



Universität
Bremen

Fachbereich 3: Mathematik und Informatik

Produktion einer Szene mit einem per Roboter Stop-Motion animierten Boot in einem computergraphischen Ozean

Bachelor-Thesis zur Erlangung des akademischen Grades Bachelor of
Science (B.Sc.) im Studienfach Digitale Medien

Name: Jonas Gelzer

Matrikelnummer: 6279436

Abgabe Datum: 08.09.2026

Erstprüfer: Prof. Dr. Ing. Udo Frese

Zweitprüfer: Dr. René Weller

Betreuer: Prof. Dr. Ing. Udo Frese

Zusammenfassung

Die vorliegende Bachelorarbeit beschäftigt sich mit der Produktion einer Stop-Motion animierten Boots-Szene und dessen Integration in einen digital simulierten Ozean. Ziel der Arbeit war es, die Bewegung eines digital simulierten Bootes zu erzeugen und anschließend auf ein physisches Modellboot mithilfe eines Roboterarms zu übertragen. Hierfür wurde zunächst ein physisches Modellboot digitalisiert und in einer virtuellen Umgebung simuliert, bevor die entstandene Bewegung wieder auf das reale Modellboot übertragen wurde.

Die daraus resultierende Bewegung des Modellbootes wurde mit einem Stop-Motion animierten Hasen ergänzt. Durch die Herstellung einer identischen räumlichen Relation zwischen der Kamera und dem Boot im digitalen und physischen Raum, konnten die Aufnahmen beider Szenen aufgrund übereinstimmender Proportionen überlagert werden. Eine Verschachtlung von digitalen und realen Bildelementen ermöglicht schließlich die finale Integration des Modellbootes in das digitale Meer.

Die erarbeitete Methodik ermöglicht das Erweitern von traditionellen Stop-Motion Techniken mit computergrafischen Elementen. Hierdurch können Szenen gedreht werden, welche durch ausschließlich physische Materialien nur eingeschränkt umzusetzen wären. Durch eine kritische Reflexion der Methodik werden ihre Einschränkungen und Verbesserungsmöglichkeiten deutlich.

Abstract

This bachelor thesis focuses on the production of a stop-motion animated boat scene and its integration into a digitally simulated ocean. The project aimed to generate the movement of a digitally simulated boat and transfer that motion to a physical model boat using a robotic arm. To achieve this, a physical model boat was first digitized and simulated in a virtual environment before the resulting motion was transferred back to the real model boat.

The resulting movement of the model boat was filmed with a stop-motion animated rabbit. By recreating an identical relationship between the camera and the boat in both digital and physical spaces, the footage from both scenes can be stacked due to matching proportions. Different elements of the digital and real image create a final integration of the model boat into the digital sea.

The developed method allows for the enhancement of traditional stop-motion techniques with computer-generated elements. This enables the filming of scenes that would be difficult to realize using exclusively physical materials. A critical reflection on the methodology highlights its limitations and potential areas for improvement.

Inhaltsverzeichnis

1	Einleitung	7
2	Theoretischer Rahmen	9
2.1	Blender	9
2.1.1	FLIP Fluids Addon	9
2.1.2	PinSolver Addon	11
2.1.3	Blenders interner Ocean Modifier	12
2.1.4	Übergang von Wasser Simulation zu Ozean: Geometry Nodes	12
2.2	Erstellung virtuelles Boot: Realityscan	13
2.3	Hintergrund der Bilder entfernen und Compositing: Comfyui	13
3	Stand der Technik	15
4	Das methodische Vorgehen	19
4.1	Erstellung des 3D-Modells	19
4.2	Szenen in Blender	19
4.2.1	Flacher Ozean	20
4.2.2	Stürmischer Ozean	21
4.3	Nachstellung der Szene per Roboterarm	23
4.3.1	Relation von Kamera und Boot	23
4.3.2	Export Position und Rotation vom Boot	25
4.3.3	Nachstellung der Position und Rotation des Modellbootes	25
4.3.4	Belichtung	25
4.4	Überlagerung der Szenen	26
5	Realisierung der Methodik	27
5.1	Erstellung 3D-Modell des Boots	27
5.2	Erstellung Ozean	28
5.2.1	Flacher Ozean	28
5.2.2	Stürmischer Ozean	29
5.3	Interaktion von Boot und Wasser	30
5.3.1	Wassersimulation	31
5.3.2	Abflachung zum Rand der Simulation	31
5.3.3	Einbau in den großen Ozean	32
5.4	Kamera in Blender Positionieren	33
5.5	Export Position und Rotation Boot	34

5.5.1	Erstellung der Script zur Steuerung des Roboterarms	35
5.6	Zusammenfügen der finalen Szene	36
5.6.1	Hintergrund freistellen	36
5.6.2	Wasser Verdeckung	37
5.6.3	Übereinanderlegen der Schichten	37
6	Durchführung des Drehs	40
6.1	Aufbau	40
6.1.1	Kamera	40
6.1.2	Roboterarm	41
6.1.3	Licht	42
6.1.4	Steuerung per Computer	43
6.2	Ablauf	44
6.3	Kritische Reflexion	48
7	Ergebnisse	50
8	Fazit	51
9	Literaturverzeichnis	52
10	Nutzung KI basierte Anwendungen	57

Abbildungsverzeichnis

1	Einbau eines realen Modellbootes in einen digitalen Ozean	8
2	PnP Algorithmus	11
3	Blenders Geometry Nodes	12
4	Kamerapositionierung zur Erstellung eines 3D-Modells	13
5	Mechanisches Meer für „Two Balloons“	15
6	Behind the Scenes von „Missing Link“	16
7	Ausschnitt aus „Missing Link“	16
8	Übergang Wassersimulation zu Ozeanoberfläche	20
9	Interaktion zwischen Boot und flacher Ozean	21
10	Projektion der Wasserpartikel	22
11	Position und Rotation vom Boot	22
12	Berechnung der Position des Bootes	24
13	Modellboot auf einem Stativ	27
14	3D-Modell des Modellbootes	27
15	Prinzip von Bump Maps	28
16	Kante zwischen Wassersimulation und großem Ozean	33
17	PinSolver Addon	33
18	PinSolver Addon: Ergebnis	34
19	Ozean ohne digitales Boot	37
20	Überlegen von freigestelltem Boot in den Ozean	38
21	Verdecken des Bootsrumpfs mit Wasser	38
22	Verdeckte Teile des Bootes über Wasser legen	38
23	Maske für den Blur-Effekt visualisiert	39
24	Ausschnitt aus dem finalen Film	39
25	Ausschnitt aus dem finalen Film	39
26	Szenenaufbau	40
27	Kamera im Szenenaufbau	41
28	Befestigungsstück für das Boot am Roboter	42
29	Franka Panda Roboterarm	42

30	Modellboot auf dem Roboterarm	42
31	Computer beim Dreh	44
32	Vordere Wasserschicht	49

1 Einleitung

Im Bereich der Animation und Filmerstellung wird durch fortschreitende Technik die visuelle Qualität von Effekten stetig verbessert. Somit können grafisch komplexe Animationen und Simulationen effizienter erstellt werden, als es bei vorherigen Filmproduktion möglich war [1, 2, 3]. Jedoch beruhen einige Filmtechniken auf manuelle physische Erstellung von solchen Effekten, wie zum Beispiel bei Stop-Motion.

Unter Stop-Motion versteht man die Ende des 19. Jahrhunderts entstandene Technik, bei welcher die Illusion von Bewegung zwischen einzelnen Bildern erzeugt wird [4]. Hierbei werden starre Objekte, welche sich nicht eigenständig bewegen können, minimal zwischen einzelnen Bildern verändert. Durch das schnelle Abspielen dieser Bilder nimmt das menschliche Gehirn eine Bewegung dieses Objektes wahr. Stop-Motion steht dabei für „stopping motion“ und bezieht sich auf eine unflüssige und ruckelartige Bewegung, die durch das aneinander Schneiden von einzelnen Bildern entsteht.

Diese Technik ähnelt sich mit modernen Filmtechniken, welche auch nur aufeinander folgende Bilder sind. Traditionelle Stop-Motion basiert jedoch, anders als moderne Filme, auf händische Veränderung zwischen den einzelnen Bildern. Auch wird in der Filmproduktion nicht jedes Bild individuell verändert, sondern eine flüssige Bewegung aufgezeichnet. So vergeht gegenteilig während des Drehs von zwei aufeinander folgenden Stop-Motion Bildern oft viel Zeit, wodurch reale Bewegungen schwierig auf diesen einzufangen sind. So stellt die Darstellung von Wasser eine besondere Herausforderung dar, da die Bewegung einer Flüssigkeit schwer kontrolliert zwischen zwei Bildern aufzunehmen ist. Daher wird Wasser häufig durch physische Materialien oder veränderbare Formen nachgestellt, welche zwischen den einzelnen Fotos austauschbar sind [5, 6]. Diese sonst so flüssige Bewegung von Wasser kann in einem Stop-Motion Film nur schwierig durch reale Flüssigkeit abgebildet werden, wodurch hier Alternativen benötigt werden.

Ausweichend jedoch könnte man die traditionelle Weise der Stop-Motion Technik mit den fortschreitenden digitalen Möglichkeiten zur Animation und Simulation von unter anderem Flüssigkeiten verbinden. Gerade Szenen, welche eine große Menge Wasser erfordern, beispielsweise bei der Nachbildung eines Ozeans, sind schwierig durch Nachbauten zu erzeugen. Bei solch einer komplexen Aufgabe kann sich hier eine Kombination mit computergrafischer Simulation anbieten. Hierbei liegt die Schwierigkeit auf einer Verbindung der beiden Techniken, sodass sie von der Bewegung der einzelnen Elemente zueinander passen.

Daraus ergibt sich folgende Forschungsfrage: Wie kann eine digital simulierte Bewegung eines Bootes auf ein reales Modellboot übertragen und dieses nach einer Stop-Motion Animation wieder zurück in den digitalen Ozean eingebaut werden?

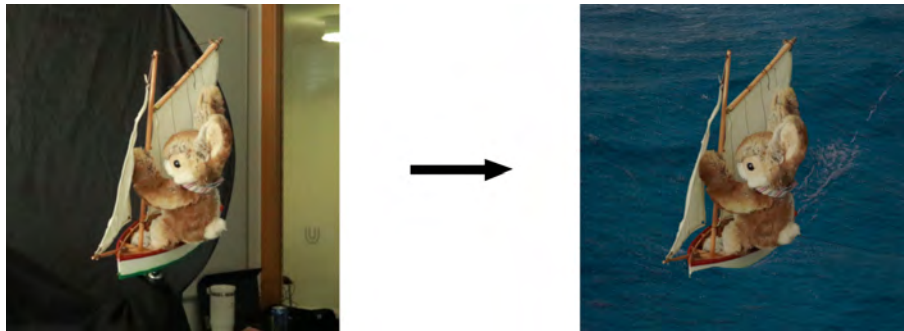


Abbildung 1: Einbau eines realen Modellbootes in einen digitalen Ozean. Links Foto vom Dreh, rechts Ausschnitt aus finalem Film in 7. *Ergebnisse* (Eigene Darstellung).

Hierfür soll die Bewegung eines Bootes vorher digital simuliert und folglich auf ein reales Modellboot mit einem Roboterarm übertragen werden. Diese physisch reproduzierte Bewegung des Bootes wird mit einem Stop-Motion animiertem Charakter auf dem Boot kombiniert. Anschließend wird dieser produzierte Film Bild für Bild in den digitalen Ozean, in welchem die Bewegung des Bootes simuliert wurde, rückwirkend eingebaut.

Ziel dieser Arbeit ist die Konzeption und Umsetzung einer Methodik, welche ein reales Stop-Motion animiertes Boot in eine digitale Ozean Umgebung einbaut. Darüber hinaus wird sich mit der Darstellung eines unendlichen Ozeans beschäftigt. Dieser sollte jedoch trotzdem eine Interaktion zwischen Boot und dem Wasser simulieren können.

Zur Erarbeitung der praktischen Vorgehensweise findet im zweiten Kapitel ein Einstieg in den Theoretischen Rahmen statt, welche die verwendeten Anwendung Erläutert. Folglich wird ein Einblick in den aktuellen Stand der Technik und verwandte Methoden zur Erstellung ähnlicher Stop-Motion Filme vorgestellt. Hier drauf basierend wird eine eigene Methodik herausgearbeitet und an zwei eigenständig produzierten Szene getestet. Abschließend wird der Erfolg der Methodik bewertet und die Bachelorarbeit wird mit einem zusammenfassenden Fazit beendet.

Beitrag dieser Bachelorarbeit:

- Berechnung der Kameraposition in dem Roboter-Koordinatensystem.
- Separate Wassersimulation mit nahtlosem Übergang in einen unendlichen Ozean.
- Export der Position eines simulierten Bootes und die psychisch Nachstellung dieser per Roboterarm.
- Einbau eines real Stop-Motion animierten Bootes in ein digitalen Ozean.

2 Theoretischer Rahmen

Um im Verlauf dieser Bachelorarbeit den beispielhaften Prozess der Produktion eines Stop-Motion animierten Bootes in einem computergrafischen Ozean zu erklären, bedarf es der Nennung und Definierung der benutzten Anwendungen. Vereinzelt basieren die Anwendung auf tiefgründigeren Algorithmen, welche jedoch im Rahmen der Bachelorarbeit nur oberflächlich angeschnitten werden.

2.1 Blender

Blender ist eine kostenfreie Open Source Software, welche durch eine große Anzahl an Funktionen die Erstellung von verschiedensten 3D-Inhalten ermöglicht [7]. Dabei bietet Blender Funktionen wie Modellieren, Animieren bis hin zum Rendern oder auch der Nachbearbeitung von Bildern und Videos an.

Über Blenders eingebauten internen Funktionen hinaus lassen sich auch von anderen Nutzern erstellte Erweiterungen hinzufügen [8]. Diese privat angeboten und teils kostenpflichtigen „Addons“ erweitern oder verändern die Umgebung von Blender über ihre Standardfunktionen hinaus. Das können rein visuelle Veränderung von Blenders Interface sein, aber auch komplett neue Möglichkeiten der Modellierung oder Simulation.

Des weiteren gibt es in Blender die Möglichkeit Python Scripte intern auszuführen [9], welche verschiedenste Daten verändern und auslesen können. Hierdurch können zum Beispiel Objekte automatisch generiert oder bearbeitet werden, Positions- und Lokationsdaten ausgelesen, oder Daten in externen Dokumenten gespeichert werden. Der Export von Daten ermöglicht plattformübergreifendes Verarbeiten und Nutzen dieser.

2.1.1 FLIP Fluids Addon

FLIP FLuids ist ein kostenpflichtiges Addon für Blender, welches von Ryan Guy entwickelt und 2016 veröffentlicht wurde [10]. Dieses Addon basiert, wie der Name verrät, auf dem Verfahren FLIP, also Fluid-implicit-particle, welches eine Verbesserung des ursprünglichen Verfahrens zur Simulation von Flüssigkeiten bietet.

Flüssigkeiten wurden herkömmlicherweise mit dem PIC, sprich dem Particle-in-cell Verfahren simuliert, welches jedoch seine Schwächen hat [11]. Hierbei wird der Bereich der zu simulierenden Flüssigkeit in ein Gitter unterteilt und war für die physikali-

schen Berechnung von Druck und Kräften verantwortlich. Einzelne Partikel haben eine Ausgangsgeschwindigkeit, welche dann bei jedem Simulationsschritt auf das Gitter übertragen wurden. Hier wurde dann eine neue Geschwindigkeit aus der Strömung und Dichte der Partikel berechnet. Diese berechnete Geschwindigkeit wurde dann wieder vollständig auf die Partikel übertragen, wodurch ihre Anfangsgeschwindigkeit vollständig mit einer neuen berechneten Geschwindigkeit ersetzt wurde. Da bei jeder Berechnung der Dichte des Gitters die Geschwindigkeit interpoliert wurden, ging bei jedem Schritt ein kleiner Teil der Bewegungsenergie verloren. Das lag daran, dass die Partikel aus einem Teil des Gitters auf die durchschnittliche Geschwindigkeit von allen Partikel des Abteils gesetzt wurden und somit schnelle Bewegungen direkt durch interpolierte Geschwindigkeiten ersetzt wurden.

"the accuracy of PIC is compromised by a large computational diffusion caused by transferring the momentum from the grid to the particles and back in modeling convection through the grid." [11, S. 315-316]

PIC erzielt somit nach Brackbill und Ruppel [11] keine besonders realistische Flüssigkeitssimulation, da die Geschwindigkeit der Partikel auf einen durchschnittlichen Wert in jedem Schritt gesetzt werden. Folglich verschieden winzige Strömungen, Spritzwasser oder andere sich in der Geschwindigkeit abhebende Details der Flüssigkeit, was den Realismus begrenzt. Um diese kleinen Details der Simulation beizubehalten benötigt es ein realistischeres Verfahren zur Berechnung der Partikelgeschwindigkeit.

