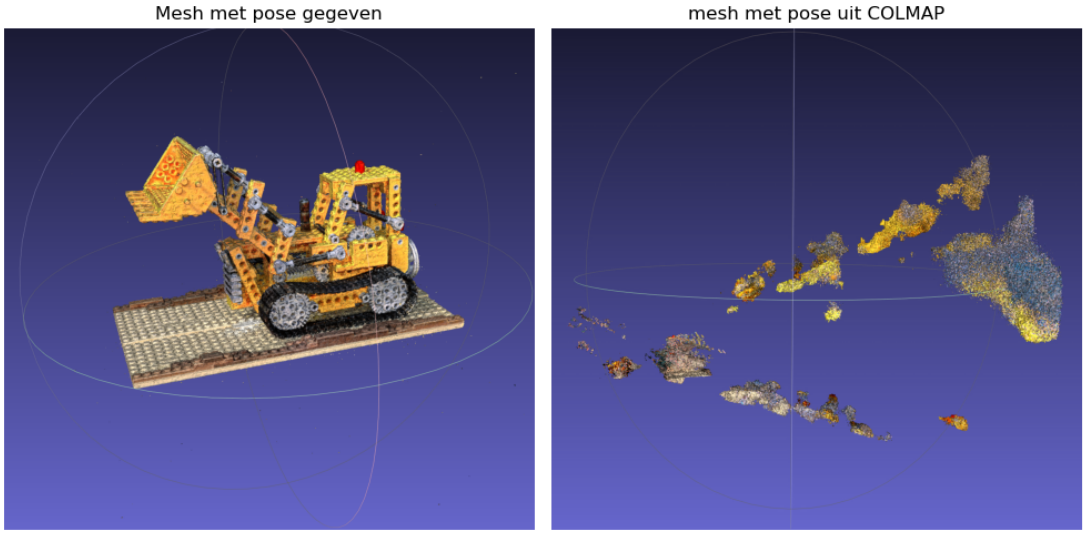
Alledaagse objecten waarvan meshes worden geproduceerd, kunnen niet op een objectieve en kwalitatieve manier worden beoordeeld. In sectie 7.3 zullen objecten ïn het wild"besproken worden. Deze sectie zal daarom gebruikmaken van een synthetische dataset. Als ground truth

wordt de bekende Lego dataset gebruikt [44]. Deze Lego set wordt in meerdere papers als referentie gehanteerd voor het evalueren van meshreconstructie. De synthetische dataset afbeeldingen van Vanilla NeRF [45] worden gebruikt. Deze zijn voorzien van camera- en positie matrices. Het radiance field wordt getraind en geëvalueerd met zowel deze matrices als met door colmap geëxtraheerde matrices.

	Met matrices	**Colmap**
Chamfer distance	14.36000812074233	-

Tabel 7.2: Vergelijking van de Chamfer distance voor synthetische datasets met gegeven camera- en positie matrices en die geëxtraheerd met Colmap.

Zoals waargenomen kan worden in tabel 7.2, ontbreekt de waarde voor COLMAP. De afwezigheid van deze waarde kan worden toegeschreven aan het onvermogen van COLMAP om een mesh te reconstrueren voor objecten zonder achtergrond. Conform de bevindingen in sectie 6.4, slaagt COLMAP er niet in om voor dergelijke objecten voldoende features te extraheren en daardoor de benodigde matrices adequaat te achterhalen. Dit impliceert dat een directe vergelijking van de meshes, verkregen uit beide methoden, niet haalbaar is. Als gevolg daarvan kan de Chamfer distance voor de synthetische datasets slechts een beperkte indicatie geven over de effectiviteit van de reconstructiemethoden, ongeacht het gebruik van Colmap.



Figuur 7.1: Geëxtraheerde meshes voor syntetische data.

In figuur 7.1 worden ter illustratie van de synthetische data beide meshes gepresenteerd. Aan de linkerzijde wordt een mesh weergegeven die als adequaat kan worden beschouwd. Aan de rechterzijde is de pose-estimation uitgevoerd met behulp van het COLMAP-algoritme. Dit onderstreept het significante belang van de matrices gerelateerd aan positie en rotatie binnen dit domein. Door slechts één variabele, namelijk de pose, aan te passen in het gehele proces kan leiden tot aanzienlijke variaties in de eindresultaten.

