INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Masterarbeit

Evaluation der Nutzung von Neuro-Evolutionären Algorithmen bei Level-Of-Detail Verfahren für 3D-Echzeitvisualisierungen

René Martin

INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Masterarbeit

Evaluation der Nutzung von Neuro-Evolutionären Algorithmen bei Level-Of-Detail Verfahren für 3D-Echzeitvisualisierungen

René Martin

Aufgabensteller:	Prof. Dr. Dieter Kranzlmüller
Betreuer:	Markus Wiedemann
Abgabetermin:	21. März 2019

Hiermit versichere ich, dass ich die vorliegende Masterarbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den 21. März 2019

. . . . . .
(Unterschrift des Kandidaten)

Abstract

When rendering complex 3D visualizations with numerous objects in real time, only a fixed amount of calculation time is available. Optimizations like using lower levels of datail for less important objects can help to keep calculation costs manageable. Different Level-Of-Detail algorithms are available. An adaptive approach described by Funkhouser and Séquin can ensure interactive frame rates [FS93]. This approach performs an evaluation about the calculation cost and benefit of each object/Level-Of-Detail pair.

Within this thesis a new way of calculating the benefit for this algorithm is introduced. The factors described in the original paper are weighted dynamically by the outputs of an artificial neural network which is trained with a neuroevolutionary algorithm. For the learn process of the artificial neural network its perfomance is evaluated by comparing a calculated image based on the network outputs with an optimal image.

First fundamentals and related work on the topics of Level-Of-Detail algorithms, artificial neural networks and color models are shown. Later the concept of the new benefit formula and a basic implementation are described. Finally a comparison between the resulting and Funkhousers and Séquins algorithms is shown.

Zusammenfassung

Um aufwändige 3D-Visualisierungen mit vielen Objekten in Echtzeit darstellen zu können, sind aufgrund begrenzter Rechenkapazität häufig Optimierungen notwendig. Eine mögliche Optimierung ist die Verwendung geringerer Detailstufen für Objekte, die nur in geringem Umfang zum Bildeindruck beitragen. Zur Auswahl, welche Detailstufe zu welchem Zeitpunkt verwendet werden soll, gibt es verschiedene Ansätze, wie z.B. einen adaptiven Ansatz für interaktive Anzeigezeiten von Funkhouser und Séquin [FS93]. Dieses Verfahren wägt die Rechenkosten aller Objekt-Detailstufen-Kombinationen gegen den Bildeindrucks-Nutzen ab.

Ziel dieser Arbeit ist es, die Berechnung des Bildeindruck-Nutzen zu optimieren, um mit dem Algorithmus in der gegebenen Rechenzeit bessere Bilder erzeugen zu können. Dafür sollen die ursprünglich willkürlich gewählten Faktoren durch ein künstliches neuronales Netz gewichtet werden. Dieses wird durch einen evolutionären Algorithmus generiert. Dabei wird die Performanz eines neuronalen Netzes danach beurteilt, wie nah ein Bild, welches mit der vom Netz gegebenen Gewichtung erzeugt wird, an einem Originalbild liegt.

In der Arbeit werden Grundlagen der verschiedenen Aspekte dargestellt, die in der neuen Berechnungsmethode relevant sind. Anschließend werden Details einer beispielhaften Implementierung besprochen und abschließend mit dem bisherigen Algorithmus von Funkhouser und Séquin verglichen.

Inhaltsverzeichnis

1 Einleitung

3D-Echtzeitvisualisierungen sind aus der modernen Informatik nicht mehr wegzudenken – von Flug-Simulationen über Datenvisualisierungen bis hin zu Unterhaltungsapplikationen, wie z.B. Computer- oder Konsolenspiele. Zusammen mit der Zahl der Anwendungsbereiche ist in den letzten Jahrzehnten auch die Komplexität der dargestellten Szenen und somit die Anforderung an die Hardware gestiegen.

Betrachtet man beispielsweise das Forschungsfeld der Pyramiden von Gizeh, so findet man bereits Modelle für Simulationen und wissenschaftliche Betrachtungen des gesamten Plateau aus den 90er-Jahren [JCS]. Bei diesen Darstellungen werden die Bauwerke auch geometrisch als Pyramiden mit einem Dreieck pro Seite dargestellt. Im Gegensatz dazu steht das 2017 veröffentlichte Videospiel *Assassins Creed: Origins*. Ein 3D-Künstler der Entwicklerstudios Ubisoft Sofia gab an, dass darin die Pyramiden aus einzelnen Modulen von Steinblöcken mit der Größe von $6m \times 6m$ bestehen [Zve]. Bereits ein einziges dieser Module besteht aus ungefähr 4000 Dreiecken. Damit die Realitätsgetreue einer Szene und somit deren Qualität wächst, ist die nächstliegendste Herangehensweise, die dargestellten Objekte detaillierter zu modellieren.

Wächst jedoch die Detailgetreue der dargestellten Objekte oder auch die Anzahl der Objekte zu stark an, so ist es bei gleichbleibender Hardwarekapazität ab einem bestimmten Punkt nicht mehr möglich, eine gewisse Berechnungszeit pro generiertem Bild bei der 3D-Echtzeitvisualisierung sicherzustellen. Dadurch entsteht eine kurzfristige Unterbrechung des Bildflusses und die Wahrnehmung des Nutzers wird negativ beeinflusst. Um diesen Effekt zu verhindern, gibt es diverse Ansätze zur Reduzierung der Berechnungszeit pro generiertem Bild. Beispielsweise kann man verschiedene Formen des Cullings, wie Backface Culling oder Frustum Culling, anwenden. Durch erstere werden Dreiecke, welche sich – von der Kamera gesehen – auf der Rückseite von Objekten befinden, vom Berechnungsprozess ausgeschlossen. Beim Frustum Culling verzichtet man auf die Berechnung von Dreiecken, welche vollständig außerhalb des Sichtfeldes der Kamera liegen. Selbst wenn man jedes im finalen Bild nicht darzustellende Dreieck durch solche Verfahren ausschließt, kann die Anzahl der tatsächlich sichtbaren Dreiecke enorm sein. Somit ist es möglich, dass trotz dieser Verfahren der Berechnungsaufwand zu hoch bleibt.

Einen weiteren Optimierungsansatz beschreiben Detailstufenauswahlverfahren. Dabei werden Informationsverluste der Szene bewusst akzeptiert, indem einige Objekte zur Laufzeit in niedrigeren Detailstufen mit weniger Polygonen gezeichnet werden, als sie im ursprünglichen Modell besitzen. Trifft man die Wahl, welche Objekte in welcher Detailstufe zu zeichnen sind, geschickt, so kann man einerseits die Rechenzeit erheblich reduzieren und andererseits die Wahrnehmung des Nutzers in einem weniger signifikantem Maß negativ beeinflussen, als dies durch Stocken des Bildflusses geschehen würde. Auch im oben genannten Videospiel *Assassins Creed: Origins* kommen solche Algorithmen zum Einsatz, um die beschriebenen Steiblock-Module im vollen Detailumfang zu zeichnen, wenn der Nutzer diese aus der Nähe betrachtet. Sollten hingegen mehrere hundert Module gleichzeitig sichtbar sein, wird die Detailstufe einiger erheblich reduziert [Zve]. Dadurch kann eine unterbrechungsfreie und

dennoch gut aussehende 3D-Echtzeitvisualisierung selbst auf begrenzter Hardwarekapazität erreicht werden.

In dieser Arbeit soll zunächst auf die bisherige Forschungsarbeit auf dem Gebiet der Detailstufenauswahlverfahren – und im besonderen Maße auf einen Algorithmus von Funkhouser und Séquin [FS93] – eingegangen werden. Dieser trifft eine Kosten-Nutzen-Abwägung für jede Detailstufe aller Objekte und kann dadurch feste Rechenzeitgrenzen garantieren. Des Weiteren werden die Forschungsgebiete der künstlichen neuronalen Netze, deren Lernverfahren sowie wahrnehmungsbasierter Farbmodelle und Farbdistanzfunktionen näher betrachtet.

Anschließend wird das Konzept einer Variation des Verfahrens von Funkhouser und Séquin vorgestellt, bei welchem die Nutzen-Berechnung jeder Detailstufe aller Objekte durch Verwendung eines neuroevolutionär erzeugten künstlichen neuronalen Netzes optimiert werden soll. Während des Lernprozesses wird das Netz dabei darauf trainiert, Bilder zu erzeugen, welche in der menschlichen Wahrnehmung möglichst nahe an einem Optimalbild liegen. Auch auf ausgewählte Details bei der Implementierung dieses Verfahrens wird in dieser Arbeit hingewiesen.

Abschließend wird die vorgestellte Abwandlung des betrachteten Detailstufenauswahlverfahrens mit dem Original bezüglich der Güte der generierten Bilder und der Verlässlichkeit ihrer Ergebnisse ausgewertet. Ebenfalls wird ein Ausblick geboten, welche Aspekte des neuen Verfahrens interessante Gegenstände weiterführender Forschungsarbeiten sein könnten.

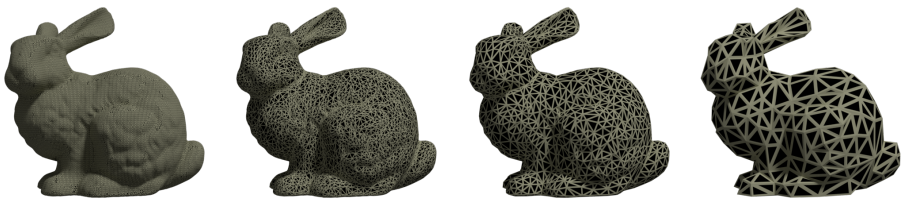
2 Grundlagen

Dieses Kapitel veranschaulicht zunächst, weshalb eine Detailstufenauswahl bei 3D-Echtzeitvisualisierungen von Vorteil ist und wie diese realisiert werden kann. Im Anschluss wird auf die Grundlagen und Anwendungsgebiete von künstlichen neuronalen Netzen und verschiedenen Farbdistanzfunktionen eingegangen.

2.1 Detailstufenauswahl bei 3D-Echtzeitvisualisierungen

Visualisierungen sind mit zunehmender Komplexität mit einem höherem Berechnungsaufwand verbunden. Besonders bei der Darstellung von 3D-Daten fällt dies ins Gewicht. Typischerweise werden 3D-Modelle als Mengen von Eckpunkten (Vertices) und planaren geometrischen Objekten (Strecken, Dreiecke, andere n-Ecke, ...) zwischen diesen Punkten dargestellt. Die Mächtigkeiten beider Mengen wachsen mit der Detailgetreue des Modells an. Dies zieht einen höheren Berechnungsaufwand für jedes Bild mit sich. Steigt er zu stark an, so ist es – mit gleichbleibender Hardwarekapazität – ab einem bestimmten Punkt nicht mehr möglich, einen kontinuierlichen Bildfluss mit genügend Bildern pro Sekunde aufrecht zu erhalten. Es entsteht eine Notwendigkeit für Algorithmen, die den Berechnungsaufwand vermindern.

Aus dieser Notwendigkeit heraus hat sich unter anderem das Forschungsfeld der Detailstufenauswahlverfahren (engl. auch *Level-Of-Detail* oder *LOD selection*) entwickelt. Erstmalig beschrieben von Clark [Cla76] soll bei diesen Verfahren die Komplexität der darzustellenden Modelle verkleinert werden. Die Reduzierung der Gesamtwirkung des Bildes wird hierbei akzeptiert, aber es wird angestrebt, diese möglichst gering zu halten. In Abbildung 2.1 sieht man ein Modell in vier verschiedenen Detailstufen. Je niedriger die Polygonanzahl des Objekts ist, desto geringer ist seine Detailgetreue. In Abbildung 2.2 sind die gleichen vier Detailstufen des Modells, aber in unterschiedlichen Entfernungen zum Betrachter dargestellt. Befindet sich ein Objekt in höherer Entfernung zum Betrachter, so ist auch dessen Abbildung auf dem zu zeichnenden Bild kleiner. Die Verwendung von niedrigeren Detailstufen ist dabei kaum wahrnehmbar.



69451 Dreiecke 13889 Dreiecke 3471 Dreiecke 693 Dreiecke

Abbildung 2.1: Modell eines Hasen aus dem Stanford 3D Scanning Repository in 4 unterschiedlichen Detailstufen.

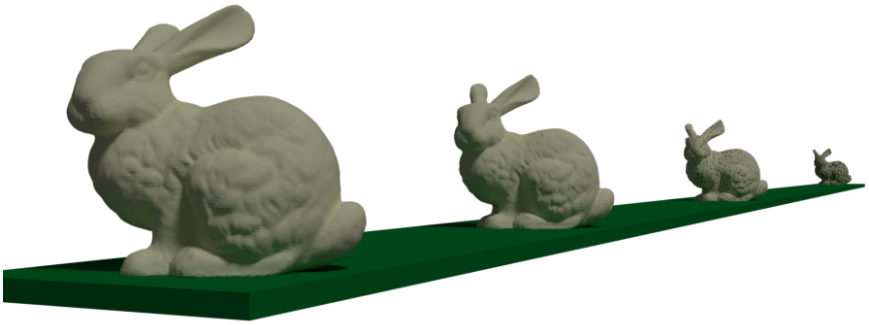


Abbildung 2.2: Modell eines Hasen aus dem Stanford 3D Scanning Repository in den gleichen Detailstufen wie in Abbildung 2.1, jedoch in verschiedenen Entfernungen zum Betrachter (links nach rechts: $11,6LE$; $19,9LE$; $38,9LE$; LE).

Detailstufenauswahlverfahren werden herkömmlicherweise in zwei Kategorien unterteilt: diskrete Verfahren und kontinuierliche Verfahren. Die Unterscheidung geschieht in Hinblick auf die Form, in der die Detailstufen der Objekte abgespeichert werden und wird in Kapitel 2.1.1, bzw. Kapitel 2.1.2 näher erläutert.

2.1.1 Diskrete Detailstufenauswahlverfahren

Diskrete Detailstufenauswahlverfahren setzen voraus, dass die Modelle der Szene bereits vor der Laufzeit in unterschiedlich detaillierten Versionen vorliegen. Zur Laufzeit werden unter Berücksichtigung bestimmter Kriterien die Detailstufen für die jeweils sichtbaren Objekte ausgewählt. Diese Auswahl geschieht von Bild zu Bild.

Erste Implementierungen, wie z.B. die von Clark vorgeschlagene [Cla76], setzen auf Heuristiken bezüglich der Distanz der Objekte zum Betrachter in der Szene. Auch Heuristiken bezüglich der durchschnittlichen Polygon-Größe, der relativen Objektgeschwindigkeit zwischen Bildern oder des Bildfokus sind denkbar. Jedoch kann bei hoher Objektanzahl keine Heuristik garantieren, dass eine fixierte Bildzeichnungsrate eingehalten wird.

Bei adaptiven Detailstufenauswahlverfahren werden zusätzlich zu solchen Heuristiken die benötigte Berechnungszeit des letzten Bildes bzw. der letzten Bilder betrachtet. Ist diese zu hoch, werden die Heuristiken zur Kostenberechnung der Detailstufen verschärft. Doch auch mit dieser Abwandlung kann eine maximale Bildberechnungszeit nicht garantiert werden. Erscheinen plötzlich viele neue Objekte auf einem Bild, so benötigt ein adaptives Detailstufenauswahlverfahren nach wie vor einige Bilder Vorlauf, um sich darauf einstellen zu können.

Im Jahre 1994 schlugen Funkhouser und Séquin einen auf Vorhersagen basierenden adaptiven Detailstufenauswahl-Algorithmus vor, der in der Lage ist, eine obere Schranke für die Berechnungszeit einzuhalten [FS93]. Bei diesem Verfahren wird für jede Detailstufe aller sichtbaren Objekte ein Kosten- und Nutzen-Wert bestimmt. Ziel ist es, für die Menge S aller sichtbaren Objekte jeweils eine Detailstufe so zu wählen, dass die Summe der Nutzen-Werte der gewählten Detailstufen maximiert wird. Dabei muss jedoch darauf Rücksicht genommen werden, dass die Summe der Kosten-Werte die gegebene obere Schranke für die Bildberechnungszeit nicht überschreitet. Diese Voraussetzung ist in Gleichung 2.1 dargestellt. Die

Funktionen zur Bestimmung der Kosten- und Nutzen-Werte sind jeweils vom Objekt O, dessen Detailstufe L und dem verwendeten Bildberechnungsalgorithmus R abhängig.

$$\begin{aligned} \text{Maximiere: } & \sum_S Nutzen(O, L, R) \\ \text{unter der Bedingung: } & \sum_S Kosten(O, L, R) \leq ZielBildBerechnungsZeit \end{aligned} \tag{2.1}$$

Bildberechnungsprozesse lassen sich grob in zwei Schritte zerteilen:

- Berechnungen pro Polygon (Koordinatentransformationen, Lichtberechnung, ...) und
- Berechnungen pro Pixel (Raster-Bildung, Tiefenspeicherung, Transparenzberechnung, Texturen-Abbildung, ...)

Diese Schritte laufen parallel ab, weshalb sich der Kosten-Wert eines Modells nach dem teureren dieser Schritte richtet. Funkhouser und Séquin modellieren die Kostenfunktion als Maximum einer Linearkombination aus Kosten pro Polygon und Kosten pro Eckpunkt, sowie den Kosten pro Pixel [FS93]. Die genannten Kosten sind dabei mit den Koeffizienten c_1, c_2 bzw. c_3 versehen, welche abhängig vom Bildberechnungsalgorithmus sowie der Hardwarekonfiguration des Benutzers sind (vgl. Gleichung 2.2).

$$Kosten(O, L, R) = max \left\{ \begin{array}{c} c_1 \cdot PolygonKosten(O, L) + c_2 \cdot EckpunktKosten(O, L) \\ c_3 \cdot PixelKosten(O) \end{array} \right\} \tag{2.2}$$

Als Heuristik für den Nutzen-Wert einer Detailstufe beschreiben Funkhouser und Séquin die in Gleichung 2.3 gezeigte Funktion [FS93]:

$$\begin{aligned} Nutzen(O, L, R) = & Größe(O) \cdot Genauigkeit(O, L, R) \cdot Semantik(O) \cdot \\ & Fokus(O) \cdot Bewegung(O) \cdot Hysterese(O, L, R) \end{aligned} \tag{2.3}$$

Alle Faktoren dieser Funktion haben den Wertebereich $[0, 1]$ und sind im Folgenden beschrieben:

- **Größe**$(\mathbf{O})$ beschreibt die Größe, die das Objekt auf dem zu berechnenden Bild einnehmen wird.
- **Genauigkeit**$(\mathbf{O}, \mathbf{L}, \mathbf{R})$ beschreibt den Unterschied, den die Darstellung in der jeweiligen Detailstufe zu einem perfekten Bild hat. Funkhouser und Séquin verwenden dafür die Gleichung 2.4. Sie setzen für $GrundUnterschied$ willkürlich den Wert $0,5$ ein und beschreiben $Samples(L, R)$ als die Anzahl der Pixel für Ray-Tracing-Algorithmen, beziehungsweise die Anzahl der Eckpunkte für Gouraud-Schattierungs-Algorithmen oder die Anzahl der Polygone für Algorithmen mit konstanter Schattierung. Allerdings darf $Samples(L, R)$ nie die Anzahl der Pixel übersteigen. m beschreibt einen Exponent, welcher abhängig von der genutzten Schattierungs-Methode ist (konstant=1, Gouraud=2) [FS93].

$$Genauigkeit(O, L, R) = 1 - Unterschied = 1 - \frac{GrundUnterschied}{Samples(L, R)^m} \tag{2.4}$$

- **Semantik**(**O**) gibt den inhaltlichen Wert eines Objektes für die Szene an. Beispielsweise ist der Tennisball einer Tennissimulation sehr viel wichtiger als eventuelle Dekorationen um das Spielfeld.
- **Fokus**(**O**) bezeichnet die Nähe des Objektes zum Fokuspunkt des Betrachters auf dem Bild. Ist ein Teil der Szene besonders auffällig, so ist der Betrachter dazu geneigt, dort hinzuschauen. Dadurch wird eine niedrigere Detaildichte im Rest des Bildes weniger wahrgenommen.
- **Bewegung**(**O**) beschreibt die relative radiale Geschwindigkeit eines Objekts zum Betrachter. Objekte, die sich mit einer hohen Geschwindigkeit über mehrere Bilder hinweg bewegen, werden nur verschwommen wahrgenommen und tragen daher weniger zum Gesamteindruck des Bildes bei.
- **Hysterese**(**O**, **L**, **R**) steht für die Notwendigkeit, dass ein Objekt seine aktuelle Detailstufe beibehält. Wechselt ein Objekt in aufeinanderfolgenden Bildern seine Detailstufe zu häufig, so wird dies vom Betrachter negativ wahrgenommen.