Das Fluid-implicit-particle, kurz FLIP Verfahren, ist der Nachgänger des PIC Verfahrens und basiert somit auf diesem [11]. Hier werden jedoch die Anfangsgeschwindigkeiten der einzelnen Partikel nicht auf einen interpolierten Wert gesetzt, sondern in ihrer Geschwindigkeit lediglich angepasst. Diese Änderung der Geschwindigkeit wird durch die Dichte der Partikel in einem Gitter berechnet und dann auf die ursprüngliche Geschwindigkeit hinzugefügt. Wenn Ein Part des Gitters sehr viele Partikel besitzt, dann gibt es auch eine hohe Geschwindigkeit. Diese wird aber nicht wie bei dem PIC Verfahren auf alle Partikel gleich angewendet, sondern jeder Partikel behält seine eigentliche Geschwindigkeit und bekommt eine zusätzliche Geschwindigkeit dazu. Infolgedessen bleibt die Bewegung der Partikel konstant und wird nicht vollständig ersetzt, wodurch kleine Realismus ausmachende Details erhalten bleiben.

Obwohl sowohl Blenders internes Mantaflow zur Berechnung von Flüssigkeiten, als auch FLIP Fluid auf dem FLIP Verfahren basieren, unterscheiden sie sich hinsichtlich ihrer Funktionen [12]. Mantaflow ermöglicht die Simulierung von allen Flüssigkeiten wie unter anderem Rauch und anderen gasförmigen Substanzen, wobei FLIP Fluids rein auf die Berechnung von Flüssigkeiten fokussiert ist. Hierdurch ermöglicht dieses Addon mehr gestalterische Kontrolle über Schaumpartikel, Oberflächenspan-

nung, Spritzer, Viskosität und ähnlichem, was dem Nutzer mehr Freiheit in der Gestaltung ermöglicht [13].

2.1.2 PinSolver Addon

PinSolver ist ein weiteres Addon für Blender und wurde im Jahre 2026 von „K_Naoki“ veröffentlicht [14]. Es ermöglicht die automatische Positionierung einer Kamera in einem virtuellen Raum. Dafür wird ein Bild eines realen Objektes, sowie ein exaktes 3D-Modell von diesem benötigt. Der Nutzer kann nun einzelne Merkmale des Objektes in sowohl dem Foto, als auch in dem 3D-Modell an der gleichen Stelle markieren. Das Addon errechnet folglich anhand dieser Punkte die genau Position, von welcher das Foto aufgenommen wurde und kann somit die reale Relation von Kamera und Objekt in Blender wiedergeben. Hierfür müssen jedoch auch Brennweite und Fokus von der Kamera in dem virtuellen Raum mit der Kamera des echten Bildes übereinstimmen.

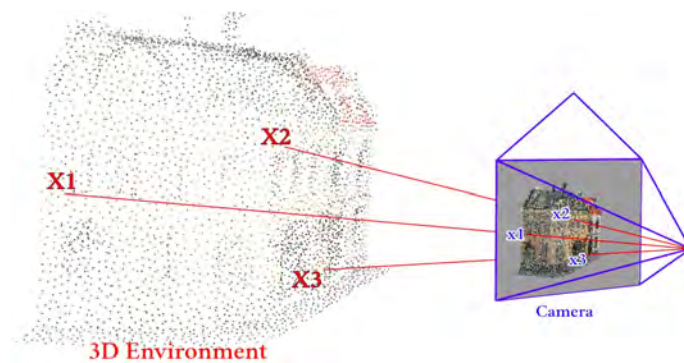


Abbildung 2: Pnp Algorithmus [15].

Beim Perspektiv-n-Point Problem geht es um das Bestimmen der Position und Orientierung einer Kamera anhand korrelierender Punkte eines 3D-Objektes und sein dazugehörigen 2D-Punkte aus einem Kamerabild [16]. Das Addon greift über OpenCV als Schnittstelle auf verschiedene Implementierung dieses Problems zu [14, 17], welche anhand einzelner ausgewählter Punkte die ursprüngliche Position einer Kamera zurück berechnen können. PinSolver verwendet primär die „SOLVEPNP_ITERATIVE“ Implementierung [18], wobei hier anhand der initialen Kamerapose die ausgewählten 3D-Punkte auf die Bildebene projiziert werden. Diese werden mit den korrelierenden 2D-Punkten verglichen und aus dieser Abweichung wird ein Projektionsfehler berechnet. Folglich wird die Kamera iterative umpositioniert, um den berechneten Fehler in jedem Schritt zu minimieren. Hierdurch landet die Kamera am Ende an einer Position im Raum, von welcher aus das ursprüngliche Bild von dem Objekt gemacht wurde [17, 16].

Abbildung 2 zeigt eine erfolgreiche Berechnung der ursprünglichen Kameraposition,

bei welcher die Punkte des 3D-Objektes mit den passenden Punkten des 2D-Bildes übereinstimmen. Die Kamera steht folglich in gleicher Relation zu dem digitalen Objekt, wie sie auch zu dem realen Objekt stand.

2.1.3 Blenders interner Ocean Modifier

Blender bietet verschiedene bereits eingebaute Modifikatoren an, welche die Topologie und Form von Objekten je nach Bedürfnis verändern können. So gibt es Modifier, welche Objekte spiegeln, abrunden, drehen, ausfüllen, oder ihnen eine neue Oberfläche generieren [19]. Einer dieser Modifikatoren ist der *Ocean Modifier*, welcher eine großflächige Ozean Oberfläche generieren kann [20].

Dieser Blender-interne Modifikation generiert eine 2D-Textur Map, welche als Displacement auf ein Objekt angewendet wird und folglich diese so in der Höhe verformt, dass eine Wellenartige Oberfläche entsteht [21]. Die Wellen können dabei in Größe, Dichte, Geschwindigkeit und Tiefe angepasst werden und da es sich nur um eine Verformung eines Objektes handelt, bietet diese keine realistische Flüssigkeitsinteraktion mit andern Objekten. Dafür ist diese jedoch stark anpassbar und kann somit große hochqualitative Flächen von Wasser ohne Simulation und somit ohne viel benötigte Rechenleistung erstellen.

2.1.4 Übergang von Wasser Simulation zu Ozean: Geometry Nodes

„Geometry Nodes“ sind ein weitere in Blender integrierte Funktion, welche wie der Name sagt, auf Nodes basiert [22]. Diese Nodes sind elementare Bausteine, welche grafische miteinander verknüpft werden können, um ähnlich wie Modifier Veränderung an der Topologie eines Objektes durchzuführen. Hierbei sind die Nodes flexibel programmierbar und ermöglichen das einfache Erstellen von komplexen Bearbeitung. Diese einzelnen Verarbeitungseinheiten ergeben einen Datenfluss, welcher in jedem einzelnen graphischen Baustein nach belieben angepasst werden kann. Hierdurch liefern die Nodes ein direktes visuelles Feedback auf einzelne Veränderungen.

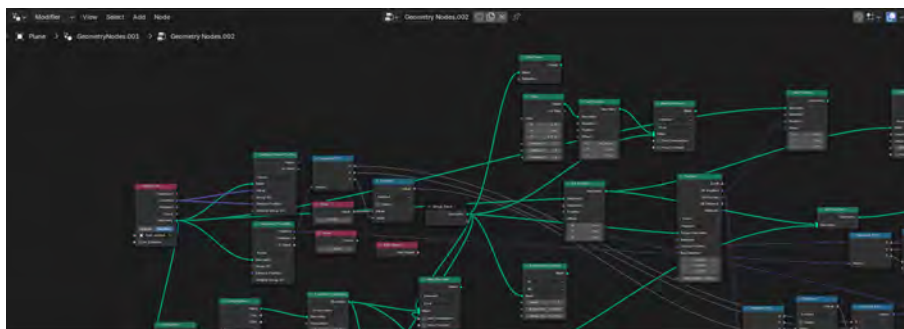


Abbildung 3: Geometry Nodes Oberfläche in Blender (Eigene Darstellung).

Wie in Abbildung 3 zu erkennen, lässt sich eine große Anzahl an verschiedener Nodes zu einem größeren Baum zusammensetzen. Dieser ermöglicht komplexe Bearbeitung und Veränderung von Objekten, anhand von mathematischen Berechnungen, neu Positionierung oder diversen Verformungen. Dies gestattet das Erstellen von Formen, welche über die normalen Funktionen von Blender herausragen.

2.2 Erstellung virtuelles Boot: Realityscan

Realityscan, ehemals RealityCapture, ist ein Software Programm, welches die Erstellung von 3D-Modellen anhand einer Reihe von Bildern ermöglicht [23]. Dabei können sowohl einzelne Objekte, also auch ganze Gebäude oder Landschaften in ein digitales Modell umgewandelt werden. Hierfür benötigt es Bilder verschiedener Perspektiven und Winkeln. Diese ermöglichen es die genaue Position, von welcher das Bild ursprünglich aufgenommen wurde zu bestimmen. Das Programm überlagert durch das Erkennen von Licht-Reflexion die Bilder digital, sodass aus den 2D-Bildern ein 3D-Objekt entsteht. Ein Beispiel für solch eine Generierung eines 3D-Objektes kann in Abbildung 4 entnommen werden, welches dann in andere Programme und Anwendungen exportiert werden kann.

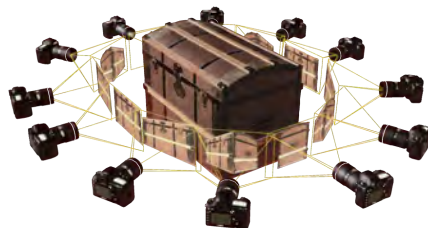


Abbildung 4: Kamerapositionierung zur Erstellung eines 3D-Modells [23].

2.3 Hintergrund der Bilder entfernen und Compositing: Comfyui

Comfyui ist ein kostenfreies Open Source Programm, welches eine grafische Oberfläche für die lokale Nutzung von Modellen der künstlichen Intelligenz (kurz KI-Modellen) bietet [24]. Ein weiteres Node-basiertes Stecksystem ermöglicht dem Nutzer das Hintereinanderschalten und Ausführen verschiedener Modelle. Diese Nodes bieten einen hohen Grad an Anpassungsfähigkeit, was bei jeder Veränderung wieder für eine direkte virtuelle Rückmeldung des Outputs sorgt.

Comfyui hat bereits einige integrierte Nodes, wobei sich aber auch von anderen Nutzern erstellte Bausteine hinzufügen lassen [25], welche die Möglichkeiten der Generierung weiter erhöhen. Open Source KI Modelle können hiermit lokal ausgeführt werden, sei es für die Generierung oder Bearbeitung von Bildern, bis zu Erstellung ganzer Videos oder Audiodateien. Zudem können bestimmte Nodes auch Anpassungen in der Größe, Form, oder Farbe durchführen, oder Bilder überlagern und verschachteln. Sogenannte erstellte „Workflows“ können aufgrund ihrer Größe komplexe Bildgenerierungs- und Bearbeitungsoperationen automatisiert durchführen und bieten gleichzeitig eine Nutzerfreundliche grafische Bedienungsfläche.

3 Stand der Technik

Stop-Motion ist eine Filmtechnik, welche Ende des 19. Jahrhunderts und somit ohne computergrafische Unterstützung ihren Lauf genommen hat [4]. Durch diese Alter wird bei einem klassischen Stop-Motion Dreh viele Elemente und Effekte durch physische Nachbauten dargestellt. Das geschickte Manipulieren realer Gegenstände kann verschiedene Bewegungen, oder zumindest eine Illusion von ihnen erzeugen.

Besonders Wasser ist aufgrund seiner schnellen Bewegung schwer kontrolliert zwischen zwei einzelnen Bildern einzufangen. Somit wird die Darstellung von Wasser in solchen Filmen oft durch physische Materialien oder Formen erzeugt.

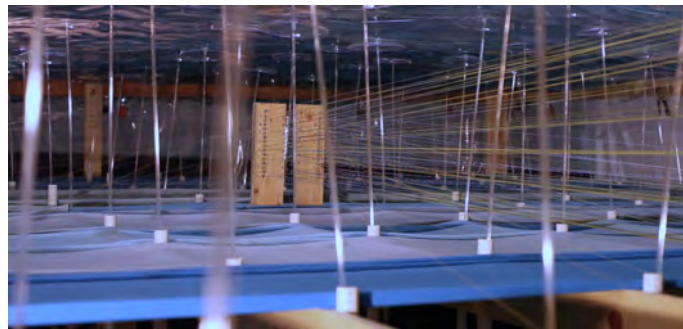


Abbildung 5: Mechanisches Meer für „Two Balloons“ [26].

Ein Beispiel für solche verwendeten Tricks ist der Kurzfilm „Two Balloons“, bei welchem der Ozean durch eine etwa sechs mal fünf Meter große Kunststoffplane nachgestellt wurde. Diese liegt auf einem komplexen Netzwerk aus Stäben und Fäden, welche durch einen zentralen Motor gesteuert werden. Hierbei werden die Stäbe an gewissen Stellen hoch gedrückt, was auf der Plane wellenartige Bewegung erzeugt. Die Aufnahmen von dem Meer wurden später im Computer kachelweise hintereinander gesetzt, um ein scheinbar unendlichen Ozean zu erstellen. Auch Bugwellen und Schaum von der Bewegung des Bootes wurden im Nachhinein mit der Hilfe des Computers hinzugefügt [26, 27].

Stop-Motion Szenen, welche ganz ohne computergrafische Nacharbeit gefilmt werden, stellen die Interaktion mit dem Wasser durch physische Materialien nach. Bei aufwendigeren Produktionen wird beispielsweise jeder einzelne Frame einer Flüssigkeitsbewegung durch eine 3D-gedruckte Form erstellt [6]. Alternativ wird auch die Illusion von bewegendem Wasser durch Reflexion von verschiedenen Glasstrukturen [28] erzeugt, oder Flüssigkeiten werden ganz nachgestellt. Hierfür werden Materialien wie Heißkleber, Watte, oder auch verschiedene Gele verwendet, welche Wasser mit

dementsprechendem Realismus Bild für Bild nachstellen [29, 30, 31].

Um das Nachbauen von komplexer Wasserbewegung zu umgehen, wird auch gerne die Kamera so positioniert, das die eigentliche Interaktion von einem Objekt mit Wasser nicht direkt im Bild zu sehen ist, oder gezielt verdeckt wird [32]. Hierdurch wird der Dreh von komplexe Szenen durch geschickte Kamerapositionierung umgangen.

Das Filmstudio „LAIKA“ auf der anderen Seite ist bekannt für hochqualitative Stop-Motion Filme [33], in welchen es auch komplexe Wasserszenen gibt. So wurde beispielsweise in dem Film „Kudo and the two String“ eine Boot-Ozean Szene durch computergrafische Hilfe erstellt. Hier wurde im voraus mit physischen Nachbauten von Wellen und einer Wasseroberfläche sich überlegt, wie die Szene und ihr Stil aussehen soll. Diese kreierte Textur und Stil wurde dann in dem digitalen Meer übernommen, wodurch dieser optisch dem physischen Nachbau sehr ähnlich sieht [34]. Das Boot, welches auf diesem Ozean fährt, wurde auf einem „6-axis hexapod model moving rig“ [35] platziert und somit bewegt. Dieses Rig kann sich auf sechs Achsen bewegen und somit das Boot oben drauf beliebig in alle Richtungen rotieren. Hierdurch wurde zuvor eine Bewegung des Rigs programmiert, wodurch das Boot Bild für Bild in Position und Rotation verändert werden kann. Diese Bewegung simuliert eine rükelartige und schwankende Fahrt auf dem in dem Film stürmischen Ozean [33]. Im Nachhinein wurde der erstellte Ozean hinter das Boot gelegt und bietet mit der übernommen physisch erstellten Wassertextur den gezielten „händischen“ Stil.



Abbildung 6: Ausschnitt aus Behind the Scenes Video von „Missing Link“ [6].



Abbildung 7: Ausschnitt aus dem Film „Missing Link“ [36].

Ebenfalls verwendet LAIKA's nächst erschienener Kinofilm „Missing Link“ ein sechs achsiges Rig für die Steuerung von einem Modellboot. Hierbei wurde anstatt eine Bewegung zu animieren diese sogar vorher digital simuliert [37]. Diese Wassersimulation liefert sowohl die Bewegung des Bootes, welche von dem Rig wiedergegeben wurde, als auch eine Grundlage für die spätere detailreichere Simulation [38]. Diese Bewegung des Bootes erlaubten ein späteres Einbauen in einen See, welcher detailreich simuliert wurde und es somit eine Interaktion von dem realen Modellboot mit

dem digitalen Ozean gibt. Durch LAIKA's Motion-Control-System ist die Position von Kamera und Boot in jedem Frame bekannt und dann somit im digitalen Rahmen nachgestellt werden. Somit kann eine digitale Kamera die exakte Bewegung wie die echt abfahren, wodurch das simulierte Wasser exakt zu dem echt aufgenommenen Modellboot passt. LAIKA verwendet in der Produktion ihrer Film eine große Anzahl von Motion Control Systemen und Robotern, welche das drehen von komplexen Szenen und Effekten ermöglichen [39, 40].