7.3 Mesh objecten in het wild

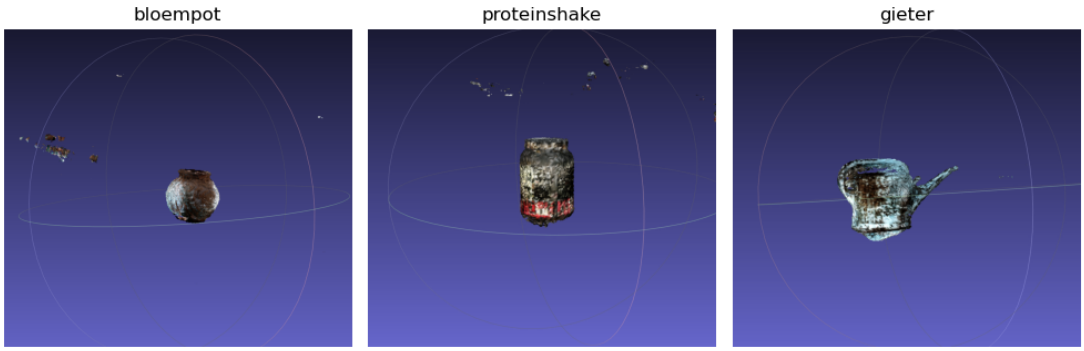
Met “objecten in het wild” bedoelen we complexe scènes. Anders gezegd: beelden met een drukke achtergrond. Het zijn taferelen uit het dagelijks leven. De resultaten worden uitvoerig besproken, maar een objectieve evaluatie van de meshes is niet mogelijk omdat er geen ground truth voorhanden is. De volledige pipeline is nu toegepast op video’s. De resultaten zijn

gegenereerd met dezelfde hyperparameters (encoder, model en density threshold raymarching). Figuur 7.2 toont de uiteindelijke resultaten voor verschillende objecten.

Bij het uitvoeren van de volledige pipeline op een videobeeld, ontstaat er een mesh met suboptimale resultaten. Bij een complex object, waarvan de mesh opnieuw is opgebouwd, zijn er duidelijk zichtbare artefacten. Het meest opvallende probleem is de brokkelige uitvoer. Ook vertonen de kleuren artefacten in alle drie de voorbeelden. Het laatste object, namelijk de gieter, heeft de beste meshreconstructie. Dit in tegenstelling tot de eerste twee, waar nog enkele zwevende objecten rondom het object te zien zijn. De reden voor deze artefacten kan tweevoudig zijn:

1. De opgenomen video en daarmee de resulterende afbeeldingen.
2. Posities geassocieerd met de trainingsafbeeldingen.

Dit kan ook een oorzaak zijn van de suboptimale mesh en het brokkelige effect. De synthetische data in figuur 7.1 vertoonde dit probleem niet.



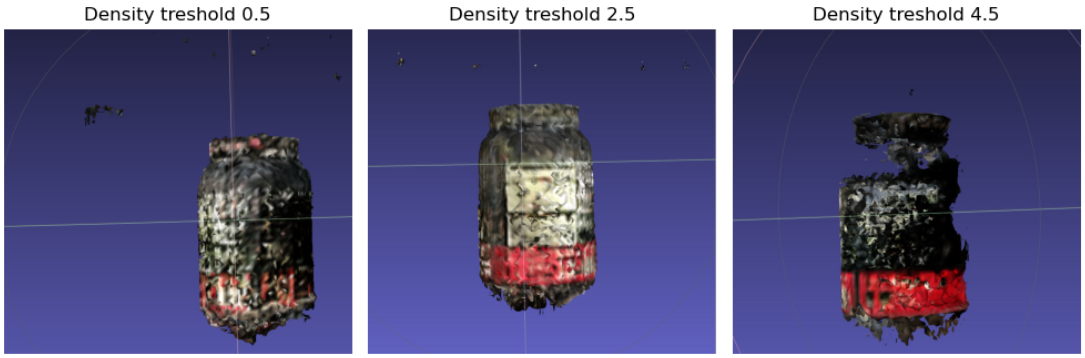
Figuur 7.2: Drie objecten waaruit meshes zijn gehaald.

In de huidige studie wordt nadrukkelijk aandacht besteed aan de invloed van de 'density threshold' binnen de radiance field ten aanzien van de mesh-constructie. Zoals geïllustreerd in Figuur 7.3, worden drie representaties gepresenteerd met verschillende density thresholds. Bij een oppervlakkige observatie van de eerste twee representaties, met respectievelijke thresholds van 0.5 en 2.5, lijken er geen significante discrepanties te zijn. Echter, een gedetailleerde analyse van de centrale regio van de middelste representatie onthult diepe inconsistente gebieden. De derde representatie, met een aanzienlijk hogere threshold, vertoont een opvallend verlies van bepaalde mesh-segmenten.