Funkhouser und Séquin betonen auch, dass diese Funktion nur eine experimentelle Heuristik für den Nutzen-Wert darstellt, bis akkuratere Modelle der menschlichen Wahrnehmung kodierbar wären [FS93]. Auch die genaue Bestimmung der einzelnen aufgeführten Faktoren wird von ihnen nicht näher erläutert.

2.1.2 Kontinuierliche Detailstufenauswahlverfahren

Im Vergleich zu diskreten Detailstufenauswahlverfahren beschreibt bei kontinuierlichen Verfahren die Speicherstruktur ein kontinuierliches Detailspektrum, bei dem die gewünschte Detailstufe erst zur Laufzeit abgerufen wird. Besonders bei großen Modellen, wie Terrains oder Gebäuden, ist es schwierig, sinnvolle diskrete Detailstufen zu finden. Kontinuierliche Detailstufenauswahlverfahren sind außerdem in der Lage, die Detailgetreue eines Modells scheinbar fließend zu ändern, ohne dass beim Nutzer eine Form der Hysterese auftreten könnte.

Erstmals stellte Hoppe 1996 ein solches Verfahren für progressive Modelle vor [Hop96]. Dabei wird das Modell in seiner niedrigsten Detailgetreue sowie die Reihenfolge, welche Eckpunkte zur Steigerung der Detaildichte in neue Kanten aufgeteilt werden müssen, abgespeichert. Beim Senken der Detaildichte wird diese Reihenfolge umgekehrt und Kanten zwischen zwei Eckpunkten werden zu einem Eckpunkt kollabiert.

Im Jahre 1997 stellte Hoppe eine Verbesserung des Verfahrens vor, die die Detaildichte des Modells basierend auf dem Blickwinkel des Betrachters variiert. Gobetti und Bouvier veröffentlichten im Jahr 2000 erstmals einen Ansatz, bei dem der Datenüberhang bei einem kontinuierlichen Detailstufenauswahlverfahren auf 8% reduziert werden konnte [GB00].

Mit modernen Grafikkarten ist jedoch das Verändern von geometrischen Informationen teurer, als das Anzeigen verschiedener statischer Detailstufen. Daher wird in der Regel weiterhin auf diskrete Detailstufenauswahlverfahren zurückgegriffen.

2.2 Künstliche Neuronale Netze

Erstmals 1943 von Warren McCulloch und Walter Pitts vorgestellt, bezeichnen künstliche neuronale Netze den Ansatz, das menschliche Gehirn mit einem mathematischen Modell

nachzustellen [MP43]. Solche Modelle können durch verschiedene Strukturen und Lern-Algorithmen an spezielle Aufgaben angepasst werden. Während sie häufig in der modernen künstlichen Intelligenz Anwendung finden, wächst auch die Zahl ihrer Verwendung in anderen Gebieten, wie z.B. der Mustererkennung, Rauschfilterung, Vorhersage, Optimierung oder sensomotorischen Koordination in der Robotik.

Künstliche neuronale Netze beschreiben eine Informationsverarbeitungsstruktur, welche dem menschlichen Nervensystem (neuronales Netz) nachempfunden ist. Das menschliche Nervensystem besteht aus Nervenzellen (auch Neuronen genannt) und Synapsen, welche Verbindungen zwischen je zwei Nervenzellen beschreiben. Wird ein Reiz am Körper ausgelöst, erzeugt dieser an einigen Neuronen einen elektrischen Impuls, welcher durch Synapsen übertragen wird und somit verschiedene andere Neuronen bis hin zum Gehirn „aktiviert“ [Mal93, S. 832ff.].

Künstliche neuronale Netze adaptieren diese Struktur. So bestehen sie aus mehreren, in mindestens zwei Schichten organisierten, virtuellen Neuronen und gewichteten Verbindungen (Synapsen) zwischen den Neuronen. Eine visuelle Repräsentation eines solchen künstlichen neuronalen Netzes ist in Abbildung 2.3 zu sehen. Es gibt bei jedem künstlichen neuronalen Netz eine Eingabeschicht, deren Neuronen nur über ausgehende Synapsen und eine Ausgabeschicht, deren Neuronen nur über eingehende Verbindungen verfügen. Außerdem gibt es noch beliebig viele so genannte versteckte Schichten, die für den Nutzer des künstlichen neuronalen Netzes nicht sichtbar sind. Je nachdem, wie die Neuronen der verschiedenen Schichten miteinander verbunden sind, unterscheidet man in verschiedene Netz-Strukturen [Mal93, S. 842f.]. Eine der am häufigsten verwendeten und in Abbildung 2.3 gezeigten Struktur nennt sich „Feed-Forward-Netz“. Sie zeichnet sich dadurch aus, dass jedes Neuron lediglich mit Neuronen der nachfolgenden Schicht verbunden ist. In anderen Strukturen können auch Schleifen auftreten. Die Struktur der Neuronen und ihre Organisation in Schichten wird auch als Topologie des künstlichen neuronalen Netzes bezeichnet.

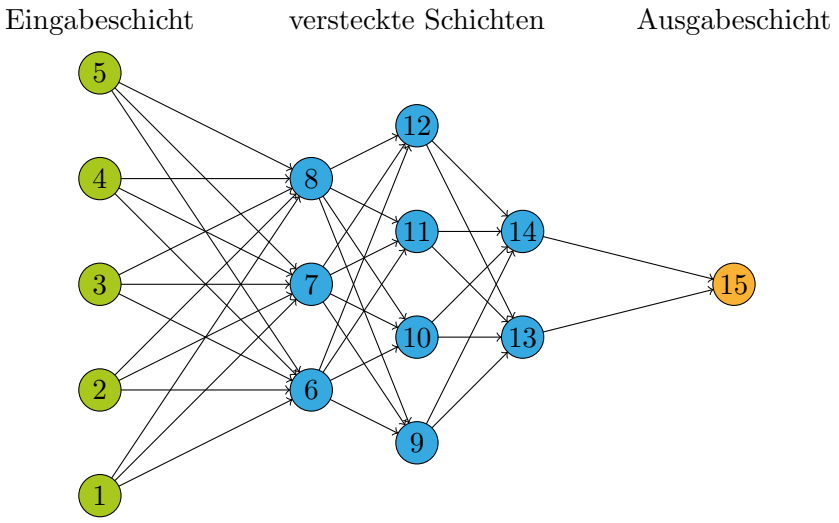


Abbildung 2.3: Visuelle Repräsentation der Struktur von Neuronen und Synapsen eines künstlichen neuronalen Netzes; es handelt sich um ein Feed-Forward-Netz, da alle Neuronen lediglich mit Neuronen der nächsten Schicht verknüpft sind

Bei einer Anfrage des Nutzers an das künstliche neuronale Netz gibt dieser an jedes Neuron der Eingabeschicht je eine Eingabegröße, welche das Problem charakterisiert. Anschließend gibt jedes Neuron zu jedem Zeitschritt eine gewisse Aktivierungsenergie an die verbundenen Neuronen. In Abbildung 2.4 ist ein einzelnes Neuron schematisch dargestellt. Die Funktionen $x_1, x_2, ..., x_n$ bezeichnen die von den verbundenen Neuronen abgegebene Aktivierungsenergien zum Zeitschritt t. $w_1, w_2, ..., w_n$ hingegen beschreiben die Gewichtungen der einzelnen Synapsen. Die Ausgabe-Funktion y gibt an, welche Aktivierungsenergie das betrachtete Neuron zu jedem Zeitschritt abgibt [Mal93, S. 842]. In Abbildung 2.4 wurde als Ausgabe-Funktion die des Schwellenwertmodells gewählt. Übersteigt dabei die Summe der gewichteten Aktivierungsenergien zum Zeitpunkt t einen gewissen Schwellenwert S, so gibt das betrachtete Neuron zum Zeitpunkt $t + 1$ seinerseits eine Aktivierungsenergie von 1 ab. Die Summe der gewichteten ankommenden Aktivierungsenergien in der Ausgabeschicht können schließlich nach genügend Zeitschritten vom Nutzer des künstlichen neuronalen Netzes wieder ausgelesen werden. Hier kann man ebenfalls mit Schwellenwerten arbeiten oder die Ergebnisse auf das gesamte Intervall $[0, 1]$ abbilden. Dieser Ansatz wird auch als „Fuzzy Logic“ bezeichnet. Bei den Nutzereingaben spricht man auch vom Eingabevektor und bei den gelesenen Resultaten von Neuronen der Ausgabeschicht vom Ausgabevektor.

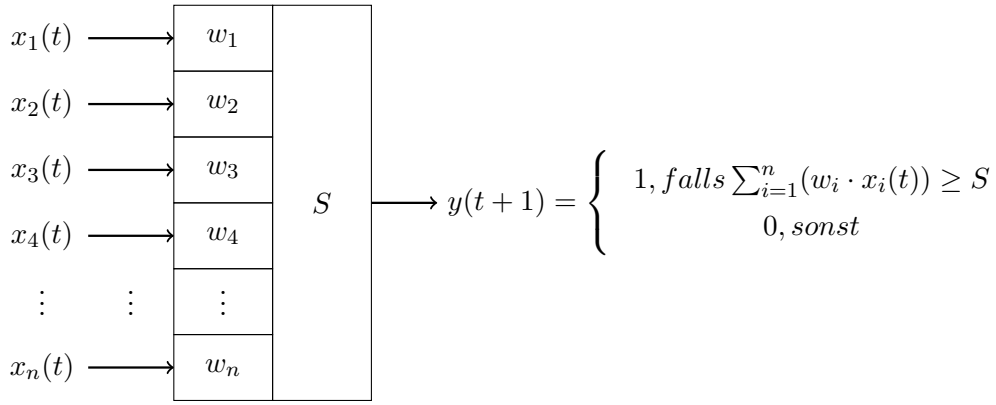


Abbildung 2.4: Einzelnes Neuron einer versteckten Schicht im Schwellenwertmodell

Da die Ausgabe-Funktion des Schwellenwertmodells nicht stetig ist, eignet sich diese nur für wenige Lernverfahren. Alternativ werden stückweise lineare Funktionen, die Sigmoid-Funktion oder Tangens Hyperbolicus verwendet [Mal93, S. 843f.]. Letztere beide sind differenzierbar, was von einigen Lernverfahren vorausgesetzt wird.

Beim Lernprozess eines künstlichen neuronalen Netzes braucht man zunächst einen Datensatz von Eingabevektoren und dazu passenden erwarteten Ausgabevektoren. Als Lernen bezeichnet man das Verändern der Gewichte der Synapsen im Netz, so dass das Netz zu den gegebenen Eingabevektoren Ausgabevektoren generiert, die näher an den erwarteten liegt als zuvor. Häufig wird das Netz dazu mit kleinen zufälligen Gewichten initialisiert. Anschließend werden alle Eingabe- und Ausgabevektor-Paare mehrfach durchlaufen. Jeder dieser Durchläufe wird als Lernepoche bezeichnet. Bei jeder Epoche werden nun die Gewichte so geändert, dass die berechneten Ausgabevektoren näher an den erwarteten Eingabevektoren liegen. Für die tatsächliche Anpassung der Gewichte in jeder Epoche gibt es verschiedene Lernregeln. Eine der elementarsten Regeln, auf der ebenfalls verschiedene andere Regeln beruhen, ist die Hebb-Regel [Mal93, S. 853]. Bei ihr werden die Gewichte aller eingehenden

Synapsen eines Neurons um das Produkt $sw \cdot a \cdot y$ verschoben. Dabei ist a die summierte gewichtete Aktivierungsenergie, y die daraus berechnete abgegebene Aktivierungsenergie und sw eine konstante positive Schrittweite. Diese beeinflusst, wie schnell sich ein Neuron an gewisse Aktivierungsenergien gewöhnt. Aus dieser Lernregel entstanden die Delta-Regel und die allgemeinere und heute vielfach angewandte Back-Propagation-Regel.

Zudem gibt es unüberwachte Wettberwerbslernverfahren. Wettbewerbslernen bedeutet, dass je ein Neuron der Ausgabeschicht für je eine zu unterscheidende Klasse steht. Unüberwacht bedeutet, dass keine zuvor bekannten Ausgabevektoren existieren und das Lernverfahren selbst eine Klasseneinteilung der Eingabevektoren vornimmt. Ein Beispiel für ein solches künstliches neuronales Netz ist das „Adaptive-Resonance-Theory-Netz“ [CGR91].

Wie bereits erwähnt, kann man künstliche neuronale Netze für Probleme aus sehr verschiedenen Anwendungsgebieten verwenden. Allerdings ist es aufgrund der vielen Parameter der Struktur der Netze und Lernmethoden oft schwierig, ein Netz zu erstellen und anzulernen, welches für das jeweilige Problem gut funktioniert. Es kann z.B. bereits zu Problemen kommen, wenn man zu viele Neuronen in versteckten Schichten verwendet und sich das Netz zu sehr auf die zu lernenden Eingabevektoren spezialisiert. Auch durch die zufällige Initialisierung der Gewichte kann man bei mehreren Lernprozessen mit gleichbleibenden Parametern manchmal zu Netzen gelangen, welche das Problem gut lösen können und manchmal auch nach mehreren hundert Lernepochen nur Netze erhalten, welche selbst die zu erlernenden Eingabevektoren nicht gut klassifizieren können.

2.2.1 Neuroevolutionär entwickelte künstliche neuronale Netze

Die Topologie eines künstlichen neuronalen Netzes wird herkömmlicherweise experimentell der Anwendung angepasst, wodurch die Entwicklungszeit gesteigert und unter Umständen optimalere und kleinere Netze nicht gefunden werden. Neuroevolution bezeichnet eine Methode zum Modifizieren von Gewichten oder Topologien in künstlichen neuronalen Netzen unter Verwendung von genetischen bzw. evolutionären Algorithmen, um spezielle Aufgaben zu erfüllen [Mii10].

Im Jahre 2002 stellten Kenneth O. Stanley und Risto Miikkulainen den NEAT-Algorithmus vor (NEAT = NeuroEvolution of Augmenting Topologies), welcher sowohl Gewichte anpasst als auch Topologien generiert [SM02]. Der evolutionäre Algorithmus ist in mehreren Epochen organisiert. In jeder Epoche wird eine Menge an Genomen - die Population jeder Epoche - generiert. Jedes Genom kann eindeutig in ein künstliches neuronales Netz (auch Phänotyp des Genoms genannt) überführt und dahingehend überprüft werden, inwiefern das gegebene Problem lösbar ist. Der Lernprozess zeichnet sich unter anderem dadurch aus, dass neben der Anzahl der Eingabe- und Ausgabeneuronen lediglich eine Fitnessfunktion zu definieren ist, anhand derer die Netze mit den höchsten Erfolgsaussichten jeder Epoche selektiert werden können.

Die Population der ersten Epoche besteht ausschließlich aus Genomen, welche Netze ohne versteckten Neuronen, in denen jedes Eingabeneuron mit jedem Ausgabeneuron verbunden ist, codieren. Nur die zufällig gewählten Gewichte der Synapsen dieser Epoche unterscheiden die Netze voneinander. Die Generierung von Genomen einer neuen Epoche geschieht auf Grundlage mindestens eines bereits getesteten Genoms und kann auf verschiedene Weisen durchgeführt werden:

- durch herkömmliche Gewichtsänderung im ursprünglichen Genom,

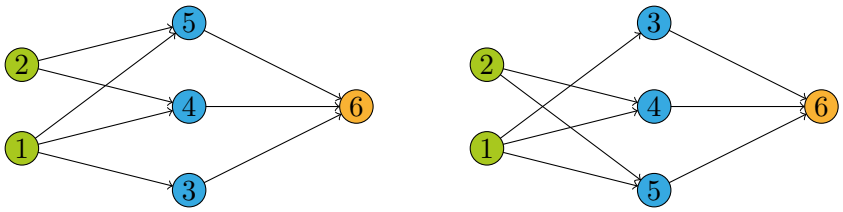


Abbildung 2.5: Darstellung zweier möglicher Genome $[3, 4, 5]$ (links) und $[5, 4, 3]$ (rechts), die im selben Phänotyp resultieren.

- durch Mutation des ursprünglichen Genoms in Form von Hinzufügen neuer Synapsen oder Neuronen, welche eine bereits bestehende Synapse aufspalten oder
- durch Rekombination mehrerer Genome früherer Epochen.

Ein zentrales Problem bei neuroevolutionären Algorithmen wird von Stanley und Miikkulainen als Competing-Conventions-Problem bezeichnet [SM02]. Damit wird der potentielle Verlust von Informationen bei der Rekombination von zwei Genomen beschrieben, die den selben Phänotyp darstellen. In Abbildung 2.5 sieht man die Darstellung zweier solcher Genome. Die Rekombination von $[3, 4, 5]$ und $[5, 4, 3]$ kann z.B. auch in $[3, 4, 3]$ oder $[5, 4, 5]$ resultieren. Dadurch ginge jeweils $1/3$ der Informationen von den ursprünglichen Genomen verloren.

Um dem Competing-Conventions-Problem vorzubeugen, versieht der NEAT-Algorithmus jegliche strukturelle Mutation bei der Codierung der Genome mit einem über allen Epochen und Genomen globalen inkrementellen Index. Durch diesen Mutations-Index ist es möglich, sicher zu stellen, dass Informationen, die sich zwei Genome teilen, durch Rekombination immer erhalten bleiben kann [SM02].

Beim Hinzufügen neuer Synapsen zu einem Genom sinkt in der Regel die Fitness des Phänotyps, da die Gewichte der neuen Synapse noch nicht angepasst werden konnten. Um jedoch trotzdem mögliche Innovation solcher Mutationen sicherzustellen, verwendet der NEAT-Algorithmus Artenbildung [SM02]. Dabei werden strukturell ähnliche Genome zu Arten zusammengefasst und vornehmlich miteinander kombiniert. Die Anzahl der Genome einer Art in neuen Epochen wird begrenzt durch den Fitness-Vorteil der Art gegenüber anderen Arten. Dadurch kann sichergestellt werden, dass keine einzelne Art dominiert. Experimente zeigen, dass die Erfolgsrate von NEAT durch die Anwendung von Artenbildung zusätzlich um bis zu 73% gesteigert werden kann [Nod10].

