Entwicklung einer VR-basierten Methode für 3D-Datennotation

Cmore Automotive GmbH

Patrick Grüner

16. März 2018

Inhaltsverzeichnis

Abstract

Abstract goes here

Dedication

To mum and dad

Declaration

I declare that..

Acknowledgements

I want to thank...

Glossar

Debugging-Tool

TODO 11

Game Engine

TODO 14

Labeling/Datennotation/Klassifizierung

TODO 4, 5, 6, 7, 8, 10, 11, 14, 22, 23

Open Source

TODO 22

Post Processing

TODO 12

Predictive Analytics

TODO 14

Roomscaling

TODO 9, 10, 11

Scripting

TODO 12

Shader

TODO 12

Software Development Kit

TODO ix, 6

Taktrate

TODO 11

Tiefenkamera

TODO 5

Tiefenkamera

TODO 5

User Interface

TODO 12

Workflow

TODO 21

Akronyme

AR	Augmented Reality i, 4, 5, 7, 8, 10
IDS	Interne Datenstruktur 22, 23
KNN	Künstliche Neuronale Netze 14, 17, 18, 20
SDK	Software Development Kit 6
UI	User Interface 21
VR	Virtual Reality i, 4, 6, 8, 10, 11, 12, 14

1

Einleitung

Der Einsatz künstlicher Intelligenzen ist aktuell in allen Bereichen der Informatik auf dem Vormarsch. Dies gilt vor allem für die Automobilindustrie, da das Thema des autonomen Fahrens nicht ohne intelligente Algorithmen realisierbar ist. Eine wichtige Rolle bei der Entwicklung solcher Algorithmen spielt dabei das maschinelle Lernen. Dabei wird versucht, ausgehend von vielen lehrreichen Beispieldaten, die Lösung einer Aufgabe zu lernen und auf andere unbekannte Daten zu verallgemeinern. So kann ein System auch auf vorher ungesehene Daten reagieren, was mit einer statischen Programmierung nur schwer oder gar nicht möglich ist. Der Erfolg dieses Prinzips hängt genauso von der Qualität der Trainingsdaten ab wie der des Algorithmus. Deshalb wird in die Erstellung dieser Daten sehr viel Arbeit gesteckt.

Im Bereich der Fahrerassistenzsysteme und des autonomen Fahrens ist es wichtig, dass das Auto seine Umgebung so gut wie möglich wahrnehmen kann. Deshalb werden für Algorithmen, die in diesen Bereichen angewendet werden, Daten von Sensoren verwendet um das Umfeld des Autos wahrzunehmen. Dies sind meist Kamera-, Rader- oder Lidarsensoren. Die Eingangsdaten dieser Sensoren müssen nun so aufbereiten werden, damit ein lernendes Computersystem etwas damit anfangen kann. Dazu werden alle, für das Anwendungsfeld wichtigen Teile mit entsprechenden Klassifikationen versehen. Zum Beispiel werden auf einem Kamerabild alle Personen und Fahrzeuge als eben diese markiert, sodass das Fahrzeug lernen kann wie Personen und Fahrzeuge aussehen und diese dann später im Straßenverkehr selbstständig erkennt. Radar- und Lidarsensoren liefern stattdessen Punktwolken als Eingangsdaten (vgl. Abbildung **??**), das Prinzip bleibt allerdings das gleiche. Auch hier müssen alle wichtigen Teile, in diesem Fall Punkte, mit entsprechenden Klassen versehen werden. Dieser Vorgang nennt sich Labeln.

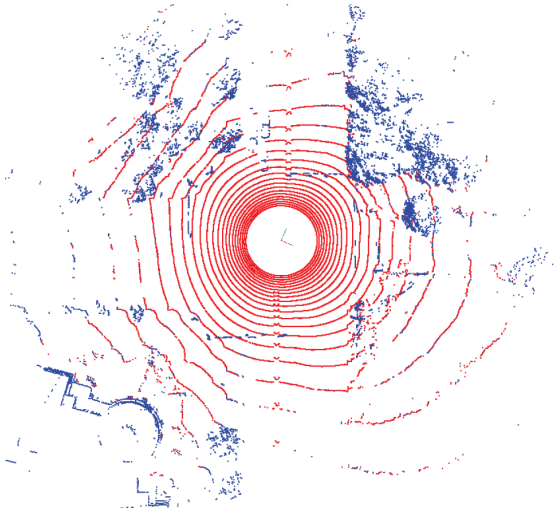

Abbildung 1.1.: Vogelperspektive auf eine Lidar-Punktwolke, wobei dir roten Punkte alle Boden- und die blauen alle Nicht-Bodenpunkte darstellen

1.1. Motivation

In einem bislang aufwändigen Verfahren müssen diese Punktwolken so aufbereitet und mit einer Bedeutung versehen werden, dass die Algorithmen an diesen Beispielen lernen, Unterscheidungen und Erkennungen selbst durchführen zu können und dabei möglichst wenige Fehler machen. Dabei kommen aktuell Softwarewerkzeuge zum Einsatz welche die Sensordaten für einen menschlichen Benutzer visualisieren und dieser sie dann manuell mit unterschiedlichen Bedeutungen versehen kann. Wegen der großen Menge an Daten, die für das Trainieren der Algorithmen benötigt wird, muss dieses manuelle Annotieren möglichst effizient durchführbar sein.

Ein zweidimensionales Kamerabild kann sehr einfach auf einem Computermonitor dargestellt und bearbeitet werden. Eine besondere Herausforderung stellt jedoch die Verarbeitung von 3-D-Daten, wie einer Punktwolke, dar. Hier stößt man mit den Möglichkeiten eines zweidimensionalen Computermonitors schnell an die Grenzen einer effizienten Darstellung und Bearbeitung. So ist es beispielsweise für die Annotierung solcher Punktwolken und ihrer Einzelmesswerte schwierig, die optimale Perspektive auf die Daten zu finden, die eine effiziente Erfassung durch den Menschen und eine entsprechende Bearbeitung zulässt. Dies erfordert von den Benutzern sehr viel Übung und Erfahrung, durch entsprechende Drehungen der Punktwolkendarstellung sich in dieser zurechtzufinden und Messpunkte realen Objekten zuzuordnen.

Eine alternative dazu soll diese Masterarbeit bieten, welche eine Applikation beschreibt, die das Visualisieren und Annotieren von Punktwolken einfacher und effizienter gestaltet. Mittels neuartiger Technologien wie Virtual- und Augmented Reality soll die Grenze zwischen 2-D und 3-D aufgehoben werden, sodass man sich als interaktiver Teilnehmer im dreidimensionalen Raum durch die Sensordaten navigieren, mit ihnen interagieren und sie annotieren kann.

1.2. Ziel der Arbeit

Ziel dieser Arbeit ist es zunächst ein passendes Medium für eine solche Applikation zu finden, das nicht nur die Anforderungen der Applikation selbst erfüllt, sondern auch an einem handelsüblichen Büro-Arbeitsplatz verwendet werden kann. Anschließend soll eine Applikation erstellt werden die folgende Grundfunktionalität erfüllt:

1. Mit der Anwendung soll es Möglich sein ein, bei CMORE gängiges, Datenformat einzulesen und alle nötigen Informationen daraus zu extrahieren
2. Aus den extrahierten Informationen soll eine Punktwolke generiert und innerhalb des gewählten Mediums visualisiert werden.
3. Der Benutzer muss die Möglichkeit haben durch die Punktwolke zu navigieren.
4. Die Applikation muss die Möglichkeit bieten jeden einzelnen Punkt mit einer Klassifikation versehen zu können. Dabei sollen Alleinstellungsmerkmale des gewählten Mediums benutzt werden, um eine vorzeigbare Verbesserung gegenüber der herkömmlichen Computer-Monitor-Annotation zu generieren.
5. Alle getätigten Annotationen müssen in die eingelesenen Dateien zurückgeschrieben bzw. in neue Dateien exportiert werden.

Anschließend sollen die Basisfunktionen erweitert und zusätzliche Funktionen hinzugefügt werden. Abhängig von der verbleibenden Zeit soll die Anwendung gemäß folgender Priorität erweitert werden:

1. Es sollen neue Möglichkeiten zur Annotation von Punkten implementiert werden
2. Die Möglichkeit, eigene Klassifikationen für Punkte zu erstellen, soll

Am wichtigsten wäre hierbei die Applikation um effiziente Möglichkeiten zu erweitern um Punkte zu labeln. Außerdem sollte es für den Benutzer möglich sein individuelle Klassifikationen anzulegen, um unterschiedliche Anwendungsgebiete abzudecken. Optional sind neue Navigationsmöglichkeiten und einlesbare Datenformate.

2

Vorbereitungen

Kurze Einleitung was alles vorbereitet wurde.

2.1. Wahl des passenden Mediums

Um das Labeling von Daten, insbesondere Punktwolken, im dreidimensionalem Raum zu verwirklichen, gibt es zwei wesentliche Technologien die dafür relevant erscheinen. Diese sind Augmented Reality (AR) und Virtual Reality (VR).

Unter Augmented Reality versteht man die direkte oder indirekte Sicht auf die reale, physische Welt, welche durch digitale Inhalte erweitert wird. Dies wird vor allem durch Smartphones oder AR-Brillen realisiert. Bei einem Smartphone hat man beispielsweise einen indirekten Blick auf die reelle Umgebung durch das Display, welches ein Live-Bild der Kamera anzeigt. Diesem Bild können nun digitale Inhalte hinzugefügt werden. Bei einer Brille (vgl. Abbildung 2.1) sieht man durch die Gläser direkt auf die physische Welt. Hierbei fungieren die Gläser als Display, welche in der Lage sind holographische Objekte anzuzeigen. Dies führt zu einer, für den Benutzer, sehr immersiven Vermischung der realen und der digitalen Welt. Üblicherweise ist es bei diesen Brillen sogar möglich durch Gesten, die mittels Händen durchgeführt werden, mit den Hologrammen zu interagieren.