Deze observaties suggereren dat de density threshold een fundamentele parameter is in het reconstructieproces van de mesh. Hoewel men initieel zou kunnen concluderen dat een minimale threshold optimale resultaten zou opleveren, blijkt dit niet inherent correct te zijn. Een te lage threshold kan resulteren in een overmatige toename van vertices en triangles, met name in het centrale, niet direct zichtbare deel van het object. Dit kan onnodige complexiteit introduceren in de mesh-structuur. Bijkomend introduceert een extreem lage threshold potentieel meer zichtbare artefacten rondom het object. Ter illustratie: een density threshold van 4.5 resulteert in een afwezigheid van dergelijke losse artefacten rond het object.

7.3.1 Korte reflectie

In het voorgaande en het huidige hoofdstuk zijn meerdere observaties gedocumenteerd. Via de beschreven pipeline is het mogelijk een mesh te onttrekken uit een radiance field, startend vanuit een videobron. Echter, de voorlopige resultaten geven aan dat er aanvullend onderzoek nodig is om de kwaliteit van de uitkomsten te verhogen. Hoewel deze pipeline flexibel is, blijkt dat fouten in één onderdeel negatieve gevolgen kunnen hebben voor het gehele resultaat.



Figuur 7.3: Objecten met verschillende density tresholds.

Aspecten zoals beeldkwaliteit en segmentatie zijn hierbij belangrijk 6.5.2. Ook de kenmerken van de camera voor elke afbeelding, de resultaten van de mesh verkregen uit synthetische data illustreren dit duidelijk 7.2.

Eerdere analyses suggereren dat NeRF wellicht niet ideaal is voor mesh-extractie. Hierbij zou de aandacht kunnen verschuiven naar SDF-netwerken. Een recent artikel van Nvidia benadrukt het potentieel van deze SDF-netwerken [46]. Het kan nuttig zijn om een andere netwerkarchitectuur voor radiance fields te overwegen, met meer focus op modellering en minder op rendering. Neurale rendering is een opkomend vakgebied en verdere studie is essentieel om het volledig te begrijpen en te ontwikkelen.

Hoofdstuk 8

Conclusies

Met dit hoofdstuk naderen we de conclusie van dit uitgebreide proefschrift. Onze ambitieuze doelstelling om videobeelden om te zetten in een mesh-structuur, ondersteund door de neural radiance field technologie, heeft een veelbelovende fase bereikt. Hoewel de resultaten nog niet volmaakt zijn, hebben we aangetoond dat de beoogde transformatie realiseerbaar is. De in dit proefschrift uiteengezette pipeline demonstreert zowel robuustheid als aanpassingsvermogen. Elk segment van dit onderzoek leent zich voor individuele studie en optimalisatie. Het is reeds geïllustreerd dat verbeteringen binnen een specifiek segment substantiële invloed kunnen hebben op de algehele output. Dit introduceert echter ook complexiteiten, vooral bij het identificeren van specifieke invloedsfactoren op de totale resultaten. Deze modulariteit heeft dus zowel voordelen als uitdagingen.

Bij gelegenheden kan een gevestigd algoritme superieure resultaten opleveren in vergelijking met het vanaf nul trainen van een nieuw model, zoals in het geval van segmentatie. Bovendien hebben externe factoren zoals beeldkwaliteit en matrices gerelateerd aan camera positionering en oriëntatie hun eigen invloed. Echter, de vooruitzichten zijn hoopvol. Neural radiance fields zijn een opkomende technologie, en er is een exponentiële groei in gerelateerd onderzoek. Ik ben optimistisch dat de toekomstige ontwikkelingen deze uitdagingen zullen adresseren en het proces van het genereren van 3D modellen van alledaagse objecten aanzienlijk zal versnellen.

Reflecterend op dit traject, van initiatie tot huidige fase, ben ik overwegend positief over de behaalde resultaten. De potentie om dergelijke technieken te operationaliseren kan een paradigmaverschuiving in de 3D-modelleringsindustrie teweegbrengen. Deze vooruitgang is niet slechts een hypothetisch scenario; het lijkt een onvermijdelijke toekomst. Dankzij dit onderzoek heb ik niet alleen diepgaande kennis opgedaan over NeRF maar heb ik ook inzichtelijke expertise verworven binnen het bredere domein van computergraphics en machine learning.