2.3 Farbmodelle und Farbdistanzfunktionen

Farbinformationen bei Echtzeitvisualisierungen werden häufig im Rot-Grün-Blau-Modell (im folgenden RGB-Modell) berechnet und abgespeichert. In diesem Farbsystem werden Farbtöne als die Summe der Anteile der drei physikalischen Grundfarben (Rot, Grün, Blau) dargestellt. Wenn man in diesem Farbraum den Unterschied zwischen zwei Farben C_1 und C_2 bestimmen will, ist die euklidische Farbdistanzfunktion naheliegend. In Gleichung 2.5 ist diese Funktion dargestellt, wobei R_{C_n}, G_{C_n} und B_{C_n} jeweils für die prozentualen Rot-,

Grün- bzw. Blau-Anteile der Farbe C_n stehen.

$$\Delta_{C_1C_2} = \sqrt{(R_{C_1} - R_{C_2})^2 + (G_{C_1} - G_{C_2})^2 + (B_{C_1} - B_{C_2})^2} \tag{2.5}$$

Das menschliche Auge weist jedoch eine unterschiedliche Empfindlichkeit bezüglich der drei physikalischen Grundfarbtöne auf [BD80]. Aufgrund der erhöhten Anzahl an Rezeptoren für grüne Farbtöne (besonders im Vergleich zur Anzahl der Rezeptoren für blaue Farbtöne) wird zwischen Farben mit hohem Grünanteil kaum ein Farbunterschied wahrgenommen, auch wenn sich der Blauanteil stark unterscheidet. Dieses Problem wird in Abbildung 2.6 verdeutlicht. Obwohl die Farben C_1 und C_2 sehr viel ähnlicher wahrgenommen werden als C_3 und C_4, ist deren euklidische Distanz größer. Man spricht davon, dass das RGB-Modell nicht empfindungsgemäß einheitlich ist.

C_1 R : 0% G : 100% B : 0%	C_2 R : 0% G : 100% B : 32,9%	$\Rightarrow \Delta_{C_1C_2} = 32{,}9\%$
C_3 R : 0% G : 46,3% B : 84,7%	C_4 R : 0% G : 69,4% B : 61,6%	$\Rightarrow \Delta_{C_3C_4} \approx 32{,}3\%$

Abbildung 2.6: Darstellung von vier Farbtönen C_1 bis C_4, sowie Angabe deren Farbanteile nach dem RGB-Modell. Auf der rechten Seite sind die euklidischen Farbdistanzen der Farbtöne der jeweiligen Zeile angegeben.

Des Weiteren überlappen sich zum Teil die Wellenlängenbereiche, welche die verschiedenen Farbrezeptorarten im menschlichen Auge wahrnehmen können. Da die Farbanteile im RGB-Farbsystem jedoch orthogonal zueinander sind, kann dieses auch nicht durch die Skalierung der Achsen in einen empfindungsgemäß einheitlichen Farbraum überführt werden. Bereits 1931 beschäftigte sich die „Commission internationale de l'éclairage“ (Internationale Beleuchtungskommission, kurz CIE) damit, einen alternativen Farbraum zu entwerfen, welcher sich besser nach der menschlichen Wahrnehmung richtet. Aus diesen Bestrebungen entstand experimentell das CIE-Normvalenzsystem (auch XYZ-Farbsystem), aber auch dieses gilt als nicht empfindungsgemäß einheitlich.

Im Jahre 1976 stellte die CIE das L*a*b*-Farbmodell (kurz CIELab-Modell) vor, in dem erstmals die Farben abhängig davon, wie ein Normalbeobachter sie unter Standard-Lichtbedingungen sieht, dargestellt werden [ISO08]. Ähnlich wie im RGB-Farbsystem stehen hier drei Achsen orthogonal zueinander, wobei

- L* die Helligkeit (Luminanz) der Farbe (0=Schwarz, 100=weiß),
- a* die Nähe der Farbe zu Grün (ca. -170) bzw. Rot (ca. 100),

- b* die Nähe der Farbe zu Blau (ca. -100) bzw. Gelb (ca. 150) angibt.

Der ISO Standard für das CIELab-Modell definiert ebenfalls eine euklidische Farbdifferenzfunktion ΔE (vgl. Gleichung 2.6). Hierbei stehen $L^*_{C_n}$, $a^*_{C_n}$ und $b^*_{C_n}$ jeweils für die Achsenwerte der Farbe C_n im CIELab-Modell.

$$\Delta E_{C_1C_2} = \sqrt{(L^*_{C_1} - L^*_{C_2})^2 + (a^*_{C_1} - a^*_{C_2})^2 + (b^*_{C_1} - b^*_{C_2})^2} \tag{2.6}$$

Auch die Farbdifferenzfunktion ΔE weist noch Mängel auf, da einige Farben mit hoher Sättigung einen geringer empfundenen Abstand aufweisen, als die Funktion beschreibt. In Abbildung 2.7 ist beispielsweise der Abstand zwischen zwei sehr ähnlich empfundenen Gelbtönen und zwei Grautönen dargestellt. Die ΔE-Farbdistanz sagt jedoch aus, dass die Gelbtöne zueinander mit $29,74$ sogar eine geringfügig höhere Distanz als die Grautöne zueinander mit 29 haben. Das CIE definierte bis heute mehrere neue Farbdifferenzfunktionen auf dem CIELab-Modell. Die neueste dieser Funktionen wird als CIEDE2000-Funktion bezeichnet [SWD], geht jedoch auch mit einem hohen Rechenaufwand einher.

C_1 $L^* : 94.59$ $a^* : -6.45$ $b^* : 95.36$	C_2 $L^* : 95.59$ $a^* : -5.63$ $b^* : 65.65$	$\Rightarrow \Delta E_{C_1C_2} \approx 29,74$
C_3 $L^* : 50$ $a^* : -5$ $b^* : 0$	C_4 $L^* : 79$ $a^* : -5$ $b^* : 0$	$\Rightarrow \Delta E_{C_3C_4} = 29$

Abbildung 2.7: Darstellung von vier Farbtönen C_1 bis C_4, sowie Angabe deren Luminanz (L*) und Farbnähe zu Grün/Rot (a*) bzw. Blau/Gelb (b*) nach dem CIELAB-Modell. Auf der rechten Seite sind die ΔE-Farbdistanzen der Farbtöne der jeweiligen Zeile angegeben.

Die Einführung eines weiteren Farbraums namens DIN99-Modell sollte das CIELab-Modell so transformieren, dass wieder eine euklidische Distanzfunktion verwendet werden kann, um den Berechnungsaufwand zu reduzieren [DIN01]. Optimierungen dieses Farbmodells von Cui at al. erreichen auch ähnlich gute Ergebnisse wie die CIEDE2000 Farbdifferenzfunktion [CLR+] und wurden 2017 in den Standard als DIN99o-Formel übernommen.

Nimmt man eine Menge verschiedener Farbtöne, so kann man diese nach ihrer Distanz von einer Referenzfarbe geordnet in einem Farbdistanzspektrum abbilden. In den Abbildungen 2.8 und 2.9 sind jeweils solche Farbdistanzspektren für die euklidische Farbdistanzfunktionen im RGB- bzw. D99-Modell und ΔE bzw. CIEDE2000 im CIELab-Modell dargestellt. Interessant sind besonders der Beginn und das Ende des Spektrums: zu Beginn sollten keine Farben auftauchen, die sich von der Referenzfarbe stark unterscheiden und am Ende sollten keine Farben der Referenzfarbe ähneln.

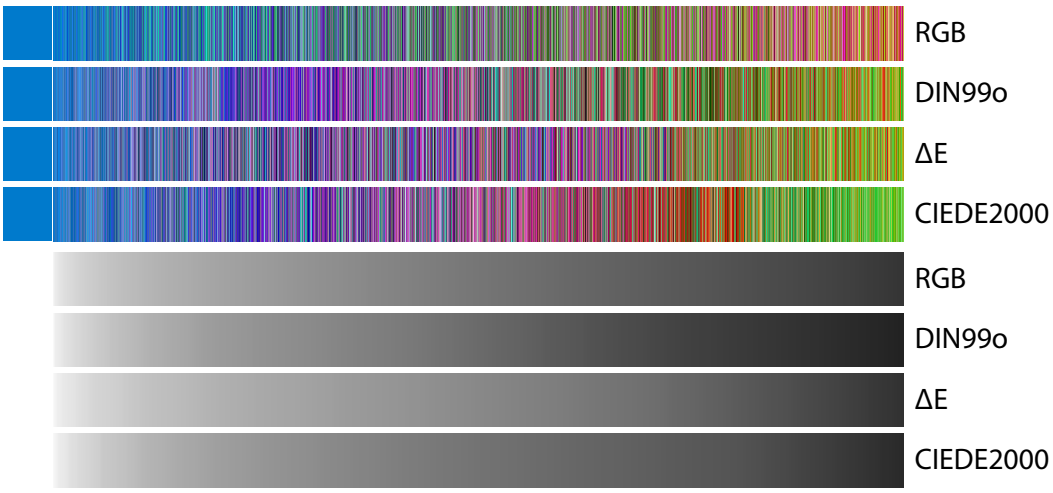


Abbildung 2.8: 4 Farbdistanzspektren zur Referenzfarbe Ozeanblau $(0\%, 47.8\%, 0.8\%)$ (RGB-Modell) mit verschiedenen Farbdifferenzfunktionen, sowie die Abstandsverteilung zu den jeweiligen Farbdistanzspektren (dunkler = höhere Distanz)

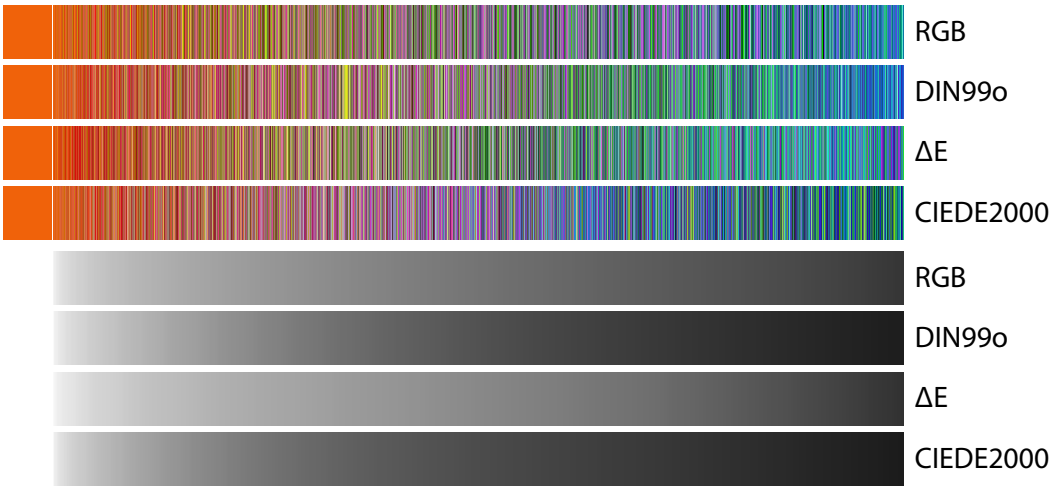


Abbildung 2.9: 4 Farbdistanzspektren zur Referenzfarbe Orange $(94.1\%, 38.4\%, 3.9\%)$ (RGB-Modell) mit verschiedenen Farbdifferenzfunktionen, sowie die Abstandsverteilung zu den jeweiligen Farbdistanzspektren (dunkler = höhere Distanz)

3 Konzept

Das 1993 von Funkhouser und Séquin vorgestellte Verfahren zur diskreten Detailstufenauswahl wurde bereits in Kapitel 2.1.1 vorgestellt. In der Originalarbeit wird zur Berechnung des Nutzen-Wertes die ebenfalls oben erläuterte Gleichung 2.3 herangezogen.

Funkhouser und Séquin geben für einige Faktoren des Nutzen-Werts keine genaue Formel zur Berechnung an und betonen zudem, dass die Gewichtung dieser nicht genauer bestimmt werden könne, bis ein genaueres Modell der menschlichen Wahrnehmung vorläge [FS93]. Ziel dieser Arbeit soll es einerseits sein, eine neue Nutzen-Funktion einzuführen, welche adaptiv die Gewichtung verschiedener Bewertungsverfahren angibt, sowie ein beispielhaftes Modell zur Abbildung der menschlichen Wahrnehmung von computergenerierten Bildern mit eingeschränkter Detailstufe vorzustellen.

3.1 Neue Nutzen-Funktion

Die bisherige Funktion zeichnet sich durch mehrere Faktoren aus dem Wertebereich $[0, 1]$ aus. Dadurch wird sichergestellt, dass der Nutzen-Wert sinkt, sobald einer der Faktoren sehr gering oder gar 0 beträgt – auch wenn andere Faktoren vergleichsweise hoch sind. Führt man Exponenten $\lambda_1, \lambda_2, ..., \lambda_6 \in \mathbb{R}^+$ zur Gewichtung dieser Faktoren ein, kann dies bei einer neuen Nutzen-Funktion weiterhin sichergestellt werden (vgl. Gleichung 3.1). Für $\lambda_n < 1$ wird der n-te Faktor gestaucht, bzw. verstärkt für $\lambda_n > 1$.

$$\begin{aligned} Nutzen(O, L, R) = \; & Größe(O)^{\lambda_1} \cdot Genauigkeit(O, L, R)^{\lambda_2} \cdot Semantik(O)^{\lambda_3} \cdot \\ & Fokus(O)^{\lambda_4} \cdot Bewegung(O)^{\lambda_5} \cdot Hysterese(O, L, R)^{\lambda_6} \end{aligned} \tag{3.1}$$

Der Faktor $Größe(O)$ wird durch die Anzahl der Pixel auf dem Bildschirm bestimmt, die das Objekt in seiner höchsten Detailstufe belegt. Um diese Anzahl in den Wertebereich $[0, 1]$ abzubilden, wird sie in Relation zur Größe des gesamten Bildes gesetzt.

Die $Genauigkeit(O, L, R)$ sollte die Nähe eines Modells zu dessen perfektem Abbild angeben. Häufig liegen jedoch keine Informationen zu einem solchen Abbild vor und nur die Feststellung der Unterschiede zur höchsten Detailstufe ist möglich. Bei Ray-Tracing-Algorithmen wird dieser Faktor durch die Anzahl der belegten Pixel des Modells beschränkt. Bei anderen Bildberechnungsverfahren ist der Faktor durch die Polygon- bzw. Eckpunkt-Anzahl v oder ebenfalls die Anzahl der belegten Pixel p (wenn $p > v$) begrenzt. Somit ist die Anzahl der belegten Pixel - unabhängig vom verwendeten Bildberechnungsverfahren R - eine obere Schranke für den noch nicht normierten Faktor $Genauigkeit^*(O, L, R)$:

$$Genauigkeit^*(O, L, R) \leq AnzahlBelegterPixel \tag{3.2}$$

Die Gleichheit in dieser Ungleichung wird erreicht, wenn die Anzahl der sichtbaren Polygone die der belegten Pixel übersteigt. In diesem Fall wäre der Faktor $Genauigkeit(O, L, R)$

proportional zur $Größe(O)$. Die Ungleichheit hingegen tritt nur auf, wenn die Anzahl der sichtbaren Pixel höher als die der Polygone liegt. Moderne Hardware ist allerdings seit Veröffentlichung dieses Detailstufenauswahlverfahrens 1993 leistungsfähiger geworden. Solche Verfahren kommen daher heute erst bei Szenen mit komplexeren Objekten und viel mehr Polygonen zum Einsatz. Man kann in vielen Fällen davon ausgehen, dass der eben genannte Fall der Gleichheit in der Ungleichung eintritt. In den anderen Fällen, belegt das Objekt vermutlich einen erheblichen Teil des Bildes und kommt als Kandidat für einen Detailstufenreduktion weniger in Betracht. Somit wäre eine zusätzliche Verminderung des Nutzen-Wert dieses Objekts auch wenig sinnvoll. Aus diesen Gründen ist davon auszugehen, dass der Faktor $Genauigkeit(O, L, R)$ nicht signifikant mehr Informationen als der Faktor $Größe(O)$ zur Nutzen-Funktion beiträgt und er damit gestrichen werden könnte.

Der $Semantik(O)$ Faktor eines Modells wird zur Entwicklungszeit einer Visualisierung bereits festgelegt und gibt den inhaltlichen Nutzen des Objekts für die Szene wieder. Hierbei wäre es wenig sinnvoll, diesen Wert während der Laufzeit zu stauchen oder zu verstärken. Somit ist auch eine Gewichtungskonstante für diesen Faktor überflüssig.

$Fokus(O)$ gibt die Nähe zum vom Nutzer fokussiertem Punkt F auf dem zu berechnenden Bild an. Um den Faktor zu berechnen, wird zunächst der Mittelpunkt A des Modells auf dem zu berechnenden Bild bestimmt und dann die Länge von $\overline{FA}$ in Relation zur höchsten Entfernung e_{max} von F zu den Bildeckpunkten gesetzt. Dieses Verhältnis ist auch in Gleichung 3.3 dargestellt, wobei E_1, E_2, ... E_4 die Eckpunkte des Bildes beschreiben.

$$Fokus(O) = \frac{|\overline{FA}|}{e_{max}}, e_{max} = max \left\{ \begin{array}{c} |\overline{FE_1}| \\ |\overline{FE_2}| \\ |\overline{FE_3}| \\ |\overline{FE_4}| \end{array} \right\} \tag{3.3}$$

In der Veröffentlichung von Funkhouser und Séquin wird der Faktor $Bewegung(O)^{\lambda_5}$ als scheinbare Geschwindigkeit des Modells in Relation zu dessen durchschnittlichen Polygongröße beschrieben [FS93]. Leider wird weder erläutert, ob die scheinbare Geschwindigkeit im 3D-Raum der Szene oder auf dem berechneten Bild gemessen wird, noch wie sichergestellt werden kann, dass diese Relation immer in $[0, 1]$ liegt. Daher soll hier stattdessen die durchschnittliche relative Geschwindigkeit der Polygone betrachtet werden. Für die relative Geschwindigkeit eines Polygons sollte dabei die zurückgelegte Strecke des Polygons auf dem Bild im Verhältnis zur maximal möglichen Bildstrecke in dieser Richtung gemessen werden. Bei Objekten, deren Eigenrotation sehr gering ist, lässt sich dies auch zur relativen Geschwindigkeit des Modell-Mittelpunktes auf dem Bildschirm vereinfachen.

Zur Berechnung der $Hysterese(O, L, R)$ für eine Detailstufe L betrachten Funkhouser und Séquin den Unterschied zwischen L und der gewählten Detailstufe im vorherigen Bild. Diese Formel betrachtet allerdings nicht, ob die Detailstufe eines Objekts sich in den zuletzt berechneten Bildern tatsächlich geändert hat. Je häufiger dies geschieht, desto niedriger sollte definitionsgemäß der $Hysterese(O, L, R)$-Faktor ausfallen. Um dies zu berücksichtigen, soll hier jeweils der Unterschied zwischen aufeinander folgenden gewählten Detailstufen in den letzten $k \in \mathbb{N}$ Bildern in Relation zum maximalen Unterschied zwischen den Detailstufen des Objekts gesetzt und abgeschwächt werden, je länger das verglichene Bild zurück liegt. Sei $\Delta_{A,B}$ den Unterschied zwischen zwei Detailstufen A und B, $L_{alt}(t - n)$ die gewählte Detailstufe im n-t letzten Bild und L_{max} bzw. L_{min} die maximale bzw. minimale Detailstufe beschreibt, so ergibt sich dafür die Funktion in Gleichung 3.4. Wählt man $k = 1$, so erhält

man die ursprüngliche Formel von Funkhouser und Séquin.

$$Hysterese_k(O,L,R) = \frac{\Delta_{L,L_{alt}(t-1)}}{\Delta_{L_{max},L_{min}}} \cdot \sum_{i=1}^{k} \left(\frac{1}{i+1} \cdot \frac{\Delta_{L_{alt}(t-i),L_{alt}(t-i-1)}}{\Delta_{L_{max},L_{min}}} \right) \tag{3.4}$$

Geht man davon aus, dass die wahrnehmbaren Unterschiede zwischen jeweils zwei aufeinander folgenden Detailstufen für das Modell gleichbleibend sind, so könnte man Δ_{L_a,L_b} mit der Differenz zwischen a und b beschreiben. Dabei steht L_n für die n-te Detailstufe des Modells.