Virtual Reality ist eine Technologie bei der spezielle VR-Brillen zum Einsatz kommen, welche keine Sicht mehr auf die reale Umgebung ermöglichen (vgl. Abbildung 2.2). Das Blickfeld des Menschen wird hierbei komplett von einem Display abgedeckt, das sich im Inneren der Brille befindet. So kann dem Benutzer eine komplette virtuelle Welt angezeigt werden. Mittels kompatiblen Controllern (vgl. Abbildung 2.2b) kann sich durch diese Welt bewegt bzw. mit ihr interagiert werden. Die Controller werden dabei von einem Sensor erfasst und somit können reale Bewegungen der Controller in die virtuelle Welt der Brille übersetzt werden.

2.1.1. Eignung von Augmented Reality und Virtual Reality im Bezug auf 3D-Datennotation

Im folgenden Abschnitt wird auf die positiven und negativen Aspekte der beiden Technologien eingegangen, um abzuwägen welche besser für 3D-Labeling geeignet ist.

Eignung von Augmented Reality

Der, im Rahmen dieser Arbeit, entwickelte Prototyp zur 3D-Datennotation ist nicht nur als Nutzungssoftware geplant, sondern dient auch als Anschauungsmaterial, beispielsweise für Messen. Zieht man diese Tatsache in Betracht, eignet sich Augmented Reality sehr gut für diesen Zweck. Es wirkt durch die Vermischung von realen und digitalen Inhalten nämlich sehr futuristisch. Zudem ist das direkte einblenden holographischer Inhalte, durch eine AR-Brille, zum Zeitpunkt der Erstellung dieser Arbeit, noch wenig verbreitet (Microsoft Hololens wurde nur wenige Tausend mal verkauft [1]), was bei unwissenden Benutzern zu einem beeindruckenden Effekt führt. Des Weiteren bietet die Augmented Reality Technologie viele, für zukünftige Labeling-Methoden relevante, Möglichkeiten. Den heutigen AR-Brillen ist es durch Tiefenkameras möglich Objekte und deren Distanzen zur Brille wahrzunehmen. Folglich könnte ein zukünftiger Ansatz sein, eine Punktwolke der Umgebung, wie sie in 3.4 zu sehen ist, mit einer AR-Brille zu erstellen. Diese kann anschließend vor Ort, in der Umgebung in der die Punktwolke erstellt wurde, klassifiziert werden.

Ein wichtiger Punkt der ebenfalls betrachtet werden muss ist die Bedienung der Brille bzw. die Interaktion mit den digital dargestellten Objekten und Informationen. Die Steuerung von AR-Brillen erfolgt handelsüblich über Gesten, welche mit der Hand bzw. den Händen getätigt werden. Ob diese Art der Interaktion für den hier gewünschten Anwendungsfall passend wäre, lässt sich im Voraus schwer feststellen. Denkbar wäre, dass sich der Benutzer durch intuitive Gesten, die man auch bei realen Objekten nutzt, schnell an die Bedienung des Tools gewöhnen würde. Andererseits könnte die Selektion der Elemente einer Punktwolke zu ungenau sein, da die Brille die Position der Hand nicht richtig interpretiert bzw. diese nicht richtig zur Position der digitalen Inhalte interpretiert. Eine genaue Einschätzung der Bedienung für solch einen Anwendungsfall kann allerdings nur gegeben werden indem man einen Labeling-Prototypen erstellt und ihn anschließend über längere Zeit testet. Dies ist im Zeitrahmen einer Masterarbeit jedoch nicht möglich. Falls sich nämlich die Steuerung als ungeeignet herausstellt bleibt nicht genug Zeit für eine Neuentwicklung auf einer anderen Plattform.

Darüber hinaus gibt es natürlich auch Aspekte die gegen die AR-Technologie sprechen. Die Klassifizierung einer Punktwolke mit dieser Technik ist an einem normalen Büro-Arbeitsplatz nicht möglich, zumindest nicht in einem Maßstab in dem es sinnvoll wäre. Die holographischen Punkte kollidieren, wenn sie darauf programmiert sind, mit der Umgebung und so würde Darstellung des Umfeldmodells verfälscht werden. Auch wenn sie das nicht

tun, dann wäre die Umgebung selbst immer noch ein Hindernis für den Benutzer, denn man möchte sich ja durch die Punktwolke bewegen. Des Weiteren sind bei AR-Brillen die vorherrschenden Lichtbedingungen ein großer Faktor, welche die Funktionalität gewisser Anwendungsfälle beeinflussen können. Wird beispielsweise eine Lichtquelle zu sehr von einem Objekt reflektiert, kann dieses von der Tiefenkamera der Brille nicht mehr richtig erfasst werden. Somit stimmt das Tiefenkamera nicht mehr mit der Realität überein. Zudem wird nicht nur die Funktionalität von Lichteinflüssen gestört, sondern auch die Darstellung der Hologramme. Bei zu viel Licht sind diese deutlich schwerer zu erkennen, vergleichbar mit der Nutzung eines Laptops im Freien, bei dem der Bildschirminhalt wegen zu heller Umgebung ebenfalls schlecht erkennbar ist. Für die Entwicklung selbst ist es ebenfalls hinderlich das es, zum Zeitpunkt der Erstellung dieser Arbeit, wenig Dokumentationen und Hilfestellungen, zum Beispiel in Entwicklerforen, gibt. Dies ist jedoch üblich für Technologien, die noch nicht lange auf dem Markt sind. Zu guter Letzt ist noch der finanzielle Faktor zu berücksichtigen. Durch die Aktualität der Technologie ist diese auch sehr teuer. Der Preis für eine AR-Brille kann dabei den Betrag von 5.000 Euro überschreiten, wie im Abschnitt 2.1.2 näher erläutert wird.

Eignung von Virtual Reality

Die Virtual Reality Technologie bietet den großen Vorteil, beliebig große Räume virtuell begehbar zu machen, ohne sich in der realen Welt selbst bewegen zu müssen. Dies ermöglicht die Bewegung durch Punktwolken, die im dreidimensionalen Raum dargestellt sind, an jedem üblichen Arbeitsplatz eines Büros. Des weiteren wirkt die Steuerung der VR-Brillen mittels den zugehörigen Controllern zum teil sehr ausgereift. Die Positions- und Bewegungserkennung des Benutzers wird als „*äußerst präzise*“ beschrieben [2]. Dies ist sehr wichtig um die Handhabung der Anwendung für den späteren Endnutzer so angenehm wie möglich zu machen. Für die Entwicklung von VR-Applikationen selbst ist zu sagen, dass es mittlerweile sehr Umfassende Dokumentationen, Anleitungen und Beiträge zu den jeweiligen Plattformen und Software Development Kits (SDKs) gibt. Dies hängt damit zusammen, dass sich die Virtual Reality Technologie schon etwas länger auf dem Markt befindet und die Zahl der Entwickler für VR-Applikationen stetig steigt. Dies ist vermutlich nicht zuletzt der Tatsache zu verdanken, dass die Preise für VR-Brillen gesunken sind. Wie im späteren, ausführlicher beschreibenden, Abschnitt 2.1.2 gezeigt, beträgt der aktuelle Preis einer Oculus Rift 449 Euro. Sowohl die zahlreichen Informationen für Entwickler als auch der niedrige Preis der Hardware sprechen, neben den zuvor genannten Aspekten, für die Wahl von Virtual Reality als Plattform für die Entwicklung von 3D-Labeling.

Was mit der VR-Technologie nicht ohne Weiteres funktioniert, ist die physische Begehung einer, zu klassifizierenden, Punktwolke. Zwar bietet die Technik die Möglichkeit durch zusätzliche Sensoren die Bewegungen des Benutzers in die virtuelle Welt zu übersetzen, jedoch funktioniert dies nur auf begrenztem Raum und ist unmöglich an einem üblichen Büro-Arbeitsplatz durchführbar. Der reale begehbare Raum muss nämlich durch diese Sensoren abgesteckt werden und ist somit durch deren Reichweite und Genauigkeit begrenzt.

Ein weiter Negativaspekt ist, dass viele Benutzer von VR-Brillen über Übelkeit klagen. In manchen Tests sind es mehr als 50 Prozent aller Teilnehmer, vor allem wenn es sich um Frauen handelt [3].

2.1.2. AR-Brillen im Vergleich

Zum Zeitpunkt der Recherche über eine passende AR-Brille, gibt es zwei potentielle Geräte die in Frage kommen. Diese sind Microsofts Hololens und die Meta 2. Andere Brillen werden, aufgrund der im voraus erkennbaren Hindernisse, nicht berücksichtigt, wie beispielsweise die Google Glass, die ein viel zu kleines Display hat. Wiederum andere sind zu diesem Zeitpunkt noch nicht auf dem Markt wie das Project Aurora, welches ebenfalls von Google ist, oder die Brille castAR. Im Folgenden werden die zwei potentiellen Brillen, im Hinblick auf die Nutzung für 3D-Datennotation, näher analysiert.

Microsoft Hololens

Die Hololens-Brille von Microsoft (vgl. Abbildung 2.1a) punktet vorwiegend in Sachen Darstellung und räumlicher Interaktion, was aus eigener Erfahrung bekannt ist. Eingeblendete Hologramme wirken auf dieser Brille sehr stabil. Ein Ruckeln oder Nachpositionieren der digital dargestellten Inhalte wurde nie bemerkt. Ebenfalls gut funktioniert die Vermessung des Raums durch die Kameras der Brille. Hologramme interagieren dadurch sehr gut mit der physischen Umgebung, indem sie beispielsweise wie ein realer Gegenstand auf dem Tisch positioniert oder an die Wand gehängt werden können. Lediglich schlecht ausgeleuchtete Bereiche eines Raumes machen dieser Funktion Probleme, sodass die Brille beispielsweise eine Wand entdeckt wo gar keine ist. Die eben genannten Merkmale der Hololens, also eine stabile Darstellung und eine gute Einbettung einer Punktwolke in einen vorhandenen Raum, ist sehr wichtig um diese zuverlässig klassifizieren zu können und somit eine gute Methode für 3D-Datennotation zu entwickeln.