In afsluiting van dit hoofdstuk wil ik een aantal persoonlijke reflecties en lessen delen die ik tijdens de uitvoering van dit project heb vergaard. De verleiding om een nieuw model te ontwikkelen kan sterk zijn, maar soms is het pragmatisch en efficiënter om voorgetrainde modellen te adopteren. Het ontwerpen van een architectuur is slechts een deel van de uitdaging. De significantie van data en preprocessing voor een machine learning model werd door mij in eerste instantie onderschat. Ik koos voor een praktische benadering, ervan uitgaande dat een directe implementatie me dichter bij de kern van de materie zou brengen. Echter, deze keuze ging niet zonder zijn eigen reeks uitdagingen. Confronterende momenten met deep learning modellen die onverwachte hindernissen opwierpen, waren niet ongewoon. Ik werd af en toe geconfronteerd met modellen die weerbarstig waren en niet conform verwachtingen presteerden. Dit leidde tot uitgebreide troubleshootingsessies rond compatibiliteitskwesties en de technische beperkingen van high-resolution beeldverwerking.

Bibliografie

[1] Yann LeCun e.a. „Handwritten Digit Recognition with a Back-Propagation Network". In: *NIPS*. 1989. URL: https://api.semanticscholar.org/CorpusID:2542741.

[2] Vincent Dumoulin en Francesco Visin. *A guide to convolution arithmetic for deep learning*. 2018. arXiv: 1603.07285 [stat.ML].

[3] Ian J. Goodfellow e.a. *Generative Adversarial Networks*. 2014. arXiv: 1406.2661 [stat.ML].

[4] Sergey Ioffe en Christian Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. 2015. arXiv: 1502.03167 [cs.LG].

[5] Alain Horé en Djemel Ziou. „Image Quality Metrics: PSNR vs. SSIM". In: *2010 20th International Conference on Pattern Recognition*. 2010, p. 2366–2369. DOI: 10.1109/ICPR.2010.579.

[6] Tage Sørensen e.a. „A method of establishing group of equal amplitude in plant sociobiology based on similarity of species content and its application to analyses of the vegetation on Danish commons". In: 1948. URL: https://api.semanticscholar.org/CorpusID:135206594.

[7] Lee R. Dice. „Measures of the Amount of Ecologic Association Between Species". In: *Ecology* 26.3 (1945), p. 297–302. ISSN: 00129658, 19399170. URL: http://www.jstor.org/stable/1932409 (bezocht op 19-08-2023).

[8] Zhou Wang e.a. „Image quality assessment: from error visibility to structural similarity". In: *IEEE Transactions on Image Processing* 13.4 (2004), p. 600–612. DOI: 10.1109/TIP.2003.819861.

[9] Andras Hajdu, Lajos Hajdu en Robert Tijdeman. „Approximations of the Euclidean distance by chamfer distances". In: *CoRR* abs/1201.0876 (2012). arXiv: 1201.0876. URL: http://arxiv.org/abs/1201.0876.

[10] Dan C. Ciresan e.a. „Deep Neural Networks Segment Neuronal Membranes in Electron Microscopy Images". In: *NIPS*. 2012. URL: https://api.semanticscholar.org/CorpusID:7725346.

[11] Olaf Ronneberger, Philipp Fischer en Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015. arXiv: 1505.04597 [cs.CV].

[12] Weichao Xu e.a. „Real-time object detection and semantic segmentation for autonomous driving". In: feb 2018, p. 44. DOI: 10.1117/12.2288713.

[13] Bartlomiej Wronski e.a. „Handheld Multi-Frame Super-Resolution". In: *ACM Trans. Graph.* 38.4 (jul 2019). ISSN: 0730-0301. DOI: 10.1145/3306346.3323024. URL: https://doi.org/10.1145/3306346.3323024.

[14] Alireza Taravat e.a. „Advanced Fully Convolutional Networks for Agricultural Field Boundary Detection". In: *Remote Sensing* 13.4 (2021). ISSN: 2072-4292. DOI: 10.3390/rs13040722. URL: https://www.mdpi.com/2072-4292/13/4/722.

[15] Jonathan Long, Evan Shelhamer en Trevor Darrell. *Fully Convolutional Networks for Semantic Segmentation*. 2015. arXiv: 1411.4038 [cs.CV].

[16] Ross Girshick e.a. *Rich feature hierarchies for accurate object detection and semantic segmentation*. 2014. arXiv: 1311.2524 [cs.CV].

[17] Liang-Chieh Chen e.a. *Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs*. 2016. arXiv: 1412.7062 [cs.CV].

[18] Hengshuang Zhao e.a. *Pyramid Scene Parsing Network*. 2017. arXiv: 1612.01105 [cs.CV].

[19] Vijay Badrinarayanan, Alex Kendall en Roberto Cipolla. *SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation*. 2016. arXiv: 1511.00561 [cs.CV].

[20] Kaiming He e.a. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV].