„The Pirates!“ ist ein weiterer Stop-Motion basierender Film, welcher eine Vielzahl an Szenen von einem Boot auf einem scheinbar unendlichen Ozean enthält. Auch hier wurde ein physisches Schiff auf einem Gimbal platziert, welche für die Bewegung des Schiffs zuständig war [41]. Eine Besonderheit hier ist der vorherige digitale Nachbau des Gimbals. Somit konnten die echten Limitierungen und Einschränkung der möglichen Bewegung in dem digitalen Raum direkt beachtet werden, bevor eine Animation des Bootes erstellt wurde. Diese Animation wurde dann auf das physische Gimbal übertragen, wodurch das Boot den geplanten Bewegung folgt [42]. Der Ozean wurde im Nachhinein digital hinzugefügt, wurde aber stilistisch so gestaltet, dass er wie ein mechanischer Ozean aussieht [42].

Ein weiterer Film, welcher zwar digitales Wasser verwendet, es aber wie mechanisch nachgebautes aussehen lässt ist „James and the Giant Peach“. Der Stil des Wassers hat drei Monate zum entwerfen benötigt [43] und wurde passend zu der Stop-Motion Welt angepasst. Ein digitale Wassersimulation bietet ein sehr realistisches Aussehen von Wasser, welches hier, wie aber auch bei vielen anderen Stop-Motion Filmen, bewusst durch ein anderen Stil ausgetauscht wurde [44].

Im Allgemeinen hat sich die Stop-Motion Produktion zu einer hohen Verwendung von Motion-Control-Systemen entwickelt. Zwischen 1993 und 2000 haben große Filmstudios angefangen Motion-Control-Systeme für ihre Kamerabewegung einzuführen, um ihnen mehr Freiheit beim Dreh zu ermöglichen [45]. Dies erlaubt beliebige Kamerafahrten, ohne eine Kamera zwischen einzelnen Bildern zielgenau per Hand verändern zu müssen. Wenn die Kamera zwischen zwei Bildern zu stark bewegt wird, erzeugt das Fehler in dem finalen Video, womit die Einführung von Motion-Control gesteuerten Kameras die Videoqualität erheblich erhöht hat.

Bereits in der maritimen Forschung werden Hexapods oder Gimbals für die physische Übertragung digital simulierter Schiffsbewegung verwendet [46]. Diese Roboterplattformen ermöglichen eine unabhängige Bewegung in sechs Freiheitsgraden. Hierdurch können Simulation in der realen Welt kontrolliert Reproduziert und somit genauer untersucht werden. Diese gezielte Nachstellung der Bewegung wird hier jedoch nicht für die Produktion von Stop-Motion Film verwendet, sondern durch die Pro-

jektion einer kontinuierlichen Bewegung. Diese kann mit Vorhersagen und anderweitigen Simulationen verglichen werden, um Aussagen zu numerischen Simulationen oder Modellen zu treffen [47]. Somit ist diese Methode für die Forschung geeignet, da sie bei einem echten Schiff kaum Kontrolle über alle Bedingung haben und sich hiermit einen kontrolliertes Umfeld für solche Bewegungen bauen.

Bei den Stop-Motion Produktionen jedoch wird dieser Bewegungsablauf nicht an einem Stück, sondern in einzelnen Frames wiedergegeben. Hierfür muss eine fließende Bewegung zuerst auf einzelne Frames heruntergerechnet werden, welche dann gezielt nachgestellt werden können.

4 Das methodische Vorgehen

Die folgende Methodik dient einer beispielhaften Animation eines Bootes auf einem Ozean. Dabei gibt es sowohl eine Szene von einem Boot auf einem flachen und stillen Ozean, also auch eine Szene mit einem stürmischen Ozean. Ziel dieser Methodik ist die Simulation zweier realistischer Szenen in einer computergrafischen Umgebung und die darauffolgende Integrierung eines realen Stop-Motion animierten Boots in diese. Hierfür werden berechnete Positionen eines digitalen Bootes mit einem Roboterarm wiedergegeben, um die virtuelle Szene physisch nachzustellen. Das Geschehen auf dem echten Modellboot wird per Stop-Motion Technik animiert und dann wieder in den computergrafischen Ozean eingebaut. Aufgrund einer exakten realen Wiedergabe der zuvor berechneten Bewegung, sollte die in echt gedrehte Szene identische Proportionen und Relationen haben, wie die Szene aus dem Computer. Dies erlaubt ein Überlagern der einzelnen Objekte der unterschiedlichen Szenen, um folglich das reale Modellboot in den digitalen Hintergrund einzubauen.

4.1 Erstellung des 3D-Modells

Zu Beginn benötigt es ein 3D-Modell eines Bootes in Blender, welches von den Maßen und Proportionen mit dem Modellboot für den realen Dreh übereinstimmt. Hierdurch kann eine digitale Wassersimulation durchgeführt werden, die zu der Form von dem Modellboot passt.

Hier bietet es sich an, das reale Modellboot zu digitalisieren um somit ein möglichst übereinstimmendes Modell zu erstellen. Auch muss das virtuelle Boot genau die gleichen Maßen in einem digitalen Raum haben wie das echte, hierfür ist es wichtig das Modellboot zu vermessen und das 3D-Modell auf genau diese Größe zu skalieren.

4.2 Szenen in Blender

Darauf folgt die Erstellung der Szenen in einer computergrafischen Umgebung. Ich habe mich aufgrund vergangener Erfahrung auf die Anwendung Blender festgelegt, die Methodik wird jedoch nicht auf diese Umgebung beschränkt. Beide Szenen spielen auf einem Ozean mit scheinbar unendlicher Größe, sodass nur großflächiges Wasser und kein festes Land zu sehen sind. Des Weiteren treibt das Boot, welches zuvor durch einscannen erstellt wurde, auf der Wasseroberfläche und passt sich jeweils dem Wellengang der Szene an.

4.2.1 Flacher Ozean

Bei der ersten Szene handelt es sich um einen flachen und stillen Ozean. Eine detailreiche und physikalisch korrekte Simulation einer große Fläche Wasser ist mit einem hohen Rechen- und Speicheraufwand verbunden, wodurch eine so großflächige Wassersimulation nicht sinnvoll und unpraktikabel ist [2]. Darüber hinaus interagiert das Boot nur mit einem kleinen Bereich des Ozeans, wodurch eine aufwendige Berechnung des restlichen Wassers überflüssig scheint [48]. Alternativ wird deshalb eine vereinfachte Darstellung der Wasseroberfläche verwendet, bei welcher es sich um eine einzelne flache Oberfläche handelt.

Diese erhält eine Material, welches die Illusion einer Wasseroberfläche durch unterschiedliche Reflexion des Lichtes erzeugt. Damit das digitale Boot nun mit diesem unechten Wasser interagiert benötigt es einen kleinen separaten Bereich, welcher diese Berechnung durchführt. Eine kleine Box um das Boot und dessen Fahrtweg herum simuliert die Verdrängung und Bewegung von Wasser anhand einer detaillierten physikalischen Simulation.

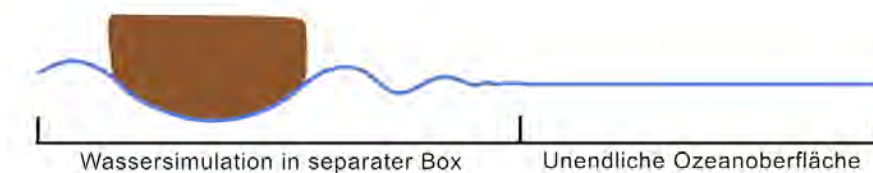


Abbildung 8: Übergang von Wassersimulation zu Ozeanoberfläche (Eigene Darstellung).

Die eigenständige Wassersimulation muss noch in die unechte Wasseroberfläche integriert werden und sollte nahtlos in diese übergehen. In Abbildung 8 befindet sich auf der linken Seite der kleinen Kasten in welchem die eigentliche Wassersimulation stattfindet. Damit dieser nahtlos in die glatte Fläche übergeht, muss er zum Rand der Box hin auf die gleiche Höhe der großflächigen Oberfläche abflachen. Hierdurch wird eine Interaktion von Boot und Meer geschaffen, ohne ein vollständigen Ozean simulieren zu müssen.

In dieser flachen Ozean Szene fährt das digitale Boot mit einer konstanten Geschwindigkeit auf einer geraden Linie entlang, ohne jegliche Kurven zu fahren. Auch Schaukelt das Boot nur minimal, um zu zeigen wie ruhig und windstill das Meer ist.

4.2.2 Stürmischer Ozean

Der stürmische Ozean soll, gegenteilig zu dem ruhigen Ozean, einen hohen und starken Wellengang haben. Dieser wird anhand von wellenartigen Verformungen der vorher glatten Oberfläche erzeugt. Diese Wellen sind jedoch keine Simulation von Wasser, sondern stellen nur die Oberfläche eines Ozeans nach, ohne diesen berechnet zu haben.

Die Interaktion von dem Boot mit dem Wasser muss somit wieder separat simuliert werden, was auch wieder nur in einer kleinen Box um das Boot herum geschieht. Das Boot sollte mit der verformten Wasseroberfläche in diesem kleinen Bereich interagieren, jedoch ist dies nicht mit den mir zur Verfügung stehenden Anwendung möglich. Die Bewegung der Wellen von außen lassen sich nicht als Wasseroberfläche in dieser Box setzen, wodurch Steigen und Sinken des Wasserstandes nicht in der Interaktion von Boot und Wasser berücksichtigte wird. Die Energie und Bewegung der Wellen können nicht auf die separate Wassersimulation Einfluss nehmen.

Deshalb wurde aufgrund beschriebener Limitierungen an dieser Stelle die Methodik angepasst und die Interaktion von dem Boot mit dem Ozean wird durch eine neue Methode vorgetäuscht. Somit findet keine physisch korrekte Simulation statt, jedoch ist diese Veränderung in dem finalen Film nicht wirklich unterscheidbar.

Nun wird das Boot unabhängig von dem Wellengang wieder auf einem flachen Ozean simuliert, so wie es in 4.2.1 *Flacher Ozean* gemacht wurde.

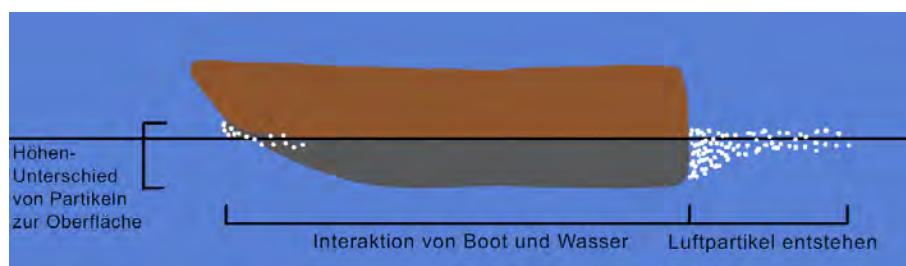


Abbildung 9: Interaktion zwischen Boot und flachen Ozean (Eigene Darstellung).

Hierbei bekommt jeder Partikel der Wassersimulation aufgrund von Verdrängung und Strömung eine neue Position für jeden einzelnen Frame. Von dieser Position aus lässt sich eine Abweichung der Höhe zu der Wasseroberfläche berechnen. Wird ein Partikel vom Boot verdrängt, liegt er unterhalb der Oberfläche, eine erzeugt Welle liegt dann oberhalb der Wasseroberfläche.

Diese Simulation wird nun, wie in Abbildung 10 dargestellt, von oben auf die große verformte Oberfläche projiziert. Dabei kollidiert jeder Partikel aus der Wassersimulation mit einer Stelle auf der Wellenoberfläche.

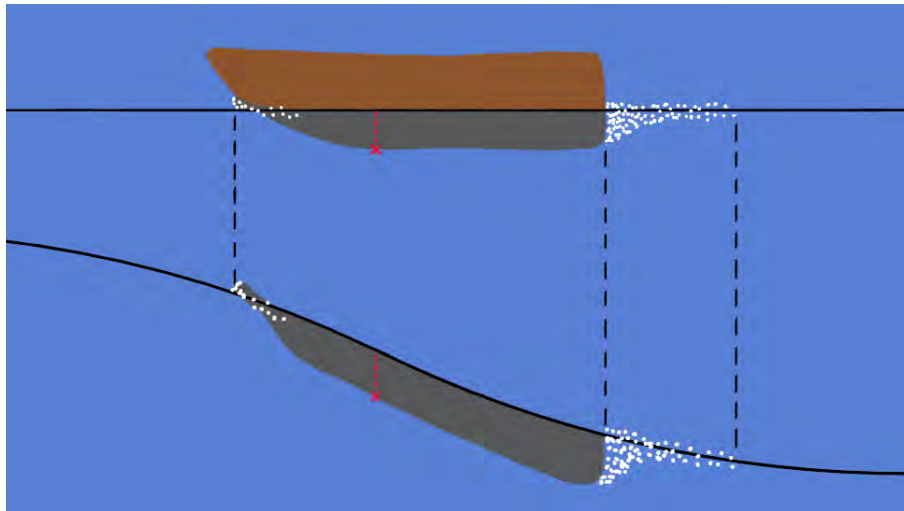


Abbildung 10: Prinzip der Projektion der Wasserpartikel (Eigene Darstellung).

Genau dieser Schnittpunkt jedes Partikels dient als neue Position der zuvor flachen Wassersimulation. Hier wird noch die zuvor berechnete Abweichung zur Wasseroberfläche berücksichtigt, wodurch wieder Partikel die zuvor oberhalb der Oberfläche lagen ein Stück nach oben versetzt werden. Da sich auch die von dem Boot zuvor verdrängten Partikel weiterhin unterhalb der Wasseroberfläche befinden, wird ein zusätzliche Aussparung in Form eines Bootes sichtbar.

Es entsteht somit eine Interaktion von dem Boot mit der Oberfläche des stürmischen Meeres, ohne das diese direkt simuliert wurde. Die Verdrängung von Wasser und die Bildung von Partikeln wird nur aus einer anderen Simulation dynamisch auf die Oberfläche gesetzt. Hierbei wird jedoch nur die eigentliche Verdrängung vom Boot dargestellt und nicht die Bewegung von den Wellen auf die Partikel berücksichtigt.

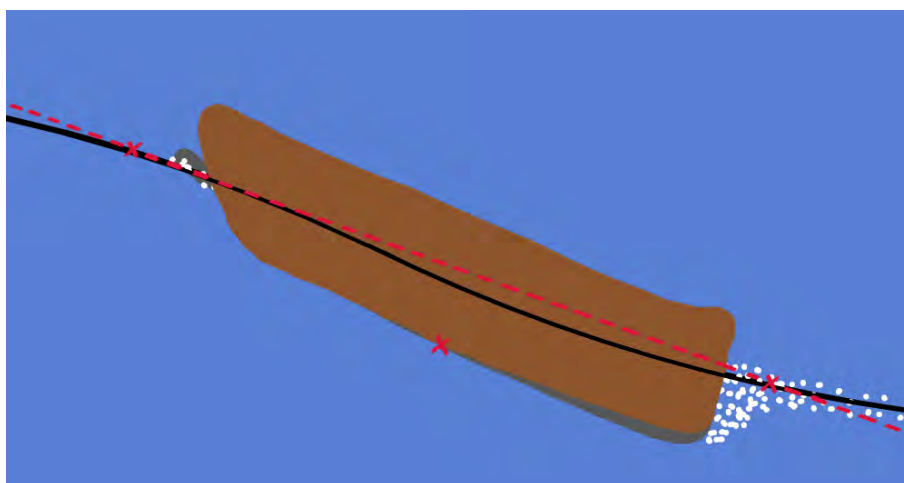


Abbildung 11: Prinzip der Position und Rotation vom Boot (Eigene Darstellung).

Da das Boot nicht direkt mit dem großen Ozean interagiert hat, muss es in Höhe

und Position auch passend auf den Ozean gesetzt werden. Hierfür wird wie in Abbildung 11 gezeigt das Boot an einem Punkt der Unterseite auf die Oberfläche des jetzt verformten Ozeans gesetzt. Das Boot sitzt nun auf der Oberfläche des Wasser und da dort zuvor die berechnete Wasserverdrängung eingebaut wurde, sitzt es schon in seiner passenden Vertiefung.

Um passend die Neigung des Bootes anzupassen, werden vier Punkte um das Boot herum auf die Wasseroberfläche gesetzt. Dabei liegt diese vorne links, vorne rechts, hinten links und hinten rechts von dem Boot. Aus diesen vier Punkten wird eine Ebene berechnet, welche die aktuelle Neigung an der Stelle des Bootes wiedergibt. In Abbildung 11 lässt sich diese Ebene in der gestrichelten roten Linie wiederfinden, welche die Orientierung des Bootes angibt. Somit wurde eine separate flache Wassersimulation in eine wellige Oberfläche eingebaut und folglich das Boot passend integriert.

4.3 Nachstellung der Szene per Roboterarm

Diese virtuellen Szenen sollen nun mit einer echten Kamera und dem echten Modellboot mit identischen Proportionen nachgestellt werden. Das 3D-Boot entspricht aufgrund einer vorherigen Digitalisierung des Modellboots mit diesem perfekt überein. Um diese Szene somit exakt nachzustellen, muss die digitale Relation von Kamera und Boot in echt nachgebaut werden. Die Kamera muss genau den gleichen Winkel und Abstand auf das echte Modellboot haben, wie es es die Kamera auf das digitale hat. Somit entsteht ein digitales Foto von dem 3D-Modell in einer digitalen Ozean Umgebung und ein Foto von dem echten Boot, welche von den Proportionen exakt übereinstimmen. Dies ermöglicht ein Überlagern der beiden Szenen, sodass das echte Modellboot in die digitale Szene nahtlos eingebaut werden kann und es zeitgleich so aussieht als fahre das Modellboot in dem digitalen Ozean.