Die Gestensteuerung der Hololens wäre für den Prozess der Klassifizierung von dreidimensionalen Punktwolken eher ungeeignet. Durch Kopfbewegung müsste zum gewünschte Ziel navigiert und dieses dann mit einer Tipp-Geste in der Luft ausgewählt werden. Da dieser Vorgang schon beim Eingeben eines etwas komplexeren Passwortes Probleme bereitet, wäre er bei der Klassifikation von hunderten Punkten für den Benutzer sehr mühsam und langwierig. Ein weiter Punkt der negativ ins Gewicht fällt ist das kleine Sichtfeld der Brille. Tests sprechen hierbei von einer horizontalen Sichtweite die lediglich 30 bis 40 Grad beträgt [4]. Der Mensch kann dagegen horizontal fast 180 Grad wahrnehmen. Diese Eigenschaft würde den Benutzer daran hindern eine Punktwolke bzw. wenigstens einen Teilbereich dieser, als ganzes wahrzunehmen. Dadurch würde ein Großteil des Reizes verloren gehen, den die 3D-Datennotation bietet. Zudem ist die Hololens sehr teuer. Der Preis für die Development Edition beträgt 3.299 Euro und in der Commercial Suite-Variante kostet die Brille sogar 5.489 Euro [5]. Für eine prototypische Entwicklung die im Rahmen einer Abschlussarbeit angefertigt wird, ist das für ein Unternehmen ein sehr hoher Preis.

(a) Microsoft Hololens AR-Brille (b) Meta 2 AR-Brille

Abbildung 2.1.: Die zwei potentiellen AR-Brillen

Meta 2

Für die Meta 2-Brille (vgl. Abbildung 2.1b) spricht das hohe Sichtfeld, welches dem Hersteller nach 90 Grad beträgt und laut eines Tests „*hält, was es verspricht*“ [6]. Wie im vorherigen Abschnitt angemerkt, ist ein hohes Sichtfeld sehr wichtig um dem Benutzer eine immersive Sicht auf eine Punktwolke zu geben. Die Steuerung der Brille wirkt ebenfalls gut und intuitiv. So ist es möglich Hologramme durch einfaches Greifen mit einer Hand auszuwählen, ohne diese zwangsläufig mit Kopfbewegungen anzuvisieren. Dies würde das selektieren vieler holographischer Gegenstände nacheinander erleichtern. Der Preis der Meta 2 ist, im Gegensatz zur Hololens geringer. Jedoch ist 1,495 Dollar immer noch ein hoher Betrag [7].

Deutlich schlechter als die Hololens schneidet die Meta 2 bei der Performance ab. Hologramme die in der Luft schweben, wie es auch Punkte einer Punktwolke tun würden, wackeln häufig und bewegen sich zum Teil von der Stelle, wenn sich der Benutzer selbst an ihnen vorbei bewegt [7]. Auch der, schon angesprochene und gute Ansatz der Greif-Interaktion, funktioniert in der Praxis nicht tadellos. Wenn sich nämlich Hände, welche von der Meta 2 erkannt werden, im Blickfeld der Brille befinden funktioniert die stabile Darstellung der Hologramme noch schlechter als schon angemerkt, weshalb sich die Hologramme so nicht mehr zuverlässig greifen lassen [8]. Die Markierung vieler digitaler Elemente wird somit sehr erschwert.

2.1.3. VR-Brillen im Vergleich

Die, in dieser Arbeit, entwickelte Methode zur 3D-Datennotation ist vorwiegend für einen normalen Arbeitsplatz im Büro konzipiert, also einen Platz an dem man einen Desktop-Computer zur Verfügung hat. Deswegen macht es keinen Sinn VR-Brillen in Betracht zu ziehen die von einem mobilen Gerät betrieben werden (Samsung Gear VR), da diese deutlich weniger Leistung und eine schlechtere Bedienung haben als Geräte für einen Desktop-PC. Auch die Nutzung der Playstation VR ist nicht sinnvoll, da zum Betrieb eine Playstation 4 nötig ist, welche in den wenigsten Unternehmen vorhanden sein dürfte.

Für das Klassifizieren im dreidimensionalen Raum ist eine gute Darstellung und Bedienung notwendig. Bei VR-Brillen, die einen Desktop-PC als Recheneinheit haben wird beides geboten. Durch die hohe Leistung eines performanten Computers kann der dreidimensionale Raum hochauflösend dargestellt werden. Auch die genaue, für Spiele konzipierte,

Steuerung über Controller kommt einem Labeling-Tool zugute. Zum Zeitpunkt der Erstellung dieser Arbeit, gibt es zwei relevante VR-Brillen aus dieser Sparte, die Oculus Rift und die HTC Vive.

Oculus Rift

Mit 470 Gramm ist die Oculus Rift (vgl. Abbildung 2.2b) deutlich leichter als andere Modelle, wie beispielsweise die HTC Vive mit 600 Gramm oder die Playstation VR mit 610 Gramm [9]. Dies ist für längeres Arbeiten mit ihr sehr vorteilhaft, da schwere Brillen oft schon nach kurzer Zeit störend beim Tragen sind. Des weiteren lässt sich die Brille problemlos an jedem handelsüblichen Büro-Arbeitsplatz verwenden. „*Oculus empfiehlt, die Kameras im Abstand von zwei Metern nebeneinander zu platzieren – bei einem klassischen PC-Arbeitsplatz also links und rechts neben dem Monitor. Das funktioniert in der Praxis gut, man kann sich mit diesem Aufbau ungefähr jeweils einen Schritt in alle Richtungen bewegen*“ [10]. Auch in Sachen Steuerung ist die Oculus Rift anderen Modellen überlegen. Durch die sogenannten Touch Controller, mit denen man virtuell nicht nur Greifen, sondern auch Daumen und Zeigefinger bewegen kann, gelingen ihr „*deutlich feinere Bewegungen*“ als beispielsweise der HTC Vive [10]. Dies bietet für die Entwicklung einer Lösung zur Markierung vieler Objekte zahlreiche Möglichkeiten.

Was bei der Oculus Rift nicht so gut funktioniert, wie bei anderen Brillen ist das Roomscaling, also das Erfassen der Bewegungen des Benutzers durch einen größeren Raum. Hierfür gibt es bei der Rift zwei Methoden, einmal mit zwei Kameras (kleine Überwachungsfläche) die sich im Abstand von drei Metern in zwei Raumecken gegenüberstehen und eine mit drei Kameras (größere Überwachungsfläche). Beide Methoden funktionieren nicht einwandfrei und auch nicht so gut wie bei der HTC Vive [10].

HTC Vive

Die HTC Vive (vgl. Abbildung 2.2a) punktet mit dem ausgeklügelten Lighthouse Tracking System, welches in Zusammenarbeit mit VALVE entwickelt wurde [11]. Das System kann die Kopfbewegungen des Benutzers durch mehr als 70 Sensoren, welche sowohl in die Brille integriert als auch extra positioniert sind, auf bis zu ein zehntel eines Grades messen [12]. Auch die Controller werden durch dieses System erfasst, was zu einer genauen Übertragung der physischen Handbewegungen in die virtuelle Welt führt. Dies kommt der Auswahl zahlreicher Objekte, wie es bei der Klassifizierung von Punkten der Fall ist, sehr entgegen. Auch eine gute Darstellung ist dank zweier Displays gewährleistet, die jeweils eine Auflösung von 1080 auf 1200 Pixel haben. Bei diesen Displays kommt es weder zu Verzerrungen des Bildes („*lens distortion*“ [12]) noch zu Pixelfehlern. Ebenfalls nützlich, vor allem am Arbeitsplatz, ist die Bluetooth-Funktion der Vive. Damit ist es möglich sein Smartphone mit der Brille zu verbinden und somit eingehende Emails oder Ähnliches zu lesen und zu beantworten, ohne die Brille absetzen zu müssen.

(a) HTC Vive AR-Brille mit Controllern

(b) Oculus Rift VR-Brille mit Controllern

Abbildung 2.2.: Die zwei potentiellen VR-Brillen

Der große Unterschied zwischen Vive und Rift liegt bei der Bedienung durch die unterschiedlichen Controller. Hier hat die HTC-Brille das Nachsehen, weil sie durch Anatomie der Controller weniger Möglichkeiten zur Interaktion in der virtuellen Umgebung bietet. Die sogenannten Wands bieten nämlich keinerlei Funktionen die es ermöglichen eine Hand nachzuahmen, um beispielsweise einzelne Finger zu bewegen und somit virtuelle Objekte greifen zu können. Im Hinblick auf eine durchdachte Steuerung für die Auswahl und Markierung vieler Punkte in einer Punktwolke gibt es somit weniger Möglichkeiten diese zu entwickeln.

2.1.4. Zusammenfassung und Entscheidung

Für die Wahl des passenden Mediums, auf dem das dreidimensionale Labeling entwickelt werden soll, sind mehrere Faktoren entscheidend. Diese Faktoren ergeben sich danach, wie sehr sich etwas für die genannte Anwendung eignet. Zunächst ist es wichtig eine Punktwolke deutlich und nach Möglichkeit im richtigen Maßstab darzustellen. Hier kommt die Augmented Reality-Technologie, durch die Interaktion mit der Umgebung und der lichtempfindlichen Darstellung digitaler Inhalte, sicherlich an ihre Grenzen. In der virtuellen Realität dagegen kann eine solche Wolke problemlos bei jeder Bedingung angezeigt werden, da die reale Umgebung nicht berücksichtigt wird. Sogar der Maßstab kann gut eingehalten werden, da dem virtuellen Raum keine Grenzen gesetzt sind, sodass beliebig große Umfeldmodelle an jedem Arbeitsplatz dargestellt werden können.

Auch die Steuerung spricht für den Einsatz von Virtual Reality. Zwar ist das Konzept der Greif-Steuerung der Meta 2 ein guter Ansatz, da diese aber noch nicht zuverlässig und genau funktioniert, ist sie für die Selektion vieler Objekte nicht gut geeignet. Dagegen bietet die Steuerung durch Controller bei VR-Brillen deutlich mehr Möglichkeiten eine gute Bedienung für das gewünschte Tool zu entwickeln. Die Tasten eines solchen Controllers können vom Entwickler nämlich nach Wunsch programmiert werden wogegen die Gesten für AR-Brillen nicht verändert bzw. nicht neu erstellt werden sollten [13].