Mit diesen Überlegungen zu den einzelnen Faktoren ergibt sich nun die neue Nutzen-Funktion in Gleichung 3.5.

$$\begin{aligned} Nutzen(O,L,R) = \; & Größe(O)^{\lambda_1} \cdot Fokus(O)^{\lambda_2} \cdot Bewegung(O)^{\lambda_3} \cdot \\ & Hysterese(O,L,R)^{\lambda_4} \cdot Semantik(O) \end{aligned} \tag{3.5}$$

Die Variablen λ_1, λ_2, ..., λ_4 zur Gewichtung hängen einerseits von der Szene, vor allem aber von der Interaktion des Nutzers mit dieser ab. Bei einem Flugsimulator scheint es zum Beispiel sehr viel sinnvoller, auf der Start- bzw. Landebahn höheren Wert auf die Größe der Umgebungsobjekte zu legen und während eines Flugs die relative Geschwindigkeit näher zu betrachten. Die Interaktionsmöglichkeiten des Nutzers mit der Szene können auch zwischen verschiedenen Anwendungen stark variieren. Bei Visualisierung von Messdaten reicht es teilweise aus, dass der Nutzer sich lediglich durch die Szene bewegt. Andererseits ist es bei Visualisierung für Stadtplanungen beispielsweise denkbar, dass Objekte verschoben oder auch neu erstellt werden können. Daher ist es nur schwer möglich, eine allgemeingültige Formel zur Bestimmung der Gewichtungsvariablen zu beschreiben.

3.2 Gewichtungsbestimmung der Nutzen-Faktoren

Um die Exponenten λ_1, λ_2, ..., λ_4 aus Gleichung 3.5 zu bestimmen, soll in dieser Arbeit maschinelles Lernen bemüht werden. Unter Zuhilfenahme eines künstlichen neuronalen Netzes kann man diese Variablen in Abhängigkeit diverser Parameter, die die Interaktion des Nutzers mit der Visualisierung beschreiben, optimieren.

Wie bereits in Kapitel 2.2 beschrieben wurde, gibt es diverse mögliche Topologien und entsprechenden Lernverfahren für künstliche neuronale Netze. Aufgrund der guten Resultate, die der NEAT-Algorithmus bei Problemen verschiedener Themengebiete, wie z.B. Robotik oder Kontrollmechanismen bei chemischen Prozessen, liefern konnte [SM02, Mii10], wird er auch in dieser Arbeit Anwendung finden.

Als Parameter für die Eingabe-Neuronen dient zunächst ausschließlich das Sichtfeld des Betrachters der Szene. Andere mögliche Parameter werden in Kapitel 6 besprochen. Das Sichtfeld des Betrachters der Szene definiert sich durch dessen Position, sowie einen normierten Vektor in Blickrichtung und einen darauf orthogonal stehenden, ebenfalls normierten Vektor, welcher die Richtung der Oberseite des Bildes bestimmt. Zerlegt man die Position und Vektoren jeweils in ihre einzelnen Komponenten, so erhält man bei 3D-Visualisierungen neun Parameter für Eingabe-Neuronen. Im NEAT-Algorithmus werden vier Ausgabe-Neuronen eingestellt, welche direkt als Exponenten λ_1, λ_2, ..., λ_4 dienen sollen.

Die Eingabe des NEAT-Algorithmus erfordert zudem eine Reihe von möglichen Eingabevektoren, für welche die entstehenden Netze getestet werden sollen. Hierfür empfiehlt es

sich, Sichtfelder des Betrachters auszuwählen, welche verstärkt vorkommen. Denkbar sind hier z.B. aufgezeichnete Bewegungspfade von Testnutzern innerhalb der Szene.

Um die entstehenden Netze des NEAT-Algorithmus zu bewerten, dient eine Fitness-Funktion. Im hier vorgestellten Problem müsste die Fitness-Funktion bewerten, wie positiv ein berechnetes Bild B wahrgenommen wird. Hierbei ist zu beachten, dass bei B die Detailstufenauswahl basierend auf dem Verfahren nach Funkhouser und Séquin mit der veränderten Nutzen-Funktion (Gleichung 3.5) stattfindet. Dabei bestimmt das zu bewertende künstliche neuronale Netz die Exponenten λ_1, λ_2, ..., λ_4.

Zur Bestimmung einer solchen wahrnehmungsbasierten Fitness-Funktion wäre korrekterweise eine Probandenstudie jeder Szene notwendig. Um dies möglichst gut abzubilden, sollen für die zu lernenden Sichtpunkte die prozentualen Bildfehler beim Vergleich von „perfekten“ Bildern zu den jeweils erzeugten Bildern – unter Verwendung der Parameter des zu evaluierenden künstlichen neuronalen Netzes – berechnet werden. Bezeichnet man die Ausgaben eines solchen Netzes N als N_{λ_1}, N_{λ_2}, ..., N_{λ_4} und die Menge aller zu trainierenden Sichtpunkte als V, so kann man diese Fitness-Funktion mit Gleichung 3.6 beschreiben. Dabei beschreibt $Bild_{NeuerFunkhouser}(\lambda_1, \lambda_2, \lambda_3, \lambda_4)$ das gezeichnete Bild unter Verwendung des Algorithmus von Funkhouser und Séquin mit der neuen Nutzen-Funktion und $Bild_{perfekt}$ ein „perfektes“ Bild.

$$\begin{aligned} Fitness(N) &= 1 - MittlererBildFehler \\ &= 1 - \frac{1}{|V|} \cdot \sum_{V} BildFehler \\ &= 1 - \frac{1}{|V|} \cdot \sum_{V} (Bild_{NeuerFunkhouser}(N_{\lambda_1}, N_{\lambda_2}, N_{\lambda_3}, N_{\lambda_4}) - Bild_{perfekt}) \end{aligned} \tag{3.6}$$

Hierbei soll als „perfektes“ Bild, ein berechnetes Bild gelten, bei dem alle sichtbaren Objekte in ihrer höchsten Detailstufe gezeichnet wurden. Nimmt man nun als Differenz von zwei Bildern B_1 und B_2 mit Breite w und Höhe h die durchschnittliche wahrgenommene Farbdistanz der Pixel an, so wird die Fitness-Funktion ohne zusätzliche Studien an die menschliche Wahrnehmung angenähert. Bezeichnet man mit $B_n(x, y)$ die Farbinformation von Bild B_n in der x-ten Spalte und y-ten Zeile, so erhält man für die Bilddistanz die Gleichung 3.7. Damit die Bilddifferenz einen prozentualen Wert im Bereich $[0, 1]$ zurück gibt, muss die Funktion für wahrgenommene Farbdistanz ebenfalls normiert sein.

$$B_1 - B_2 = \frac{1}{w \cdot h} \sum_{x=1}^{w} \sum_{y=1}^{h} WahrgenommeneFarbdistanz(B_1(x, y), B_2(x, y)) \tag{3.7}$$

Als Funktion für die wahrgenommene Farbdistanz wären die in Kapitel 2.3 beschriebenen CIEDE2000- oder DIN99-Formel denkbar. Da sie vergleichbare Ergebnisse liefern, wird aus Performanz-Gründen im weiteren die Farbdistanzfunktion im DIN99-Modell verwendet.

4 Implementierung

In diesem Kapitel sollen einige Implementierungsdetails für das in Kapitel 3 beschriebene Konzept genauer beleuchtet werden.

4.1 Verwendete Software

Die Applikation wurde für Windows-Betriebssysteme entwickelt und getestet, verwendet jedoch außer dem Dateisystem keine tiefer greifenden Betriebssystem-Funktionen und könnte vermutlich ohne viel Aufwand auf Unix-basierte Systeme übertragen werden.

Zum Berechnen der Bilder wurde die am Leibniz-Rechenzentrum entwickelte *sceenLib* verwendet. Diese organisiert die Elemente einer Szene in einem Szenen-Graph und unterstützt mehrere Detailstufen pro Knoten in diesem Graphen. Zusätzlich wird eine Implementierung des ursprünglich vorgeschlagenen Algorithmus von Funkhouser und Séquin geboten. Das Bildberechnungsverfahren dieser Bibliothek basiert auf OpenGL.

Als Implementierung für das NEAT-Verfahren wurde sich für AccNEAT von Sean Doughtery entschieden (`https://github.com/sean-dougherty/accneat`). Entstanden aus der ursprünglichen NEAT-Implementierung von Keneth Stanley, wurde das Verfahren in dieser Bibliothek durch weitere genetische Mutationen, verbesserte Suchverfahren, Laufzeitoptimierungen und Parallelisierung nach Angaben des Autors um ein vielfaches beschleunigt. Im Rahmen dieser Arbeit wurde diese Bibliothek für Windows-Betriebssysteme adaptiert (siehe `https://github.com/Artenuvielle/accneat`).

4.2 Aufbau der Implementierung

Die Applikation zum Testen des Konzepts dieser Arbeit ist in drei Schritten aufgebaut:

- eine **Vorberechnungsphase**, in der Kosten-Werte und Nutzen-Faktoren aus gewissen Blickwinkeln bereits berechnet werden, da sie später noch häufiger gebraucht werden,
- eine **Lernphase**, in der durch den NEAT-Algorithmus (siehe Kapitel 2.2.1) ein künstliches neuronales Netz angelernt wird, und
- final die eigentliche **Laufzeit**, in der das entwickelte künstliche neuronale Netz zur Bestimmung der Nutzen-Funktion-Gewichte für das Detailstufenauswahlverfahren verwendet wird.

Die Ergebnisse der ersten beiden Phasen können zudem einzeln zwischengespeichert werden, um sie nicht zu jeder Ausführung erneut berechnen zu müssen.

4.2.1 Vorberechnungsphase

Das Detailstufenauswahlverfahren von Funkhouser und Séquin stellt bereits eine obere Schranke für die Berechnungszeit des Bildes über die Kosten-Werte der Objekte sicher. Diese Kosten-Berechnung soll in dieser Arbeit unverändert bleiben. Dadurch kann man den Aufwand für das Detailstufenauswahlverfahren zum Vergleich von Nutzen-Funktionen als nichtzeitkritisch auffassen.

Häufig wird beispielsweise der $Größe(O)$-Faktor aus Gründen des Berechnungsaufwands durch die Anzahl der belegten Pixel des kleinsten umschließenden Rechtecks oder ähnlichen Verfahren approximiert. Um den Fehler durch Approximationen beim Vergleich von Nutzen-Funktionen möglichst gering zu halten, wurde sich in dieser Arbeit dazu entschieden, die Werte weitgehend exakt zu bestimmen. Dies geschieht unter Missachtung der dafür benötigten Rechenzeit. Bei tatsächlicher Anwendung der neuen Nutzen-Funktion müsste man jedoch weiterhin auf solche Approximationen zurückgreifen.

Für die exakte Bestimmung der Nutzen-Faktoren ist es im eben genannten Beispiel für den $Größe(O)$-Faktor notwendig, das Bild bereits einmal zu berechnen und die Anzahl der sichtbaren Pixel für das jeweilige Modell zu bestimmen. Aus der Menge der vom Modell belegten Pixel lässt sich auch der Mittelpunkt des Objekts auf dem Bild für die Berechnung der Nutzen-Faktoren $Fokus(O)$ und $Bewegung(O)$ bestimmen. In dieser Arbeit wird davon ausgegangen, dass der fokussierte Punkt des Betrachters nicht zusätzlich gemessen wird. Daher soll zur Näherung die Bildmitte als fokussierter Punkt betrachtet werden.

Da aus jedem zu betrachtenden Blickwinkel V_i für alle Objekte die Pixel ausgezählt werden müssen, wird in der Implementierung zu dieser Arbeit V_i jeweils einmal für jedes Objekt O_j berechnet. Dabei wird jedes Objekt mit einer konstanten Farbtonschattierung versehen, wobei lediglich O_j weiß und alle anderen Objekte in der selben Farbe wie der Hintergrund der Szene dargestellt werden. Ein solches Bild ist in Abbildung 4.1 zu sehen. Die Anzahl der weißen Pixel auf dem resultierenden Bild kann schließlich zum Bestimmen der Nutzen-Faktoren $Größe(O)$, $Fokus(O)$ und $Bewegung(O)$ genutzt werden.

Während der Lernphase muss der NEAT-Algorithmus jedes erzeugte künstliche neuronale Netz mit der Fitness-Funktion evaluieren. Jede Auswertung der Fitness-Funktion erfordert jedoch für jeden Blickwinkel eine Anwendung des Detailstufenauswahlverfahrens. Somit würde die Anzahl der zu berechnenden Bilder bereits bei kleinen Populationsgrößen und Epochenanzahlen in enorme Höhen steigen. Da allerdings die Ausgaben der im NEAT-Algorithmus zu evaluierenden künstlichen neuronalen Netze lediglich die Gewichtung der Nutzen-Faktoren verändert, können diese Faktoren bereits vor der Lernphase für jeden zu betrachtenden Blickwinkel in der Vorberechnungsphase bestimmt werden.

Auch das Festhalten der Kosten-Werte jeder Detailstufe jedes Objekts in der Vorberechnungsphase ist möglich. Bei der Implementierung wurde hierbei mit einer tatsächlichen Zeitmessung für das Berechnen eines Bildes mit dem jeweiligen Modell in der jeweiligen Detailstufe experimentiert. Die Experimente zeigten jedoch, dass diese Messung zu stark von der sonstigen Auslastung der Hardware durch andere Applikationen und dem Betriebssystem, sowie der Komposition der Szene abhängt.

Schließlich wurden die Kosten-Werte mit der von Funkhouser und Séquin vorgeschlagenen Funktion berechnet (vgl. Gleichung 2.2, Kapitel 2.1.2). Dabei wurden die Konstanten c_1, c_2 und c_3 experimentell bestimmt (siehe Tabelle 4.1). Die Hardwarekonfiguration für diese experimentelle Bestimmung basierte auf einem Intel i7-3770 Prozessor (4x3.40GHz) und einer NVIDIA GeForce GTX 660 Grafikkarte (2GB Grafikkartenspeicher).

Abbildung 4.1: Ansicht einer Kameraposition bei der Vorberechnung; nur das Objekt, dessen Größe bestimmt werden soll wird weiß, alle anderen Objekte schwarz dargestellt

Konstante	Wert (benötigte Zeit in ms)
c_1	2.5068e-07
c_2	5.0135e-07
c_3	1.13557e-07

Tabelle 4.1: experimentell bestimmte Konstanten für die Kosten-Wert-Berechnung nach Funkhouser und Séquin

Auch die in Kapitel 3.2 besprochenen „perfekten“ Bilder für alle zu evaluierenden Blickwinkel werden in dieser Vorberechnungsphase erzeugt. Dazu wird jegliches Modell auf seine höchste Detailstufe gesetzt und das daraus berechnete Bild abgespeichert.

4.2.2 Lernphase

In der Lernphase wird der NEAT-Algorithmus mit einer voreingestellten Populationsgröße, maximaler Anzahl an Epochen und den in der Vorberechnungsphase erhaltenen Daten gestartet.

In jeder Epoche wird eine neue Population von Genomen erstellt und von jedem dieser Genome die Fitness – wie in Kapitel 3.2 beschrieben – geprüft. In der hier vorgestellten Implementierung sind jedoch die Ausgaben der generierten Netze im Intervall $[0, 1]$. Bei der Verwendung dieser Werte als Gewichtung der Nutzen-Faktoren kann zwar keine Verstärkung

mehr auftreten, aber da alle Faktoren der neuen Nutzen-Funktion mit Gewichten versehen sind, wird angenommen, dass es auch reicht diese lediglich unterschiedlich stark zu dämpfen.

Übersteigt die Fitness eines Genoms eine ebenfalls vordefinierte Konstante $ZielFitness$, so findet der NEAT-Algorithmus bereits ein Ende. Andernfalls wird mit einer weiteren Epoche fortgefahren, solange noch nicht die maximale Anzahl an Epochen erreicht ist. Über alle getesteten Genome wird zusätzlich zu den für den NEAT-Algorithmus noch notwendigen Genomen, jenes mit dem höchsten gefundenen Fitness-Wert gespeichert.

Endet der NEAT-Algorithmus, so kann das gespeicherte Genom mit der höchsten Fitness und das damit beschriebene künstliche neuronale Netz für die Laufzeit genutzt werden. Mit der Konstanten $ZielFitness$ kann der Entwickler der Visualisierung bestimmen, wie nah das gewichtete Detailstufenauswahlverfahren versuchen soll, an die „perfekten“ Bilder heranzukommen. Für diese Arbeit wurde die Konstante mit einem künstlich hohen, nicht erreichbaren Wert festgelegt, um zu betrachten, wie gut die Fitness nach einer gewissen Rechenzeit werden kann.

4.2.3 Laufzeit

Während der Laufzeit kann das in der Lernphase erhaltene künstliche neuronale Netz verwendet werden, um nun auch die Gewichte für die Nutzen-Faktoren an nicht trainierten Stellen der Szene zu bestimmen. Anstelle dieser Phase werden alle im kommenden Kapitel 5 besprochenen Vergleiche und Auswertungen zwischen neuer und bisherigen Nutzen-Funktion vorgenommen.

5 Auswertung

Im Folgenden sollen anhand verschiedener Test-Szenen Vergleiche zwischen der bisherigen Nutzen-Funktion nach Funkhouser und Séquin [FS93] und der neuen, im Kapitel 3 vorgestellten Nutzen-Funktion gezogen werden. Zunächst werden dafür die den Lernprozess des künstlichen neuronalen Netzes am stärksten beeinflussenden Parameter betrachtet und sich aus deren Wahl ergebende Probleme besprochen. Basierend auf diesen Erkenntnissen soll im Anschluss die Güte der durch das neu beschrieben Verfahren generierten Bilder mit den vom ursprünglichen Algorithmus generierten Bildern verglichen sowie die neue Gewichtung der Nutzen-Faktoren betrachtet werden.

Alle in diesem Kapitel beschriebenen Tests wurden mit den folgenden Software- bzw. Hardwarespezifikationen durchgeführt:

- Betriebssystem: Windows 10 Pro 64-Bit-Version (10.0, Build 17134)
- Prozessor: Intel® Core™ i7-3770, 4x3.40GHz
- Grafikkarte: NVIDIA GeForce GTX 660, 2GB virtueller Arbeitsspeicher
- Hauptspeicher: 16GB

Da jedoch – wie bereits in Kapitel 4.2.1 beschrieben – der Vergleich der verschiedenen Nutzen-Funktionen nicht zeitkritisch ist, sollte die Hardwarekonfiguration, abseits vom Verhältnis der Kosten-Wert-Konstanten, keinen Einfluss auf die Messergebnisse haben. Alle Bilder für die in diesem Kapitel durchgeführten Tests wurden mit einer Auflösung von 1280x1024 Pixel berechnet.

5.1 Beschreibung der Testszenen

Für die Evaluation der verschiedenen Nutzen-Funktionen wurden drei Testszenen erstellt. Die erste Testszene ($Szene_1$) soll das Verhalten der Verfahren bei vielen gleichen Objekten darstellen. In der zweiten Testszene ($Szene_2$) werden teilweise unterschiedliche Objekte in verschiedenen Farben dargestellt, um im Vergleich zu $Szene_1$ auch Farbwahrnehmung mit einbeziehen zu können. Da sowohl $Szene_1$ als auch $Szene_2$ nicht sehr realitätsnah sind, soll mit Testszene 3 ($Szene_3$) ein modellhafter Ausschnitt der realen Welt exemplarisch betrachtet werden.

Alle Testszenen sollten zur besseren Vergleichbarkeit eine einheitliche statische Hintergrundfarbe aufweisen. Bei deren Wahl ist es vorteilhaft, darauf zu achten, welche Farben im Vordergrund der Szenen an sich zu sehen sind, damit eine gewisse wahrgenommene Distanz stets vorhanden ist. In realen Szenen wird dieser Effekt meist durch den Kontrast zum Hintergrund hervorgerufen. Als Hintergrundfarbe aller Szenen wurde daher ein Karminrot (Rotanteil 60%, Grünanteil 20%, Blauanteil 20%) gewählt.