Hierfür wird bei dem Dreh eine Kamera an eine feste Position gestellt und zeigt in die Richtung von dem Roboterarm. Auf diesem wird das besagte Modellboot befestigt, welches auf die zuvor berechneten Positionen gesetzt wird. Folglich stellt das Boot die digital simulierte Bewegung nach, ohne dass die Position von dem Boot per Hand verändert werden muss.

4.3.1 Relation von Kamera und Boot

Der Roboterarm bewegt sich in seinem eigenständigen Koordinatensystem, in welchem sich auch die reale Kamera an unbekannter Stelle befindet. Diese Relation zwischen der festen Kamera und dem festen Roboterarm ist für die später Wiedergabe der berechneten Position vom Boot wichtig.

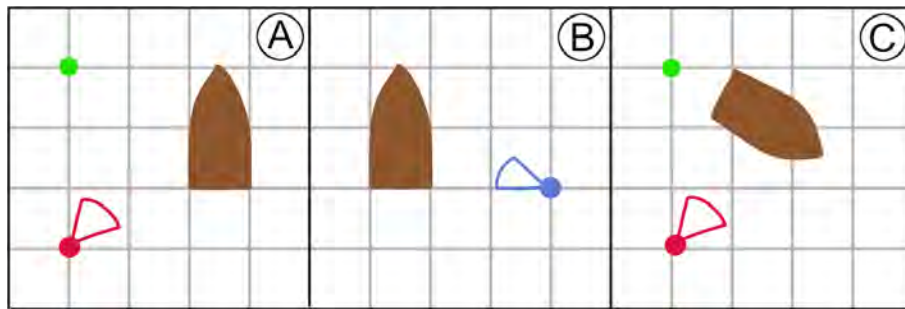


Abbildung 12: Berechnung der Position des Bootes (Eigene Darstellung).

Darstellung A in Abbildung 15 zeigt den ursprünglichen Szenenaufbau zur Berechnung der Relation zwischen Roboter und Kamera. Hierbei ist der grüne Punkt der Ursprung des Koordinatensystem des Roboters und das braune Boot wurde in dem Arbeitsbereich des Roboters positioniert. Der Rote Punkt ist die Position der echten Kamera, welche in die Richtung von dem Boot guckt.

Um nun die Position der Kamera in dem Koordinatensystem des Roboters herauszufinden, benötigt es ein Objekt mit bekannter Position, in dem Fall das Boot. Diese Position muss aus den Daten des Roboterarms ausgelesen werden.

Wenn die Kamera nun ein Foto von dem Boot gemacht hat und das zuvor Erstellte 3D-Modell von diesem an der gleichen Stelle im digitalen Raum positioniert wird wie in echt, dann lässt sich die Position der Kamera berechnen. Mithilfe bestimmter Algorithmen, kann das aufgenommene Foto mit dem 3D-Modell des Bootes überlagert werden, wodurch der Winkel und die Richtung der Aufnahme an einen festen Punkt zurückgeführt werden können. Von genau diesem wurde das Foto aufgenommen, was die Position der Kamera ergibt. Da die ausgelesene Position von dem Boot aus dem Koordinatensystem des Roboters stammt, befindet sich die berechnete Position der Kamera auch in genau diesem Koordinatensystem.

Wird die Kamera in Blender nun an einer anderen Stelle positioniert, wie es in Darstellung B zu erkennen ist, lässt sich die Position vom Boot relativ zur Kamera berechnen. Das Boot hat also eine feste Position in dem lokalen Koordinatensystem der Kamera. Wird die Kamera in Blender nun an die zuvor berechnete Position der echten gesetzt, lässt sich das virtuelle Boot an die gleiche Position des lokalen Kamera Koordinatensystems positionieren wie zuvor. Hierdurch ergibt sich eine neue Position von dem Boot, und zwar umgerechnet in das Koordinatensystem des Roboters.

In Darstellung C wurde die Kamera aus Darstellung B wieder an ihre reale Position und Orientierung gesetzt und das Boot an die gleiche Stelle aus dem lokalen Kamera Koordinatensystem. Der Roboter kann diese berechnete Position von dem Boot anfahren und somit diese Szene aus Blender mit einer gleichen Relation in echt nach-

stellen.

Hierdurch lässt sich eine beliebige Positionierung von Kamera und Boot in einer digitalen Umgebung auf eine feste Kamera umrechnen. Der Roboterarm positioniert das Boot flexible an der berechneten Stelle und sorgt damit für einen identischen Nachbau der Szene. Das ermöglicht ein späteres Überlagern der echten Szene, mit der virtuellen Szene aus Blender, da beide eine genau gleiche Relation von Boot und Kamera haben und somit ein identisches Foto von dem Boot erstellen.

Nachdem die neue Position von dem Boot in dem Koordinatensystem des Roboters berechnet wurde, hat das Boot auch eine andere Rotation. Diese hängt von der Positionierung der Kamera in der Szene ab, kann aber auch aus dem Roboterkoordinatensystems ausgelesen werden. Befindet sich die Kamera wie in Darstellung B rechts von Boot, hat das Boot eine andere Rotation, als würde die Kamera von links auf diese gucken. Die Rotation muss separat ausgelesen werden, da diese unabhängig von der Positionierung des Bootes passiert.

4.3.2 Export Position und Rotation vom Boot

Da das Boot in jedem Frame eine unterschiedliche Position und Rotation hat, müssen diese Daten für jeden einzelnen Frame berechnet und exportiert werden. Hierfür berechnet ein Blender-internes Script die Position und Rotation, welche der Roboter anfahren müsste, und erstellt hier draus ein Script für die Steuerung des Roboters.

4.3.3 Nachstellung der Position und Rotation des Modellbootes

Das Modellboot wird auf dem Roboterarm befestigt und per zuvor erstellten Script an die richtige Position gestellt. Hierfür kann jeder einzelne Frame einer Animation als gezielte Position angefahren werden, wodurch die Bewegung Bild für Bild und somit im Stop-Motion Stil angesteuert werden kann.

4.3.4 Belichtung

Damit das überlagerte Bild von dem echten Modellboot auch farblich in die digitale Szene passt, muss die Belichtung dementsprechend nachgestellt werden. Die richtige Belichtung des echten Bootes bietet erhöhten Realismus bei dem späteren Einbau.

Bei der flachen Ozean Szene gibt es eine diffuse Beleuchtung von allen Seiten, wobei es aufgrund von gutem Wetter auch eine helle Sonne gibt. Diese primäre Lichtquelle muss bei dem Dreh der echten Szene von der gleichen Seite auf das Modellboot scheinen, wie in der digitalen Szene. Da das Modellboot immer anders zwischen den

Szenen auf dem Roboterarm rotiert ist, muss die Richtung der nachgemachten Sonne dynamisch angepasst werden.

Bei der stürmischen Szene gibt es als Besonderheit vereinzelt Blitze zu sehen, welche die digitale Umgebung stark erhellen. Auch das Modellboot muss an genau diesen Stellen mit weißem Licht beleuchtet werden, damit es bei späteren Einbau von der Helligkeit in die Szene passt.

4.4 Überlagerung der Szenen

Durch vorausgehende Berechnung der richtigen Position und Rotation des Modellboots sollte das reale Foto von der Position, Größe und Rotation zur Kamera exakt mit dem digitalen Szene übereinstimmen. Das Boot befindet sich somit an genau der gleichen Stelle auf beiden Bildern und stimmt auch von der Form übereinander, da das reale Modellboot zu einem identischen 3D-Modell verwandelt wurde.

Das Modellboot in dem echten Foto muss von seinem Hintergrund ausgeschnitten werden, sodass es lediglich einen transparenten Hintergrund hat. Folglich kann diese Foto über die digitale Szene gelegt werden und sollte nahezu perfekt in diese passen. Darüber hinaus müssen passende Stellen von dem Bootsrumpf wieder mit Wasser verdeckt werden, um es auf einer dreidimensionalen Ebene in das Meer einzubauen. Dieser Prozess muss für jeden einzelnen Frame aus der Szene wiederholt werden, um somit den ganzen Film von dem echten Dreh in die digitale Welt einzuarbeiten.

Hierdurch wird die Illusion erzeugt, dass das reale Modellboot sich in dieser Szene befindet und aktiv mit dem Wasser interagiert. Es wird perfekt in eine digitale Umgebung eingebunden, welche nur schwierig physisch bei einem echten Dreh nachstellbar wäre. Der digitale Ozean ist nahezu unendlich, was bei einer Stop-Motion Produktion nur mit computergrafischer Hilfe möglich wäre. Auch benötigt es keine Nachstellung von der Interaktion zwischen Modellboot und Wasser, da diese auch digital simuliert wurde.

Durch diese erarbeitete Methodik lässt sich ein reales Modellboot in beliebige computergrafische Umgebungen einsetzen und zeigt auch Interaktion mit dieser. Das identische 3D-Modell simuliert eine für das Modellboot realistische Interaktion mit seiner Umgebung, ohne dass diese in echt gefilmt werden muss. Hierdurch müssen keine komplexen Nachbauten von Wasser durch unterschiedliche Formen oder andere physische Materialien angefertigt werden. Auch Szenen mit einer großen Fläche von Wasser sind im physischen erschwert zu bauen, wodurch die Verbindung mit einem computergrafischen Hintergrund eine Szenengestaltung ermöglicht, wie sie durch rein physische Stop-Motion Technik nicht möglich wäre.

5 Realisierung der Methodik

Bei der Realisierung der zuvor erarbeiteten Methodik wurden die in 2. *Theoretischer Rahmen* genannten Anwendungen und Tools genutzt. Da es sich bei der vorliegenden Arbeit um eine praxisorientierte Bachelorarbeit handelt, wird ab hier die Ich-Form verwendet. Hierdurch können die von mir durchgeführten Arbeitsschritte eindeutig hervorgehoben werden.

5.1 Erstellung 3D-Modell des Boots



Abbildung 13: Modellboot auf einem Stativ (Eigene Aufnahme).



Abbildung 14: 3D-Modell von dem Boot in RealityScan (Eigene Aufnahme).

Für die Erstellung eines computergrafischen Bootes habe ich das reale Modellboot mit RealityScan digitalisiert. Hierfür habe ich insgesamt 149 Fotos von dem Modellboot in Abbildung 13 aus verschiedenen Winkeln gemacht. Realityscan hat darauf hin das Modell in Abbildung 14 erstellt, welches bis auf ein paar Löcher im Segel genau mit dem echten Modellboot übereinstimmt.

Das 3D-Boot wird hauptsächlich zur Berechnung der Interaktion mit dem digitalen Wasser benötigt, wodurch es nicht perfekt aussehen muss. Darüber hinaus dient es noch für die spätere Orientierung von der Überlagerung des echten Modellboots mit dem Wasser der Szene. Imperfektionen des Modells sind nicht weiter schlimm, da das 3D-Boot in dem finalen Film nicht zu sehen ist, sondern mit einem Foto des echten Bootes ersetzt wird.

5.2 Erstellung Ozean

Die Erstellung der zwei Ozeane erfolgte in Blender und basiert auf einer Verbindung des FLIP Fluid addons, mit dem Blender-internen Ocean Modifier. FLIP Fluids wird für die Berechnung der eigentlichen Interaktion zwischen digitalem Boot und Wasser in einem kleinen separaten Bereich verwendet. Der Ocean Modifier verformt eine Oberfläche wellenartig, um einen großflächigen Ozean zu erstellen, ohne das dieser eine direkte Flüssigkeitssimulation ist.

Da das Boot nur mit einem kleinen Teil des digitalen Ozeans interagiert, reicht es auch nur in diesem eine Interaktion zu simulieren. Der restliche Ozean wird großflächig dargestellt, wogegen der Bereich nah um das Boot herum hochauflösend simuliert wird. Dies ist auch die typische Vorgehensweise bei der Erstellung von digitalen Ozean in Filmen oder Videospielen [1, 2, 3, 48].

5.2.1 Flacher Ozean

Für den unendlichen stillen Ozean habe ich eine flache Ebene als Grundlage genommen. Diese habe ich mit einer Textur versehen, welche den Anschein von kleinen Wellen erweckt. Es sieht somit wie eine Wasseroberfläche aus, ist aber keine richtige Wassersimulation. Das Material arbeitet mit einer „Bump Map“, um die Illusion von einer unebenen welligen Oberfläche zu erzeugen.

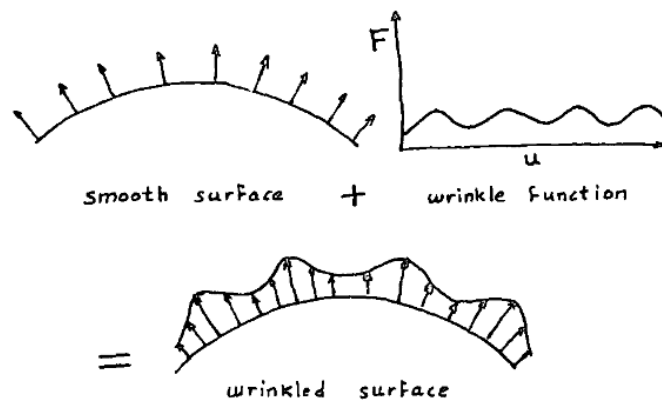


Abbildung 15: Prinzip von Bump Maps [49].

Bump Maps verändern die Normale einer Oberfläche und somit die Richtung, in welche Licht reflektiert wird [49]. Durch diese abweichende Reflexion von Licht, entsteht der Eindruck einer Unebenheit, da eine glatte Oberfläche das Licht in gleichmäßiger Richtung widerspiegeln würde. Die ursprüngliche Geometrie der glatten Fläche bleibt dabei bestehen, wodurch diese Unebenheiten rein aus Veränderung der Lichtreflexionen entstehen und nicht eigenständig berechnet werden müssen.

Bei der Erstellung so einer Wassertextur habe ich mich an dem Material von Linka Palm [50] orientiert und dieses stark auf meine Szene angepasst. Hierbei wurden Größe, Skalierung, Aussehen und Details passend für die Szene verändert.

Für die Beleuchtung der Szene habe ich ein „HDRI“ für ein realistische Lichtbild verwendet. Hier steht HDRI für High Dynamic Range Image und enthält Helligkeitsinformationen einer kompletten Umgebung. Dabei ist ein HDRI typischerweise ein 360° Bild einer Szene, in welche die Helligkeitswerte von jedem Pixel gespeichert sind. Die einzelnen Pixel dienen als Lichtquelle mit der Stärke ihrer gespeicherten Helligkeit. Somit ermöglicht ein HDRI eine realistische Belichtung einer kompletten Szene, ohne einzelne Lichtquellen selber platzieren zu müssen [51].

Bei dem verwendeten HDRI für diese Szene handelt es sich um ein 360° Bild von einem Ozean bei schönem Wetter [52]. Der Himmel ist Blau und es gibt vereinzelt kleine Wolken. Darüber hinaus sorgt eine untergehende Sonne für eine orange und warme Umgebung.

Passend zu dem ruhigen Meer fährt das Boot langsam auf gerader Linie. Dabei macht es keinen großartigen Bewegungen, schaukelt aber leicht von links nach rechts. Diese seitliche Rotation habe ich mit dem Blender internen Noise Modifier hinzugefügt, welcher dem Boot eine zusätzliche periodische Neigung auf seiner Längsachse gibt.

5.2.2 Stürmischer Ozean

Für den Ozean bei Gewitter ist es jedoch unpraktikabel die Wellen per Bump Map zu erzeugen, da bei dieser nur der Anschein einer Welle entsteht. Das digitale Boot soll auf die Oberfläche gesetzt werden, welche bei Verwendung einer Bump Map immer noch flach ist. Hierdurch würde das Boot in einer flachen Linie und nicht auf der Wasseroberfläche fahren. Deshalb habe ich hier den Ocean Modifier von Blender für diese wellenartige Verformung verwendet. Diese können zyklisch animiert werden, wodurch sie sich bewegen und in der Höhe und Position verändern.

Bei der Beleuchtung dieser Szene habe ich ein weiteres HDRI verwendet, diesmal von einer grau bewölkten Umgebung [53], um ihr das schlechte Wetter zu verleihen.

Um die Umgebung des Gewitters weiter zu verstärken, habe ich Blitze an verschiedenen Stellen mit abweichender Größe und Helligkeit eingefügt. Hierbei wurde sich an dem Tutorial von CG Boost [54] orientiert, wobei eine punktförmige Lichtquelle an der Stelle des Blitzes kurzzeitig die ganze Szene aufleuchten lässt.

Auch in dieser Szene fährt das Boot mit einer konstanten Geschwindigkeit, muss aber noch von der Höhe auf die Wasseroberfläche gesetzt werden. Hierfür lass ich

ein leeres Objekt in Blender mit dem Shrinkwrap Modifier auf die Wasseroberfläche setzten. Dabei wird dieses Objekt von oben auf die Wasseroberfläche projiziert und wird so nah wie möglich in jedem Punkt auf die Oberfläche gesetzt. Hierdurch sitzt das Objekt flach und dynamisch auf der Oberfläche und folgt der Bewegung dieser. Um nicht das Boot auch in jedem Punkt flach auf die Oberfläche zu setzten (und somit die Struktur zu verstören) binde ich es an das zuvor erstellte Objekt. Hierdurch folgt es dem leeren Objekt in der Höhe, welches selber auf der Oberfläche des Wasser sitzt.