Zudem gibt es, durch die deutlich frühere Markteinführung, viel mehr Dokumentationen und Hilfestellungen im Bereich Virtual Reality, was die Entwicklung deutlich erleichtert.

Aufgrund der eben genannten Vorteile von VR gegenüber AR wird für die Entwicklung, im Rahmen dieser Arbeit, **Virtual Reality** verwendet.

Bleibt noch die Frage zu klären welche Brille verwendet werden soll. Diese Frage ist schwerer zu beantworten als die vorherige. Die zwei potentiellen VR-Brillen, also Oculus Rift und HTC Vive, sind beide geeignete Kandidaten. Die Vive punktet durch gutes Roomscaling und gutes Tracking der Brille und der Controller, wogegen die Oculus Rift guten Tragekomfort und gut durchdachte Controller-Bedienung bietet.

Wichtig für die Entwicklung einer 3D-Datennotation ist aber nicht, ob man sich physisch, also mittels Roomscaling durch die Punktwolke bewegen kann, sondern wie man die klassifizierbaren Punkte markieren bzw. auswählen kann. Da hierfür die Oculus Touch Controller mehr Möglichkeiten bieten wird für die Entwicklung eines VR-Labeling-Tools die **Oculus Rift** verwendet.

2.2. Wahl des passenden Rechners

Für die Berechnungen, die notwendig sind um Inhalte in der Oculus anzuzeigen, ist nicht die Brille selbst verantwortlich. Diese Aufgabe übernimmt ein extra Computer. Die Brille fungiert dementsprechend nur als Anzeigegerät. Auch die Sensoren sind an den PC angeschlossen um das Tracking der Brille und der Controller zu gewährleisten. Da vor allem die Grafikberechnungen sehr aufwendig sind, kann nicht jeder handelsübliche PC oder Laptop dazu verwendet werden eine solche VR-Brille zu betreiben.

Wichtig sind also zunächst die Basiskomponenten wie ein schneller Prozessor und eine leistungsstarke Grafikkarte. Aber auch ein schneller Arbeitsspeicher mit ausreichender Kapazität ist wichtig damit notwendige Daten so schnell wie möglich zur Verfügung stehen. Für die Wahl, welche Komponenten eingesetzt werden sollen, kann sich an den empfohlenen Spezifikationen des Herstellers orientiert werden (Oculus Rift Recommended System Specs [14]). Diese sind jedoch vorwiegend für den Endnutzer gedacht, bei dem es wichtig ist ein einzelnes Programm für die Brille auszuführen. Für die Entwicklung ist dagegen mehr Leistung notwendig. Soll zum Beispiel eine erstellte Software getestet werden, muss nicht nur die Software selber, sondern eventuell auch die Entwicklungsumgebung und zusätzliche Debugging- und Analyse-Tools ausgeführt werden. An dieser Stelle sollte also nicht gespart werden.

Sind die Basiskomponenten ausgewählt, sollte noch geprüft werden ob noch weiteres Zubehör wichtig ist um diese zu betreiben. Da gerade Prozessoren mit hoher Taktrate viel Wärme produzieren, muss stets für ausreichende Kühlung gesorgt werden. Bei der Rechner-Konfiguration für diese Masterarbeit wurden beispielsweise, neben einem hochwertigen Prozessorkühler, noch zwei weitere Gehäuselüfter verbaut, um die Abwärme aus dem inneren des Computers heraus zu leiten. Zu guter Letzt ist es wichtig ein passendes Mainboard auszuwählen, auf dem alle Komponenten verbaut werden. Für die Oculus Rift werden vier USB-Anschlüsse empfohlen, welche das Mainboard haben muss [14]. Ebenfalls muss das Board genug Platz bieten um die gewählten Komponenten zusammen anbringen zu können. Leistungsstarke Grafikkarten und Prozessorlüfter können zum Teil sehr groß

ausfallen. Auch für weitere Aufrüstungen sollte das Board genügend Platz bieten, falls es, zu einem späteren Zeitpunkt, notwendig wäre zusätzliche Komponenten (zweite Grafikkarte) hinzuzufügen. Unter Berücksichtigung der eben genannten Aspekte wurde folgende Rechner-Konfiguration für die Entwicklung verwendet:

Bezeichnung der Komponente	**Verwendete Komponente**
Prozessor:	Intel® Core™ i7-7700K
Grafikkarte:	Gainward GeForce GTX 1070 Phoenix
Mainboard:	ASUS PRIME Z270-A
Arbeitsspeicher(RAM):	HyperX DIMM 16 GB DDR4-2400 Kit
Festplatte:	Samsung 850 Pro 2,5" 512 GB
Netzteil:	Cooler Master G550M 550W
Prozessorlüfter:	Noctua NH-D9L
Gehäuselüfter:	2x Coolink SWiF2-1200 120x120x25
Gehäuse:	Cooler Master N300

Tabelle 2.1.: Rechner-Konfiguration für die Entwicklungen dieser Arbeit

2.3. Wahl der passenden Entwicklungsplattform

Für die Erstellung von Virtual Reality Software gibt es zwei Entwicklungsumgebungen, die gut für das Entwickeln von Virtual Reality geeignet sind, weil sie schon seit längerem VR-Support bieten und dies auf umfassende Weise. Diese sind Unreal Engine 4 und Unity Engine. Beide Plattformen bieten alles nötige um VR-Software zu entwickeln, weshalb die Wahl zwischen ihnen eher subjektiv, aufgrund der eigenen Bedürfnisse und Erfahrungen, zu fällen ist.

Im Allgemeinen kann man sagen, dass die Unreal Engine mehr Möglichkeiten für professionelles Game Engineering bietet, da viel Wert auf visuelle Qualität gelegt wird, beispielsweise durch integrierte Post Processing-Techniken und bessere Shader. Sie bietet neben dem Scripting auch die Gelegenheit, durch C++ Programmierung Module der Engine zu verändern bzw. neue hinzuzufügen. Das gestalten von User Interfaces ist mit dem UMG UI Designer ebenfalls gut gelöst [15].

Die Unity Engine bietet vor allem Anfängern einen leichten Einstieg in die Welt der Grafik- und Spieleprogrammierung. Im Editor können ganz einfach sogenannte Game Objects erstellt werden, denen dann Funktionalitäten und Eigenschaften angehängt werden können. Dies geschieht meist in Form von Skripten, welche oft schon vorgefertigt zur Verfügung stehen. Wenn dies nicht der Fall ist können diese angepasst oder neu erstellt werden. Die große Community der Unity Engine ist hierbei eine große Hilfe.

Entscheidung

Ein Vorteil den die Unity Engine, im Falle dieser Arbeit, bietet ist Nutzung von C# als Scripting-Sprache. Da C.LABEL (vgl. 3.4) ebenfalls in C# programmiert wurde können Komponenten, wie etwa das Einlesen von Daten in die Anwendung, leicht übernommen werden. Des Weiteren habe ich selbst schon, im Rahmen eines Universitätsprojekts, mit dieser Engine gearbeitet, weshalb ein längeres einarbeiten in die Entwicklungsumgebung nicht nötig wäre. Die Vorteile der Unreal Engine sind, wie schon erwähnt, die visuelle Präsentation und die dabei gebotene Performance. Diese sind vor allem für das erstellen aufwändiger Landschaften und deren Inhalten von Vorteil.

Für die Entwicklung der, in dieser Arbeit, angestrebten Anwendung spielen jedoch Dinge wie Grafik und beeindruckendes Leveldesign keine große Rolle. Viel wichtiger ist sowohl die Möglichkeit der Kompatibilität zur C.LABEL-Anwendung als auch die vorhandene Entwicklungserfahrung einer Engine. Gerade letzteres ist auf Grund des begrenzten Zeitrahmens einer Masterarbeit von Vorteil, da ohne große Einarbeitung mehr Zeit zum Entwickeln von Funktionalität und Optimierungen bleibt. Aus diesen Gründen wird im weiteren Verlauf dieser Arbeit die **Unity Engine** als Entwicklungsumgebung verwendet.

3

Grundlagen

Dieses Kapitel befasst sich mit einigen grundlegenden Themen, die mit dem Inhalt dieser Arbeit zu tun haben. Dies soll dem besseren Verständnis für die folgenden Kapitel dienen. Der Abschnitt 3.1 beispielsweise, befasst sich mit dem Prinzip des maschinellen Lernens auf Basis von neuronalen Netzen, was verdeutlichen soll warum die Daten der Punktwolken überhaupt gelabelt werden müssen. Anschließend wird die CMORE Inhouse Software C.LABEL in 3.4 vorgestellt um den aktuellen Stand der Technik im Bereich Punktwolkenannotierung zu zeigen. 3.2 stellt die Game Engine Unity vor, die benutzt wurde um die Virtual Reality Applikation zu erstellen.

3.1. Maschinelles Lernen

Der Ausdruck Maschinelles Lernen (engl.: *Machine Learning*) ist heutzutage ein Überbegriff für Methoden, die versuchen durch Optimierung eine Funktion zu erlernen. Bei solch einer Funktion kann es sich sowohl um eine simple binäre Entscheidung auf eine Fragestellung handeln als auch um komplexere Dinge wie das Übersetzen von Texten in eine andere Sprache, das Erkennen von Personen auf Bildern, oder eben das Klassifizieren von Objekten in einer Punktwolke, was für diese Arbeit relevant ist. Die Komplexität der Funktion bestimmt auch die zu verwendende Methode zur Lösung des Problems. Im Zuge des Azure Machine Learning Studios veröffentlichte Microsoft eine Graphik (vgl. Abbildung 3.1), mit der man den richtigen Lernalgorithmus für gewisse Predictive Analytics-Methoden bestimmen kann. Für die Objektklassifizierung im Automobilbereich ist es wichtig die Klasse eines Objekts mit hoher Genauigkeit aus einer großen Anzahl verschiedener Klassen zu bestimmen. Basierend auf der Graphik 3.1 ist die beste Methode dafür ein Künstliches Neuronales Netz(KNN). Die Benutzung solcher Netze ist üblich, wenn es um Klassifikation von vielen verschiedenen Dingen geht und soll im folgenden näher erläutert werden.