[21] Tao Zhou e.a. „RGB-D salient object detection: A survey". In: *Computational Visual Media* 7.1 (jan 2021), p. 37–69. DOI: 10.1007/s41095-020-0199-z. URL: https://doi.org/10.1007%2Fs41095-020-0199-z.

[22] Xuebin Qin e.a. „U2-Net: Going Deeper with Nested U-Structure for Salient Object Detection". In: deel 106. 2020, p. 107404.

[23] Jeong Joon Park e.a. *DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation*. 2019. arXiv: 1901.05103 [cs.CV].

[24] *Signed distance function - wikipedia*. https://en.wikipedia.org/wiki/Signed_distance_function. [Accessed 23-08-2023].

[25] *Ray marching - Wikipedia — en.wikipedia.org*. https://en.wikipedia.org/wiki/Ray_marching. [Accessed 23-08-2023].

[26] Alex Yu e.a. *Plenoxels: Radiance Fields without Neural Networks*. 2021. arXiv: 2112.05131 [cs.CV].

[27] Ben Mildenhall e.a. „NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis". In: *ECCV*. 2020.

[28] Jeong Joon Park e.a. „DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation". In: *CoRR* abs/1901.05103 (2019). arXiv: 1901.05103. URL: http://arxiv.org/abs/1901.05103.

[29] Matthew Tancik e.a. „Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains". In: *NeurIPS* (2020).

[30] Thomas Müller e.a. „Instant Neural Graphics Primitives with a Multiresolution Hash Encoding". In: *ACM Trans. Graph.* 41.4 (jul 2022), 102:1–102:15. DOI: 10.1145/3528223.3530127. URL: https://doi.org/10.1145/3528223.3530127.

[31] William E. Lorensen en Harvey E. Cline. „Marching Cubes: A High Resolution 3D Surface Construction Algorithm". In: *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '87. New York, NY, USA: Association for Computing Machinery, 1987, p. 163–169. ISBN: 0897912276. DOI: 10.1145/37401.37422. URL: https://doi.org/10.1145/37401.37422.

[32] E. V. Chernyaev. „Marching Cubes 33: Construction of topologically correct isosurfaces". In: (nov 1995).

[33] Jianping Shi e.a. „Hierarchical image saliency detection on extended CSSD". In: *IEEE transactions on pattern analysis and machine intelligence* 38.4 (2015), p. 717–729.

[34] Qiong Yan e.a. „Hierarchical saliency detection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2013, p. 1155–1162.

[35] Pytorch. *BCEWithLogitLoss*. https://pytorch.org/docs/stable/generated/torch.nn.BCEWithLogitsLoss.html. [Accessed 06-07-2023].

[36] Almar Klein e.a. *imageio/imageio: v2.31.1*. Versie v2.31.1. Jun 2023. DOI: 10.5281/zenodo.8025955. URL: https://doi.org/10.5281/zenodo.8025955.

[37] Johannes Lutz Schönberger en Jan-Michael Frahm. „Structure-from-Motion Revisited". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[38] Johannes Lutz Schönberger e.a. „Pixelwise View Selection for Unstructured Multi-View Stereo". In: *European Conference on Computer Vision (ECCV)*. 2016.

[39] Johannes Lutz Schönberger e.a. „A Vote-and-Verify Strategy for Fast Spatial Verification in Image Retrieval". In: *Asian Conference on Computer Vision (ACCV)*. 2016.

[40] Onur Ozyesil e.a. *A Survey of Structure from Motion*. 2017. arXiv: 1701.08493 [cs.CV].

[41] S.M. Seitz e.a. „A Comparison and Evaluation of Multi-View Stereo Reconstruction Algorithms". In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*. Deel 1. 2006, p. 519–528. DOI: 10.1109/CVPR.2006.19.

[42] *OpenCV: Epipolar Geometry — docs.opencv.org*. https://docs.opencv.org/3.4/da/de9/tutorial_py_epipolar_geometry.html. [Accessed 23-08-2023].

[43] *GitHub - danielgatis/rembg: Rembg is a tool to remove images background — github.com.* `https://github.com/danielgatis/rembg`. [Accessed 11-08-2023].

[44] *Blend swap — Lego 856 Bulldozer — blendswap.com.* `https://www.blendswap.com/blend/11490`. [Accessed 19-08-2023].

[45] Mildenhall et. al. *NeRF Data - Google drive — drive.google.com.* `https://drive.google.com/drive/folders/128yBriW1IG_3NJ5Rp7APSTZsJqdJdfc1`. [Accessed 17-08-2023]. 2020.

[46] Jacob Munkberg e.a. „Extracting Triangular 3D Models, Materials, and Lighting From Images". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Jun 2022, p. 8280–8290.