Somit sitzt das Boot zwar auf der Oberfläche, dreht sich aber noch nicht passend zum Wellengang mit. Hierfür habe ich vier weitere leere Objekte auf die Oberfläche des Wasser gesetzt, und zwar vorne links, vorne rechts, hinten links und hinten rechts. Aus diesen vier Positionen wird in jedem Frame per Python Script eine Ebene berechnet und aus dieser dann die Rotation ausgelesen. Diese wird dann wiederum auch in jedem einzelnen Frame neu auf das Boot übertragen, wodurch dieses den Wellenbewegungen folgt.

Bei dieser Methode wird jedoch nur die aktuelle Position des Wasser und die Neigung direkt auf das Boot übertragen und nicht zum Beispiel die abflachende Beschleunigung beim eintauchen in das Wasser. Ein echtes Boot sitzt nicht genau auf der Wasseroberfläche, sondern beschleunigt bei einem steigenden Position erst langsam nach oben und bei einer fallenden Höhe taucht es ein Stück tiefer in das Wasser ein.

Um diesen Effekt nachzustellen, habe ich in dem Script ein Widerstand eingebaut, wodurch das Boot nicht direkt auf seine berechnete Position und Rotation gesetzt wird. Es werden die Position und Rotation nicht vollständig auf den berechneten Wert gesetzt, sondern etwa 40% in die Richtung der neuen Werte angepasst. Somit macht das Boot keine schnellen springenden Bewegung, sondern Beschleunigt in die Bewegung rein und flacht dann langsam wieder ab. Diese Nachstellung ist physisch nicht korrekt, erzeugt aber einen sehr ähnlichen Effekt, welcher optisch realistische Ergebnisse liefert.

5.3 Interaktion von Boot und Wasser

Beide Ozeane bestehen aktuell nur aus eine Fläche, welche entweder durch eine Textur, oder den Ocean Modifier aussehen wie die Oberfläche von einem Meer. Auch fährt das Boot aktuell nur über die Oberfläche des Ozeans, ohne mit diesem zu interagieren. Hierfür habe ich eine FLIP Fluid Simulation in der Größe von dem Fahrtweg des Bootes erstellt. Diese berechnet separat die Interaktion von dem Boot mit dem Ozean.

5.3.1 Wassersimulation

Bei dem flachen Ozean reicht die Simulation von dem Boot auf einer flachen Wasseroberfläche. Bei dem stürmischen Ozean hingegen sollte die Bewegung der Wellen mit in die separate Wassersimulation aufgenommen werden, um eine realistische Interaktion berechnen zu können. Jedoch lässt sich dies nicht in den mir zur Verfügung stehenden Blender Funktionen und Erweiterungen umsetzen. Somit wurde die Methodik von einer realistischen Flüssigkeitssimulation hin zu einer Projektion einer separaten Simulation abgeändert. Somit wird für beide Szenen eine Interaktion mit einer flachen Wasseroberfläche simuliert, auf welcher das Boot mit der Geschwindigkeit aus der jeweiligen Szene fährt.

Da das entstandene 3D-Modell des Modellbootes sehr löchrig war, habe ich mit einem weiteren Shrinkwrap Modifier von Blender ein anderes Objekt auf dieses geschrumpft. Folglich entsteht ein Objekt in Form von dem ursprünglichen Boot, jedoch ohne jegliche Löcher welche Probleme bei der Interaktion mit Wasser bereiten. Vorher ist Wasser in das Boot gelaufen und es hat nicht korrekt das Wasser verdrängt, weshalb dieses lückenlose Objekt für eine korrekte Berechnung benötigt wurde.

Damit das reale und somit auch identische digitale Boot mit einer Länge von etwa 44 cm in der Simulation größer wirkt, habe ich die Box der Wassersimulation höher skaliert. Eine Einstellung des FLIP Fluid addons erlaubt die relative Größe der Simulation zu erhöhen, wobei die eigentliche Größe der Objekte gleich bleibt. Das ist wichtig, da sonst bei dem Export der Position des Bootes sonst falsche Werte rauskommen, das simulierte Wasser aber trotzdem kleiner aussieht. Dadurch wirkt das Boot größer, als es mit seinem knappen halben Meter in Wirklichkeit ist.

5.3.2 Abflachung zum Rand der Simulation

Damit nun die separate Wasserinteraktion von dem Boot in die große Ozean Oberfläche übergeht, habe ich diese zum Rand hin abgeflacht. Hierfür wurde dem Tutorial von Blender Cinematic (fxartist_b3d) [55] gefolgt, in welchem der Höhenunterschied zum Rand der Simulation langsam auf die ursprüngliche Höhe der Wasseroberfläche gesetzt wird. Es entsteht eine flüssige Abflachung der simulierten Wellen, sodass diese sauber auslaufen, ohne unrealistisch oder ruckartig zu verschwinden. Hierdurch wird der Effekt von einem nahtlosen Übergang aus Abbildung 8 erzielt. Diese Methode funktioniert für beide Szenen.

5.3.3 Einbau in den großen Ozean

Um die beiden einzelnen Teile des Ozean übergangsfrei ineinander übergehen zu lassen, wurde aus dem großen Ozean ein Loch in der Größe der Wassersimulation ausgeschnitten. Bei dem flachen Meer lässt sich aufgrund der zum Rand hin auslaufenden Höhe der Simulation diese sauber in das Loch einsetzen.

Bei dem stürmischen Ozean habe ich mit Geometry Nodes die flache Interaktion von Boot und Wasser von oben auf diesen projiziert. Hierdurch wurde die flache Wasseroberfläche so verformt, das sie genau den gleichen Wellengang hat wie der stürmische Ozean. Auch die eigentliche Interaktion, welche für Wasserverdrängung und Blasenbildung gesorgt hat, wurde bei der Projektion beachtet. Durch das Speichern des Höhenunterschiedes von jedem Partikel zu der flachen Wasseroberfläche, können diese nach der Projektion später wieder dementsprechend verschoben werden.

Diese verformte Fläche hat somit an bestimmten Stellen eine zusätzliche Vertiefung oder Erhöhung des Wassers, welche die Interaktion von Boot und dem stürmischen Meer darstellen, ohne das diese passiert ist. Da das simulierte Wasser gerade von oben auf den großen Ozean gesetzt wurde, bleiben die Proportionen von dem verdrängten Wasser gleich, wodurch die Größe von dem Loch im stürmischen Ozean auch der Größe von dem Boot entspricht.

Da die ursprüngliche flache Wassersimulation zum Rand hin abflacht, gibt es auch bei der Projektion zum Rand hin keine weiteren Höhenverschiebung von der welligen Oberfläche. Somit nähert sich der Rand von der Simulation der originalen Höhe des stürmischen Ozeans. Hierdurch kann die auf die Oberfläche projizierte Wassersimulation ohne sichtbaren Übergang in ein gleichgroßes Loch des stürmischen Ozeans gesetzt werden.

Bei der Umsetzung dieser Methode hat sich in der Praxis bei beiden Szenen jeweils eine sichtbare Kante zwischen der eigentlichen Simulation und dem großflächigen Ozean gebildet. Bei dem flachen Ozean könnte das auf eine minimale Lücke zwischen den Objekten zurückzuführen sein, beim stürmischen Ozean auf die Projektion von oben.

Der verwendete Ocean Modifier für die stürmische Szene verformt eine Oberfläche sehr detailreich, ohne seine eigentliche Topologie zu verändern. Bei der Projektion der einzelnen Partikel der Wassersimulation werden diese jedoch auf die Höhe der Topologie gesetzt. Bei dieser sind auch trotz einer Erhöhung auf viele hundert tausende noch deutliche glatte Flächen zu erkennen. Somit werden die Partikel auf die undetailreiche eigentliche Topologie der Ozean-Oberfläche gesetzt. Hierdurch entsteht eine Kante zwischen dem sehr hochauflösendem Ozean und der Projektion auf

eine Oberfläche mit erkennbar großen glatten Flächen.



Abbildung 16: Kante zwischen Wassersimulation und großem Ozean (Eigene Aufnahme).

Eine Erhöhung der Auflösung auf einige Millionen Vertex hat das Problem zwar verkleinert, da die Topologie kleinere Flächen bekommen hat. Dies hat aber auch die Geschwindigkeit der Berechnung deutlich beschränkt. Bei der Abwägung von benötigter Rechenleistung zu Sichtbarkeit der Kante, wurde sich für eine effizientere Berechnung eine größeren sichtbaren Kante entschieden.

5.4 Kamera in Blender Positionieren

Die Kamera aus dem echten Dreh steht an einer unbekannte Position in dem Koordinatensystems des Roboters, an welcher auch die Kamera in Blender positioniert werden soll. Da ich die Position von dem auf dem Roboter befestigten Boot auslesen kann, ermöglicht mir das die Berechnung der Kameraposition mithilfe des PinSolver Addons. Wichtig hierfür ist eine Übereinstimmung von Brennweite, Fokus und Seitenverhältnis von der echten und digitalen Kamera. Wenn diese voneinander abweichen, lässt sich die Position der einen Kamera nicht präzise berechnen.

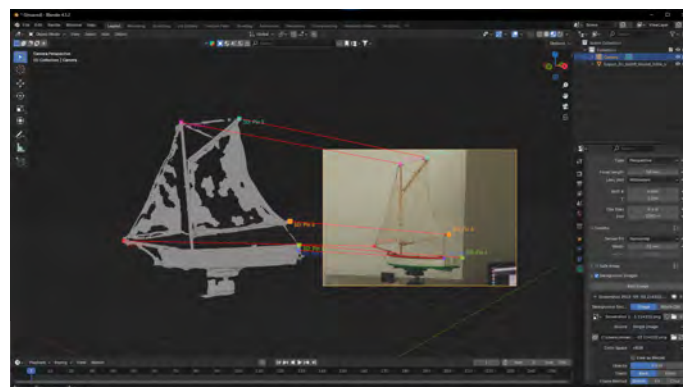


Abbildung 17: PinSolver Addon in Blender (Eigene Aufnahme).

Hierfür nehme ich ein Foto von dem Boot mit der fest positionierten Kamera auf und platziere in Blender das 3D-Modell von dem Boot an der ausgelesenen echten Roboter Position. Für das Addon müssen die Positionen von einzelnen Merkmalen auf sowohl dem Foto, als auch auf dem digitalen Boot markiert werden. Abbildung 17 zeigt meine

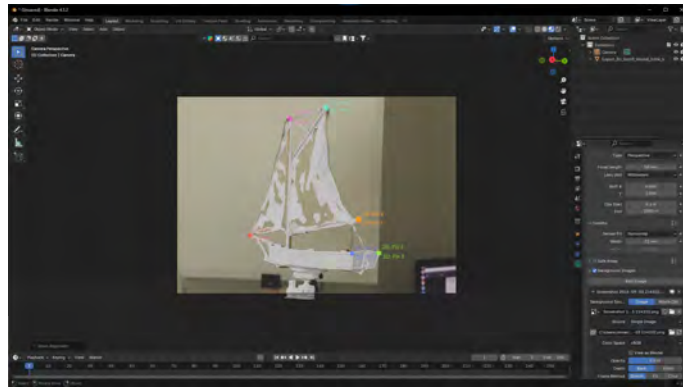


Abbildung 18: PinSolver Addon in Blender: Ergebnis (Eigene Aufnahme).

Auswahl an markierten Punkten, wobei Abbildung 18 eine neue Position für die Kamera berechnet hat, von welcher das ursprüngliche Foto aus aufgenommen wurde.

Somit wurde der Aufbau der echten Szene in Blender übertragen, da jetzt die Kamera an ihrer echten Stelle positioniert ist. Auch wurde die Position von dem echten Boot aus dem Koordinatensystem des Roboters ausgelesen, wodurch die berechnete Position der digitalen Kamera auch in dem Koordinatensystem des Roboters liegt.

5.5 Export Position und Rotation Boot

Wird die digitale Kamera an einer anderen Stelle in dem digitalen Raum positioniert, dann liegt das digitale Boot an beliebiger Stelle in dem lokalen Koordinatensystem der Kamera. Diese Relation von Boot zu Kamera lässt sich per Blender Script in einer 4x4 Matrix berechnen, welche die Position, Rotation und Translation von dem Boot angibt, wobei die Kamera der Ursprung des Koordinatensystems ist.

Setzt man nun die Kamera an die zuvor berechnete Position, an welcher sie bei dem echten Dreh steht, lässt sich das digitale Boot an die gleiche relative Position aus dem Kamera Koordinatensystem setzen. Hierdurch ändert sich Winkel und Blickrichtung auf das Boot nicht, da es immer noch an genau der gleichen relativen Stelle in dem lokalen Kamera Koordinatensystems steht.

Da die sich ursprüngliche Position der Kamera in dem Koordinatensystem des Roboters befindet, wurde nun auch das Boot in dem gleichen Koordinatensystem platziert. Diese Position von dem Boot ist somit die benötigte Position, welche der Roboter anfahren muss, um das Boot in gleicher Relation wie in der digitalen Szene zu platzieren. Somit wurde die beliebigen digitale Positionen von Boot und Kamera einer Szene auf eine feste echte Kamera umgerechnet um somit die nötige Position von dem Modellboot herauszufinden.

Diese Berechnung habe ich mit einem Python Script in Blender umgesetzt, wobei die Kamera zwischen den Frames nicht direkt an ihre echte Position gesetzt wird, sondern per Matrix-Berechnung dieses nur indirekt passiert. Eine 4x4 Matrix hat genau die Lokationsdaten der Kamera gespeichert und wenn diese mit der lokalen 4x4 Matrix des Boots im Raum der Kamera multipliziert wird, ergibt sich eine neue Matrix. Aus dieser lässt sich die benötigte Position des Bootes im Koordinatensystem des Roboters auslesen, ohne das wirkliche räumliche Bewegung die Szene zwischen den Frames immer verschiebt. Hierdurch lassen sich saubere Kamera und Boot Bewegung erstellen, ohne das nach jedem einzelnen Bild beide Objekte verschoben werden.

Um aus der neu berechneten Position des Bootes auch zuverlässig die Rotation bestimmen zu können müssen die Rotationsachsen von dem digitalen Boot mit dem echten übereinstimmen. Das reale Modellboot zeigt bei einer Rotation von 0° um alle Achsen mit seiner Spitze in Richtung der Y-Achse. Damit das digitale Boot gleiche Rotationswerte liefert, welche eine identische Rotation erzielen, habe ich das Koordinatensystem des digitalen Bootes entsprechend transformiert. Somit ich das Boot auch so ausgerichtet, das es entlang der Y-Achse schaut und dabei auch 0° um die Achsen rotiert ist.

Diese Ausrichtung ist wichtig, damit bei der Umpositionierung des Bootes anhand der Kamera-Relation auch die Rotation ausgelesen werden kann. Hiermit wurden das Koordinatensystem des Roboters und das in Blender auf die gleiche Orientierung gesetzt. Wird jetzt beispielsweise das Boot in Blender um 10° auf der X-Achse gedreht, entspricht das auch eine benötigten Rotation des echten Bootes um 10° auf der X-Achse. Indem ich die Rotation der beiden Boote synchronisiert habe, konnte ich mir komplexere Umrechnung der Rotationen in zwei unterschiedlichen Koordinatensystemen sparen.

5.5.1 Erstellung der Script zur Steuerung des Roboterarms

Damit die berechnete Position von dem Boot in jedem Frame von dem Roboter nachgestellt werden kann, benötigt es für jeden Frame ein passendes Script zur Steuerung. Hierfür musste ich zuerst herausfinden, in welchem Format der Roboter gesteuert wird und habe ein Script zur Steuerung aus dem Webinterface exportiert. In diesem stehen sowohl die Rotationswerte für jedes einzelne Gelenk, als auch eine feste Position und Rotation anhand einer 4x4 Matrix.

Glücklicherweise reicht es in dem Script diese Matrix zu ersetzen und der Roboterarm berechnet eigenständig eine neue Rotation der Gelenke aus, um die Position der Matrix zu erreichen. Somit konnte ich ein weiteres Python Script schreiben, wel-

ches für jeden Frame der Animation ein Script für den Roboterarm erstellt. In diesem wird die 4x4 Matrix durch eine Matrix der neu berechneten Position des Bootes ersetzt. Hierbei musste die Matrix transformiert werden, da Blender die Matrix anders ausgibt, als der Roboter diese für seine Positionierung ausließt. Auch das passiert in dem Python Script, welches somit für jeden Frame der Animation eine „task“ Datei für den Roboter erstellt.

5.6 Zusammenfügen der finalen Szene

Durch das Zusammenfügen der Aufnahmen des echten Modellbootes mit dem Render der digitalen Szene entsteht eine einheitliche Szene. Das Modellboot soll realistische in den unendlichen Ozean eingebunden werden und gleichzeitig mit diesem interagieren, ohne das er das physisch macht.