3.1.1. Künstliche Neuronale Netze

Das Prinzip der Künstlichen Neuronalen Netze ist, wie vieles in der Informatik, einer Struktur aus der realen Welt nachempfunden. Das komplexeste neuronale Netz ist wohl das

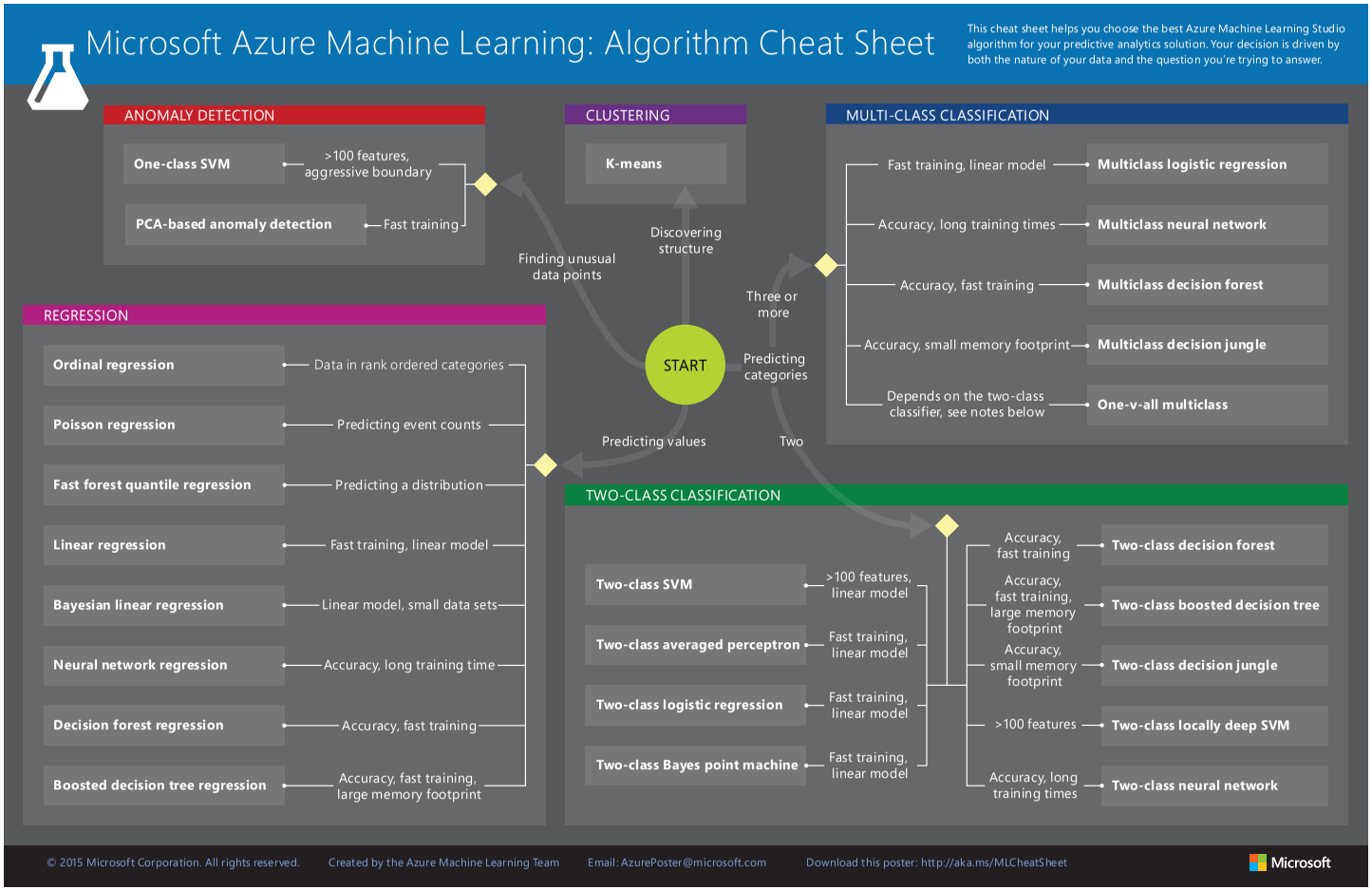

Abbildung 3.1.: Diagramm um den besten Machine Learning Algorithmus für eine Predictive Analytics-Methoden zu finden

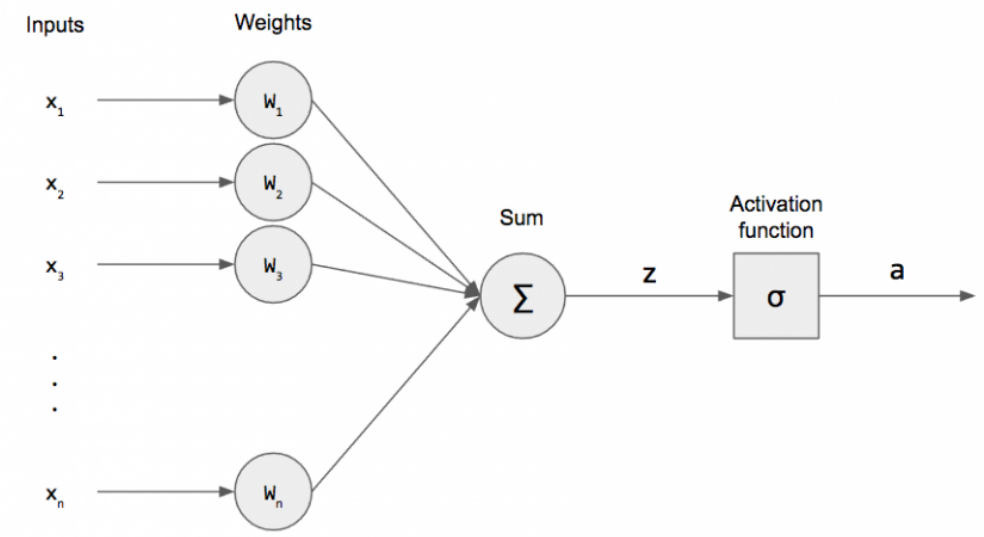

Abbildung 3.2.: Einfaches Perzeptron nach dem Prinzip von Frank Rosenblatt

menschliche Gehirn mit etwa 86 Milliarden Nervenzellen, welche auch Neuronen genannt werden. Daher kommt auch der Begriff des Neuronalen Netzes. Diese Nervenzellen sind durch Synapsen mit bis zu mehreren Tausend anderen Zellen verbunden. Diese Verbindungen bilden zusammen das Netz. Die Kommunikation zwischen den Neuronen erfolgt in der Regel über elektrische Impulse oder chemische Botenstoffe. Beides wird über Synapsen zu den nächsten, verbundenen Neuronen weitergeleitet und bewirkt dort einen Reiz. Bei der Übertragung von chemischen Botenstoffen kann dieser Reiz zusätzlich gewichtet werden, das heißt es kann ein stärkerer oder schwächerer Reiz ausgesendet werden. Alle ankommenden Reize werden dann am Empfangsneuron akkumuliert und ergeben den Eingangsreiz des Neurons. Dieses, vereinfacht dargestellte Prinzip, wird auch bei den KNNs verwendet. Übersteigt dieser Reiz eine bestimmte Schwelle kommt es zum entscheidenden Ereignis: Das Neuron *feuert*. Dadurch gibt das feuernde Neurone einen Impuls an alle anderen Neuronen weiter mit dem es verbunden ist. Bei dafür vorgesehenen Neuronen oder verbänden davon, kann dieser Impuls auch ein Ereignis auslösen. Diese Ereignisse können physischer Natur sein, also beispielsweise Bewegungen, oder auch psychischer Natur, zum Beispiel das Erkennen von Dingen. Werden diese Nervenzellen mit einem Ausreichenden Impuls angesprochen, wird das entsprechende Ereignis ausgelöst.

Die entscheidende Komponente dieses Prinzips ist die adaptive Gewichtung der weitergegebenen Reize. Adaptiv deswegen, da die Gewichtung der Reize sich während eines Menschenlebens verändern können. So erkennen Kinder oft Gefahrensituation nicht, da ihr Gehirn nicht ausreichend geschult ist um diese zu erkennen. In diesem Kontext bedeutet das, dass an die „Gefahren-Neuronen“ kein ausreichend starker Reiz gesendet wird. Durch sammeln von Erfahrungen optimiert sich dieser Wert, was in späteren Jahren zu einer anderen Reaktion des Gehirns führen kann als früher. Das adaptieren dieser Gewichte auf einen passenden Wert bezeichnen wir als Lernen. Um Computersystemen einen derartigen Lerneffekt zu ermöglichen wurde dieses Biologische Prinzip nun mathematisch imitiert.

Künstliche Neuronale Netze basieren auf dem Prinzip des Perzeptrons, das von Frank Rosenblatt 1958 vorgestellt wurde [16]. Dieses Perzeptron repräsentiert die Funktion eines Neuron auf eine mathematische Weise (vgl. 3.2). Die Eingangsdaten eines Perzeptrons bestehen aus einer n-großen Menge an Werten $x_1 \dots x_n$, welche eine gleichgroße Menge an jeweiligen Gewichten $W_1 \dots W_n$ haben. Wenn das zu analysierende Ziel beispielsweise ein Bild ist, wäre n die Menge aller Pixel des Bildes und bei Punktwolken eben die Anzahl der Punkte. Die Eingangswerte werden von einer mathematischen Funktion empfangen und zu einem skalaren Wert z zusammengefasst. Hierbei wird meist die Gewichtete Summe benutzt die folgendermaßen berechnet wird:

$$z = \sum_{i=0}^{n} W_i \cdot x_i \tag{3.1}$$

Im biologischen Vorbild repräsentiert diese Summe den eingehenden Reiz in eine Nervenzelle, die x-Werte sind dabei die Impulse der vorherigen Neuronen und die W-Werte repräsentieren die Gewichtung, also die Stärke der Eingangsreize. Der biologische Schwellwert wird im Perzeptron durch eine Mathematische Funktion $\sigma(z)$ dargestellt. Wie in [17] beschrieben gibt dafür es zwei beliebte Funktionen. Die *Fermifunktion* (3.2) , auch Logische Funktion genannt, welche einen Ausgabe-Wertebereich von (0,1) hat und den *Tangens Hyperbolicus* ($tanh(z)$) mit einem Wertebereich von (-1,1). Beide sind differenzierbar, was wichtig für das Lernen mit Backpropagation ist. Was das bedeutet wird in 3.1.2 näher erläutert. Wenn der Wert, den diese Funktion liefert, nicht noch durch eine weitere Ausgangsfunktion verändert wird, ist er der Ausgangswert des Perzeptrons.

$$\sigma(z) = \frac{1}{1 + e^{-}z} \tag{3.2}$$

In KNNs können mehrere dieser Perzeptrons eine sogenannte Schicht bilden. Je mehr Schichten ein neuronales Netz hat desto *tiefer* ist es, daher auch der Begriff *Deep Learning*, also tiefes Lernen. In der Regel bestehen KNNs aus mindestens 3 Schichten, nämlich einer Eingangsschicht, einer oder mehrerer Zwischenschichten und einer Ausgangsschicht.