$Szene_1$ enthält 25 Kopien eines Modells, welches einer deformierten Kugel ähnelt. Bei den gerundeten Oberflächen einer Kugel sind besonders Unterschiede in der Gleichmäßigkeit des

Schattierungsverlaufs bei verschiedenen Detailstufen der bläulichen Oberflächen wahrnehmbar. Jedes dieser Modelle besteht in seinen vier Detailstufen aus jeweils 2200, 35.224, 211.352 bzw. 1.409.024 Dreiecken. Die Kopien sind in einem 5x5 Raster angeordnet. In Abbildung 5.1 ist eine Übersicht über $Szene_1$ zu sehen.

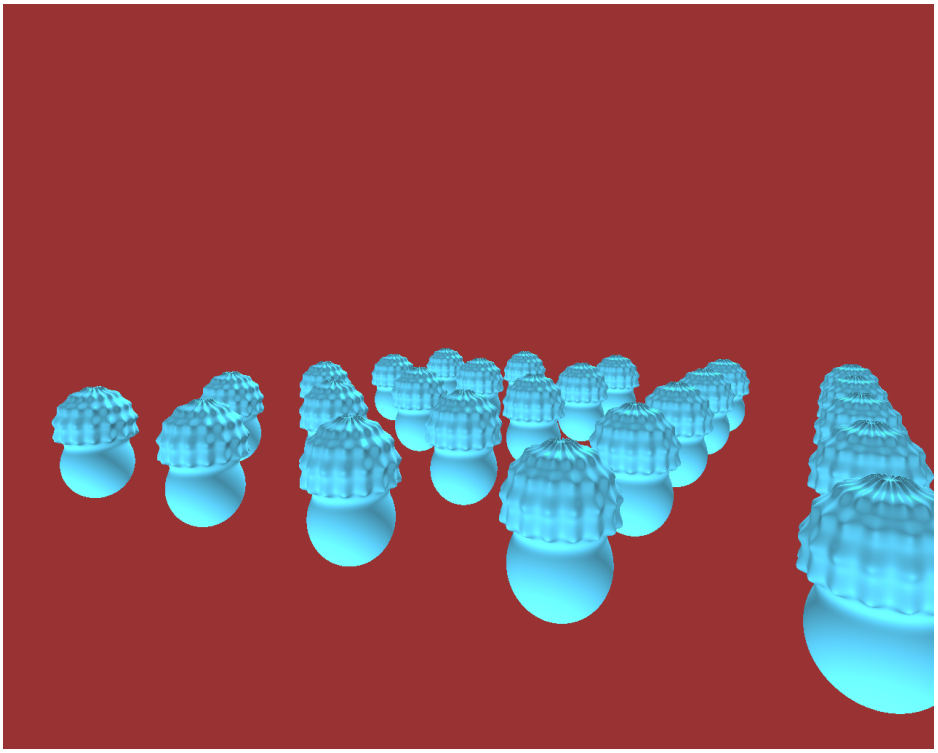


Abbildung 5.1: Überblick über die Testszene $Szene_1$, alle Objekte sind in ihrer höchsten Detailstufe dargestellt.

$Szene_2$ wurde aus frei verfügbaren Modellen des Standford Scanning Repository zusammengestellt. Die Modelle eines Hasen und eines Gürteltiers wurden in den Farbtönen Rot, Grün, Blau und Gelb jeweils 15 mal kopiert und auf den Punkten eines 15x15 Raster einmalig zufällig verteilt. Auch deren Rotation um die nach oben gerichtete Y-Achse wurde leicht variiert. Im Vergleich zu $Szene_1$ kann hier das Verhalten der Detailstufenauswahlverfahren bei nicht symmetrischen Objekten betrachtet werden. Besonders das Modell des Gürteltiers trägt bei einigen Blickwinkeln aufgrund seiner ausgestreckten vorderen Extremitäten zu interessanten Überdeckungen der Modelle bei. Die Objekte liegen in sechs verschiedenen Detailstufen vor, deren Detailgetreue in Tabelle 5.1 abgelesen werden kann. In Abbildung 5.2 ist eine Übersicht über $Szene_2$ zu sehen.

$Szene_3$ beschreibt eine Komposition von Modellen, wie sie in einer tatsächlichen Visualisierung einer Stadt vorkommen könnte. Die Testszene entstand aus dem frei zur Verfügung stehenden Cartoon Lowpoly City Free Game Pack 3DMModell von Anton Moek (Verfügbar unter `https://www.turbosquid.com/3d-models/uvw-polygons-unity-3d-model-1294342`). Die Objekte dieser Szene wurden künstlich mit einem Oberflächen-Unterteilungsmodifikator komplexer gestaltet, damit mehr Berechnungszeit notwendig ist. Ein Überblick über $Szene_3$ ist in Abbildung 5.3 gegeben.

Modell	Detailstufe	Anzahl der Dreiecke
Hase	0	69
	1	346
	2	1.736
	3	8.681
	4	43.405
	5	69.451
Gürteltier	0	344
	1	1.728
	2	8.648
	3	43.242
	4	216.214
	5	345.944

Tabelle 5.1: Anzahl der Dreiecke der verschiedenen Detailstufen der Modelle in $Szene_2$

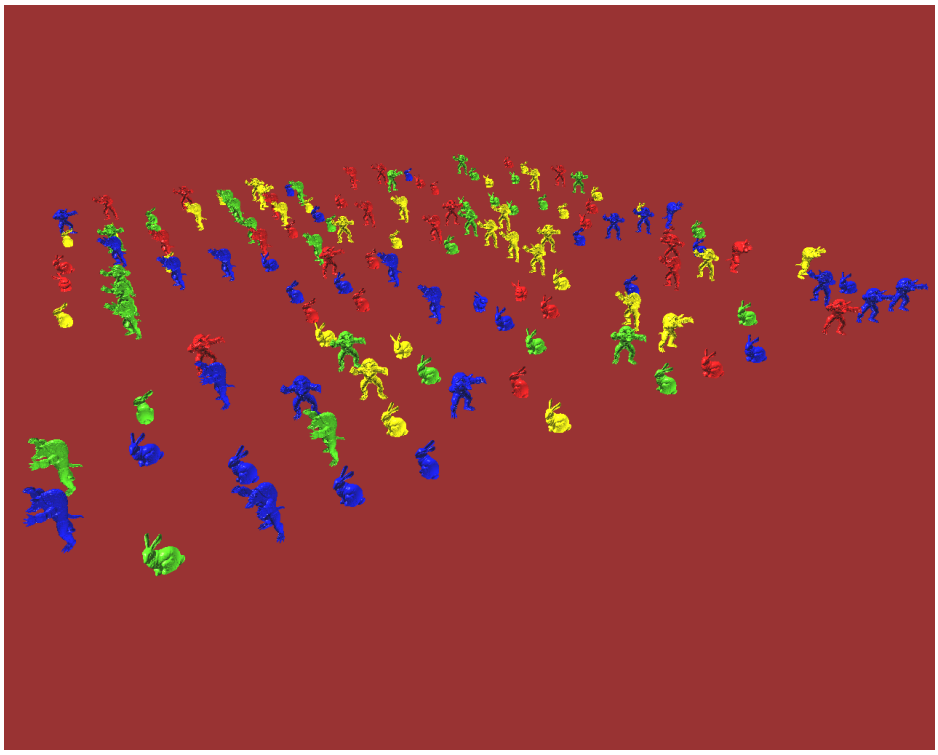


Abbildung 5.2: Überblick über die Testszene $Szene_2$, alle Objekte sind in ihrer höchsten Detailstufe dargestellt.

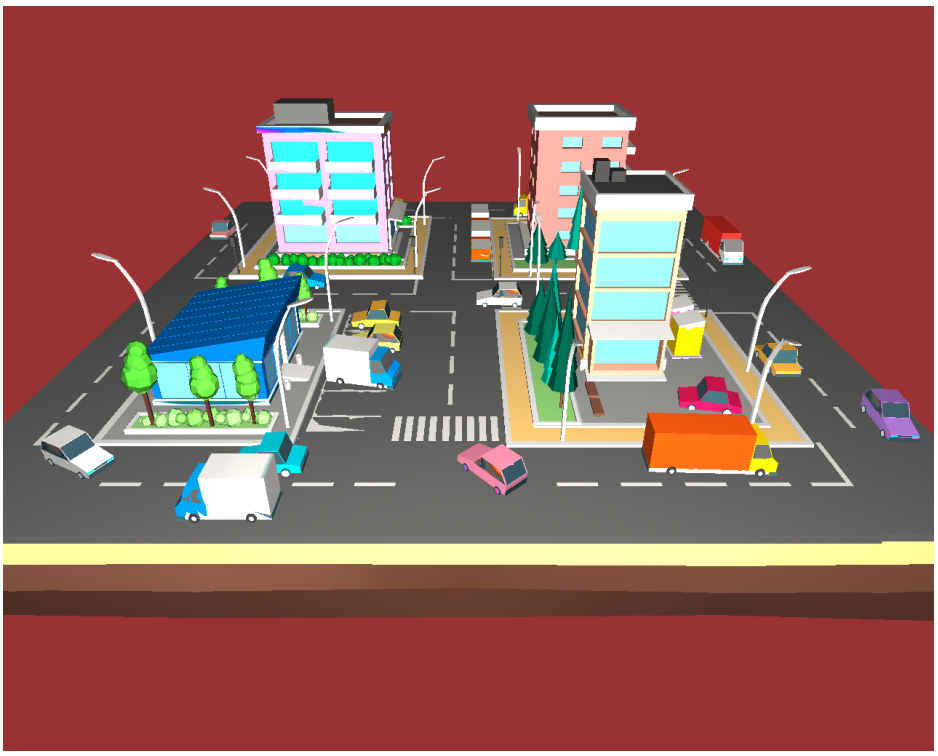


Abbildung 5.3: Überblick über die Testszene $Szene_3$, alle Objekte sind in ihrer höchsten Detailstufe dargestellt.

In jeder der Testszenen wurde eine Reihe von Kamerafahrten definiert. Diese stellen exemplarisch die Interaktion möglicher Betrachter mit den Szenen dar. Als Blickwinkel zum Entwickeln eines künstlichen neuronalen Netzes durch den NEAT-Algorithmus werden pro Testszene eine diskrete Anzahl an Punkten entlang drei bis vier dieser Kamerafahrten verwendet. Diese Kamerafahrten werden im folgenden auch als die *angelernten* Kamerafahrten bezeichnet. Die übrigen Fahrten können als Vergleich der Performanz des Detailstufenauswahlverfahren unter Verwendung der alten oder neuen Nutzen-Funktion dienen und werden *nicht angelernte* Kamerafahrten genannt.

5.2 Auswertung des Lernprozesses

Der NEAT-Algorithmus hat diverse Parameter, die z.B. die Erzeugung der künstlichen neuronalen Netze oder deren Unterteilung in Spezies beeinflussen. Diese Parameter tragen maßgeblich dazu bei, wie schnell ein besonders gut bewertetes künstliches neuronales Netz gefunden werden kann. Besonders die Anzahl der Epochen und Populationsgröße jeder Epoche bestimmen, wie lange mit den jeweiligen Parametern gesucht werden soll. In diesem Unterkapitel sollen zunächst lediglich die Ergebnisse der Lernprozesse bezüglich dieser Parameter verglichen werden um herauszufinden, welche Konfiguration am sinnvollsten ist, um im Anschluss mit der bisherigen Nutzen-Funktion verglichen zu werden.

In Abbildung 5.4 ist ein Diagramm basierend auf den geringsten mittleren Bildfehlern in $Szene_1$ bei verschiedenen Testreihen in Abhängigkeit der berechneten Epochen abgebildet.

Die Testreihen unterscheiden sich in ihrer maximalen Epochenanzahl und Populationsgröße. Während Testreihe 1 100 Epochen mit je 10 Genomen betrachtet, zeigt Testreihe 2 80 Epochen mit je 20 Genomen und Testreihe 3 40 Epochen mit je 50 Genomen. Für jede dieser Testreihen wurde der Lernprozess zehnfach durchgeführt. Im Diagramm sind jeweils die Durchschnitte der geringsten mittleren Bildfehler dargestellt. Die Standardabweichung zu diesen Durchschnittswerten werden in Abbildung 5.5 visualisiert.

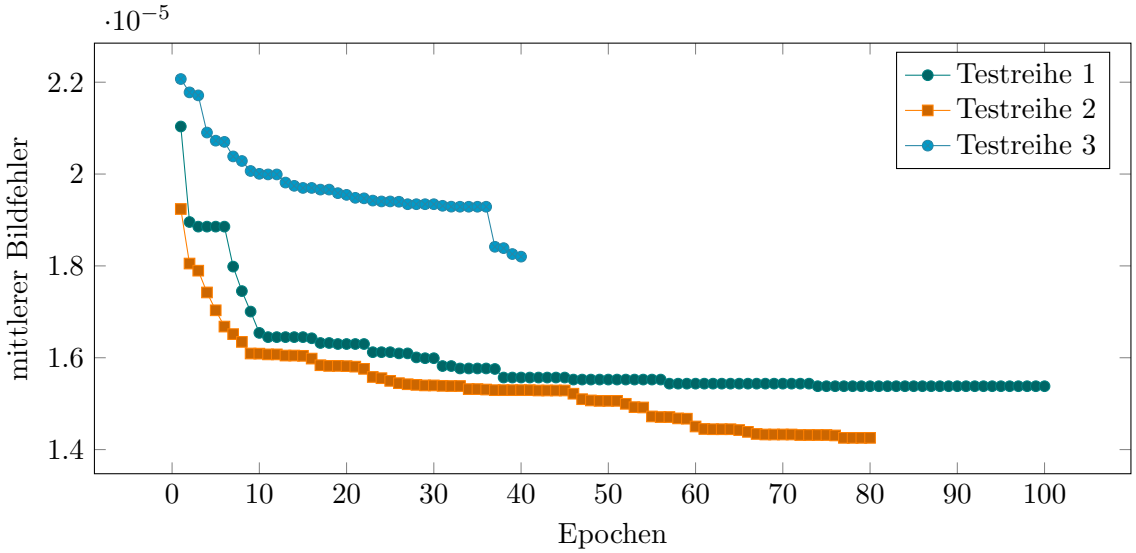


Abbildung 5.4: Durchschnitte der geringsten mittleren Bildfehler bei zehnfacher Durchführung des NEAT-Lernprozesses in Abhängigkeit von der Anzahl absolvierter Epochen in $Szene_1$

In den dargestellten Testreihen 1, 2, bzw. 3 wurden insgesamt 1000, 1600, bzw. 2000 Genome evaluiert. In Abbildung 5.4 ist deutlich zu erkennen, dass bei Testreihe 1 ab Epoche 60 kaum noch Verbesserungen auftreten. Aufgrund der niedrigen Populationsgröße kommt es kaum zur Neubildung von Spezies und somit entstehen weniger Innovationen. Testreihe 3 zeigt, dass durch eine hohe Populationsgröße zwar häufig Innovationen auftreten, sich jedoch auch mehr Spezies bilden, welche keine Innovation hervorbringen und bei geringer maximaler Epochenanzahl nicht rechtzeitig ausgeschlossen werden können. Dies wird besonders im Übergang von Epoche 36 zu 37 ersichtlich, in welchem in jdem Durchlauf erstmals nicht zielführende Spezies abgestoßen wurden. Des Weiteren legt das Diagramm die Vermutung nahe, dass der geringste Mittlere Bildfehler der getesteten Netze im Laufe der Epochen einem exponentiellen Zerfall folgt. Dieser Vermutung folgend, würde man bei einer weiteren Erhöhung der maximale Epochenanzahl in Testreihe 3 ab einem gewissen Punkt zu besseren Ergebnissen als Testreihe 2 gelangen. Treten solche rasanten Verbesserungen des mittleren Bildfehlers wie zwischen Epoche 36 und 37 in Testreihe 3 auch in späteren Epochen noch häufiger auf, ist diese Vermutung auch mit den dargestellten Daten vereinbar. Allerdings würde mit der Erhöhung der maximalen Epochenanzahl für Testreihe 3 auch der insgesamt nötige Rechenaufwand stark steigen. Die Betrachtung dieser Testreihen zeigt also, dass – für einen gegebenen maximalen Rechenaufwand von 1600 zu betrachtenden Genomen – das Verhältnis von Populationsgröße zu maximaler Epochenanzahl in Testreihe 2 am effizientes-

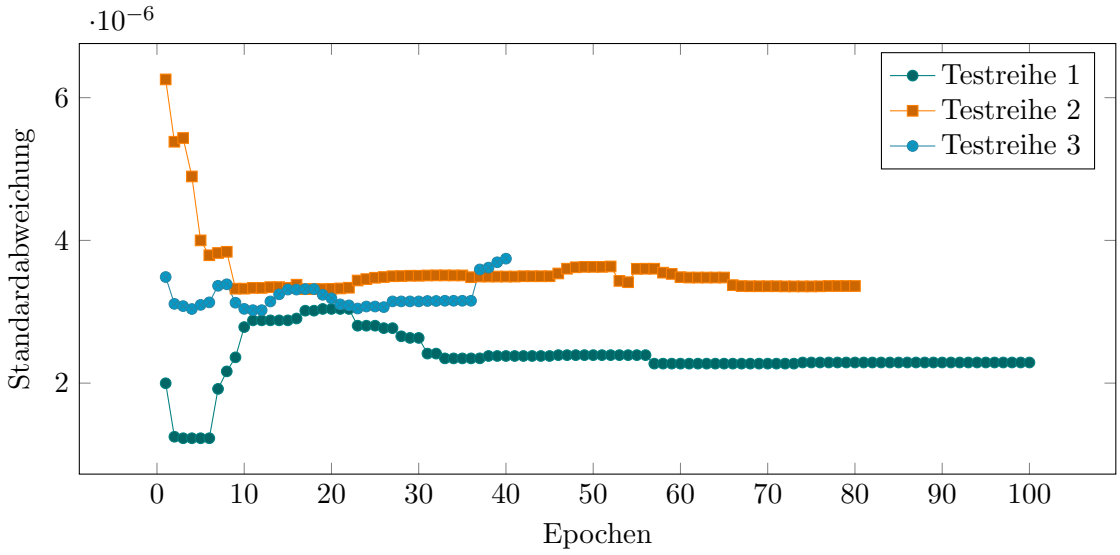


Abbildung 5.5: Standardabweichungen der durchschnittlichen Bildfehler bei zehnfacher Durchführung des NEAT-Lernprozess in Abhängigkeit von der Anzahl absolvierter Epochen in $Szene_1$

ten genutzt wird.

Die in Abbildung 5.5 dargestellte Standardabweichung der Durchschnitte der geringsten mittleren Bildfehler zeigt, dass alle drei Testreihen annähernd gleichbleibende Ergebnisse liefern. Man kann also davon ausgehen, dass unabhängig vom Verhältnis der Populationsgröße zur maximalen Epochenanzahl die Güte des geringsten mittleren Bildfehlers selbst bei Wiederholung des Lernprozesses gleichbleibend ist. Allerdings ist auch hier erkennbar, dass die Standardabweichung für ein effizienteres Verhältnis geringer ausfällt.

Um die oben genannte These, dass der geringste mittlere Bildfehler im Laufe der Epochen einem exponentiellen Zerfall folgt, zu stützen, sind in Abbildung 5.6 die Durchschnitte der geringsten mittleren Bildfehler im Laufe der absolvierten Lernepochen von vier weiteren Testreihen dargestellt. Die Durchschnitte beziehen sich ebenfalls wieder jeweils auf die zehnfache Durchführung des NEAT-Lernverfahrens. Die jeweils verwendeten Parameter der Testreihen 4 bis 6 sind in der Tabelle 5.2 zu sehen.