5.6.1 Hintergrund freistellen

Um die Bilder von dem Dreh nun mit der digitalen Szene zu kombinieren, muss in den realen Aufnahmen zuerst der Hintergrund entfernt werden. Diesen Prozess habe ich in einem Comfyui workflow implementiert, bei welchem ich auf Segmentierung trainierte KI Modelle verwendet habe. Unter der Segmentierung eines Bildes versteht man die Einteilung einzelner Pixel in unterschiedliche Klassen, wobei bei einer binären Segmentierung dabei zwischen beispielsweise Vorder- und Hintergrund unterschieden wird [56]. Hier benötigen gerade dünnen Strukturen und feinen Details, wie zum Beispiel Haaren, eine hochauflösende und detailreiche Grenze zwischen den beiden Klassen.

Einige Modelle zur binären Segmentierung sind mit Nodes von 1038lab [57] aufrufbar. Diese bieten eine Schnittstelle zu verschiedenen Modellen mit unterschiedlichen Stärken und Anwendungsbereichen. Hier habe ich je nach Situation die beiden verschiedenen Modelle BEN2 und RMBG-2.0 verwendet. RMBG-2.0 basiert auf dem Bilateral Reference (BiRef) Verfahren, bei welchem eine detailreiche Segmentierung bei einem geringen rechen aufwand erfolgen soll.

BiRef verarbeitet das Eingabebild zuerst auf einer niedrigen Auflösung, um somit eine grobe Segmentierung durchzuführen [58]. Anschließend wird diese an den Kanten verfeinert, um Details hochauflösend zu trennen. Hierfür nutzt der Decoder die Informationen aus der groben Einteilung und holt sich diese Stellen nochmal aus dem hochauflösenden Eingabebild. Es werden die Kanten zwischen den beiden Gruppen verfeinert, ohne das das gesamte Bild in hoher Auflösung durchlaufen werden muss, sondern sich nur gezielt auf diese Bereiche fokussiert wird.

Aufgrund dieser Vorgehensweise kann BiRef feine Segmentierungsgrenzen ziehen und hierbei effizienter arbeiten, als bei einer vollständigen hochauflösenden Verarbeitung des Eingabebildes.

Jedoch hat diese detailreiche Einteilung von Pixeln in Vorder- oder Hintergrund nicht für alle Bilder problemlos funktioniert, wodurch viel Nacharbeit benötigt wurde. So habe ich eine große Anzahl an der von der KI ausgegebenen Masken per Hand nachzeichnen müssen, um falsch gesetzten Grenzen zu beheben. Die Fotos von dem Dreh sind durch eine große Anzahl von Haaren, leicht transparenten Segeln und vielen kleinen Lücken schwierig präzises automatisch zu segmentieren.

5.6.2 Wasser Verdeckung

Die Freigestellten Bilder des Modellbootes werden auf die Ozean Szene gelegt, welche ohne das digitale Boot gerendert wurde. Für einen realistischen Einbau, muss das echte Boot mit dem Rumpf auf in dem digitalen Wasserhintergrund verschwinden. Hierfür benötigt es eine weitere Schicht, welche nur den Part vom Wasser beinhaltet, der den Rumpf des Bootes zu einem Teil bedeckt.

Hierfür habe ich mit Blenders Geometry Nodes eine Kopie der Ozeanoberfläche erstellt, welche nur die Wasseroberfläche entlang des Bootes in zwei Hälften teilt. Hierbei wird je nach Szene nur die Hälfte, welche zu der Kamera zeigt sichtbar und die andere verschwindet. Hierdurch kann dieser Render wiederum über das bereits ins Meer eingebaute Boot gelegt werden und verdeckt dieses auf der einen Seite mit Wasser. Je nach Kamerarichtung wurde die angezeigte Seite dementsprechend angepasst und bei Szenen in welchen das Boot von Hintern gezeigt wird, wurde der Ozean entgegen der Bootachse geteilt. Somit wurde nur das Wasser, welches hinter dem Boot liegt sichtbar und kann auch wieder den Rumpf passend verdecken.

5.6.3 Übereinanderlegen der Schichten

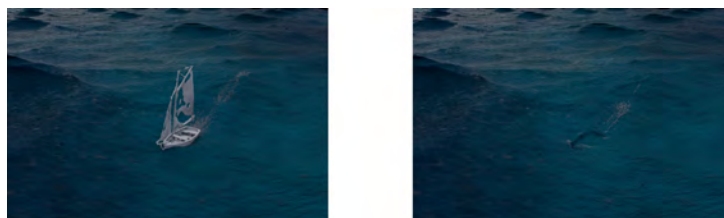


Abbildung 19: Ozean ohne digitales Boot (Eigene Darstellung).

Die verschiedenen einzelnen Schichten müssen passend überlagert werden, um ein finales Bild zu erzeugen. Hierbei kommt ein Bild des gesamten Ozeans aus Abbildung 19 in den Hintergrund und darauf das Hintergrunds freie Foto des Modellbootes.

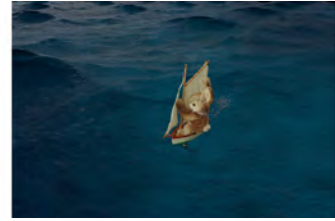


Abbildung 20: Überlegen von freigestellten Boot in den Ozean (Eigene Darstellung).

Hierbei liegt das Boot sehr unnatürlich auf dem Ozean, da der Rumpf nicht in das Wasser eintaucht. Auch Teile von der Halterung, auf welcher das Boot befestigt ist, wurde nicht immer automatisch erkannt und sind sichtbar.

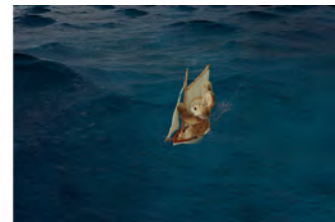


Abbildung 21: Verdecken des Bootrumpfs mit Wasser (Eigene Darstellung).

Eine separat gerenderter Ausschnitt des Ozeans, welcher nur auf einer Seite des Bootes liegt, wird über den Hasen gelegt. Da nur die Seite des Meeres, welche zur Kamera zeigt auf dem Bild liegt, verdeckt es passend dem Rumpf des Schiffs. Hierbei enthält dieser Halbe Ozean auch das Wasser, welches unter dem Boot liegt und folgend falsch den Hasen verdecken würde. Dieser zu viel gerenderte Part war nicht möglich konstant in den Blender *Geometry Nodes* zu entfernen, weshalb hier in vielen Frames dieser per Hand entfernt wurde.

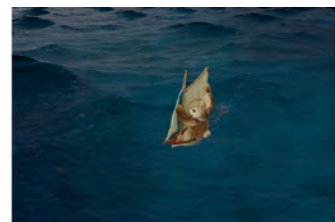
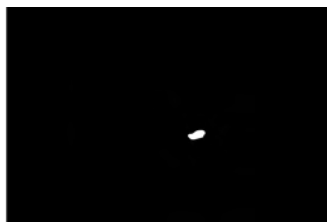


Abbildung 22: Verdeckte Teile des Hasens über Wasser legen (Eigene Darstellung).

Zum Schluss wurde noch der Part von dem Hasen, welcher ungewollt von dem halben Ozean verdeckt wurde per Hand wieder hinzugefügt. Hierfür habe ich für jeden Frame eine separate Maske für die überstehenden Körperteile des Hasen gezeichnet und diese über das finale Bild gelegt. Damit das Boot nicht zu scharfe Kanten hat, wurde diese an dem Übergang von Boot zu Hintergrund leicht verschwommen gemacht. Hierfür habe ich ein Blur-Effekt auf genau diese Stellen angewendet, um somit einen realistischeren Einbau in die Szene zu erzeugen. Zu scharfe Kanten weisen auf ein

im nach hinein eingefügtes Objekt hin, da es aber eine realistische Szene sein sollte, muss das Boot ohne sichtbare Kanten in dies eingebaut werden.



Abbildung 23: Maske für den Blur-Effekt visualisiert (Eigene Darstellung).

Besonders die stürmische Szene benötigte viel Nacharbeit per Hand, da komplexe Verschachtlung von bis zu vier verschiedenen Ebenen nicht automatisiert vorgenommen werden konnte. Somit musste ich in fast jedem Frame eine Maske per Hand zeichnen, oder eine gegebene Maske stückweise verändern. Darüber hinaus haben die unterschiedlichen Ebenen sich nicht perfekt überlagert, wodurch diese auch teils in Größe und Position angepasst werden mussten. Am häufigsten kam es vor, dass das Bild von dem echten Boot nicht perfekt in die dafür vorgesehene Vertiefung in dem Ozean passt. Auch hier mussten manuell viele Frames nachgearbeitet werden.



Abbildung 24: Ausschnitt aus dem finalen Film (Eigene Aufnahme).



Abbildung 25: Ausschnitt aus dem finalen Film (Eigene Aufnahme).

6 Durchführung des Drehs

6.1 Aufbau



Abbildung 26: Aufbau der Szene für den Dreh (Eigene Aufnahme).

Für den Dreh der eigentlichen Stop-Motion Szenen benötigt es einen Roboterarm auf welchem das Boot sitzt und von eine fest positionierte Kamera. Hierbei hat der Roboter die Aufgabe, die digital berechneten Positionen für das Boot präzise nachzustellen, wobei meine Aufgabe die Stop-Motion Bewegung von dem Hasen zwischen den einzelnen Bildern ist. Ein Computer dient der Steuerung von sowohl dem Roboter, als auch der Kamera und ein Scheinwerfer sorgt für die Nachstellung des Lichts aus der digitalen Szene.

6.1.1 Kamera

Für den Dreh der Szene habe ich eine Canon EOS M50 mit verstellbaren Objektiv verwendet. Die Kamera wurde auf einem Stativ in der Ecke platziert und aufgrund von vorheriger Berechnung der Relation zu dem Roboterarm danach nicht mehr bewegt. Höchstens den Fokus habe ich zwischen den Szenen umgestellt, welcher aber aufgrund von einer geringen Anzahl an Markierungen nur stufenweise umgestellt werden konnte. Da ich sowohl in Blender als auch bei der echten Kamera den exakt gleichen Wert für den Fokus brauche, konnte ich diesen nur von den Markierungen ablesen. Somit war ich auf die Fokusslängen mit einer Markierung beschränkt, da ich sonst diese Zahl nicht auch in Blender einstellen könnte. Somit konnte ich nur mit einem Fokus von 18, 24, 35, 50, 70, 100 oder 150mm arbeiten.



Abbildung 27: Kamera *Canon EOS M50* im Szenenaufbau (Eigene Aufnahme).

6.1.2 Roboterarm

Bei dem benutzten Roboterarm handelt es sich um einen Franka Panda Roboterarm. Dieser bietet mit seinen sieben Gelenken eine flexible Ansteuerung von allen Positionen in einem Radius von 85,5cm um die Basis herum [59]. Hierbei hat jedes Gelenk unterschiedliche zulässige Gelenkwinkel, wobei das siebte und somit letzte Gelenk nur in einem Bereich von 166 bis -166 Grad drehen kann. Hierdurch wird nur ein Arbeitsbereich von 332 Grad Drehung in dem letzten Gelenk möglich, was späteres Drehen noch einschränken wird.

Der Roboterarm kann mit Libfranka in ROS programmiert werden, oder in einer eigenen Weboberfläche grafisch bedient werden. Hier können die Bausteine zur Steuerung auch heruntergeladen werden und nach Anpassungen wieder importiert werden. In diesen Bausteinen wird die an zufahrende Position und Rotation gespeichert, welche sich somit beliebig austauschen lässt.

Das letzte Gelenk verfügt über eine Flansch-Schnittstelle an welchem Werkzeuge oder Objekte befestigt werden können. Da das Boot von einem vorherigen Dreh bereits auf einem Kugelgelenk saß, musste dieses noch mit dem Flansch verbunden werden. Hierfür habe ich in Blender ein Modell entworfen, welches nach dem 3D-Drucken die Befestigung von dem Boot an dem letzten Roboter Gelenk ermöglicht hat. Diese wurde in drei Versionen weiter optimiert und lässt sich auf der einen Seite in den Roboter und auf der anderen Seite an das Boot schrauben.

Wenn eine Position in dem Raum angefahren wird, dann bestimmt dies die Zielposition von dem letzten Gelenk. Da aber nun das Boot auf dem letzten Gelenk angebracht ist, befindet sich das Modellboot über dieser an zufahrenden Position. Damit jedoch das Boot an der vorgebenden Zielposition landet, wurde der Abstand zwi-

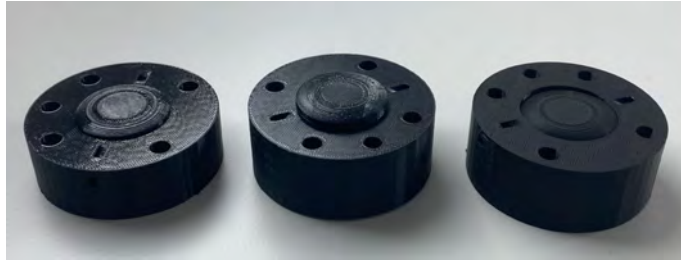


Abbildung 28: Drei Versionen von dem 3D-gedruckten Befestigungsstück (Eigene Aufnahme).

schen dem letzten Gelenk und der Unterseite des Bootes mitberechnet. Hierdurch wird nicht mehr das letzte Gelenk an eine angefahren Position gestellt, sondern auf eine entsprechend berechnete Position von der Unterseite des Bootes.



Abbildung 29: Franka Panda Roboterarm (Eigene Aufnahme).



Abbildung 30: Modellboot auf Roboterarm befestigt (Eigene Aufnahme).

Den Roboterarm habe ich mit einem schwarzen Tuch abgedeckt, um den Kontrast zwischen Boot und Hintergrund zu erhöhen. Dadurch konnte ich später einfacher den Hintergrund raus schneiden, da ein weißes Segel mit einem weißen Hintergrund nur schwierig zu erkennen wäre. Auch habe ich aus dem gleichen Grund einen schwarzen Hintergrund hinter dem Boot positioniert.

6.1.3 Licht

Die Belichtung aus der digitalen Szene muss möglichst akkurat bei dem echten Dreh nachgestellt werden. Bei dem stillen Ozean gibt es eine orange leuchtende Sonne, welche ich mit einem warmen Scheinwerfer nachgestellt habe. Diesen habe ich je nach Orientierung von dem Boot in seiner Position angepasst. Da das Boot zwischen den einzelnen Szenen anderes rotiert ist, das Licht jedoch immer von hinten auf das

Boot trifft, wurde der Scheinwerfer für eine gleichbleibende relative Beleuchtung umgestellt.

Ich habe in einem geschlossenen Raum gedreht, welcher durch Jalousien von außen abgedunkelt werden konnte. Für das Tageslicht in der flachen Ozean Szene habe ich diese also aufgelassen. Hierbei ergab sich das Problem von ungleichmäßiger Belichtung von außen. Da zwischen den einzelnen Bilder einige Zeit vergangen ist, kam von außen aufgrund von Wolken und ähnlichem immer unterschiedlich helles Licht in den Raum. Hierdurch wirkt die Belichtung zwischen den einzelnen Bilder sehr inkonstant und musste in der Nacharbeitung angepasst werden.

Auch konnte ich für die dunkleren Szene die Jalousien nicht vollständig abdunkeln und auch hier ist unregelmäßiges Licht in den Raum getreten. Das nachzustellende Gewitter hat auch an vereinzeln Stellen Blitze, welche ich bei dem echten Dreh wieder mit einem Scheinwerfer nachgestellt habe. Hierbei wurde für die Dauer eines Blitzes das Boot an entsprechender zeitlicher Stelle von entsprechender Richtung hell beleuchtet.

Um diese wechselhaften Lichtverhältnisse von den gedrehten Szenen zu beheben, wurde in dem ComfyUI Workflow jedes Foto farblich auf ein festes Farb-Histogramm gesetzt. Diese beinhaltet eine Verteilung unterschiedlicher Farb- und Helligkeitswerte und deren Häufigkeit, wodurch eine einheitlicherer Helligkeit in einer Szene erzeugt wurde. Hierdurch wurde jedes Foto auf die universelle Helligkeit gesetzt, welche beim Dreh nicht vorhanden war, obwohl auch leichtes Flackern immer noch in dem finalen Video zu erkennen ist.

6.1.4 Steuerung per Computer

Sowohl der Roboterarm an sich, als auch die Kamera waren an einen externen Computer angeschlossen, welche ich für die zentrale Steuerung genutzt habe. Den Roboterarm konnte ich über das Webinterface steuern, in welchem ich eigene aus Blender exportierte „task“ Scripte für das Anfahren von Positionen hochgeladen habe.

Auch die Kamera konnte ich mit einem Kabel an den Computer anschließen, wodurch ich von diesem aus sowohl Fotos machen, als eine Live-Vorschau der Kamera bekommen habe. Diese habe ich in der Stop-Motion fokussierten Animationssoftware „Dragon Frame“ mit dem aktuellen Render des digitalen Bootes abgeglichen. Die Software ist stark auf die Produktion von Stop-Motion ausgelegt und ermöglicht somit die Aufnahme einzelnen Bilder um daraus zusammen hängige Animationen zu erstellen. Diese Bilder werden schon während des Drehs angezeigt und können mit einer Vorlage, wie bei mir der digitale Render, überlagert werden.