Eingangschicht

Die Eingabeschicht ist der Startpunkt des Informationsflusses in einem künstlichen neuronalen Netz. Eingangssignale werden von den Neuronen dieser Schicht aufgenommen und gewichtet an alle Neuronen der ersten Zwischenschicht weitergegeben. Die Anzahl der Neuronen dieser Schicht hängt, wie schon erwähnt, von den Eingangsdaten ab. Werden Bilder mit einer Auflösung von 50x50 in das Netz gespeist, hat dieses Netz in der Regel 50x50, also 2500 Neuronen in der Eingangsschicht.

Zwischenschicht

Zwischen der Eingabe- und der Ausgabeschicht befindet sich in jedem künstlichen neuronalen Netz mindestens eine Zwischenschicht, die auch verborgene Schicht(engl.: *hidden layer*)

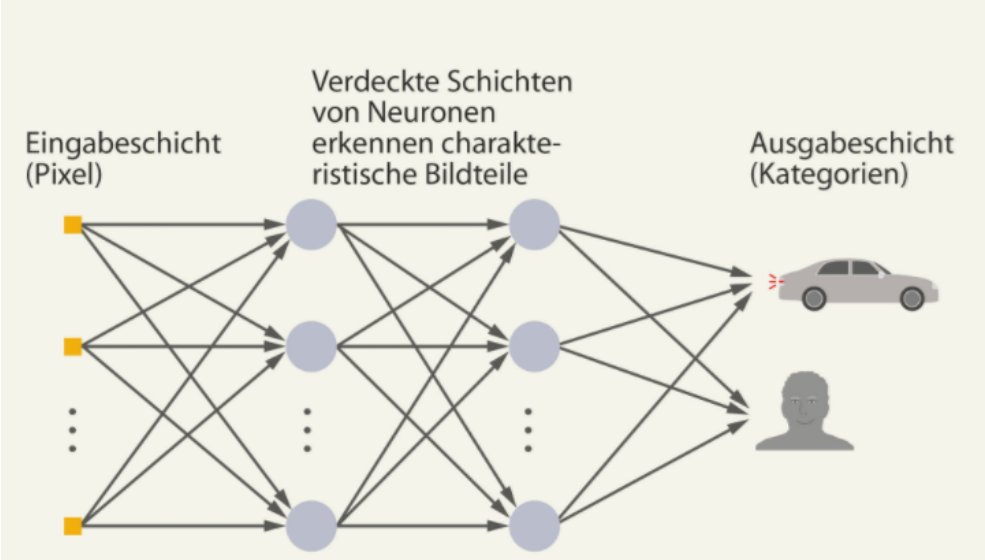

Abbildung 3.3.: Stark vereinfachte Darstellung eines KNNs zur Erkennung von Autos und Personen

genannt wird. Theoretisch ist die Anzahl der möglichen verborgenen Schichten in einem künstlichen neuronalen Netzwerk unbegrenzt. In der Praxis bewirkt jede hinzukommende Schicht jedoch auch einen Anstieg der benötigten Rechenleistung, die für den Betrieb des Netzes notwendig ist.

Ausgangsschicht

Die Ausgabeschicht liegt hinter den Zwischenschichten und bildet die letzte Schicht in einem Künstlichen Neuronalen Netz. In der Ausgabeschicht angeordnete Neuronen sind jeweils mit allen Neuronen der letzten Zwischenschicht verbunden. Die Ausgabeschicht stellt den Endpunkt des Informationsflusses in einem KNN dar und enthält das Ergebnis der Informationsverarbeitung durch das Netzwerk. Beim Beispiel der Objektklassifikation besteht die Ausgabeschicht nun aus n vielen Endknoten, wobei n die Menge an möglichen Klassifikationen ist. Die Eingangssumme dieser Knoten, den sie aus den vorherigen Neuronen bekommen, entsprechen der Wahrscheinlichkeit, dass es sich bei den Eingangsdaten um die jeweilige Klasse handelt. Abbildung 3.3 soll dieses Prinzip nochmals veranschaulichen.

3.1.2. Trainieren von Neuronalen Netzen

Lernen ist ein weiter Begriff. Es bedeutet, dass ein System sich in irgendeiner Form verändert, um sich z.B. an Veränderungen in seiner Umwelt anzupassen. Die typische Veränderung von künstlichen neuronalen Netzen ist das adaptieren der Verbindungsgewichte $W_1 \ldots W_n$. Diese Adaption geschieht nach Regeln, die man in Algorithmen definiert – ein Lernverfahren ist also immer ein Algorithmus, den man einfach mithilfe einer Programmiersprache implementieren kann. Diese Algorithmen lassen sich grob in 3 verschieden Arten einteilen, nämlich Unüberwachtes Lernen (*Unsupervised Learning*), Bestärkendes Lernen (*Reinforcement Learning*) und Überwachtes Lernen (Supervised Learning). Für Objektklassifizierung wird in der Regel letzteres verwendet, darum wird im Folgenden nur auf das Supervised

Learning eingegangen.

Der Lernprozess bei dieser Methode ist, wie der Name schon sagt, überwacht. Das kommt daher, dass das Netz von einer, vom Menschen vordefinierten, Menge an Trainingsbeispielen lernt. Genauer gesagt sind diese Beispiele eine Menge P an Eingabemustern mit jeweiliger korrekter Lösung, sodass das Netz nach der Ausgabe eine gewisse Fehlerbeschreibung zurückgeben kann. Für das Beispiel der Punktwolkenklassifizierung bestünde dabei das Trainingsset aus vielen verschiedenen Punktwolken, die alle ein Objekt darstellen, wobei die Art des Objektes, also die Lösung der Klassifizierung, bekannt ist.

Die Werte der Gewichte $W_1 \dots W_n$ werden bei einem untrainierten Netz zu Beginn zufällig gewählt. Dadurch gibt ein solches Netz in der Regel völlig falsche Lösungen für die ersten Trainingsbeispiele aus. Das ist aber nicht weiter schlimm, da nun der Lerneffekt einsetzen soll. Durch die Bekanntheit der Lösung des Beispiels kann der Fehler zwischen dem Ergebnis des Netzes und der gewünschten Lösung genau bestimmt werden. Dieser Fehler kann durch zurückrechnen auf jedes Neuron projiziert werden und so kann der Fehler, den jedes einzelne Neuron verursacht bestimmt werden. Daraus kann dann der Betrag errechnet werden, um den sich das Gewicht W_i des Neurons verändern muss, damit bei zukünftigen Eingaben bessere Ergebnisse erzielt werden. Dieses Zurückrechnen nennt sich *Backpropagation*. Für das Verständnis des Grundprinzips ist die Formel zu Berechnung der Gewichtsänderung nicht wichtig, sie wird aber zur Vervollständigung im Folgenden angegeben:

$$\triangle w_{ij} = -\eta \delta_j o_i \tag{3.3}$$

- $\triangle w_{ij}$ ist die Änderung des Gewichts w_{ij} der Verbindung von Neuron i zu Neuron j
- η ist ein fester Wert, der eine Lernrate ausdrücken soll, mit der die Stärke der Gewichtsveränderung festgelegt wird
- δ_j ist das Fehlersignal des Neurons j das sich aus der partiellen Ableitung $\frac{\partial E}{\partial net_j}$ ergibt
  - E ist dabei der quadratische Fehler der bei der Berechnung des Ergebnisses entstand, er wird durch die Formel $E = \sum_{i=1}^{n} (t_i - o_i)^2$ berechnet
    - t_i steht darin für den gewünschten Zielwert
  - net_j ist die Netzeingabe $\sum_{i=1}^{n} x_i w_{ij}$
- o_i steht sowohl bei der Berechnung von $\triangle w_{ij}$, als auch von E für die errechnete Ausgabe des Neurons i

Zusammenfassend kann man nun sagen, dass der Lernprozess eines künstlichen Neuronalen Netzes nach folgendem Schema abläuft:

1. **Eingabe** eines Elements p aus der Trainingsmenge P in die Eingabeschicht des Netzes

2. **Vorwärtspropagierung** der Eingabe durch das Netz gesamte Netz, sodass man bei der Ausgabeschicht ein Ergebnis erhält

3. **Vergleich** des Ergebnisses mit der richtigen Lösung und Berechnung des Fehlers

4. **Zurückpropagieren** des Fehlers, um somit die Gewichte der Neuronen anzupassen

Wie unschwer zu erkennen ist basiert der Backpropagation-Algorithmus auf dem Vorhandensein eines Trainingsdatensatzes. Damit das Netz nach dem Training für neue, unbekannte Eingaben die richtige Lösung errechnet ist es elementar wichtig, dass dieser Datensatz eine hohe Qualität und Quantität hat. Qualität bedeutet in diesem Fall, dass die Trainingsdaten repräsentativ für den jeweiligen Anwendungsfall sein müssen. Ein Personenerkennungsalgorithmus der im Straßenverkehr Personen identifizieren soll liefert keine guten Ergebnisse, wenn er ausschließlich mit Bildern von Personen in Innenräumen gefüttert wird. Mit Quantität ist Anzahl der verschiedenen Elemente im Trainingssatz gemeint. KNNs verbessern ihre Ausgabe in der Regel mit größerer Menge an unterschiedlichen Trainingsdaten.

