Text-To-3D

Aktuelle Themen KI (WS22/23)

Jason Schühlein (js450) - Hochschule der Medien

2023-03-17

Inhaltsverzeichnis

1 Einleitung

In den vergangenen Monaten haben Text-To-Image Modelle wie Stable Diffusion [1] oder DALL-E 2 [2] die Industrie verändert. Auch erste Text-To-Video Modelle wie Make-A-Video [3] wurden veröffentlicht. Der logische Schritt vorwärts ist nun Text-To-3D. Im Vergleich zu Text-To-Image ist Text-To-3D noch ein vergleichsweise unterentwickeltes Feld, aber auch hier hat sich viel getan in letzter Zeit.

Der Nutzen von Text-To-3D Modellen liegt auf der Hand, sie könnten die Medienlandschaft ähnlich stark prägen wie Text-To-Image Modelle, wenn auch eher mit Fokus auf industrielle Anwendungen und weniger an den Endverbraucher gerichtet wie es Text-To-Image..

2 Hintergrund

Im Folgenden betrachten wir kurz eine Einordnung in den Forschungsbereich der *3D Generation* sowie die vorherrschenden 3D Repräsentationen. Dann widmen wir uns kurz dem traditionellen 3D Workflow und behandeln schließlich die für Text-To-3D nötigen Grundlagen.

2.1 Einordnung

Der Forschungsbereich der *3D Generation* beinhaltet die Unterkategorien *3D Shape Generation* sowie *3D-aware Image Generation*. *3D Shape Generation* bezieht sich hierbei auf Modelle deren Output direkt ein 3D Modell ist, wohingegen Modelle der *3D-aware Image Generation* darauf abzielen, Bilder aus (neuen) Blickrichtungen zu erzeugen und im Zuge dessen eine 3D Repräsentation lernen, die extrahiert werden kann. Der entscheidende Unterschied liegt in der Supervision beider Unterkategorien: *3D Shape Generation* benötigt 3D Daten, *3D-aware Image Generation* lediglich 2D Daten. *Text-To-3D* ist ein kleiner, auf Text konditionierter Bereich des größeren Forschungsfeldes *3D Generation*. [4]

Dass das Feld als Ganzes noch hinterherhängt liegt an einigen Faktoren: so existieren beispielsweise wesentlich weniger Datensätze an gelabelten 3D Daten als an gelabelten Bildern. Um zu verdeutlichen wie groß der Unterschied beider Supervision Varianten ist, stellen wir kurz zwei der bekanntesten Datensätze für 2D und 3D Supervision gegenüber: *LAION 5B* [5] besteht aus 5,85 Milliarden Text-Bild Paaren, wohingegen *ShapeNet* [6] lediglich 51 Tausend Text-Shape Paare enthält. Aber auch die erhöhte Komplexität und damit den höheren Rechenaufwand den die weitere Dimension von 2D zu 3D mit sich bringt spielt eine große Rolle. Viele der aktuell bestehenden KI Modelle sind außerdem insofern

beschränkt, dass sie lediglich eine handvoll an Klassen lernen und davon 3D Modelle generieren können.

2.2 3D Repräsentationen

Es existieren mehrere Repräsentationsformen von 3D Modellen, jede mit eigenen Vor- und Nachteilen sowie Anwendungsgebieten. Vorherrschend im Bereich Computeranimation sowie Visual Effects sind natürlich Meshes, diese sind aber nicht unbedingt die beste Wahl für alle KI Anwendungen.

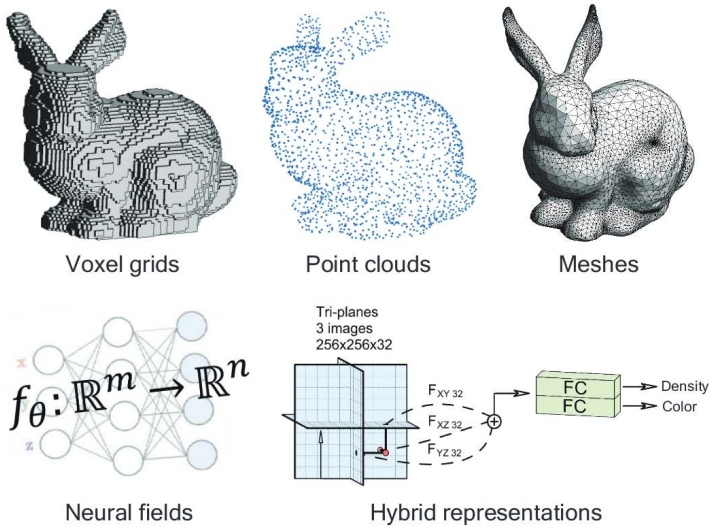

Abbildung 1: 3D Repräsentationen im Überblick [4].

2.2.1 Voxel Grids

Voxel Grids sind einfache Gitterstrukturen im dreidimensionalen Raum, vergleichbar mit Pixeln im zweidimensionalen Raum. In den einzelnen Gitterzellen werden Werte gespeichert wie Farbe und Dichte. Voxel Grids sind einfach abzufragen und werden häufig für Fluid Simulationen verwendet.

2.2.2 Point Clouds

Eine ähnlich primitive Datenstruktur bilden Point Clouds, diese sind unsortierte Sammlungen an Punkten im dreidimensionalen Raum. 3D Sensoren liefern häufig Point Clouds, was diese Form der Repräsentation sehr praktisch und gut verfügbar macht.

2.2.3 Meshes

Polygonale Meshes bilden die Basis vieler Endanwendungen, besonders im Animations und Visual Effects Bereich (siehe 3D Workflow). Meshes sind definiert durch Vertices, Edges und Faces und beschreiben damit auch die Verhältnisse zwischen den einzelnen Punkten. Großer Vorteil von Meshes gegenüber der Voxel Grids besteht in der kompakten Datenstruktur, denn ein großes oder fein aufgelöstes Voxel müsste etliche "leere" Gitterzellen speichern. Im Vergleich zu Point Clouds haben wir bei Meshes zusätzlich die Information wie welche Punkte miteinander verbunden sind. Gleichzeitig sind Meshes aber auch schwieriger von tiefen neuronalen Netzen interpretierbar und reproduzierbar als die einfacheren Point Clouds. [4]

2.2.4 Neuronale Felder

Neuronale Felder sind kontinuierliche, implizite Repräsentationen, realisiert mit einem neuronalen Netz. Ein bekannter Vertreter der Neuronalen Felder sind die *Neural Radiance Fields (NeRF)*. Dieses lernt eine neuronale Szenenbeschreibung aus mehreren Bildern der Szene aus unterschiedlichen Betrachtungswinkeln, vergleichbar mit Photogrammetrie. Konkret lernt ein NeRF die Abbildung $(x, y, z, \theta, \Phi) \rightarrow (R, G, B, \sigma)$, ein Mapping von Raumkoordinaten sowie Blickrichtung zu Farbwert und Dichte, also eine Art 3D Volumen. Aus diesem können dann neue Blickrichtungen gerendert werden welche nicht in den Trainingsdaten vorkamen. Vorgestellt wurden NeRFs von Mildenhall et al. in [7].