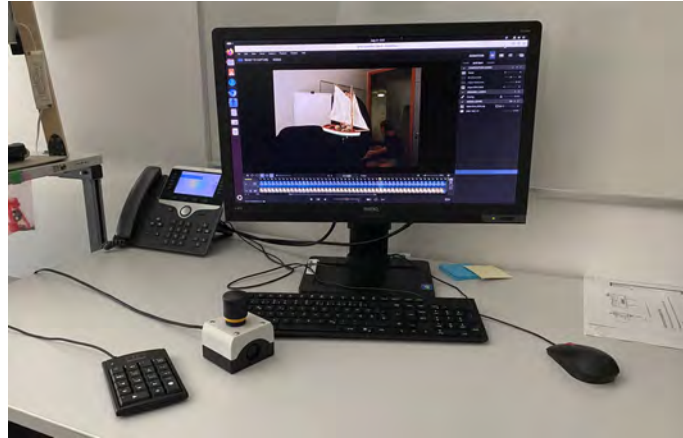


Abbildung 31: Zentraler Computer zur Steuerung (Eigene Aufnahme).

Hierdurch konnte ich minimale Anpassung an der Position und Rotation des echten Modellbootes machen, um es sauber in die digitale Szene zu integrieren. Aus dem Programm heraus lassen sich auch Aufnahmen der Kamera machen, wodurch der Computer wichtig für die zentrale Steuerung war. Ich konnte somit die Fotos anstatt direkt an der Kamera auch von dem PC aus aufnehmen und musste somit die Kamera nicht berühren.

6.2 Ablauf

Die einzelnen Szenen wurden somit Bild für Bild aufgenommen, so wie es für Stop-Motion üblich ist. Damit die Bilder einer Szene möglichst konstant sind (6.1.3 *Licht*), habe ich ganze Szenen an einem Stück gedreht. Jede Szene hat dabei einen sehr ähnlichen Aufbau, bei welcher die Kamera fest positioniert ist und das Licht von dem Scheinwerfer Szenen abhängig umgestellt wurde.

Innerhalb jeder Szene wurde für jedes Bild zuerst das Script für die Positionierung des Roboters geladen. Darauf hin hat der Roboter, wenn er die Position erreichen kann, diese auch angefahren. Folglich war dann meine Aufgabe das manuelle Animieren des Hasen auf dem Boot und das Abgleichen, ob das Boot von der Position und Rotation auch mit dem digitalen übereinstimmt. Um den Anschein von Wind und Bewegung zu kreieren, habe ich auch das Fell des Hasen, sowie das Segel zwischen den einzelnen Bilder unterschiedlich stark verändert. Hierdurch wird die Szene deutlich lebendiger und spiegelt eine Bootsfahrt auf dem Ozean realistischer wieder.

Wenn das Boot mit der Vorlage übereinstimmt und der Hase passende verändert wurde, habe ich ein Foto mit einem Fernauslöser aufgenommen und denn Prozess für das nächste Bild wiederholt. Damit ich weiß, wie ich denn Hasen in jeder Szene animieren muss, habe ich vorweg ein Storyboard für alle 13 geplanten Szenen ent-

worfen. Im folgenden werden die einzelnen Szenen aus dem Storyboard mit Details und Problemen beim Dreh aufgelistet:

1.1	50mm, Closeup Shot mit Zoom heraus	7 Sekunden	Hase liegt auf dem Boot während Arme am Körper anliegen. Am Ende Griff der linken Hand an das Segel zum Steuern.	
-----	------------------------------------	------------	--	---

Eine recht simple Szene, welche zum Testen meiner Methodik und dem späteren Einbau diene.

1.2	50mm, Shot von Links, Parallel zum Boot	5 Sekunden	Hase hat das Segel in der Hand und es kommt ein Windstoß von Links. Der Hase zieht das Segel wieder zurück in die Mitte und lehnt sich zurück und nimmt dabei die Hände hinter den Kopf.	
-----	---	------------	--	---

Hier erreichte das Boot oft den Winkel des letzten Roboter-Gelenks, welcher nicht erreichbar war. Somit musste diese Szene mehrfach auf den Arbeitsbereich des Roboters angepasst und überarbeitet werden.

1.3	70mm, Closup auf den Kopf vom Hasen	4-5 Sekunden	Hase streckt die Arme hinter den Kopf und verdeckt dabei mit seinen Ohren seine Augen. Hase entspannt und probiert zu schlafen.	
-----	-------------------------------------	--------------	---	---

Der Hase lehnt sich zurück und entspannt. Um diese Entspannung zu zeigen lasse ich den Hasen mit den Ohren seine Augen zuhalten.

1.4	35mm, Vogelperspektive, viel Abstand	5 Sekunden	Hase liegt entspannt auf dem Boot.	
-----	--------------------------------------	------------	------------------------------------	---

Bei dieser Szene wollte ich die Limitierung von den Kamerawinkeln testen. Somit stand das Boot fast vertikal auf dem Roboter, konnte aber mit genug Draht trotzdem den Hasen an gewünschter Position halten.

1.5	50mm, Closeup Shot auf Hase,	5 Sekunden	Hase hört Gewitter und Hebt den Kopf an, während er seine Ohren von seinen Augen wieder wegmacht, um etwas zu sehen.	
-----	------------------------------	------------	--	---

Der Hase hört ein lautes Geräusch und guckt hoch. Hier stell ich die Ohren gerade auf, um die Angst und Hellhörigkeit des Hasen zu symbolisieren.

1.6	35mm, Perspektive von Hinten auf das Boot	6 Sekunden	Ein Gewitter zieht auf, Hase fährt genau in die Richtung vom Gewitter. Blitze und hohe Wellen ziehen im Hintergrund auf. Es blitzt mehrfach.	
-----	---	------------	--	---

Das Boot fährt auf ein Gewitter zu. Hier habe ich erstmals Blitze aus der digitalen Szene durch passendes Anschalten eines Scheinwerfers bei dem Dreh nachgestellt.

2.1	35mm, Boot von Links	7 Sekunden	Hase steht auf Boot mitten in Gewitter. Boot wackelt stark hin und her	
-----	----------------------	------------	--	---

Hase steht jetzt auf dem Boot, wodurch eine schwer zu animierende Szene von dem aufstehenden Hasen übersprungen wurde. Dieser harte Sprung von dem Szenenaufbau ist nicht optimal, jedoch war eine Szene in welcher das Hase aufsteht aufgrund der Größenverhältnisse von Hase zu Boot schwierig umzusetzen.

2.2	24mm, Boot von hinten rechts	5 Sekunden	Boot wackelt weiter hin und her	
-----	------------------------------	------------	---------------------------------	--

Infolge einer anderen Blickrichtung auf das Boot wurde der Hase vollständig von dem Segel verdeckt. Diese Szene war schwierig später in den Ozean wieder einzubauen und hat mir zugleich optisch nicht gefallen, weshalb diese es nicht in den finalen Film geschafft hat.

2.3	24mm, Boot wieder von Links	6 Sekunden	Boot wackelt und kippt dann stark nach links. Szene endet mit der Neigung nach links	
-----	-----------------------------	------------	--	---

Das Boot schwankt stark hin und her, wodurch diese Bewegung auch auf den Hasen und das Segel übertragen werden musste.

2.4	35mm, Boot von Hinten und leicht rechts	3 Sekunden	Boot kippt sehr stark nach links, Hase hält sich stark am Mast fest um nicht herunterzufallen.	
-----	---	------------	--	---

Das Boot kippt stark nach links und der Hase hält sich an dem Mast fest, um nicht runterzufallen. Hier wurde der Hase mit Draht an dem linken Arm an dem Mast befestigt. Diese Szene soll den starken Wellengang des Ozeans nochmal deutlicher machen.

2.5	70mm, Closeup auf Mast von rechts	2-3 Sekunden	Mast bricht durch und fällt ins Wasser	
-----	-----------------------------------	--------------	--	---

Für das Durchbrechen von dem Mast, habe ich eine separate Holzstange mit gleichem Durchmesser verwendet. Diese habe ich dann an einer Stelle durchgebrochen und dann langsam umkippen lassen. Hierbei war die Schwierigkeit die Stange an unterschiedlichen Positionen in der Luft zu halten, was ich schließlich mit Metalldraht gelöst habe.

2.6	24mm, Vorne Links auf das Boot	5 Sekunden	Boot fährt ohne Mast weiter. Hase probiert sein Gleichgewicht zu halten und rudert dabei wild mit den Armen.	
-----	--------------------------------	------------	--	---

Auch hier ist wieder eine starke Abweichung in dem Szenenaufbau, da der Mast verschwunden ist, anstatt auf dem Wasser zu treiben. Für diesen Effekt hätte es noch eine weitere Methodik benötigt. Der Hase probiert stark sein Gleichgewicht zu halten und rudert hierfür mit den Armen. Da aber die Arme des Hasen weit über das Boot herausragen, war hier viel manuelle Handarbeit nötig, um die Arme wieder über das Boot-Verdeckende Wasser zu legen.

2.7	35mm, Links auf das Boot	5-6 Sekunden	Hase verliert das Gleichgewicht und fällt nach Hinten um. Dabei fällt er auf den Kopf, verliert sein Bewusstsein und treibt auf dem Boot weiter.	
-----	--------------------------	--------------	--	---

Bei dieser Szene war das Hinfallen des Hasen sehr anspruchsvoll nachzustellen, da er schrittweise in der Luft umpositioniert werden musste. Er stand nur noch auf einem Bein und ihn zwischen den Bildern konstant in der Luft zu verändern war sehr zeitaufwändig. Auch hier wurde viel Draht verwendet, um den Hasen möglichst fixiert zu bekommen. Da das Modellboot dabei etwa schräg auf dem Roboter stand, hat die Fixierung des Hasens nur schwieriger gemacht.

Viele der Szenen konnten aufgrund des limitierten Arbeitsbereichs des Roboters nicht direkt gedreht werden, sondern mussten öfters angepasst werden. Hier musste ich immer wieder ausprobieren, ob bestimmte Positionen und Rotationen des Bootes überhaupt per Roboter erreichbar sind und diese dann wieder in Blender anpassen.

Auch war die Bewegung des Hasen nicht einfach zwischen den einzelnen Bildern zu verändern, da er in jeder Szene unterschiedlich orientiert war. So war das Boot in

Szene 1.4 beispielsweise fast komplett vertikal und der Hase musste mit Draht an diesem befestigt werden. Zwischen den einzelnen Bildern musste der Hase so konstant verändert werden, dass kein einziger Frame zu sehr von dem davor abweicht. Dieses Problem bei Stop-Motion erlaubt keine Fehler, was bei den recht anspruchsvollen Szenen den Dreh deutlich erschwert hat.

Jeder Bestandteil des Aufbaus hat eine wichtige Funktion für den Dreh. Die Kamera war sowohl für die finalen Fotos, als auch die Live Vorschau des Bildes wichtig. Diese Vorschau war dann in Dragon Frame zu erkennen, wodurch ich die reale Szene mit der gerenderten Szene aus Blender übereinstimmen konnte. Der Computer war darüber hinaus für die zentrale Steuerung des Roboters zuständig, und dieser für die Positionierung des Bootes. Meine Aufgabe war das Verändern des Hasen zwischen den einzelnen Aufnahmen, um die Illusion von Bewegung und Leben zu erzeugen.

6.3 Kritische Reflexion

Um die Qualität der produzierten Ergebnisse genauer zu überprüfen, wird im folgenden das methodische Vorgehen kritisch reflektiert.

Der Dreh der beiden Szenen hat durchaus gut funktioniert und auch hat der Roboter die Bewegung des digitalen Bootes passend auf das physische Boot übertragen. Jedoch hat es hierfür ein zeitaufwändiges Testen der möglichen Positionen und Rotationen des Roboters und dessen Arbeitsbereich benötigt. Viele Szenen wurden in Blender erstellt und in Script umgewandelt, nur damit der Roboter diese nicht nachstellen kann. Somit war die Gestaltung der Szenen stark auf den Arbeitsbereich des Roboters beschränkt. Es konnten folglich keine zu nahen Aufnahmen von dem Boot gemacht werden und auch keine weit entfernten. Um Szenen mit unterschiedlichen Abständen von Kamera zu Boot zu verwirklichen, musste mit verschiedenen Fokus-Einstellung der Kamera gearbeitet werden.

Diese waren jedoch schwierig präzise an dem Objektiv der Kamera einzustellen, da diese zwischen 18mm und 150mm frei verstellbar war. Einzelne Markierungen an dem Objektiv konnten als grobe Orientierung verwendet werden, jedoch sind diese auch nur aufgedruckte Zahlen ohne jegliche Form der Orientierung.

Somit war es schwierig die exakte Fokus-Länge der Blender Kamera nachzustellen, wodurch das reale Bild nicht perfekt mit der digitalen Szene übereinstimmt. Durch diese Schätzen der aktuellen Einstellung des Objektivs, konnte es bereits bei der Berechnung der Kameraposition zu Ungenauigkeiten gekommen sein. Diese Berechnung muss extrem präzise sein, damit die digitale Kamera das gleiche Verhältnis zu dem Roboter hat, wie in echt. Minimale Abweichung der berechneten Position sorgen

folglich für eine falsche Positionierung des Bootes. Diese wird in Abhängigkeit von Boot und Kamera berechnet, was bei einer falsch berechneten Kameraposition für eine verschobene Wiedergabe der digitalen Boots-Position führt. Somit haben sich einige kleine Fehler im Aufbau durch den ganzen Dreh gezogen und folglich zeitaufwändig im Nachhinein wieder entfernt werden mussten. Des weiteren war das Boot aufgrund einer defekten Schraube nicht richtig fest auf dem Kugelgelenk befestigt, wodurch diese sich eigenständig auf der Längsachse drehen konnte. Es weichte in der Rotation teilweise von der berechneten Orientierung ab. Um das zu beheben wurde beim Dreh die Live-Vorschau der Kamera mit der gerendeten Ozean Szene überlagert, um eine falsche Rotation manuell in jedem Bild zu beheben.

Aufgrund meiner Wahl einer komplexeren Szene, hat die Automatisierung der Hintergrundfreistellung oft nicht optimal funktioniert. So besteht meine verwendete Figur aus vielen dünnen Haaren und zusätzlich hat das Boot viele kleine Lücken zwischen den schon transparenten Segeln. Es wurde viel Nacharbeit bei dem Hintergrund entfernen der einzelnen Schichten benötigt, was den Zeitaufwand enorm erhöht hat. Teilweise mussten bis zu 4 Schichten überlagert werden, wobei jede genau passend ausgeschnitten sein musste.

Die komplexe Verdeckung des Bootes durch die zweite Schicht Wasser hat auch oft nicht sauber alles richtig verdeckt, wodurch auch hier in jedem Frame per Hand teile ausgeschnitten werden mussten. Das lag primär an der Instabilität von einem bestimmten Node in Blenders Geometry Nodes, welche den Part Wasser unter dem digitalen Boot nicht sauber ausschneidet. Wenn dieser wie in Abbildung 32 noch zu sehen ist, würde er das Boot falsch verdecken und muss somit per Hand ausgeschnitten werden.

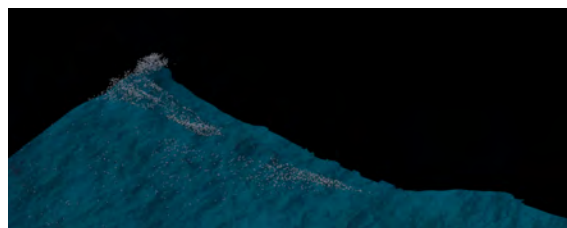


Abbildung 32: Vordere Wasserschicht falsch gerendert (Eigene Aufnahme).

Darüber hinaus mussten Körperteile von dem Hasen noch über diese zweite Schicht Wasser gelegt werden, welche auch nicht automatisch erkennbar waren. So musste auch hier Bild für Bild die Körperteile des Hasen, welche von dem zweiten Wasser verdeckt wurden wieder per Hand ausgeschnitten und über das Wasser gelegt werden. Diese sehr hellen Haare des Körpers konnten auch nicht automatisch von der weißen Farbe des Bootes getrennt werden, weshalb hier die unglückliche Farbwahl eine weitere manuelle Hintergrundentfernung benötigt hat.

7 Ergebnisse

Der in Kapitel 6 vorgestellte Dreh ergibt ein Stop-Motion Kurzfilm von einem Hasen auf einem Modellboot, welches in ein Gewitter geriet. Dabei fasst der Film eine gesamte Anzahl von 853 Bildern, welche bei einer nach Sfetcu [60] in Stop-Motion gängigen Anzahl von 12 Bewegungsänderung pro Sekunde, 71 Sekunden Material ergibt:

“With the standard practice of ‘doubles’ (double-framing — exposing 2 frames for each shot), 12 changes are usually made for one second of film movement” [60, S. 219].

Für diesen etwas über eine Minute langen Film habe ich ungefähr 30 Stunden für den reinen physischen Dreh und weitere 25 Stunden für den Einbau des echten Bootes in die digitale Szene benötigt. Hierbei wird die eigentliche Erstellung der digitalen Ozeane, Wassersimulation, Kamerafahrten, Scripte oder Workflows nicht mit eingerechnet.