Zur Erstellung dieses Datensatzes ist es nun nötig so viele repräsentative Daten wie möglich zu sammeln und diese mit der richtigen Lösung zu annotieren. Eine Methode für eine solche Annotierung wird in dieser Arbeit vorgestellt. Dieses Kapitel sollte nun veranschaulicht haben warum solche Methoden so essentiell, auf dem Weg zur Erschaffung einer künstlichen Intelligenz, sind.

3.2. Die Unity Engine

3.3. Lidar-Sensor

3.4. Das C.LABEL Tool

Beschreibung des C.Label Tools, was es kann, was in die VR übernommen werden soll und was deswegen für die weitere Auswahl an HW und Tools zu berücksichtigen ist

4

C.LABEL VR

Dieses Kapitel beschreibt die Virtual Reality Applikation **C.LABEL-VR** und ist somit der Hauptteil dieser Arbeit. Das Ziel dieser Applikation ist es die Annotierung von Punktwolken, wie sie beispielsweise in **C.LABEL**(siehe Kapitel 3.4) möglich ist, innerhalb einer dreidimensionalen Umgebung zu realisieren. Dazu müssen Datenformate, welche Informationen über Punktwolken enthalten, eingelesen und anschließend, aus diesen Daten, Punktwolken generiert werden. Der Vorgang des Imports wird in Kapitel 4.1 näher beschrieben, die Generierung der Punktwolken in 4.2. Um sich in der virtuellen Umgebung durch diese Wolken bewegen zu können, wurden diverse Möglichkeiten zur Navigation entwickelt. Auf die Funktion dieser Möglichkeiten und deren Auswirkungen auf das befinden des Menschen (*Virtual Motion Sickness*) wird in Kapitel 4.3 eingegangen.

Anschließend geht es um die Annotierung der generierten Punktwolken. Annotierung bedeutet in diesem Kontext, dass die einzelnen Punkte der Wolken mit bestimmten Klassifikationen versehen werden, welche der Art des Objektes entsprechen, dem die Punkte zugehören. Ist der Punkt beispielsweise Bestandteil eines Autos, so wird er mit der Klasse „Auto“ versehen. Um diese Aufgabe erfüllen zu können wurden mehrere Arten der Annotierung entwickelt. Welche dies sind und wie sie realisiert wurden, wird in Kapitel 4.4 gezeigt. Das letzte Kapitel 4.5 beschäftigt sich mit dem User Interface der Applikation. Dabei wird hauptsächlich auf die Funktionen des Ingame-Menüs eingegangen, welche Herausforderungen es bei der Erstellung von UIs in der virtuellen Realität gibt und wie diese bewältigt wurden. Zur Veranschaulichung der Funktionen von **C.LABEL-VR** ist in der Abbildung 4.1 der grobe Workflow in Form eines Diagramms abgebildet.

4.1. Import und Export von Daten

Umfeldmodelle, welche in Form einer Punktwolke vorliegen, werden meist von einem Radar-, oder, wie in Kapitel 3.3 gezeigt, von einem Lidar-Sesnor aufgenommen. Die Informationen über diese Modelle, zum Beispiel die Positionen der einzelnen Punkte, werden in einem passenden Datenformat abgespeichert. Um welches Format es sich dabei handelt kann sehr unterschiedlich sein. In der Regel hängt es davon ab zu welchem Zweck die

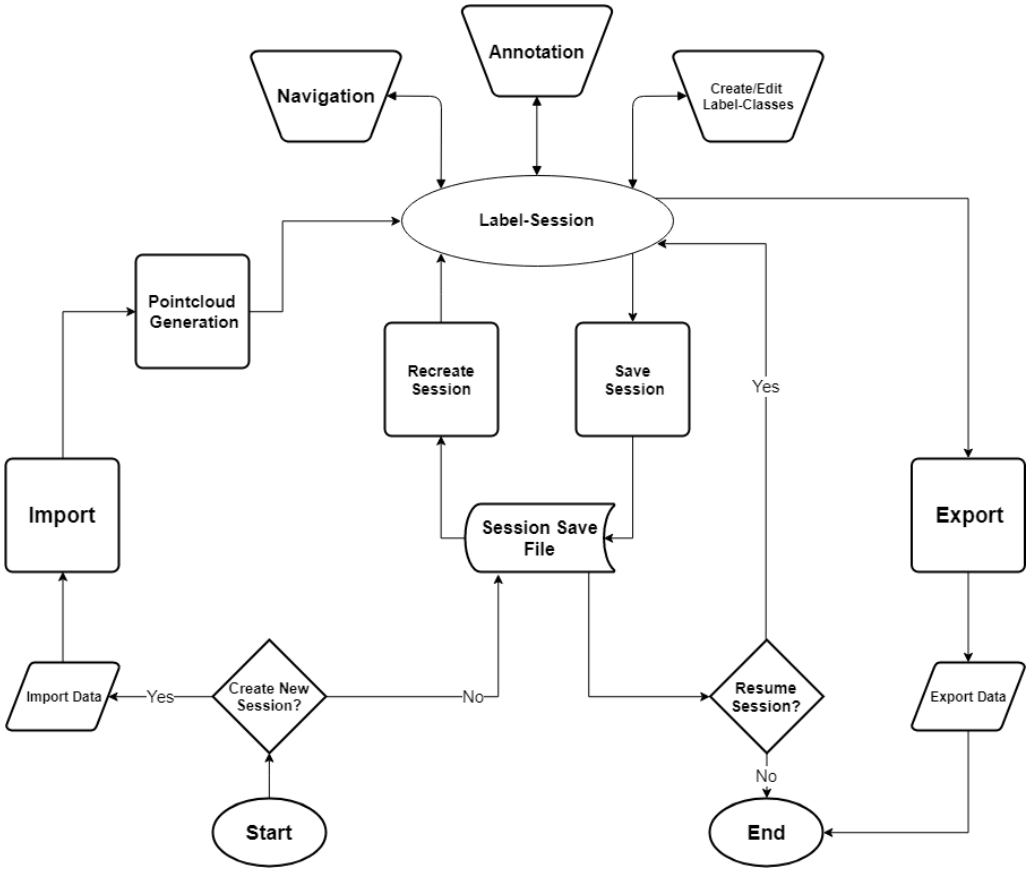

Abbildung 4.1.: in C.LABEL-VR vom Import bis zum Export

aufgenommenen Daten gebraucht werden und wie umfangreich diese Daten sind. Am häufigsten werden sie dafür verwendet um Lernalgorithmen und neuronale Netze (siehe 3.1.1) zu trainieren. Automobilhersteller haben für solche Vorhaben natürlich unterschiedliche Konzepte und somit können auch die Formate der Daten unterschiedlich sein. Um die Applikation **C.LABEL-VR** für mehrere dieser Hersteller benutzbar zu machen, muss das Programm also mit verschiedenen Datenformaten umgehen können.

Innerhalb dieser Arbeit wurden mit zwei verschiedenen Arten von Daten gearbeitet. Das erste ist das **PCD**-Format (*Point Cloud Data*). Die Open Source-Bibliothek **PCL**(*Point Cloud Library*) benutzt das Datenformat **PCD** um Informationen über Punktwolken zu verwalten. **PCL** ist eine sehr mächtige C++-Bibliothek, die in der Regel von jedem benutzt wird, der performante Algorithmen für Punktwolken entwickeln möchte. Auch **C.LABEL** (siehe 3.4) generiert aus eingelesenen Daten **PCD**-Files um Funktionen der Point Cloud Library für die Wolken zu benutzen.

Das zweite Format ist das, von der HDF-Group bereitgestellte, **HDF5**-Format (*Heterogeneous Data Format*). Es ist ausgelegt zur schnellen Erstellung und Auslesung von komplexen Datenstrukturen. Wie diese Strukturen aussehen wird später in Kapitel 4.1.2 näher erklärt. Zur schnellen Bearbeitung großer Datenmengen ist dies ein gängiges Format, wird von den meisten Kunden von **CMORE** benutzt und deshalb auch in **C.LABEL-VR** verwendet.

4.1.1. Architektur

Wichtig ist es also sowohl den Import, als auch den Export so modular wie möglich zu gestalten, dass er leicht um neue Datenformate erweitert werden kann. Das Prinzip dafür ist in Abbildung 4.2 dargestellt und wird im Folgenden näher erläutert. Die angesprochenen Erweiterungen um neue Datenformate werden als **Addons** bezeichnet. Jedoch handelt es sich dabei nicht immer nur um eine Erweiterung eine komplett neuen Datenformates. Komplexe Formate wie zum Beispiel HDF5, in denen der Entwickler die Struktur selbst festlegt, müssen für jede neue Struktur, auf unterschiedliche Weise eingelesen werden und brauchen somit auch separate **Importer-** und **Exporter-Addons**. Um die Erweiterbarkeit der Applikation sicherzustellen wurde der Import und der Export von der Hauptapplikation „abgekapselt". Dafür wurde eine interne Datenstruktur(IDS) eingeführt, welche alle nötigen Informationen enthält, die für das Labeling notwendig sind. Sie wird im späteren Kapitel 4.2.1 genauer vorgestellt. Der Vorteil dabei ist, dass man dadurch auf die gleiche Datenstruktur zurückgreifen kann und somit unabhängig vom importierten Datenformat ist. Wird also ein neues Import-Format eingeführt muss an der Hauptapplikation nichts verändert werden.

Jedes Import-Addon hat also zunächst die Aufgabe, seine jeweiligen Daten einzulesen und alle Informationen, die für die IDS notwendig sind, daraus zu extrahieren. Aus diesen Informationen können anschließend die entsprechenden Punktwolken generiert werden. Die Generierung wird in Kapitel 4.2.2 erklärt. Beim Export ist es notwendig, dass die Struktur der, vom Import-Addon eingelesenen, Daten gleich bleibt. Es sollen lediglich die Labeling-Informationen aus der IDS hinzu- bzw. eingefügt werden. Aus der IDS kann das Export-Addon diese Informationen extrahieren und anschließend in die eingelesenen Daten exportieren. Die importierten Daten werden so allerdings überschrieben. Möchte man das nicht, müssen die Daten neu erstellt und mit den Labeling-Informationen versehen werden.