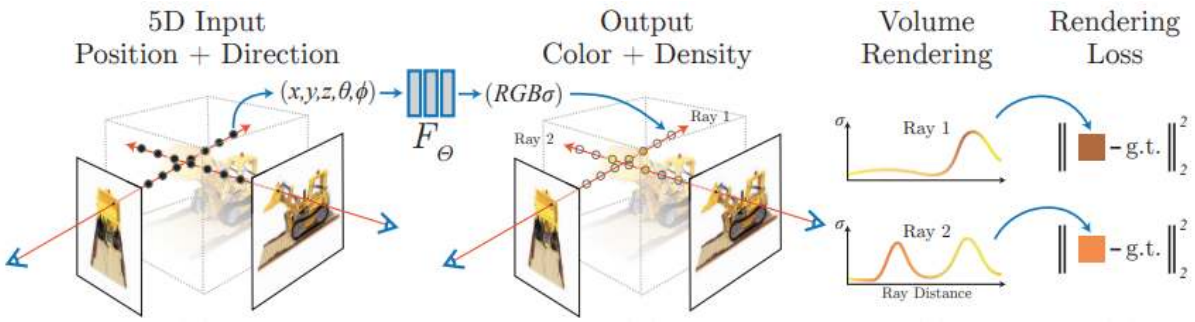

Abbildung 2: NeRF Training [7].

2.2.5 Hybride Ansätze

Um die Vorteile der bisherigen Repräsentationen zu vereinen und die Nachteile abzuschwächen werden aktuell häufig hybride Ansätze gewählt. Ein Beispiel für eine solche hybride Repräsentation ist die Verwendung von sog. *Tri-Planes* zur Verbesserung Neuronaler Felder in [8].

2.3 3D Workflow

Im einfachsten Fall findet sich auf Webseiten wie Turbosquid[1] bereits ein geeignetes 3D Mesh zum Kauf, potentiell sogar mit Rig für den späteren Animationsschritt. Sollte das Objekt real existieren, so kann man auch auf 3D Scans oder Photogrammetrie zurückgreifen um eine erste Point Cloud als Basis zu erhalten, die dann in ein Mesh umgewandelt wird. Auch hier gibt es Anbieter wie Quixel Megascans[2] die eine große Menge an hochauflösenden Scans in Form von texturierten Meshes zur Verfügung stellen. Je nach Anwendungsfall und Qualität benötigen die Modelle allerdings noch eine Aufbereitung, denn es besteht eine Vielzahl an verschiedenen Anforderungen. Im Kontext von Games ist man beispielsweise je nach Game Engine begrenzt in der Mesh Auflösung und präferiert Meshes mit einem geringeren Polycount, genannt Low-Poly-Mesh. Aber auch für Animationsfilme und VFX sind angemessene Polycounts sinnvoll, so brauchen Modelle im Hintergrund nur eine geringe Auflösung im Gegensatz zu Modellen im Vordergrund. Das Konzept dahinter nennt sich *Level of Detail (LOD)*. Weiterhin ist die Topologie der Meshes von besonderer Bedeutung, ganz besonders wenn Deformationen angewandt werden sollen bei organischen Objekten. Bei fehlerhafter Topologie (z.B. durch “non-manifold geometry”) kann es schnell zu störenden Artefakten beim Shading kommen. Ebenfalls wichtig für Shading sind die sogenannten UV Layouts welche beschreiben wie eine Textur auf ein Modell angebracht wird, vergleichbar mit dem Faltmuster oder Schnittmuster. Das UV Layout muss im Falle von Scans häufig erst noch erstellt werden oder zumindest optimiert.

Der traditionelle 3D Modeling Workflow kann unterteilt werden in 2 Bereiche: *Hardsurface Modeling* und *Sculpting* für organische Objekte. Meist wird mit Bildreferenzen oder Skizzen aus mehreren Betrachtungswinkeln gearbeitet, um das Modell möglichst wahrheitsgetreu reproduzieren zu können. Die Anforderungen bleiben dieselben, jedoch kann hier durch die künstlerische Freiheit wesentlich mehr experimentiert werden und von vornherein auf saubere Topologie geachtet werden. Auch prozedurale Workflows in Houdini oder Blender sind denkbar, hierbei wird eine Objektklasse durch Node-Trees parametrisiert. Damit kann dann eine Vielzahl ähnlicher 3D Modelle generiert werden die denselben Regeln folgen.

2.4 Technischer Hintergrund

Dieser Abschnitt bietet einen Überblick über die Schlüsseltechnologien CLIP und Diffusion auf denen aktuelle Text-To-Image aber auch Text-To-3D Modelle aufbauen.

[1] https://www.turbosquid.com
[2] https://quixel.com

2.4.1 CLIP

CLIP steht für *Contrastive Language Image Pretraining* und wurde am 26. Februar 2021 von OpenAI vorgestellt [9]. Es besteht aus einem Text Encoder der Text in ein Text Embedding umwandelt, sowie einem Image Encoder der Bilder entsprechend zu Image Embeddings umwandelt. Für einen Batch aus N Image-Text Paaren (x, y), also Bildern mit dazugehöriger Caption, werden alle Text Embeddings allen Image Embeddings gegenübergestellt. CLIP wird dann in Abbildung 3 darauf trainiert, die korrekten N Image-Text Paare zu identifizieren aus den $N \times N$ möglichen Paare. Konkret soll das Skalarprodukt korrekter Paare maximiert und falscher Paare minimiert werden.