Der Kurzfilm kann unter folgendem QR-Code aufgerufen werden:



Oder alternativ über einen dieser Links:

<https://drive.google.com/file/d/14LqKaKv2XfpCI9B1Tved0AY2zURd0EIY/view>

<https://nc.uni-bremen.de/index.php/s/F5XZYMdWWLD9aER>

8 Fazit

Im Mittelpunkt der vorliegenden Arbeit stand die Produktion einer Stop-Motion Boot Szene und der darauf folgende Einbau in einen digitalen Ozean. Die vorgenommene digitale Boots-Simulation konnte recht zuverlässig über den Roboterarm auf das Modellboot übertragen werden, welches dann per Stop-Motion Technik animiert wurde. Jedoch zogen sich kleine Fehler durch den Aufbau an welchem die Methodik getestet wurden, und sorgten für einen hohen zeitlichen Aufwand bei der Behebung.

Die Forschungsfrage kann demgegenüber beantwortet werden, dass die Integration eines Stop-Motion animierten Boot in einen digitalen Ozean durchaus realisierbar ist. Hierfür wurde das Modellboot zunächst digitalisiert und anschließend in einer virtuellen Umgebung simuliert. Diese Bewegung wurde anhand eines Roboterarms auf ein physisches Modellboot übertragen. Entsprechendes Boot wurde mit der Stop-Motion Technik animiert und schlussendlich durch Überlagerung wieder in den digitalen Ozean eingepflegt.

Die entwickelte Methodik hat sich grundsätzlich als geeignet erwiesen, unterscheidet sich jedoch von den in der Praxis üblichen verwendeten Motion-Control-Systemen von größeren Filmstudios, bei welchen die Position der Kamera zu jeder Zeit bekannt ist. Somit musste die Position der Kamera erstmals anhand eines einzelnen Bildes und dem korrelierenden 3D-Modell berechnet werden, was nur für Ungenauigkeiten gesorgt hat. Diese sorgen für minimale Abweichung in der Position der Objekte, welche beim Überlagern der Schichten dann zeitaufwändig händisch behoben werden mussten.

Des weiteren hat die Implementierung von dem Übergang zwischen der Wassersimulation und dem großflächigen Ozean grundsätzlich funktioniert, erzeugte aber visuelle Kanten durch zu geringe Auflösung. Durch Limitierung in der Rechenleistung von benutzter Technik konnte keine hochauflösende Übergang berechnet werden, was jedoch nicht gegen die Wirksamkeit der Methodik spricht.

Ob eine Optimierung der Methodik anhand eines genaueren Objektivs der Kamera, simpleren Stop-Motion animierten Objekten, größeren Arbeitsbereich der Roboters, oder Verwendung von rechenstärkerer Technik oder genaueren KI-Modellen zur Segmentierung der Bilder für eine effektivere Produktion sorgt, benötigt weiteres Testen. Die Umsetzung der erarbeiteten Methodik war aufgrund genannter Mängel sehr zeitaufwändig. Jedoch kann die Methodik aufgrund der Verwendung einer separaten Kamera eine mögliche Alternative zu kostenintensiven Motion-Control-Systemen bieten.

9 Literaturverzeichnis

- [1] Mark Sawicki und Juniko Moody. *Filming the Fantastic with Virtual Technology: Filmmaking on the Digital Backlot*. März 2020. ISBN: 9780429331282. DOI: 10.4324/9780429331282.
- [2] Darles Emmanuelle u. a. „A Survey of Ocean Simulation and Rendering Techniques in Computer Graphics“. In: *Computer Graphics Forum - CGF* 30 (Sep. 2011). DOI: 10.1111/j.1467-8659.2010.01828.x.
- [3] Qifeng Yan. „Research and Innovation of Animated Movie Production Technology Based on Computer Graphic Design“. In: *Applied Mathematics and Nonlinear Sciences* 9 (Nov. 2024). DOI: 10.2478/amns-2024-3174.
- [4] Barry J. C. Purves. *Stop-motion Animation: Frame by Frame Film-making with Puppets and Models*. 2. Aufl. Basics Animation. London: Fairchild Books, 2014. ISBN: 978-1-4725-2190-3.
- [5] Dan Sarto. *Stop-Motion and VFX Converge in the Vast Oceans of ‘Kubo and the Two Strings’*. URL: <https://www.awn.com/vfxworld/stop-motion-and-vfx-converge-vast-oceans-kubo-and-two-strings>. (accessed: 01.09.2026).
- [6] Joe Foley. *Laika reveals the secrets of how it depicts water in stop-motion animation*. URL: <https://www.creativebloq.com/art/animation/laika-reveals-the-secrets-of-how-it-depicts-water-in-stop-motion-animation>. (accessed: 01.09.2026).
- [7] Blender. *About Blender*. URL: https://docs.blender.org/manual/en/latest/getting_started/about/index.html. (accessed: 31.07.2026).
- [8] Unbekannt. *About*. URL: <https://extensions.blender.org/about/>. (accessed: 03.09.2026).
- [9] Unbekannt. *Quickstart*. URL: https://docs.blender.org/api/current/info_quickstart.html. (accessed: 03.09.2026).
- [10] rlguy. *Blender-FLIP-Fluids*. URL: <https://github.com/rlguy/Blender-FLIP-Fluids>. (accessed: 03.09.2026).
- [11] J. Brackbill und H.M. Ruppel. „FLIP: A method for adaptively zoned, particle-in-cell calculations of fluid flows in two dimensions“. In: *Journal of Computational Physics* 65 (Aug. 1986), S. 314–343. DOI: 10.1016/0021-9991(86)90211-1.

- [12] Unbekannt. *Frequently Asked Questions*. URL: <https://flipfluids.com/faq/>. (accessed: 03.09.2026).
- [13] Unbekannt. *Main Features*. URL: <https://flipfluids.com/features/>. (accessed: 03.09.2026).
- [14] Blender Extensions. *PinSolver. Interactive camera alignment*. URL: <https://extensions.blender.org/add-ons/pin-solver/>. (accessed: 06.08.2026).
- [15] Rashik Shrestha. *Perspective-n-Point(PnP)*. URL: <https://medium.com/@rashik.shrestha/perspective-n-point-pnp-f2c7dd4ef1ed>. (accessed: 03.09.2026).
- [16] Ping Wang u. a. „A simple, robust and fast method for the perspective-n-point Problem“. In: *Pattern Recognition Letters* 108 (2018), S. 31–37. ISSN: 0167-8655. DOI: <https://doi.org/10.1016/j.patrec.2018.02.028>. URL: <https://www.sciencedirect.com/science/article/pii/S0167865518300692>.
- [17] Open Source Computer Vision. *Perspective-n-Point (PnP) pose computation*. URL: https://docs.opencv.org/4.12.0/d5/d1f/calib3d_solvePnP.html?. (accessed: 03.09.2026).
- [18] kurono73. *PinSolver*/*__init__.py*. URL: https://github.com/kurono73/PinSolver/blob/main/__init__.py. (accessed: 04.09.2026).
- [19] Blender Docs. *Object Modifiers*. URL: <https://docs.blender.org/manual/en/latest/modeling/modifiers/>. (accessed: 06.08.2026).
- [20] Blender Docs. *Ocean Modifier*. URL: <https://docs.blender.org/manual/en/latest/modeling/modifiers/physics/ocean.html>. (accessed: 06.08.2026).
- [21] blender.org. *Ocean Simulation*. URL: <https://archive.blender.org/wiki/2015/index.php/Doc%3A2.6/Manual/Modifiers/Simulate/Ocean/?>. (accessed: 04.09.2026).
- [22] Blender Docs. *Geometry Nodes Introduction*. URL: https://docs.blender.org/manual/en/latest/modeling/geometry_nodes/introduction.html. (accessed: 06.08.2026).
- [23] Quixel. *Photogrammetry Basics*. URL: <https://dev.epicgames.com/community/learning/courses/blA/unreal-engine-realityscan-photogrammetry-basics-by-quixel/k4ve/unreal-engine-realityscan-what-is-photogrammetry>. (accessed: 06.08.2026).
- [24] Comfy. *ComfyUI Official Documentation*. URL: <https://docs.comfy.org/>. (accessed: 04.09.2026).
- [25] Comfy. *How to Install Custom Nodes in ComfyUI*. URL: https://docs.comfy.org/installation/install_custom_node. (accessed: 04.09.2026).

- [26] Paul Strauss. *Stop-motion Ocean*. URL: <https://theawesomer.com/stop-motion-ocean/456630/>. (accessed: 29.07.2026).
- [27] Mark Smith. *Featurette 2 | A Stop Motion Ocean*. URL: <https://vimeo.com/237123570>. (accessed: 02.09.2026).
- [28] Dan Sarto. *On the Set of Laika's 'The Boxtrolls'*. URL: <https://www.awn.com/animationworld/on-set-laikas-boxtrolls>. (accessed: 02.09.2026).
- [29] Arielle Silver. *Ghost Ships | Arielle Silver | OFFICIAL VIDEO*. URL: <https://www.youtube.com/watch?v=2KMPX4IAEBg>. (accessed: 29.07.2026).
- [30] Inspira Stop Motion. *Stop Motion Waters 8 - WAVE - Behind the Scenes Just Dance 2023 edition*. URL: <https://www.youtube.com/watch?v=yA3EfNs11yM>. (accessed: 29.07.2026).
- [31] John Montgomery. *Kubo and the Two Strings: water effects*. URL: <https://www.fxguide.com/featured/kubo-and-the-two-strings-water-effects>. (accessed: 02.09.2026).
- [32] Field Day. *The Sea Is Blue: A Stop Motion Short Film | Field Day Presents Vincent Peone*. URL: <https://www.youtube.com/watch?v=Dp2ZskwkhkA>. (accessed: 21.08.2026).
- [33] Unbekannt. *Matthew McConaughey Kubo and the Two Strings*. URL: <https://www.girl.com.au/matthew-mcconaughey-kubo-and-the-two-strings.htm>. (accessed: 01.09.2026).
- [34] The Verge. *Stop-motion animation goes high tech at Laika*. URL: <https://www.youtube.com/watch?v=plXmbLAUTRg>. (accessed: 31.07.2026).
- [35] Jesse Levitt. *Motion Control*. URL: <https://jesse-levitt-wshc.squarespace.com/motion-control-1>. (accessed: 01.09.2026).
- [36] *Missing Link*. 2019.
- [37] Ian Failes. *How stop-motion and VFX meet in Laika's 'Missing Link'*. URL: <https://beforesandafters.com/2019/04/12/how-stop-motion-and-vfx-meet-in-laikas-missing-link>. (accessed: 02.09.2026).
- [38] Vincent Frei. *MISSING LINK: Steve Emerson – VFX Supervisor – LAIKA*. URL: <https://www.artofvfx.com/missing-link-steve-emerson-vfx-supervisor-laika/>. (accessed: 02.09.2026).
- [39] Steve Switaj. *The Robots of LAIKA*. URL: <https://history.siggraph.org/wp-content/uploads/2022/09/2018-Talks-Switaj-The-Robots-of-LAIKA.pdf>. (accessed: 02.09.2026).

- [40] Dan Sarto. *Set Visit: Go Behind the Scenes of LAIKA's 'Missing Link'*. URL: <https://www.awn.com/animationworld/set-visit-go-behind-scenes-laikas-missing-link>. (accessed: 02.09.2026).
- [41] Leo Benedictus. *Aardman: Shipshape and Bristol fashion*. URL: <https://www.theguardian.com/film/2011/oct/27/aardman-pirates-adventure-scientists>. (accessed: 02.09.2026).
- [42] Unbekannt. *Interview: Rigging the cast of The Pirates! Band of Misfits with Character TD Martin Orlowski*. URL: <https://www.riggingdojo.com/2012/04/29/interview-martin-orkowski/>. (accessed: 02.09.2026).
- [43] Unbekannt. *JAMES AND THE GIANT PEACH*. URL: <https://filmeducation.org/pdf/film/JamesJP.pdf>. (accessed: 03.09.2026).
- [44] Victoria McKee. *You gotta roll with it*. URL: <https://www.independent.co.uk/arts-entertainment/you-gotta-roll-with-it-1330476.html>. (accessed: 03.09.2026).
- [45] Laura Saini. „New tools for animation and design : a haptic-based system for stop motion camera animation and curve design by algebraic-trigonometric Pythagorean Hodograph curves“. In: (Juni 2013).
- [46] Alexandra L. Zheleznyakova. „Physically-based method for real-time modelling of ship motion in irregular waves“. In: *Ocean Engineering* 195 (2020), S. 106686. ISSN: 0029-8018. DOI: <https://doi.org/10.1016/j.oceaneng.2019.106686>. URL: <https://www.sciencedirect.com/science/article/pii/S0029801819308017>.
- [47] Haiwen Tu u. a. „Performing captive model tests with a hexapod“. In: *Ocean Engineering* 171 (2019), S. 49–58. ISSN: 0029-8018. DOI: <https://doi.org/10.1016/j.oceaneng.2018.10.037>. URL: <https://www.sciencedirect.com/science/article/pii/S0029801818320754>.
- [48] Hilko Cords und Oliver Staadt. „Real-Time Open Water Environments with Interacting Objects.“ In: Jan. 2009, S. 35–42.
- [49] James F. Blinn. „Simulation of wrinkled surfaces“. In: *SIGGRAPH Comput. Graph.* 12.3 (1978), S. 286–292. ISSN: 0097-8930. DOI: [10.1145/965139.507101](https://doi.org/10.1145/965139.507101). URL: <https://doi.org/10.1145/965139.507101>.
- [50] Linka Palm. *Procedural Ocean Material*. URL: <https://www.blendkit.com/asset-gallery-detail/ea1fc7ba-86bd-4176-a494-adf971f44bd7/>. (accessed: 23.08.2026).

- [51] Lazi Akademie. *HDRI-Beleuchtung in 3D, Realistisches Umgebungslicht mit HDR-Bildern*. URL: <https://www.lazi-akademie.de/wiki/animation-vfx/3d-grundlagen/hdri-beleuchtung-3d/>. (accessed: 18.08.2026).
- [52] Free Poly. *Clouds HDRI*. URL: <https://www.blendkit.com/asset-gallery-detail/7f183564-86a2-459c-ab45-15b2825efca3/>. (accessed: 23.08.2026).
- [53] Greg Zaal. *Cedar Bridge HDRI*. URL: <https://www.blendkit.com/asset-gallery-detail/02b5a7c6-7d9f-4884-9094-46782c151fd2/>. (accessed: 23.08.2026).
- [54] CG Boost. *Animated Lightning in Blender (Tutorial)*. URL: <https://www.youtube.com/watch?v=oE-vT1ekvrs>. (accessed: 02.08.2026).
- [55] Blender Cinematic (fxartist_b3d). *How to blend Flip simulation onto Ocean | Flip Fluids addon | Ocean Modifier || Blender*. URL: <https://www.youtube.com/watch?v=TwJiGmCfaV4>. (accessed: 23.08.2026).
- [56] Hongyuan Zhu u. a. „Beyond pixels: A comprehensive survey from bottom-up to semantic image segmentation and cosegmentation“. In: *Journal of Visual Communication and Image Representation* 34 (2016), S. 12–27. ISSN: 1047-3203. DOI: <https://doi.org/10.1016/j.jvcir.2015.10.012>. URL: <https://www.sciencedirect.com/science/article/pii/S1047320315002035>.
- [57] 1038lab. *ComfyUI-RMBG*. URL: <https://github.com/1038lab/ComfyUI-RMBG>. (accessed: 29.08.2026).
- [58] Peng Zheng u. a. „Bilateral Reference for High-Resolution Dichotomous Image Segmentation“. In: *CAAI Artificial Intelligence Research* 3 (2024), S. 9150038. DOI: 10.26599/AIR.2024.9150038. URL: <https://www.sciopen.com/article/10.26599/AIR.2024.9150038>.
- [59] Franka Robotics. *USER MANUAL, PANDA RESEARCH*. URL: https://wiki.happyllab.at/images/1/13/Panda_research_-_Benutzerhandbuch_DE.pdf. (accessed: 26.08.2026).
- [60] Nicolae Sfetcu. *Animation Cartoons*. März 2006. URL: https://www.researchgate.net/publication/369680540_Animation_Cartoons.

10 Nutzung KI basierte Anwendungen

Erkennung Format der Roboterscripte

Die aus dem Webinterface exportierten Scripte für den Franka Panda Roboterarm folgen einem sehr komplexen Aufbau durch Aneinanderreihung von vielen Zahlen und Daten. Hier wurde ChatGPT für das Verstehen dieser einzelnen Bauteile verwendet.

Link: <https://chatgpt.com/>

Umwandlung Matrixen

Die Positions- und Rotationsdaten aus dem Script des Roboterarms werden in einer 4x4 Matrix gespeichert. Der Roboterarm hat diese aber anders formatiert, als wie Blender 4x4 Matrixen liebt. Hier wurde zuerst ChatGPT für diese Transformation verwendet, bevor ich ein passendes Python Script geschrieben habe.

Link: <https://chatgpt.com/>

Optimierung Python Scripte

Für die Fehlersuche und Optimierung meiner Python Scripte habe ich ChatGPT verwendet.

Link: <https://chatgpt.com/>