Testreihe	**maximale Epochenanzahl**	**Populationsgröße**
4	80	15
5	80	25
6	70	20
7	90	20

Tabelle 5.2: maximale Epochenanzahl und Populationsgröße für verschiedene Testreihen des NEAT-Lernverfahrens

Alle in Abbildung 5.6 dargestellten Testreihen weisen das bereits zuvor vermutete Verhalten auf: in den ersten Epochen der Lernphase fällt der geringste gefundene mittlere Bildfehler

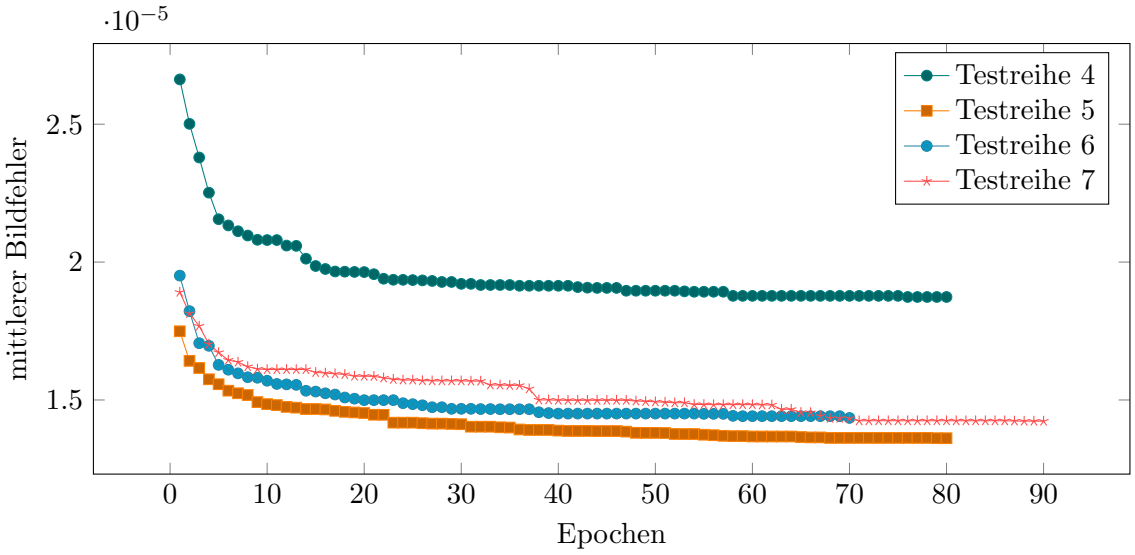


Abbildung 5.6: Durchschnitte der geringsten mittleren Bildfehler bei zehnfacher Durchführung des NEAT-Lernprozesses in Abhängigkeit von der Anzahl absolvierter Epochen in $Szene_1$

pro Epoche noch erheblich ab. Dieser Abfall nimmt im Verlauf der Lernphase bis auf wenige Ausnahmen stetig ab. Dieses Verhalten ist darauf zurückzuführen, dass in der ersten Epoche des Lernverfahrens lediglich zufällig generierte Genome evaluiert werden. Diese sind in der Regal an die gegebene Problemstellung nur sehr schlecht angepasst und somit besteht vor allem zu Beginn der Lernphase besonders viel Potential zur Optimierung der Genome. Im Laufe der Lernphase entstehen Genome, welche immer besser an die Testfälle angepasst sind und somit wird es zunehmend schwieriger neue Genome zu finden, welche einen noch geringeren mittleren Bildfehler erreichen.

Des weiteren sieht man bei den Ergebnissen von Testreihe 4 bis 7 deutlich, dass Lernverfahren, bei denen die maximale Epochenanzahl oder auch die Populationsgröße erhöht wird, jeweils zu Genomen mit geringeren mittleren Bildfehlern am Ende der Lernphase führen. In beiden Fällen ist dies darauf zurückzuführen, dass durch die Erhöhung einer der Parameter auch mehr Genome insgesamt getestet wurden und somit auch mehr Potential besteht, besser angepasste Genome zu finden. Andererseits wächst mit mehr getesteten Genomen natürlich auch die benötigte Laufzeit des NEAT-Verfahrens. Wie aber bereits mit Testreihe 3 gezeigt wurde ist auch der Verhältnis der beiden Parameter zueinander relevant. Die Standardabweichungen zu den in Abbildung 5.6 gezeigten Durchschnitten sind im Anhang dargestellt.

Abbildung 5.7 zeigt den mittleren Bildfehler in Abhängigkeit der berechneten Epochen bei den 10 Durchläufen von Testreihe 2. Man erkennt, dass beim mehrmaligen Ausführen des NEAT-Algorithmus die angelernten künstlichen neuronalen Netze ähnliche mittlere Bildfehler aufweisen. Lediglich eine aus zehn Durchführung weist deutliche negative Abweichungen auf. Betrachtet man diesen schlimmsten Fall (4. Durchlauf) näher, erkennt man, dass bereits die erste Epoche erheblich schlechtere Ergebnisse als andere Durchläufe geliefert hat. Dies ist auf eine schlechte Wahl der zufälligen Startgewichte der Netze der ersten Epoche

zurückzuführen. Mit einer höheren Populationsgröße (und für Effizienz auch damit einhergehenden höheren maximalen Epochenanzahl) ist es möglich, solche Effekte zu reduzieren. Allerdings zeigt der 1. Durchlauf, dass selbst mit einem erhöhten mittleren Bildfehler in der ersten Epoche vergleichsweise gute Ergebnisse erzielt werden können. Im kommenden sollen weiterhin die gewählten Parameter für Populationsgröße und maximale Epochenanzahl aus Testreihe 2 verwendet werden, da davon auszugehen ist, dass diese auch bei weiteren Tests repräsentative Ergebnisse liefern.

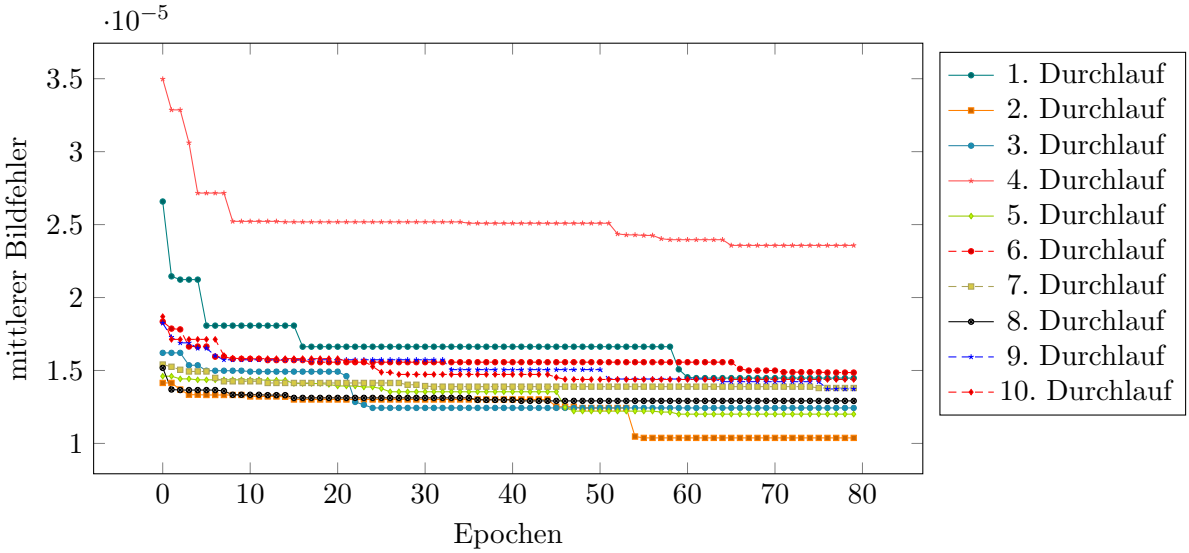


Abbildung 5.7: Geringste mittlere Bildfehler beim Lernprozess des NEAT-Algorithmus in Abhängigkeit von der Anzahl absolvierter Epochen in $Szene_1$ für eine maximale Epochenanzahl von 80 Epochen und Populationsgröße 20

Die bisher in diesem Kapitel dargestellten Messwerte und Beobachtungen entstammen allesamt lediglich Experimenten aus $Szene_1$. Die Beschriebenen Testreihen wurden im Rahmen dieser Arbeit ebenfalls bei $Szene_2$ und $Szene_3$ durchgeführt. Dabei konnten auch wieder die bisherigen Beobachtungen bezüglich der Parameter für den Lernprozess festgestellt werden. Im Anhang sind die dazu gehörigen Abbildungen dargestellt. Somit ist davon auszugehen, dass die beschriebenen Phänomene nicht nur ein Nebeneffekt der Szenenkomposition sind, sondern auf ein zugrunde liegendes Prinzip hinweisen.

5.3 Evaluation der neuen Nutzen-Funktion

Die in dieser Arbeit vorgestellte Nutzen-Funktion fügt zu dem Berechnungsaufwand pro Bild im Vergleich zur Original-Nutzen-Funktion lediglich die Aktivierung des künstlichen neuronalen Netzes, sowie die Potenzierung der Nutzen-Faktoren hinzu. Beides ist vergleichsweise vernachlässigbar. Vergleiche der Berechnungskosten verschiedener Detailstufenauswahlverfahren wurden bereits durchgeführt [FS93]. Um die Ergebnisse dieser Algorithmen zu vergleichen, sollen daher an dieser Stelle lediglich die damit berechneten Bilder betrachtet werden. Eine solche Evaluation unterliegt derselben Problematik wie die Kapitel 3.2 beschriebene Auswahl einer Fitness-Funktion: eigentlich müsste die Wahrnehmung der Bilder durch

Probanden getestet werden. Auch hier soll der gleiche Lösungsansatz wie bei der Fitness-Funktion gewählt werden. Betrachtet man den mittleren Bildfehler der erzeugten Bilder eines Detailstufenauswahlverfahrens im Vergleich zu optimalen Bildern der selben Situation, kann man den entstehenden Wert zwischen verschiedenen Algorithmen betrachten und auswerten.

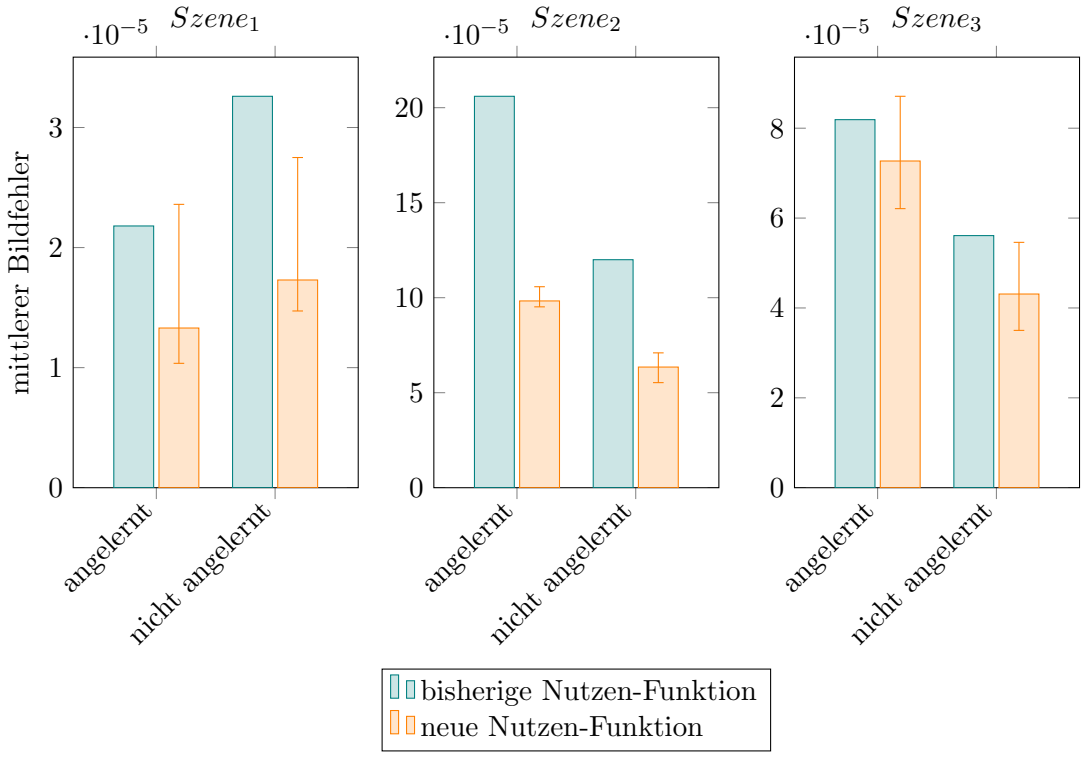


Abbildung 5.8: Vergleich des mittleren Bildfehlers zwischen der bisherigen Nutzen-Funktion, sowie dem besten und schlimmsten Fall bzw. Durchschnitt von Durchläufen mit der neuen Nutzen-Funktion aufgeschlüsselt nach Testszene, sowie angelernten und nicht angelernten Kamerafahrten

In Abbildung 5.8 wird der mittlere Bildfehler der bisherigen und neuen Nutzen-Funktion in allen Testszenen sowohl bei angelernten, als auch bei nicht angelernten Kamerafahrten dargestellt. Die bläulichen Balken zeigen jeweils die Ergebnisse der Funktion nach Funkhouser und Séquin. Die orangefarbenen Balken zeigen den durchschnittlichen mittleren Bildfehler bei jeweils 10 Durchläufen des Lernprozesses und der neuen Nutzen-Funktion. Des Weiteren ist bei den orangefarbenen Balken das Intervall des schlechtesten, sowie besten gemessenen mittleren Bildfehlers dargestellt. Bei der bisherigen Nutzen-Funktion ist ein solches Intervall nicht notwendig, da die Ergebnisse konstant sind. Man erkennt, dass in allen Szenen im Schnitt der mittlere Bildfehler durch die neue Nutzen-Funktion sowohl bei angelernten, als auch bei nicht angelernten Kamerafahren gesenkt werden kann. Bei den gewählten Fahrten in $Szene_1$ und $Szene_2$ ist damit sogar nahezu eine Senkung der Fehlers auf die Hälfte bezüglich der bisherigen Nutzen-Funktion möglich.

In $Szene_3$ fällt die Senkung des mittleren Bildfehlers geringer aus. $Szene_1$ und $Szene_2$ besitzen im Vergleich zu $Szene_3$ besonders viele rundliche Oberflächen, auf denen Schat-

tierungsverläufe auftreten. Durch solche Verläufe fallen niedrigere Detailstufen besonders auf, da die Kontraste zwischen den Schattierungen zweier benachbarter Polygone stärker sind. Während des Lernprozesses werden die künstlichen neuronalen Netze bereits auf dadurch auftretende wahrgenommene Fehler trainiert und liefern daher bessere Ergebnisse im Vergleich zur alten Nutzen-Funktion. In $Szene_3$ gibt es jedoch nur wenige rundlich wirkende Oberflächen, wodurch die neue Nutzen-Funktion den mittleren Bildfehler nicht so stark reduzieren kann.

Die größere Streuung der Ergebnisse verschiedener Durchläufe in $Szene_1$ und $Szene_3$ kommt in beiden Szenen durch jeweils einzelne zufällig schlecht verlaufene Lernprozesse zustande. Bei dem Durchlauf mit dem größten mittleren Bildfehler bei angelernten Kamerafahrten in beiden Szenen, liegt dieser Fehler über dem mittleren Bildfehler der bisherigen Nutzen-Funktion. Eben jene Durchläufe korrelieren ebenfalls mit dem größten mittleren Bildfehler bei nicht angelernten Kamerafahrten. Daraus folgt: Durchläufe, welche vergleichsweise schlechte Ergebnisse bei angelernten Kamerafahrten liefern, schneiden ebenfalls schlecht bei nicht angelernten Kamerafahrten ab. Da jedoch die Durchschnitte der Ergebnisse durchweg deutlich unter den schlechtesten Verläufen liegen, ist es in solchen Fällen ratsam, den Lernprozess zu wiederholen und gegebenenfalls die Populationsgröße und maximale Epochenanzahl zu erhöhen.

Ein möglicher Effekt, der bei Lernprozessen von künstlichen neuronalen Netzen auftreten kann, ist die übermäßige Anpassung an Testszenarien. Trainiert man ein Netz besonders lange auf eine diskrete Menge von Eingaben, so kann es passieren, dass besonders viele Neuronen gebildet werden, welche jedoch lediglich gute Fitness-Werte bei genau den Eingaben aus ebenjener Menge liefern. Ändert man die Eingaben bereits geringfügig, sinkt der berechnete Fitness-Wert drastisch und das künstliche neuronale Netz eignet sich nicht, um damit auch nicht antrainierte Fälle eines Problems zu lösen. Einem solchen Effekt könnte man bei dem neu vorgestellten Verfahren durch die Reduktion der Populationsgröße und maximaler Epochenanzahl oder Erhöhung der Menge der anzulernenden Kamerapositionen entgegenwirken. Bei den gezeigten Ergebnissen ist die übermäßige Anpassung jedoch auszuschließen, da durchweg alle Lernprozesse aller Testszenen ähnliche Ergebnisse für angelernte und nicht angelernte Kamerafahrten liefern (vgl. Abbildung 5.8).

In Abbildung 5.9 sind generierte Bilder einer nicht angelernten Kameraposition aus $Szene_2$ und Pixel, in denen diese sich vom Optimalbild unterscheiden, dargestellt. Es ist erkennbar, dass die Anzahl der Pixel, welche farbliche Unterschiede aufweisen, ungefähr gleich ist. Jedoch beträgt der mittlere Bildfehler bei dem Bild, welches mit der bisherigen Nutzen-Funktion generiert wurde $8.25e-05$ und der mittlere Bildfehler bei dem gezeigten Bild nach Generierung mit dem neuen Verfahren nur $6.62e-06$. Beide Werte sind jedoch sehr gering und es ist nur bei genauer Betrachtung des Bildes tatsächlich wahrzunehmen, dass überhaupt Unterschiede zum Optimalbild vorliegen. Genau dieser Effekt soll im Allgemeinen durch Detailstufenauswahlverfahren erreicht werden und mit dem neu vorgestellten Algorithmus sollte es möglich sein, ihn auch bei noch komplexeren Szenen als bisher zu erhalten.