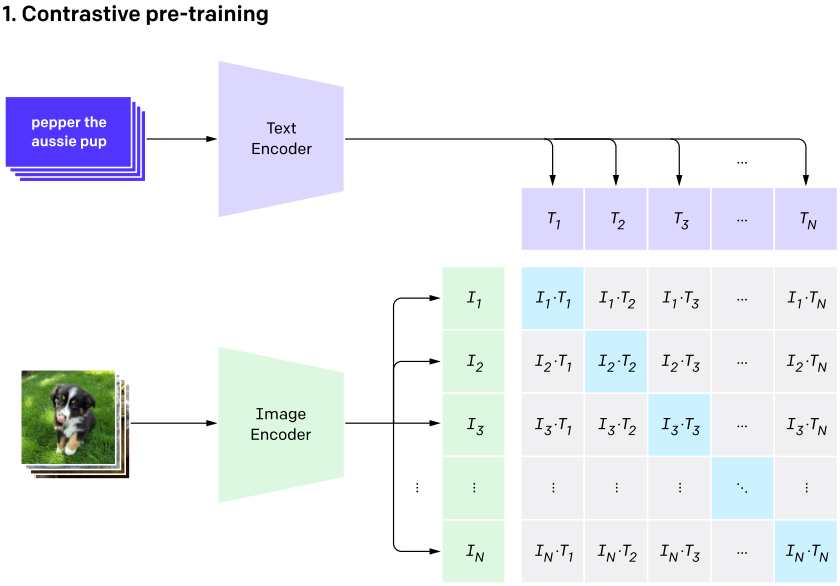

Abbildung 3: Training von CLIP [9]

Nach dem Training, also in der Inferenz, wird dann ein Bild und eine Caption gegenübergestellt. Der sogenannte CLIP Score beschreibt dann wie hoch die Ähnlichkeit beider Modalitäten ist, also wie gut Bild und Text zusammenpassen.

2.4.2 Diffusion

Diffusion Probabilistic Models (DPM) sind die Basis vieler aktueller Text-To-Image Modelle, darunter DALL·E 2 [2], Imagen [10] sowie Stable Diffusion [11]. Das Prinzip ist von der Thermodynamik inspiriert und wurde von Ho et al. in [12] vorgestellt. Diffusion Modelle gehören wie auch GANs zu den generativen Modellen, sie generieren Daten ähnlich zu den gesehenen Trainingsdaten.

Diffusion Modelle bestehen aus 2 Phasen, dem *Forward Diffusion Process* sowie dem *Reverse Diffusion Process*. Bei ersterem wird einem Bild Schritt für Schritt Gaußsches Rauschen aufaddiert, bis es nicht

mehr von purem Gaußschen Rauschen unterscheidbar ist. Der *Forward Diffusion Prozess* ist fix und wird nicht gelernt.

Im *Reverse Diffusion Process* wird ein UNet (Autoencoder mit Skip Connections) darauf trainiert einen einzelnen solchen Schritt rückgängig zu machen, hierzu bekommt es ein Bild der Forward Kette zum Zeitpunkt t und muss das Rauschen vorhersagen das von Zeitschritt $t-1$ zu t aufaddiert wurde. Zieht man das vorhergesagte Rauschen nun vom Input ab, erhält man eine Rekonstruktion des Bildes zum Zeitschritt $t-1$.

Für die Inferenz gibt man nun Gaußsches Rauschen auf den trainierten *Reverse Diffusion Process* und iteriert üblicherweise ≈ 1000 Schritte bis man ein Bild aus der Verteilung der Trainingsdaten erhält.

Um das Ergebnis der bisher *Unconditional Diffusion Models* lenken zu können, wird zusätzlich eine *Conditional* Variable in den Input gegeben. Bei einem Trainingsdatensatz bestehend aus Hunden könnte dies beispielsweise die Rasse des Hundes sein den wir nach der Inferenz sehen möchten. Die wichtigsten *Guidance* Methoden sind die *Classifier Guidance* sowie die *Classifier-free Guidance* [13].

Die *CLIP Guided Diffusion* gehört zur ersten Kategorie. Es wird in der Inferenz eines *Unconditional Diffusion Models* ein Classifier (hier CLIP) verwendet, um zu jedem Schritt im *Reverse Process* den Prompt mit dem generierten Bild zu vergleichen. Der Gradient wird dann verwendet, um das Bild in die Richtung der korrekten Klasse (bzw. eines hohen CLIP Scores) zu adaptieren.

Bei der *Classifier-free Guidance* wird kein weiteres Classifier Modell benötigt. Stattdessen wird ein *Conditional Diffusion Model* trainiert, wozu gelabelte Daten nötig sind. Allerdings wird *Conditional Dropout* angewendet, das bedeutet der Label oder der Prompt wird in $10-20\%$ der Fälle weggelassen. Das Modell lernt somit, sowohl *Conditional* als auch *Unconditional* zu arbeiten. Für die *Classifier-free Guidance* wird dann zu jeden Schritt ein Bild mit und ein Bild ohne *Conditional* generiert. Daraus wird die Differenz berechnet und verstärkt mit der *Guidance Scale*.

Größter Vorteil der Diffusion Modelle gegenüber GANs ist, dass kein Adversarial Training erforderlich ist. Allerdings haben Diffusion Modelle durch die große Anzahl an Iterationen auch wesentlich längere Inferenzzeiten.

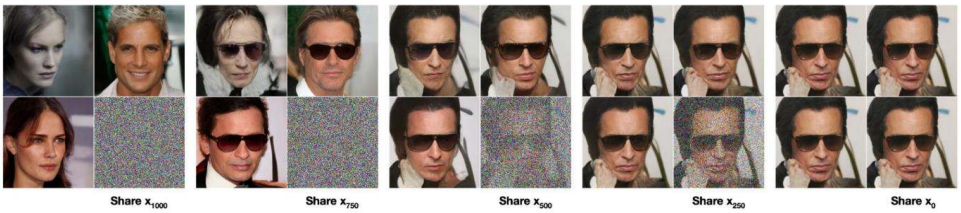

Abbildung 4: Reverse Diffusion auf dem CelebA-HQ Datensatz beginnend bei verschieden stark verrauschten Startbildern [12].

3 Entwicklungen und Meilensteine

Wir betrachten nun eine handvoll nennenswerter Meilensteine und Entwicklungen der letzten Jahre: Text2Shape aus 2018, CLIP-Forge aus 2021 sowie Dream Fields und CLIP-Mesh aus 2022.

3.1 Text2Shape

Eines der ersten nennenswerten Paper im Bereich Text-To-Shape Generation aber auch Text-To-Shape Retrieval ist *Text2Shape* von Chen et al. aus 2018 [14]. *Text2Shape* trainiert einen "joint embedding space" aus Text und eingefärbten Voxel und kombiniert diesen mit GANs um neue 3D Shapes zu generieren. Allerdings ist dieses Modell beschränkt auf die Klassen mit denen es trainiert wurde.