Die Importierten Daten, beispielsweise **HDF5**, enthalten allerdings oft deutlich mehr Informationen, als für die interne Struktur notwendig. Diese Zusatzinformationen werden **Metadaten** genannt. Für den Export bedeutet dies, dass man aus der internen Struktur von **C.LABEL-VR**, die Struktur der anfangs eingelesenen Daten nicht wieder rekonstruieren kann, da die **Metadaten** sonst fehlen würden. Um dieses Problem zu lösen müssen die zusätzlichen Daten separat gespeichert werden. Dazu wurde eine erweiterbare Datenstruktur angelegt, in der man für jede Import-Struktur eine Metadatenstruktur anlegen kann. Das entsprechende Export-Addon kann die Metadaten dann verwenden um, mit Hilfe der internen Daten, das gewünschte Exportformat zu rekonstruieren.

Zusammenfassend ist also folgendes zu tun um ein neues Datenformat oder eine neue Datenstruktur einzulesen: Zunächst muss eine Importfunktion programmiert werden, die alle Informationen aus den gewünschten Daten ausliest. Anschließend muss diese Funktion das genormte, interne Datenformat(Kapitel 4.2.1) erstellen und alle nötigen Informationen aus den eingelesenen Daten darin speichern, damit daruas die Punktwolken generiert

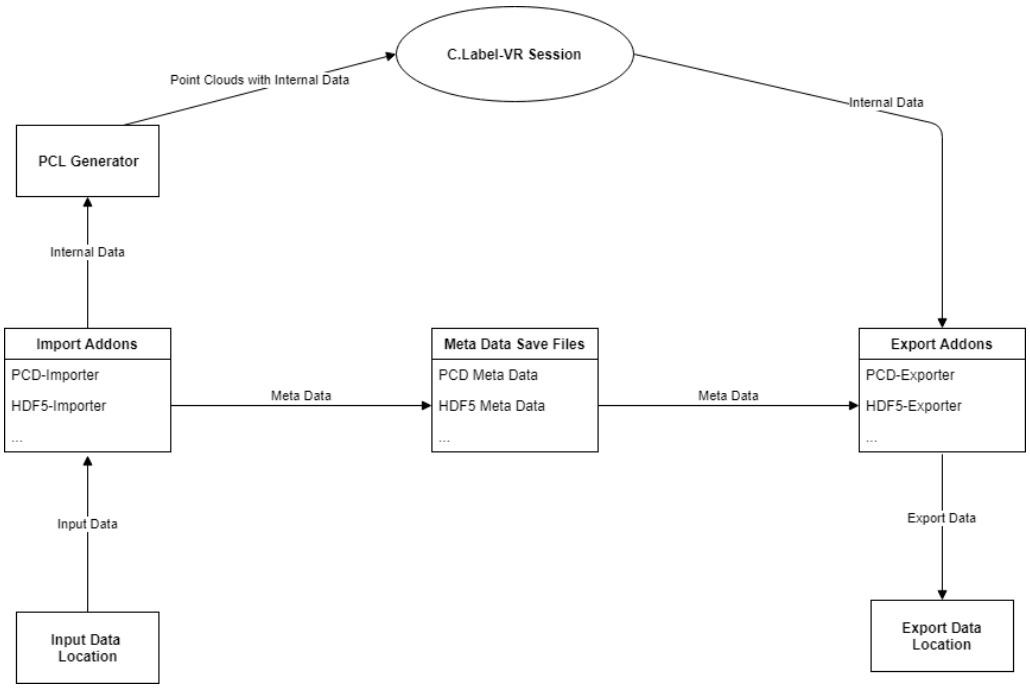

Abbildung 4.2.: Das Import-Export-Prinzip in C.LABEL-VR

werden können. Falls es Daten gibt, die über diejenigen im IDS hinausgehen muss eine Metadatenstruktur angelegt werden, in welche die zusätzlichen Daten abgelegt werden. Zuletzt ist eine Exportfunktion zu implementieren, die sowohl die Labeling-Informationen aus der IDS in die eingelesenen Daten exportieren kann, als auch aus den internen- und den Metadaten das Ausgangsdatenformat wiederherstellen kann, um die Daten als neue Files zu exportieren. Im folgenden Kapitel soll dieses Prinzip konkret an einem HDF5-Beispiel gezeigt werden.

4.1.2. HDF5-Beispiel

HDF5 Daten

Beispiel

4.2. Generierung einer Punktwolke

4.2.1. Interne Datenstruktur

4.2.2. Generierung

session pointcloud ...

Koordinatensysteme

4.2.3. Optimierung

-generell nichts teures in update mehthode

-gameobject find -getcomponent

keine update oder start mehtode bei skripten der punkte collider trigger nicht bei den punkten sondern beim anderen object generell alle pollenden aktionen von den punkten entfernen

(-clonen von prefabs statt neue instanzen von inbuilts) shared materials verwenden kein material.color weil das eine kopie des materials erzeugt, welches dann einzeln gecallt werden muss

gpu instancing bei den materials verwenden und forward rendering options

4.3. Navigation

4.3.1. VR-Krankheit

4.3.2. Freier Flug

4.3.3. Teleport

4.4. Annotieren der Punktwolke

4.4.1. Pointer Labeling

4.4.2. Clustering

ground plane fitting

4.5. Ingame Menu

4.5.1. Movement

4.5.2. Labelclasses

5

Schluss

5.1. Zusammenfassung

5.2. Herausforderungen

5.3. Ausblick

A

Appendix Title

Literaturverzeichnis

[1] P. Gothard, "Microsoft hololens: Firm admits sales figures are in the 'thousands'," Internetquelle (abgerufen am 24.11.2017), Jan. 2017. [Online]. Available: https://www.theinquirer.net/inquirer/news/3003380/microsoft-hololens-firm-admits-sales-figures-are-in-the-thousands

[2] R. Berg, "Touch-controller machen oculus rift zur besten vr-brille," Internetquelle (abgerufen am 24.11.2017), Dec. 2016. [Online]. Available: https://www.welt.de/wirtschaft/webwelt/article160455301/Touch-Controller-machen-Oculus-Rift-zur-besten-VR-Brille.html

[3] J. Munafo, M. Diedrick, and T. A. Stoffregen, "The virtual reality head-mounted display oculus rift induces motion sickness and is sexist in its effects," *Experimental Brain Research*, vol. 235, no. 3, pp. 889–901, dec 2016.

[4] J.-K. J. Hannes A. Czerulla, "Microsoft hololens im test: Tolle software, schwaches display," Internetquelle (angerufen am 27.11.2017), Jun. 2016. [Online]. Available: https://www.heise.de/ct/artikel/Microsoft-HoloLens-im-Test-Tolle-Software-schwaches-Display-3248670.html

[5] Microsoft, "Microsoft hololens choose the option that best suits your purpose." Internetquelle (abgerufen am 27.11.2017), Nov. 2017. [Online]. Available: https://www.microsoft.com/de-de/hololens/buy

[6] T. Kammann, "Augmented reality: Ar-brille meta 2 im test – besser als hololens?" Internetquelle (abgerufen am 27.11.2017), Nov. 2017. [Online]. Available: https://vrodo.de/augmented-reality-ar-brille-meta-2-im-test-besser-als-hololens/

[7] Metavision, "Meta 2 augmented reality development kit," Internetquelle (abgerufen am 27.11.2017), Nov. 2017. [Online]. Available: https://buy.metavision.com/products/meta2?hsCtaTracking=f279363d-13d6-49b1-ac86-3b857cdab7af%7Ce791c1c8-d612-476a-9ed0-d31641c628f1

[8] J. Durbin, "Meta 2: Hands-on with the ar startup's latest kit," Internetquelle (abgerufen am 27.11.2017), Apr. 2017. [Online]. Available: https://uploadvr.com/meta-2-hands-ar-svvr/

[9] J.-K. Janssen, "Oculus rift im test: Virtual reality für die massen," Internetquelle (abgerufen am 27.11.2017), Mar. 2016. [Online]. Available: https://www.heise.de/ct/artikel/Oculus-Rift-im-Test-Virtual-Reality-fuer-die-Massen-3151909.html

[10] ——, "Oculus touch im test: So echt können sich hände in vr anfühlen," Internetquelle (abgerufen am 27.11.2017), Dec. 2017. [Online]. Available: https://www.heise.de/ct/artikel/Oculus-Touch-im-Test-So-echt-koennen-sich-Haende-in-VR-anfuehlen-3552295.html

[11] R. Artus, "Lighthouse von valve erklärt," Internetquelle (abgerufen am 28.11.2017), Nov. 2017. [Online]. Available: https://vrjump.de/lighthouse-erklaert

[12] L. Painter, "Htc vive review," Internetquelle (abgerufen am 28.11.2017), Aug. 2017. [Online]. Available: http://www.techadvisor.co.uk/review/wearable-tech/htc-vive-review-2017-3635648/

[13] W. M. R. D. Forum, "Creating a new gesture," Internetquelle (abgerufen am 28.11.2017), Apr. 2016. [Online]. Available: https://forums.hololens.com/discussion/549/creating-a-new-gesture

[14] O. R. Support, "Was sind die empfohlenen mindestsystemvoraussetzungen, die für den betrieb der oculus rift erforderlich sind?" Internetquelle (abgerufen am 29.11.2017), Nov. 2017. [Online]. Available: https://support.oculus.com/170128916778795/

[15] P. Gora and L. Leibetseder, "Unreal vs unity: Ein vergleich zwischen zwei modernen spiele-engines," Favoritenstrasse 9-11/186, A-1040 Vienna, Austria, Oct. 2016, 1. [Online]. Available: https://www.cg.tuwien.ac.at/research/publications/2016/Przemyslaw_Gora_2016_UVU/

[16] C. V. D. Malsburg, "Frank rosenblatt: Principles of neurodynamics: Perceptrons and the theory of brain mechanisms," in *Brain Theory*. Springer Berlin Heidelberg, 1986, pp. 245–248.

[17] D. Kriesel, *Ein kleiner Überblick über Neuronale Netze*, 2007. [Online]. Available: erhältlichaufhttp://www.dkriesel.com