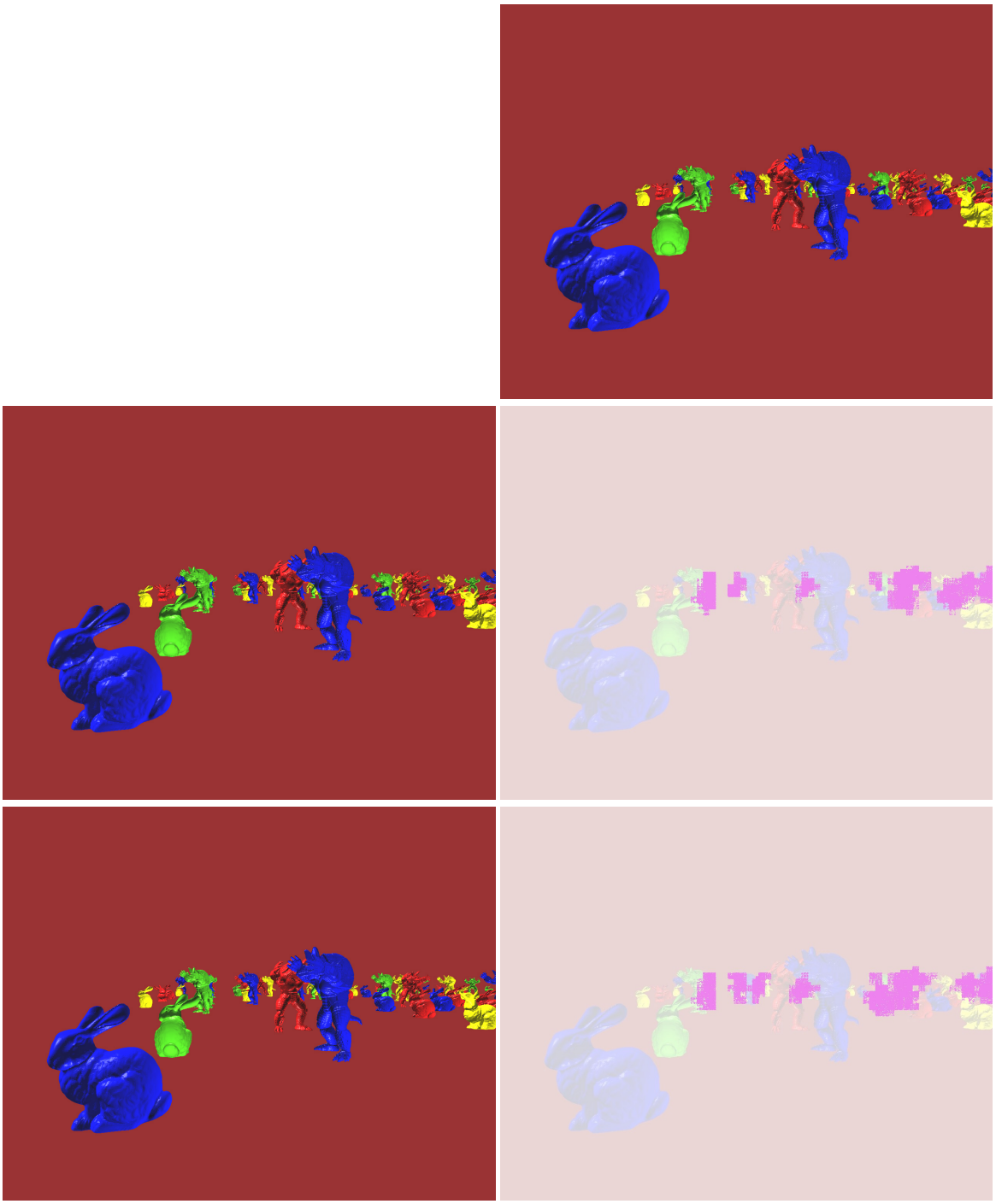


Abbildung 5.9: Generierte Bilder eines Blickwinkels einer nicht angelernten Kamerafahrt in $Szene_2$, sowie Unterschiedsbilder, auf denen Pixel die sich zum Optimalbild unterscheiden rosa hervorgehoben sind; Optimalbild (oben rechts), Bild nach bisheriger Nutzen-Funktion (mittig links), Unterschiedsbild bei bisheriger Nutzen-Funktion (mittig rechts), Bild nach neuer Nutzen-Funktion (unten links), Unterschiedsbild bei neuer Nutzen-Funktion (unten rechts)

5.4 Gewichtung der Nutzen-Faktoren

Bei der bisherigen Auswertung wurden vornehmlich die Resultate der neuen Nutzen-Funktion betrachtet. In diesem Unterkapitel sollen die Ausgaben der mit dem NEAT-Verfahren entwickelten künstlichen neuronalen Netze, welche zur Gewichtung der neuen Nutzen-Funktion verwendet wurden (siehe Kapitel 3.1), verglichen werden.

Der in dieser Arbeit betrachteten neuen Nutzenfunktion liegt die von Funkhouser und Séquin aufgestellte Behauptung, dass eine Gewichtung der Parameter von der Szene abhängig ist, zugrunde [FS93]. Zusätzlich wird davon ausgegangen, dass die Gewichtung abhängig von der Interaktion des Nutzers mit der Szene, oder zumindest von der Postion des Betrachters in der Szene ist.

In Abbildung 5.10 ist sind die Durchschnitte der Ausgaben der angelernten künstlichen neuronalen Netze bei den zehn Durchläufen des NEAT-Verfahrens dargestellt. Bei diesen Durchschnitten gehen jeweils die Ausgaben der Netze für die angelernten, als auch nicht angelernten Kamerapositionen mit ein.

In der Abbildung ist abzulesen, dass einige Faktoren der neuen Nutzen-Funktion im Durchschnitt stärker gewichtet werden als andere. So fällt zum Beispiel auf, dass der Hysterese-Faktor fast durchweg höher gewichtet wird als alle anderen Faktoren. Eine mögliche Erklärung dafür wäre, dass, durch das Weglassen des von Funkhouser und Séquin noch beschriebenen Genauigkeits-Faktor, der Hysteres-faktor der einzige verbleibende Faktor ist (vgl. Kapitel 3.1), der die verschiedenen Detailstufen eines Objekt unterschiedlich bewertet. Alle anderen Faktoren beziehen sich jeweils nur auf das Objekt in seiner höchsten Detailstufe.

Betrachtet man die ebenfalls in Abbildung 5.10 dargestellte Standardabweichung, fällt zusätzlich auf, dass häufig eine Standardabweichung von nahezu 0,5 erreicht wird. Dies resultiert daraus, dass häufig die Ausgaben der angelernten künstlichen neuronalen Netze sehr nah an 0 bzw. 1 liegen und zwischen diesen Extremen springen. In Tabelle 5.3 sind beispielsweise die Ausgaben eines angelernten künstlichen neuronalen Netzes für eine spezielle Kameraposition angegeben.

Variable	**Ausgabe des künstlichen neuronalen Netzes**	**Gewichtung für Faktor**
λ_1	2.5068e-07	Größe
λ_2	5.0135e-07	Fokus
λ_3	1.13557e-07	Geschwindigkeit
λ_4	1.0e-01	Hysterese

Tabelle 5.3: Ausgaben des künstlichen neuronalen Netzes nach dem fünften Durchlauf des NEAT-Lernverfahren für die zweite Kameraposition der zweiten angelernten Kamerafahrt

Somit werden in vielen Situationen einzelne Faktoren der neuen Nutzen-Funktion durch deren Gewichte fast vollständig aus- oder eingeschaltet.

Am Anhang zu dieser Arbeit sind zusätzlich die Durchschnitte und Standardabweichungen der Ausgaben der künstlichen neuronalen Netze nach den NEAT-Lernprozessen in $Szene_2$ und $Szene_3$ dargestellt. Alle hier beschriebenen Betrachtungen treffen auf diese Szenen ebenso zu.

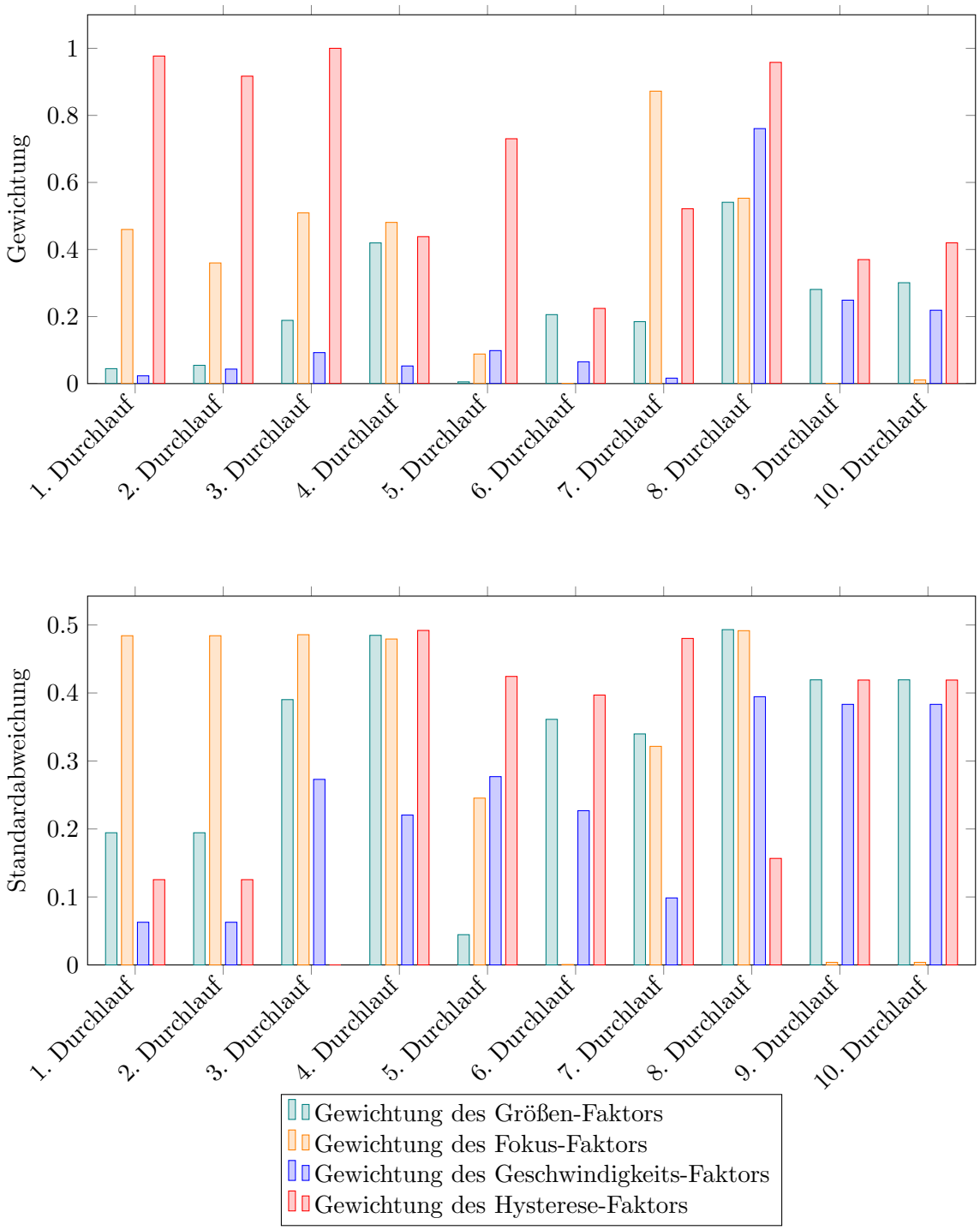


Abbildung 5.10: Durchschnitte und Standardabweichungen über alle betrachteten Kamerapositionen der Ausgaben des künstlichen neuronalen Netzes bei den zehn Durchläufen des NEAT-Algorithmus in $Szene_1$ (maximale Epochenanzahl 80, Populationsgröße 20)

6 Zusammenfassung & Ausblick

In diesem Kapitel sollen die Ergebnisse dieser Arbeit noch einmal zusammengetragen und ein Ausblick auf weitere mögliche Forschungsthemen, welche in Zusammenhang mit dem hier vorgestellten Verfahren stehen, gegeben werden.

Die bisherige Nutzen-Funktion nach Funkhouser und Séquin [FS93] wurde mit einer gewichteten Nutzen-Funktion, deren Gewichte dynamisch durch ein entsprechend trainiertes künstliches neuronales Netz bestimmt werden, ersetzt. Beim Vergleich der ursprünglichen und neuen Nutzen-Funktion bezüglich eines neu definierten, wahrnehmungsbasierten mittleren Bildfehlers konnte damit eine Senkung dieses Fehlers um bis zu 50% erreicht werden.

Die Ergebnisse der neuen Nutzen-Funktion hängen in hohem Maße von dem Resultat des NEAT-Lernprozesses ab. Je mehr Berechnungszeit man dem Lernprozess zur Verfügung stellt, desto besser können die Resultate im Durchschnitt ausfallen. Jedoch sollte dabei auch beachtet werden, dass eine übermäßige Anpassung des generierten Netzes an die gegebenen Kamerapositionen denkbar ist. Für ein Maximum von 1600 zu testenden Genomen während des Lernprozesses haben sich eine Populationsgröße von 20 und maximale Epochenanzahl von 80 am effizientesten für die hier vorgestellten Testfälle erwiesen.

Die in dieser Arbeit verwendeten Testszenen waren nur statische Szenen ohne Eigenbewegung oder anderweitige Veränderung der dargestellten Objekte. Weitere Arbeiten könnten beispielsweise dynamische Szenen betrachten. Damit verbunden wären auch andere Interaktionsmöglichkeiten mit der Visualisierung denkbar, welche – neben der hier betrachteten Position des Nutzers in der Szene – als zusätzliche Eingabeparameter für die anzulernenden künstlichen neuronalen Netze verwendet werden könnten.

Eine der größten möglichen Fehlerquellen des vorgestellten Verfahrens ist die Bestimmung des Wahrnehmungsunterschieds – dem mittleren Bildfehler – zwischen zwei Bildern. Die dafür entwickelte Formel dient als einziges Merkmal, nach dem die Genome im NEAT-Algorithmus beurteilt werden. Die entwickelte Formel stützt sich grundlegend auf drei weiter zu ergründenden Thesen:

- die Farbwahrnehmungsunterschiede verschiedener Farben werden durch die euklidische Farbdistanz im DIN99o-Farbraum korrekt abgebildet,
- der Mensch nimmt alle Bereiche des Bildes zu gleichen Teilen wahr und jedes Pixel – egal ob am Rand oder im Zentrum des Visualisierungsdarstellung – ist gleich gewichtet,
- und der Mensch nimmt lediglich Farbunterschiede pro Pixel und nicht als gleichartig gefärbte Bereiche wahr.

Zum aktuellen Zeitpunkt existieren beim ersten Punkt keine weiteren Verbesserungsmöglichkeiten. Sollten künftig, trotz der bereits langjährigen Arbeit auf dem Gebiet der Farbwahrnehmung, bessere wahrnehmungsbasierte Farbdistanzfunktionen entwickelt werden, wäre dies noch einmal auf den hier gezeigten Algorithmus zu übertragen.

Beim zweiten und dritten Punkt sollte ebenfalls noch weitere Forschungsarbeit geleistet werden. Abhängig vom Fokuspunkt der Augen eines Nutzers müssten verschiedene Pixel gewichtet werden. Beispielsweise wäre es möglich, sowohl das zu vergleichende Bild, als auch das Optimalbild vor dem Vergleich mit einer Fischaugen-Projektion zu verzerren. Das Zentrum für eine solche Verzerrung würde dann den Blickpunkt des Nutzers auf der Visualisierungsdarstellung entsprechen und die Stärke der Verzerrung könnte mit der Brennweite der Augen des Nutzers korreliert werden.

Da bei modernen Displays die Bildpunkte häufig so dicht gestreut sind, dass ein menschliches Auge den Unterschied zweier Nachbarn nicht mehr feststellen kann, ist wohl der dritte Punkt am leichtesten zu widerlegen. Verschiebt sich beispielsweise auf einem Schwarz-Weiß-Bild eine fünf Pixel lange und ein Pixel hohe horizontale Linie um einen Pixel nach unten, so entstehen nach der definierten Funktion Farbunterschiede auf 40 Pixeln, obwohl das Auge diese Verschiebung womöglich nicht wahrnimmt. Hier könnte man entweder die Nachbarschaft jedes Pixels mit abnehmender Gewichtung ebenfalls vergleichen oder sich beispielsweise Verfahren der Mustererkennung bedienen, indem man die Bilder in Farbsegmente untergliedert und das Größenverhältnis benachbarter Segmente untersucht.

Vorteilhaft an dem in dieser Arbeit vorgestellten Verfahren ist auch zu erwähnen, dass durch die Verwendung von künstlichen neuronalen Netzen zur Bestimmung der Gewichte der Nutzen-Funktion-Faktoren mit Leichtigkeit zusätzliche Faktoren in die Nutzen-Funktion aufgenommen werden können oder deren Gewichtung von anderen Parametern abhängig gemacht werden kann. Lediglich die Anzahl der Eingabe- und Ausgabe-Neuronen der erwarteten künstlichen neuronalen Netze würde sich dadurch ändern und ein erneuter Durchlauf des NEAT-Lernverfahrens könnte solche Netze problemlos generieren.

Von Stanley und Miikulainen wurde 2006 auch eine Adaption des NEAT-Lernverfahrens vorgestellt, bei dem der Lernprozess eines künstlichen neuronalen Netzes zeitgleich zur Laufzeit der Anwendung des Netzes in Echtzeit läuft [SBM05]. Dies würde allerdings auch mit der Notwendigkeit für einen alternativen wahrnehmungsbasierten Laufzeit-Bewertungsalgorithmus für den Lernprozess einhergehen.

Abschließend bleibt zu sagen, dass trotz einigen Simplifizierungen des Wahrnehmungsmodells und möglicherweise besser geeigneter Lernverfahren für das genutzte künstliche neuronale Netz in dieser Arbeit ein Verfahren gezeigt werden konnte, welches beim Detailstufenauswahlverfahren nach Funkhouser und Séquin besser wahrgenommene Bilder produzieren kann.

Abbildungsverzeichnis

Literaturverzeichnis

[BD80] Bowmaker, J K. ; Dartnall, H J.: Visual pigments of rods and cones in a human retina. In: *The Journal of Physiology* 298 (1980), Nr. 1, S. 501–511. http://dx.doi.org/10.1113/jphysiol.1980.sp013097. – DOI 10.1113/jphysiol.1980.sp013097

[CGR91] Carpenter, Gail A. ; Grossberg, Stephen ; Reynolds, John H.: ARTMAP: Supervised real-time learning and classification of nonstationary data by a self-organizing neural network. In: *Neural networks* 4 (1991), Nr. 5, S. 565–588

[Cla76] Clark, James H.: Hierarchical Geometric Models for Visible Surface Algorithms. In: *Commun. ACM* 19 (1976), Oktober, Nr. 10, 547–554. http://dx.doi.org/10.1145/360349.360354. – DOI 10.1145/360349.360354. – ISSN 0001–0782

[CLR+] Cui, G. ; Luo, M. R. ; Rigg, B. ; Roesler, G. ; Witt, K.: Uniform colour spaces based on the DIN99 colour-difference formula. In: *Color Research & Application* 27, Nr. 4, 282-290. http://dx.doi.org/10.1002/col.10066. – DOI 10.1002/col.10066

[DIN01] DIN-Normenausschuss Farbe (FNF): Farbmetrische Bestimmung von Farbabständen bei Körperfarben nach der DIN99-Formel. Berlin, DE, 2001. – Standard

[FS93] Funkhouser, Thomas A. ; Séquin, Carlo H.: Adaptive Display Algorithm for Interactive Frame Rates During Visualization of Complex Virtual Environments. In: *Proceedings of SIGGRAPH 93*, 1993, S. 247–254

[GB00] Gobbetti, Enrico ; Bouvier, Eric: Time-critical multiresolution rendering of large complex models. In: *Computer-Aided Design* 32 (2000), S. 785–803

[Hop96] Hoppe, Hugues: Progressive Meshes. In: *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*. New York, NY, USA : ACM, 1996 (SIGGRAPH '96). – ISBN 0–89791–746–4, 99–108

[ISO08] Europäisches Komitee für Normung: Colorimetry - Part 4: CIE 1976 L*a*b* Colour Space. Brüssel, BEL, November 2008. – Standard

[JCS] John C. Sanders, Peggy M. S.: *Constructing The Giza Plateau Computer Model*. https://oi.uchicago.edu/research/projects/constructing-giza-plateau-computer-model-1990-1995, Abruf: 19.01.2019

[Mal93] Mallot, Hanspeter A.: Neuronale Netze. In: Görz, Günther (Hrsg.): *Einführung in die künstliche Intelligenz*. Addison-Wesley, 1993. – ISBN 3–89319–507–6, S. 829–882

[Mii10] Miikkulainen, Risto: Neuroevolution. Version: 2010. http://nn.cs.utexas.edu/?miikkulainen:encyclopedia10-n. In: Sammut, Claude (Hrsg.) ; Webb, Geoffrey I. (Hrsg.): *Encyclopedia of Machine Learning*. New York : Springer, 2010. – ISBN 978–0–387–30164–8, 716-720