Abbildung 5: Vergleich zwischen Baseline, zwei Varianten von Text2Shape und Groundtruth [14]

3.2 CLIP-Forge

Autodesks *CLIP-Forge* aus 2021 umgeht das Problem der nötigen gelabelten 3D Daten, indem es ausschließlich ungelabelte 3D Modelle zum Training verwendet. Hier wird ein Autoencoder trainiert der die 3D Modelle erst encoded und danach wieder decoded, man erhält damit einen Latent Space der gesehenen Objektklassen. In Kombination mit CLIP kann man nun den Decoder in eine Richtung leiten

die eine hohen CLIP Score von Text und Rendering des 3D Modells hervorsagt. Allerdings sind wie auch zuvor die Klassen an 3D Modellen die man generieren kann beschränkt durch die gesehenen Klassen im Training. [15]

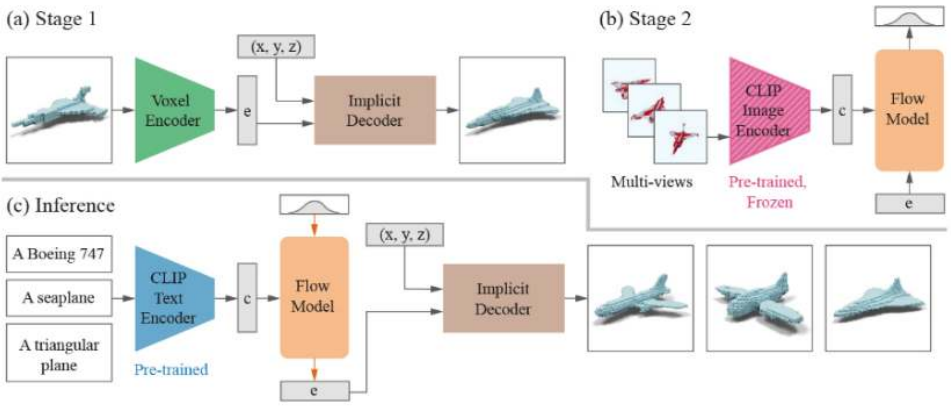

Abbildung 6: Aufbau von CLIP-Forge: Trainingsphasen a) und b), Inferenz c) [15].

In Abb. 6 ist der Aufbau von CLIP-Forge dargestellt: in Stage 1 des Trainings wird ein Voxel Autoencoder Modell trainiert auf Basis der ungelabelten 3D Trainingsdaten. Die Voxel Embeddings e werden dabei gespeichert. Sobald das Autoencoder Training abgeschlossen ist, wird in Stage 2 ein Flow Model darauf trainiert aus CLIP Image Embeddings der Trainingsdaten ein solches Voxel Embedding e zu generieren. Nun können Autoencoder und Flow Model zur Inferenz neuer 3D Shapes bzw. Voxel verwendet werden. Hierzu wird sich der "joint representation space" der CLIP Embeddings zunutze gemacht und das Flow Model mit CLIP Text Embeddings statt Image Embeddings inferiert, das resultierende Voxel Embedding e wird dann an den Decoder aus Stage 1 gegeben welcher daraus ein Voxel erzeugt (siehe Abb. 7 für weitere Generationen). [15]

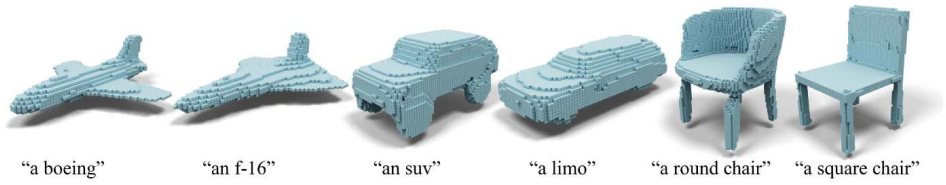

Abbildung 7: CLIP-Forge Resultate für Prompts innerhalb des Trainingsdatensatzes [15].

Die Resultate von CLIP-Forge zeigen durch die Anwendung von CLIP erste Zero-Shot Kapazitäten (siehe Abb. 8). Diese sind jedoch durch den Autoencoder beschränkt, denn dieser kann nur zuverlässig Shapes en- und decoden die in den Trainingsdaten vorgekommen sind. Zero-Shot bedeutet in diesem Kontext das Erzeugen von Daten außerhalb der gesehenen Trainingsdaten.

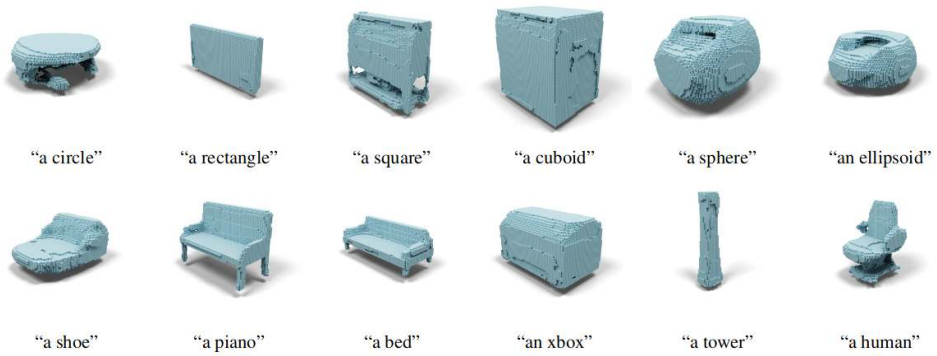

Abbildung 8: CLIP-Forge Resultate für Prompts außerhalb des Trainingsdatensatzes zeigen erste Zero-Shot Kapazitäten [15].

3.3 Dream Fields

Dream Fields verwendet eine andere 3D Repräsentation als vorherige Modelle, es greift auf NeRFs zurück und optimiert diese mittels CLIP. Vorgestellt wurde Dream Fields von Jain et al. in 2022 [16].

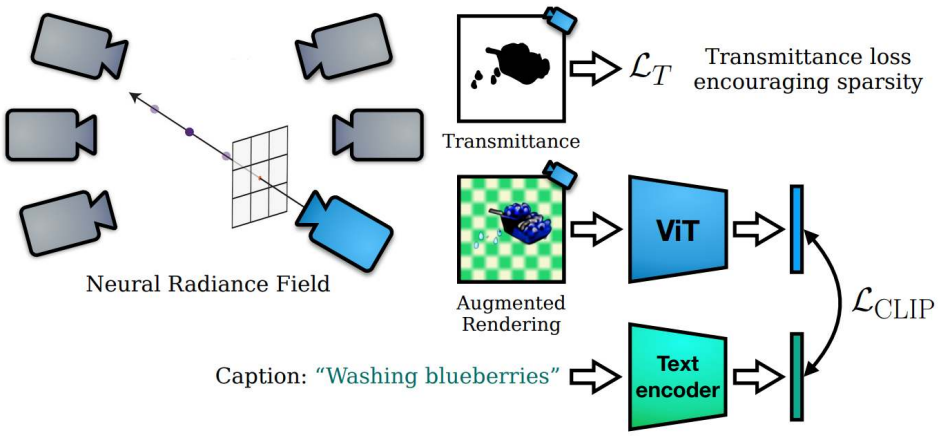

Abbildung 9: Konzept von Dream Fields [16].

Abbildung 9 zeigt das Konzept hinter Dream Fields: es wird ein NeRF zufällig initialisiert und Bilder aus verschiedenen Blickwinkeln gerendert, mit randomisiertem Hintergrund. Diese Bilder werden dann mittels CLIP Score mit dem Text Prompt verglichen und daraus der Loss $\mathcal{L}_{CLIP}$. Zusätzlich wird auch jedes Mal auch eine Transmittance Karte gerendert und ein Transmittance Loss $\mathcal{L}_T$ gebildet, der das NeRF dazu motiviert, zusammenhängende Objekte hervorzubringen. Beide Loss Funktionen werden dann verwendet um das NeRF Schritt für Schritt zu optimieren. [16]

Im Vergleich zu CLIP Forge ist Dream Fields Zero-Shot fähig, denn hier wird kein Autoencoder ange-

wandt, der CLIP in seiner Zero-Shot Fähigkeit beschneidet.

Die Ergebnisse der Dream Fields (siehe Abb. 10) sind NeRFs, diese müssten je nach Anwendungsfall erst zu Meshes umgewandelt werden. Für weitere, animierte Beispiele siehe die Projektwebsite von Dream Fields[3].

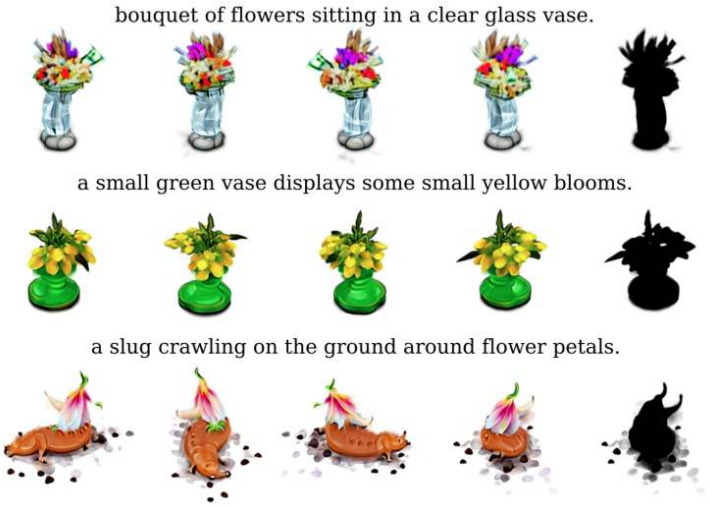

Abbildung 10: Dream Fields Resultate [16].

3.4 CLIP-Mesh

CLIP-Mesh ist vergleichbar mit Dream Fields, jedoch wird hier kein NeRF optimiert, sondern ein beliebiges Basis Mesh. Abbildung 11 zeigt den Aufbau und initialen Zustand des Modells mit einem Kugel Basis Mesh und der randomisierten Texture sowie Normal Map. Davon werden mit einem differenzierbaren Renderer mehrere einzelne Blickwinkel gerendert und dann mittels CLIP mit dem Text Prompt verglichen. Zusätzlich enthält der Loss auch einen Regularisierungsterm der dazu dient das Objekt “intakt” zu halten. Schließlich wird noch ein Diffusion Model verwendet, um das CLIP Text Embedding in ein CLIP Image Embedding zu überführen und mit dem Embedding des gerenderten Bildes zu vergleichen. [17]

Abbildung 12 zeigt Resultate von Dream Fields (oben) und CLIP-Mesh (unten), beide mit Ergebnissen vergleichbarer Qualität und auch beide Zero-Shot fähig, aber entsprechend in unterschiedlichen 3D Repräsentationen, weshalb visuell kein fairer Vergleich möglich ist. Jedoch ist anzumerken, dass das Training von Dream Fields wesentlich kostspieliger ist als das von CLIP-Mesh. Konkret benötigt Dream Fields für ein Objekt 24h auf 4 Highend GPUs wohingegen CLIP-Mesh lediglich 50min auf einer einzelnen Highend GPU benötigt und damit etwa um den Faktor 100 schneller ist. [17]

[3] https://ajayj.com/dreamfields

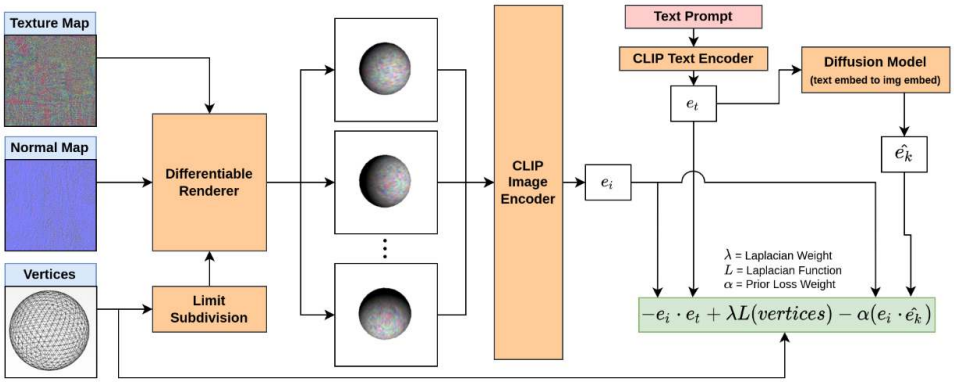

Abbildung 11: Konzept von CLIP-Mesh [17].

Abbildung 12: Resultate von Dream Fields (oben) und CLIP-Mesh (unten). Prompts: a) “mount everest” b) “a vase with pink flowers” c) “a hamburger” d) “Eiffel tower” e) “a red chair” [17].

4 State of the Art

An den bisherigen Modellen und Ansätzen kann man deutlich die Tendenz zur 2D statt 3D Supervision ablesen und im Zuge dessen wird sehr gerne auf CLIP Modelle zurückgegriffen um Text mit Bild, aber auch Bild mit Bild, vergleichen zu können. Die folgenden Modelle bilden nun State of the Art und stützen sich teilweise auf diese Tendenz, gehen aber auch einen Schritt weiter mit der Verwendung von vortrainierten Text-To-Image Modellen.

4.1 GET3D

GET3D ist ein GAN zur Shape Generation und kann neue Meshes generieren aus der Verteilung der Trainingsdaten. Vorgestellt wurde es 2022 von Gao et al. [18].

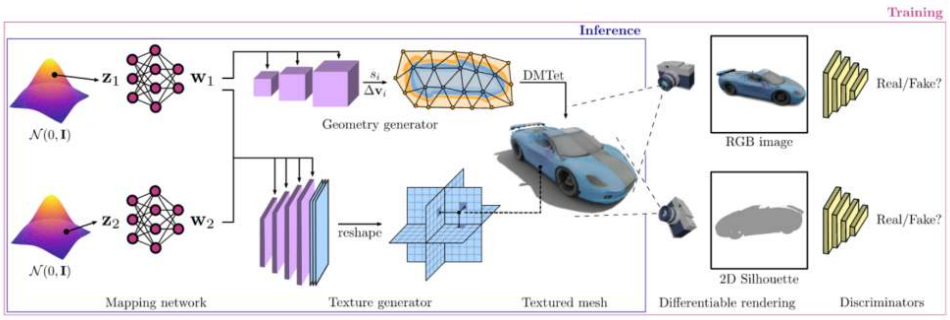

Abbildung 13: Aufbau von GET3D [18].

In Abb. 13 ist der Aufbau von GET3D abgebildet, es besteht aus zwei Generatoren und zwei Diskriminatoren. Ein Generator lernt ein 3D Signed Distance Field (SDF) und der andere Generator lernt ein Texture Field auf Basis von Tri-Planes (siehe Hybride Ansätze). Das Mesh wird mittels Deep Marching Tetrahedra (DMTet) aus dem SDF extrahiert und die Textur wird aus der Tri-Plane gesampled. Schließlich werden Bilder aus verschiedenen Blickwinkeln differenzierbar gerendert und zwei Diskriminatoren unterscheiden dann zwischen realen und künstlichen RGB Daten sowie Silhouetten. [18]

Abb. 14 zeigt Resultate trainierter GET3D Modelle, exportiert und visualisiert in Blender. Die Inferenz und damit die Generierung neuer Shapes der entsprechenden Klasse ist bei GANs vergleichsweise schnell. Allerdings dauert das Training eines einzelnen GET3D Modells 2 Tage auf 8 Highend GPUs [18].

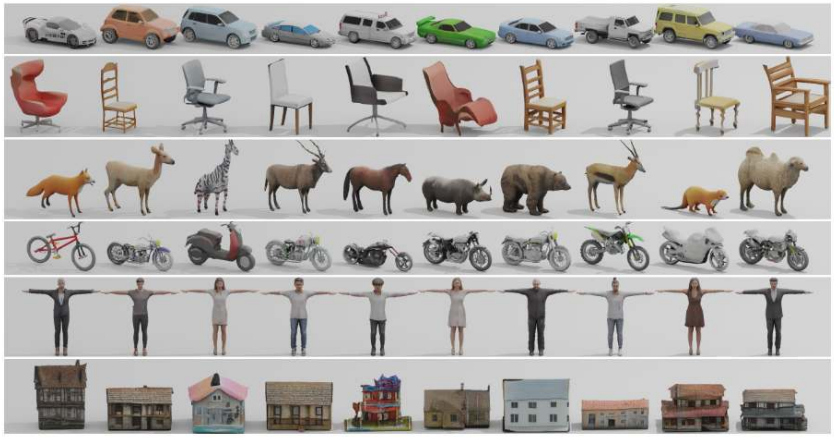

Abbildung 14: Resultate einzelner GET3D Modelle, jede Zeile entspricht einem trainierten Modell [18].

Das initiale GAN ist allerdings noch Unconditional, also generiert lediglich Modelle aus der Domäne der Trainingsdaten. Durch einen zweiten Trainingsschritt bzw. Finetuning mit CLIP, können die Generatoren entsprechend Text Conditional werden, allerdings natürlich nur innerhalb der Klasse auf die das GAN bisher trainiert wurde [18]. Aus einem GAN welches Fahrzeuge generiert kann somit ein GAN werden

welches “verbrannte Autos” generiert, nicht aber plötzlich Topfpflanzen oder andere Klassen.

Ein weiterer interessanter Punkt ist, dass GET3D einfach auf weitere Texturparameter erweiterbar ist, so können neben der Base Color zusätzlich Metallic und Roughness Texture Maps generiert werden, welche nötig sind für Disney BRDF Shader [18]. In Abb. 15 hierzu Resultate von GET3D, gerendert mit einem Disney BRDF.

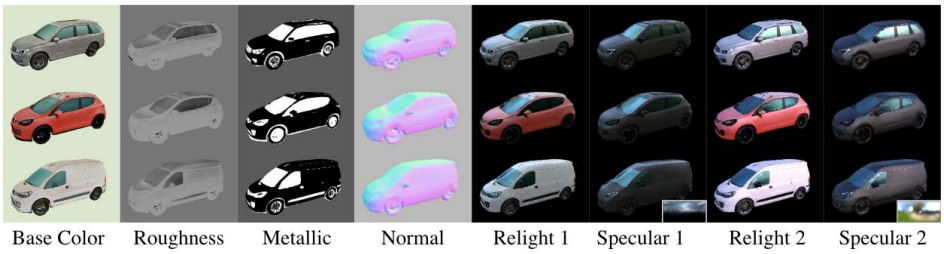

Abbildung 15: Gelernte Disney BRDF Texturparameter gefolgt von zwei Beleuchtungsszenen [18].

4.2 Dream Fusion

Dream Fusion ist die Weiterentwicklung der Dream Fields, veröffentlicht von Poole et al. in 2022 [19]. Statt CLIP wird nun ein Text-To-Image Modell als Kritiker verwendet, genauer Google’s Imagen [10].

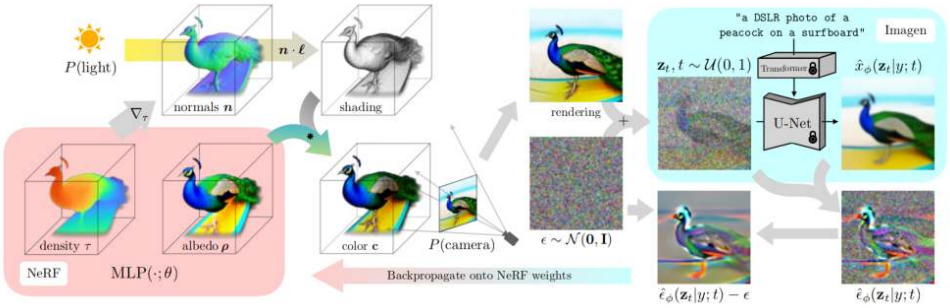

Abbildung 16: Konzept von Dream Fusion [19].

In Abb. 16 ist das Grundprinzip hinter Dream Fusion abgebildet: Es wird wie auch zuvor bei Dream Fields ein NeRF optimiert indem daraus Bilder aus unterschiedlichen Blickwinkeln differenzierbar gerendert werden, diesmal zusätzlich auch mit verschiedenen Lichtsetzungen. Anders als bei Dream Fields, wird das Resultat allerdings nicht mittels CLIP verglichen, sondern mit Rauschen addiert und als Input für ein vortrainiertes Diffusion Modell verwendet, hier Imagen. Das Diffusion Modell sagt dann das hinzugefügte Rauschen vor. Durch die Differenz mit dem tatsächlich hinzugefügten Rauschen kann man dann einen Loss ableiten, der verwendet werden kann, um das NeRF zu optimieren. [19]

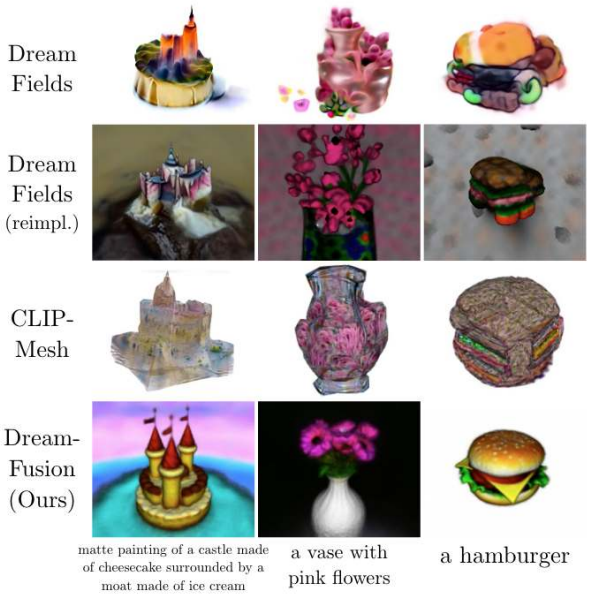

Abbildung 17: Vergleich von Dream Fields, CLIP-Mesh und Dream Fusion [19].

Die Resultate von Dream Fusion sind in Abb. 17 gegenübergestellt mit Dream Fields und CLIP-Mesh. Die Qualität ist wesentlich besser geworden bei einer Trainingsdauer von lediglich 90min. [19]

Da Imagen nicht öffentlich zugänglich ist, gibt es die eine Open Source Variante unter dem Namen Stable Dream Fusion, welche das bekannte Text-To-Image Modell Stable Diffusion [1] verwendet anstelle von Imagen. [20]

4.3 Magic3D

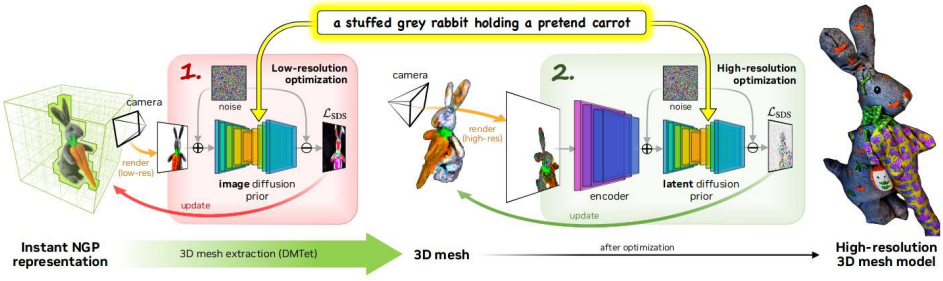

Abbildung 18: Aufbau von Magic3D [21].

Magic3D wurde Ende 2022 von Lin et al. bei NVIDIA entwickelt und kombiniert viele der vorangegangenen Konzepte indem es erst ein grobes NeRF optimiert und danach ein hochaufgelöstes Mesh [21].

Abbildung 18 zeigt den Aufbau im Detail: In der ersten Trainingsphase wird eine verbesserte Variante der NeRFs verwendet, die sog. Instant NGP. Diese werden, wie auch bei Dream Fusion, durch ein Text-To-Image Diffusion Modell optimiert und schließlich wird mittels DMTet ein Mesh extrahiert. In der zweiten Phase wird wieder ein Text-To-Image Diffusion Modell verwendet, um Renderings verschiedener Blickrichtungen zu optimieren und so das Mesh zu verbessern. Im Unterschied zur ersten Phase wird hier allerdings nicht das Bild direkt in das Diffusion Modell gegeben, sondern erst encoded und in einen Latent Space gebracht. [21]

Die Resultate von Magic3D heben sich besonders durch den Zugewinn an Details von Dream Fusion ab (siehe Abb. 19). Außerdem benötigt Magic3D nur noch 40min für das Training, knapp die Hälfte der Zeit von Dream Fusion. [21]

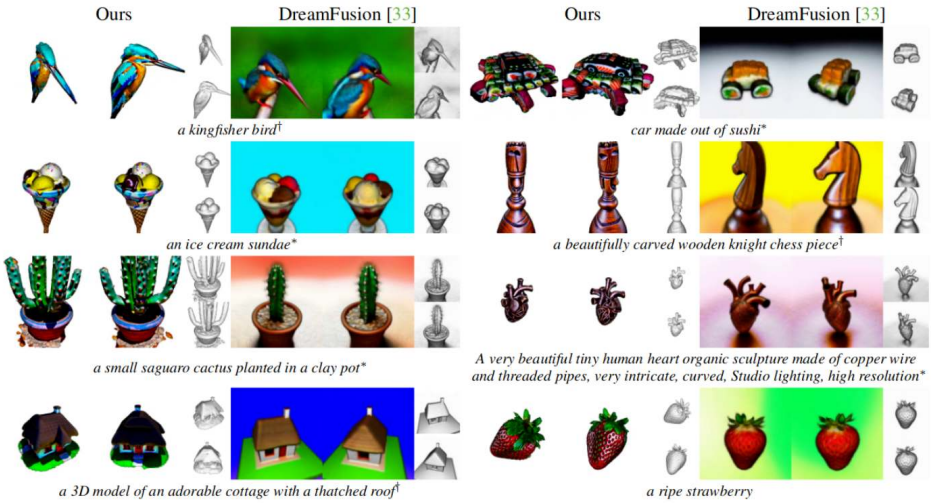

Abbildung 19: Resultate von Magic3D im Vergleich zu Dream Fusion [21].

5 Limitationen

Trotz der aktuellen Entwicklungen ist das Text-To-3D Feld noch nicht verlässlich einsetzbar in einer Produktionsumgebung. Der größte Nachteil aktueller Modelle ist die lange Inferenzzeit bis zum fertigen 3D Modell. 3D Supervised Modelle leiden weiterhin unter dem Mangel an hochwertigen und größeren 3D Datensätzen sowie dem erhöhten Rechenaufwand. Daher haben sich in vergangener Zeit viele Arbeiten mit 2D Supervised Text-To-3D Ansätzen beschäftigt, denn hier gibt es schlichtweg genügend Daten und mit CLIP sowie vortrainierten Text-To-Image auch sehr gute Optionen zur Evaluierung der generierten Modelle (bzw. Renderings dieser). Ein Drawback hingegen von 2D Supervision sind blickwinkelabhängige Artefakte wie das Multi-Face oder auch Janus Face Problem. Bei diesem weisen generierte Objekte aus verschiedenen Blickrichtungen dieselben Features auf und können dadurch

beispielsweise mehrere Gesichter haben, siehe Abb. 20.

Auch die zugrunde liegende Architektur hat enorme Auswirkungen auf die Flexibilität der Text-To-3D Modelle. GANs sind beispielsweise schwierig zu trainieren aber schnell in der Inferenz, wodurch schnell viele Varianten eines Modells erzeugt werden können. Jedoch sind GANs häufig beschränkt auf die im Training gesehen Daten und Domänen. NeRFs leiden ebenfalls unter langen Optimierungszeiten, jedoch mit fallender Tendenz aufgrund aktueller Forschungen wie Instant NGP.

Abbildung 20: Beispiel des Janus Face Problems bei dem Prompt “A DSLR photo of a squirrel” aus der Stable Dream Fusion Galerie [20].

6 Fazit

Text-To-3D hat im letzten Jahr einen deutlichen Schritt nach vorne gemacht und kommt immer näher an Inferenzgeschwindigkeiten die wir von Text-To-Image Modellen gewohnt sind, auch die Qualität hat sich wesentlich verbessert.

Anwendungen lassen sich aber auch bereits jetzt finden, so könnten Modelle wie GET3D dazu verwendet werden, schnell verschiedenste Variationen von Hintergrundassets wie Häusern zu generieren, ohne dass es eines aufwändigen prozeduralen Workflows bedarf. CLIP-Mesh könnte bereits verwendet werden um bestehende (grobe) Objekte zu stilisieren oder zu bearbeiten, indem einfach ein anderes Basis Mesh verwendet wird. Und Magic3D soll demnächst in NVIDIAs Omniverse integriert werden, was es zum ersten Modell machen würde das im großen Stil verwendet werden könnte.

Quellen

[1] „CompVis/Stable-Diffusion“. https://github.com/CompVis/stable-diffusion.

[2] „DALL·E 2 Research Preview Update“, *OpenAI*. https://openai.com/blog/dall-e-2-update/, Mai 2022.

[3] U. Singer *u. a.*, „Make-A-Video: Text-to-Video Generation without Text-Video Data“. arXiv, September 2022. doi: 10.48550/arXiv.2209.14792.

[4] Z. Shi, S. Peng, Y. Xu, Y. Liao, und Y. Shen, „Deep Generative Models on 3D Representations: A Survey“. arXiv, Dezember 2022. doi: 10.48550/arXiv.2210.15663.

[5] C. Schuhmann *u. a.*, „LAION-5B: An Open Large-Scale Dataset for Training next Generation Image-Text Models“. arXiv, Oktober 2022. doi: 10.48550/arXiv.2210.08402.

[6] A. X. Chang *u. a.*, „ShapeNet: An Information-Rich 3D Model Repository“, Stanford University — Princeton University — Toyota Technological Institute at Chicago, arXiv:1512.03012 [cs.GR], 2015.

[7] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, und R. Ng, „NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis“. arXiv, August 2020. doi: 10.48550/arXiv.2003.08934.

[8] E. R. Chan *u. a.*, „Efficient Geometry-aware 3D Generative Adversarial Networks“. arXiv, April 2022. doi: 10.48550/arXiv.2112.07945.

[9] A. Radford *u. a.*, „Learning Transferable Visual Models From Natural Language Supervision“. arXiv, Februar 2021. Zugegriffen: 7. Juni 2022. [Online]. Verfügbar unter: https://arxiv.org/abs/2103.00020

[10] C. Saharia *u. a.*, „Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding“. arXiv, Mai 2022. doi: 10.48550/arXiv.2205.11487.

[11] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, und B. Ommer, „High-Resolution Image Synthesis with Latent Diffusion Models“. arXiv, April 2022. Zugegriffen: 7. Juni 2022. [Online]. Verfügbar unter: https://arxiv.org/abs/2112.10752

[12] J. Ho, A. Jain, und P. Abbeel, „Denoising Diffusion Probabilistic Models“. arXiv, Dezember 2020. Zugegriffen: 7. Juni 2022. [Online]. Verfügbar unter: https://arxiv.org/abs/2006.11239

[13] S. Dieleman, „Guidance: A Cheat Code for Diffusion Models“. 2022.

[14] K. Chen, C. B. Choy, M. Savva, A. X. Chang, T. Funkhouser, und S. Savarese, „Text2Shape: Generating Shapes from Natural Language by Learning Joint Embeddings“. arXiv, März 2018. Zugegriffen: 1. Dezember 2022. [Online]. Verfügbar unter: https://arxiv.org/abs/arXiv:1803.08495

[15] A. Sanghi *u. a.*, „CLIP-Forge: Towards Zero-Shot Text-to-Shape Generation“. arXiv, April 2022. Zugegriffen: 21. November 2022. [Online]. Verfügbar unter: https://arxiv.org/abs/arXiv:2110.02624

[16] A. Jain, B. Mildenhall, J. T. Barron, P. Abbeel, und B. Poole, „Zero-Shot Text-Guided Object Generation with Dream Fields“. arXiv, Mai 2022. Zugegriffen: 9. November 2022. [Online]. Verfügbar unter: https://arxiv.org/abs/arXiv:2112.01455

[17] N. M. Khalid, T. Xie, E. Belilovsky, und T. Popa, „CLIP-Mesh: Generating Textured Meshes from Text Using Pretrained Image-Text Models“. September 2022. doi: 10.1145/3550469.3555392.

[18] J. Gao *u. a.*, „GET3D: A Generative Model of High Quality 3D Textured Shapes Learned from Images“. arXiv, September 2022. doi: 10.48550/arXiv.2209.11163.

[19] B. Poole, A. Jain, J. T. Barron, und B. Mildenhall, „DreamFusion: Text-to-3D Using 2D Diffusion“. arXiv, September 2022. Zugegriffen: 9. November 2022. [Online]. Verfügbar unter: https://arxiv.org/abs/arXiv:2209.14988

[20] J. Tang, „Stable-Dreamfusion: Text-to-3D with Stable-Diffusion“. 2022.

[21] C.-H. Lin *u. a.*, „Magic3D: High-Resolution Text-to-3D Content Creation“. arXiv, November 2022. doi: 10.48550/arXiv.2211.10440.