[MP43] McCulloch, Warren S. ; Pitts, Walter: A logical calculus of the ideas immanent in nervous activity. In: *The bulletin of mathematical biophysics* 5 (1943), Dec, Nr. 4, 115–133. http://dx.doi.org/10.1007/BF02478259. – DOI 10.1007/BF02478259. – ISSN 1522–9602

[Nod10] Nodine, Timothy: Speciation in NEAT / Department of Computer Science, The University of Texas at Austin. Version: 2010. http://nn.cs.utexas.edu/?nodine:ugthesis10. 2010 (HR-10-06). – Undergraduate Honors Thesis

[SBM05] Stanley, Kenneth O. ; Bryant, Bobby D. ; Miikkulainen, Risto: Real-time Neuroevolution in the NERO Video Game. In: *IEEE Transactions on Evolutionary Computation* (2005), 653-668. http://nn.cs.utexas.edu/?stanley:ieeetec05

[SM02] Stanley, Kenneth O. ; Miikkulainen, Risto: Evolving Neural Networks Through Augmenting Topologies. In: *Evolutionary Computation* 10 (2002), Nr. 2, 99-127. http://nn.cs.utexas.edu/?stanley:ec02

[SWD] Sharma, Gaurav ; Wu, Wencheng ; Dalal, Edul N.: The CIEDE2000 color-difference formula: Implementation notes, supplementary test data, and mathematical observations. In: *Color Research & Application* 30, Nr. 1, S. 21–30. http://dx.doi.org/10.1002/col.20070. – DOI 10.1002/col.20070

[Zve] Zvezdanov, Petar: *Assassin's Creed Origins - Pyramids 3D assets and textures*

Anhang

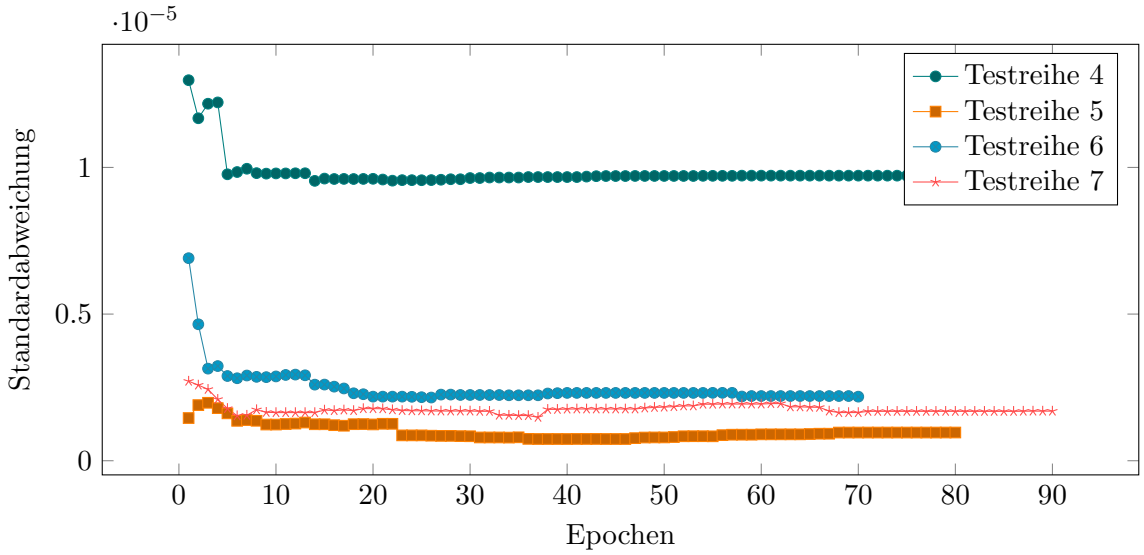


Abbildung 1: Standardabweichungen der durchschnittlichen Bildfehler bei zehnfacher Durchführung des NEAT-Lernprozess in Abhängigkeit von der Anzahl absolvierter Epochen in $Szene_1$

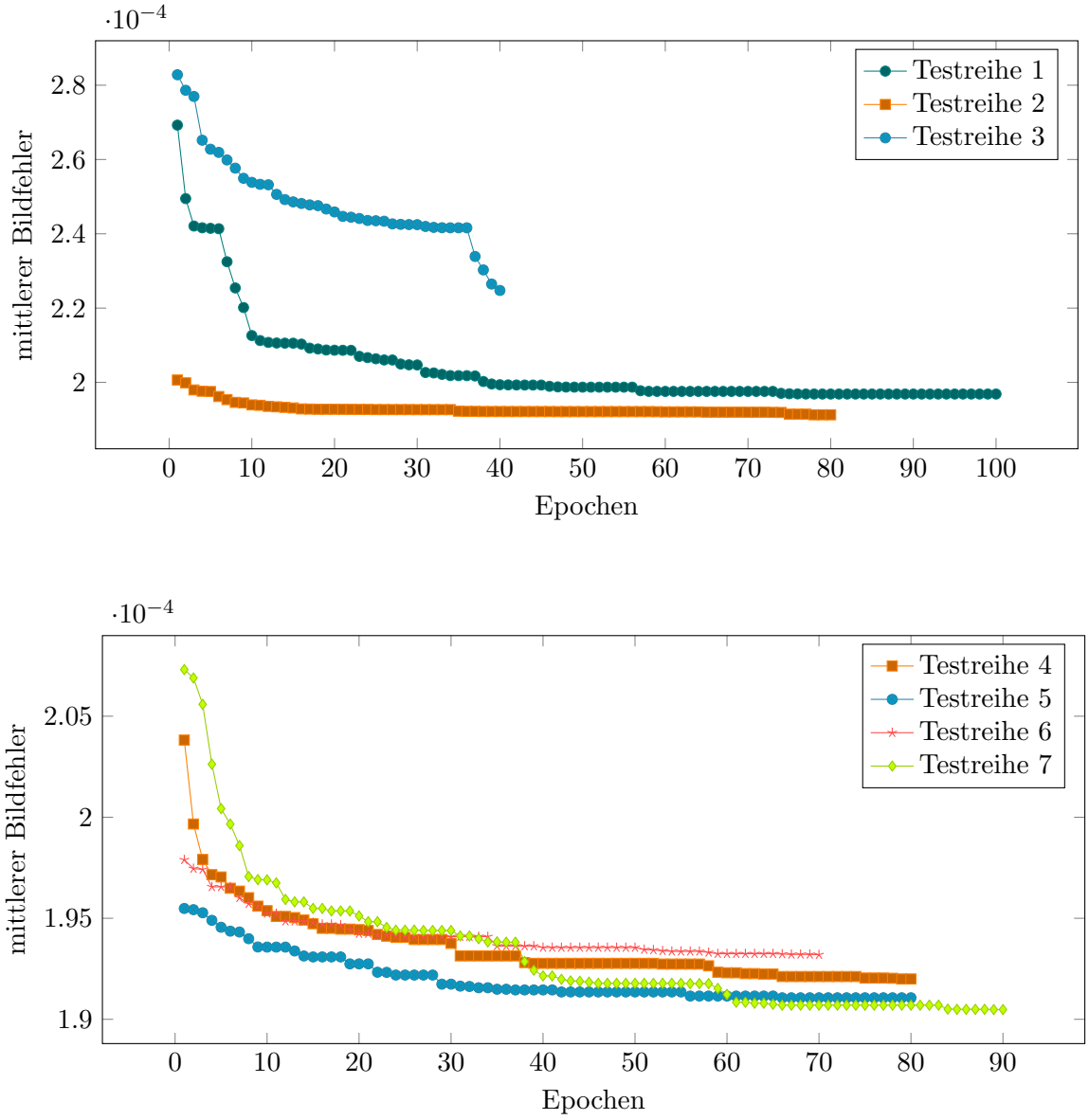


Abbildung 2: Durchschnitte der geringsten mittleren Bildfehler bei zehnfacher Durchführung des NEAT-Lernprozesses in Abhängigkeit von der Anzahl absolvierter Epochen in $Szene_2$

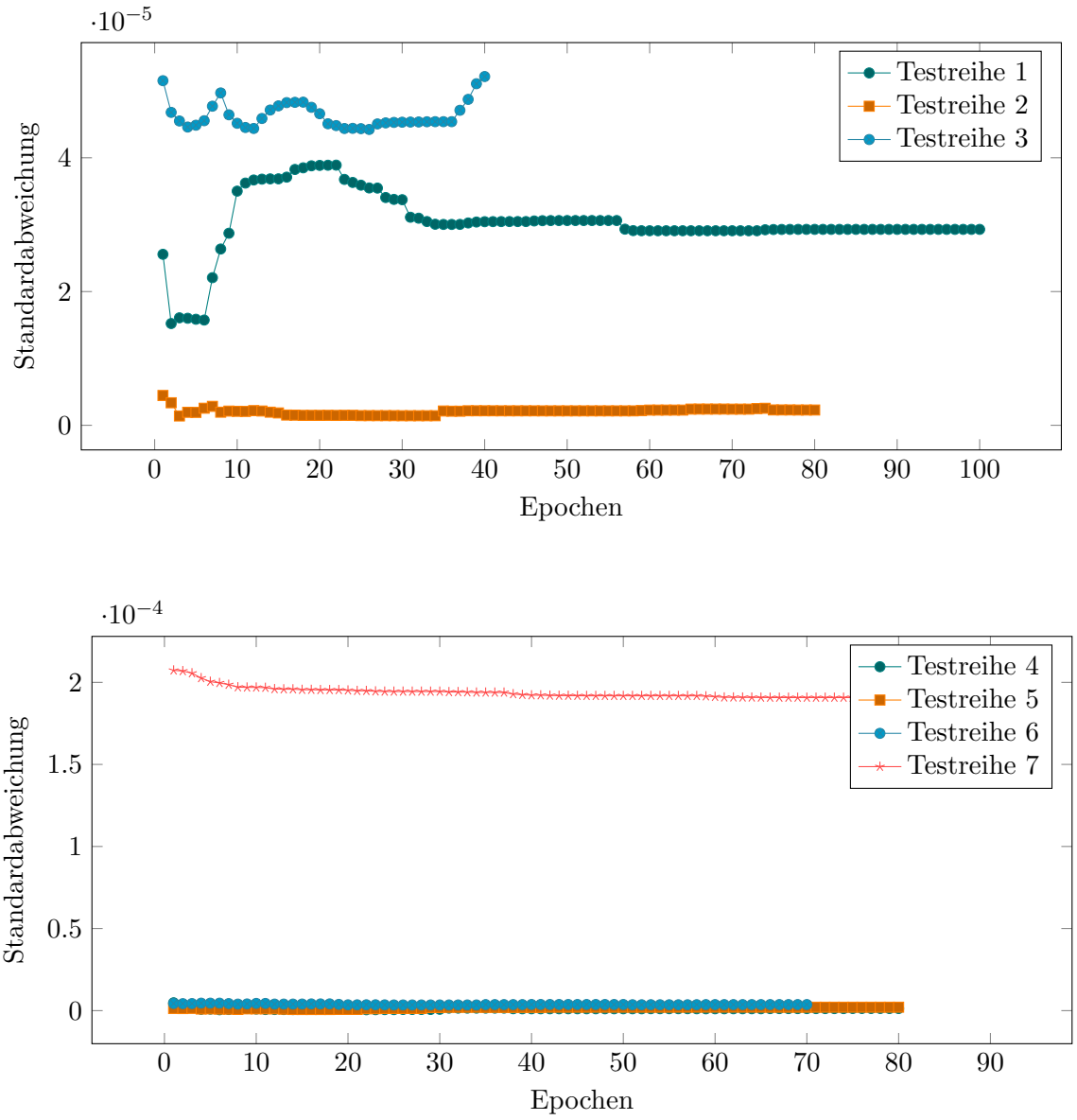


Abbildung 3: Standardabweichungen der durchschnittlichen Bildfehler bei zehnfacher Durchführung des NEAT-Lernprozess in Abhängigkeit von der Anzahl absolvierter Epochen in $Szene_2$

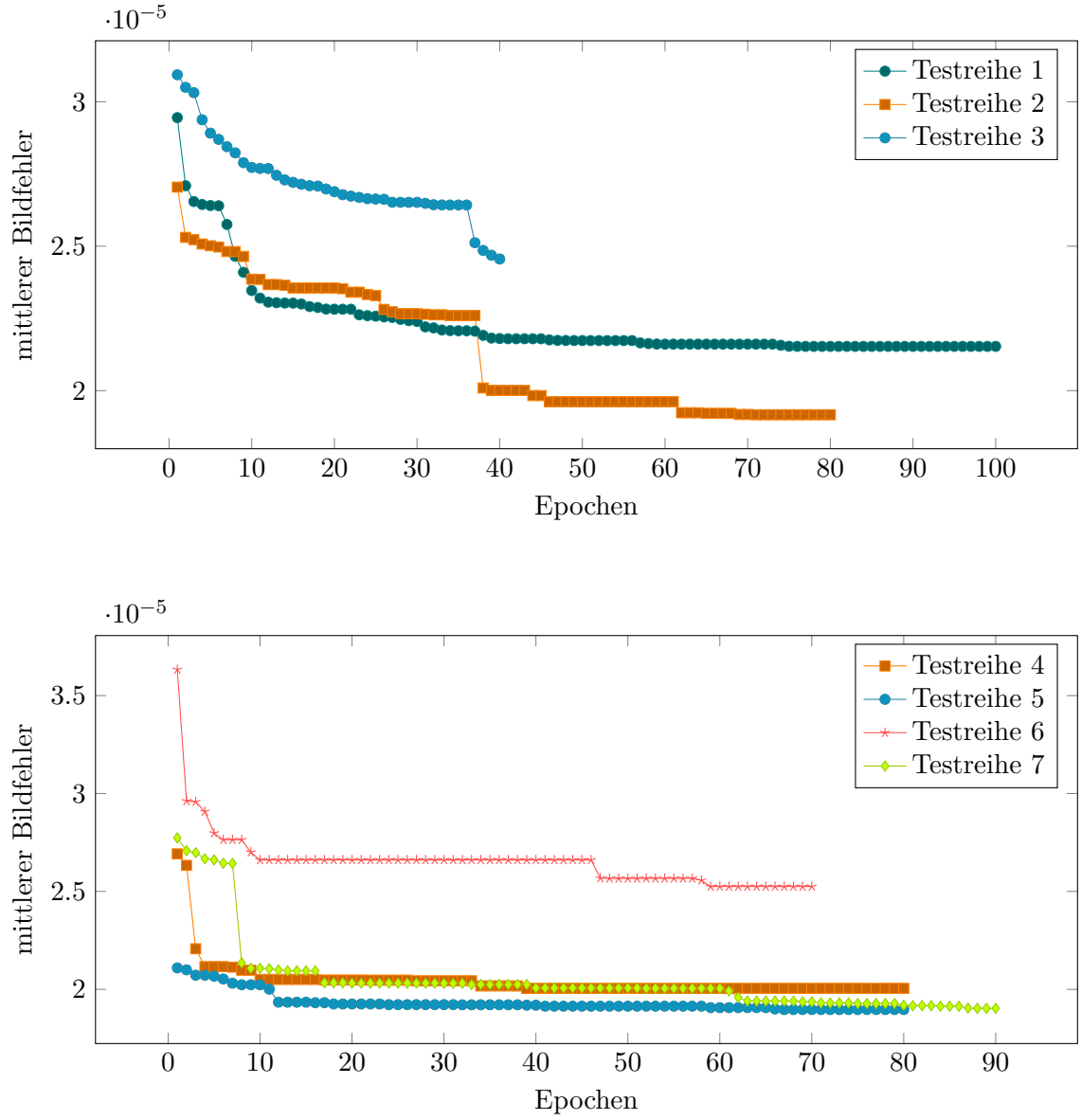


Abbildung 4: Durchschnitte der geringsten mittleren Bildfehler bei zehnfacher Durchführung des NEAT-Lernprozesses in Abhängigkeit von der Anzahl absolvierter Epochen in $Szene_3$

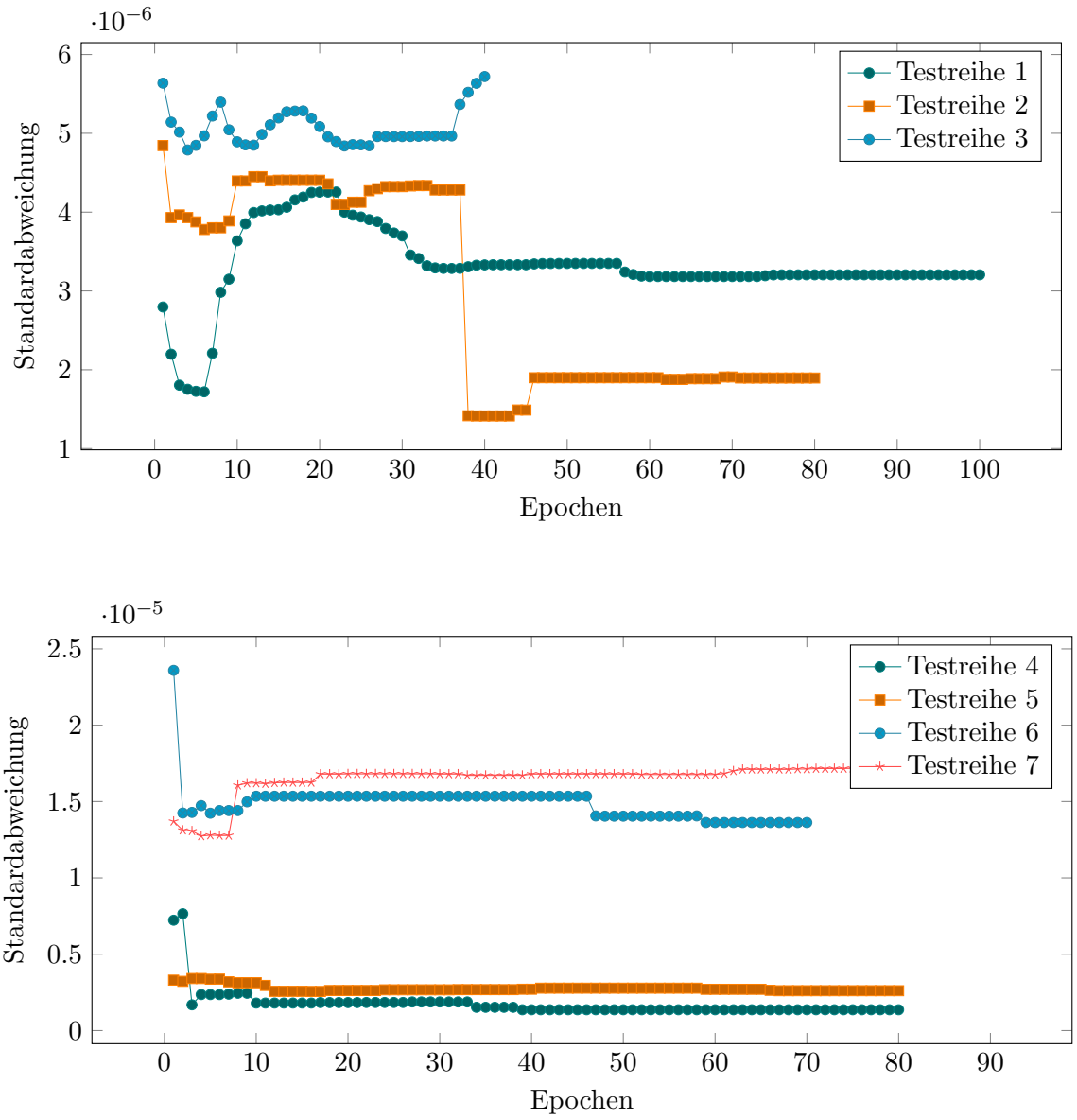


Abbildung 5: Standardabweichungen der durchschnittlichen Bildfehler bei zehnfacher Durchführung des NEAT-Lernprozess in Abhängigkeit von der Anzahl absolvierter Epochen in $Szene_3$

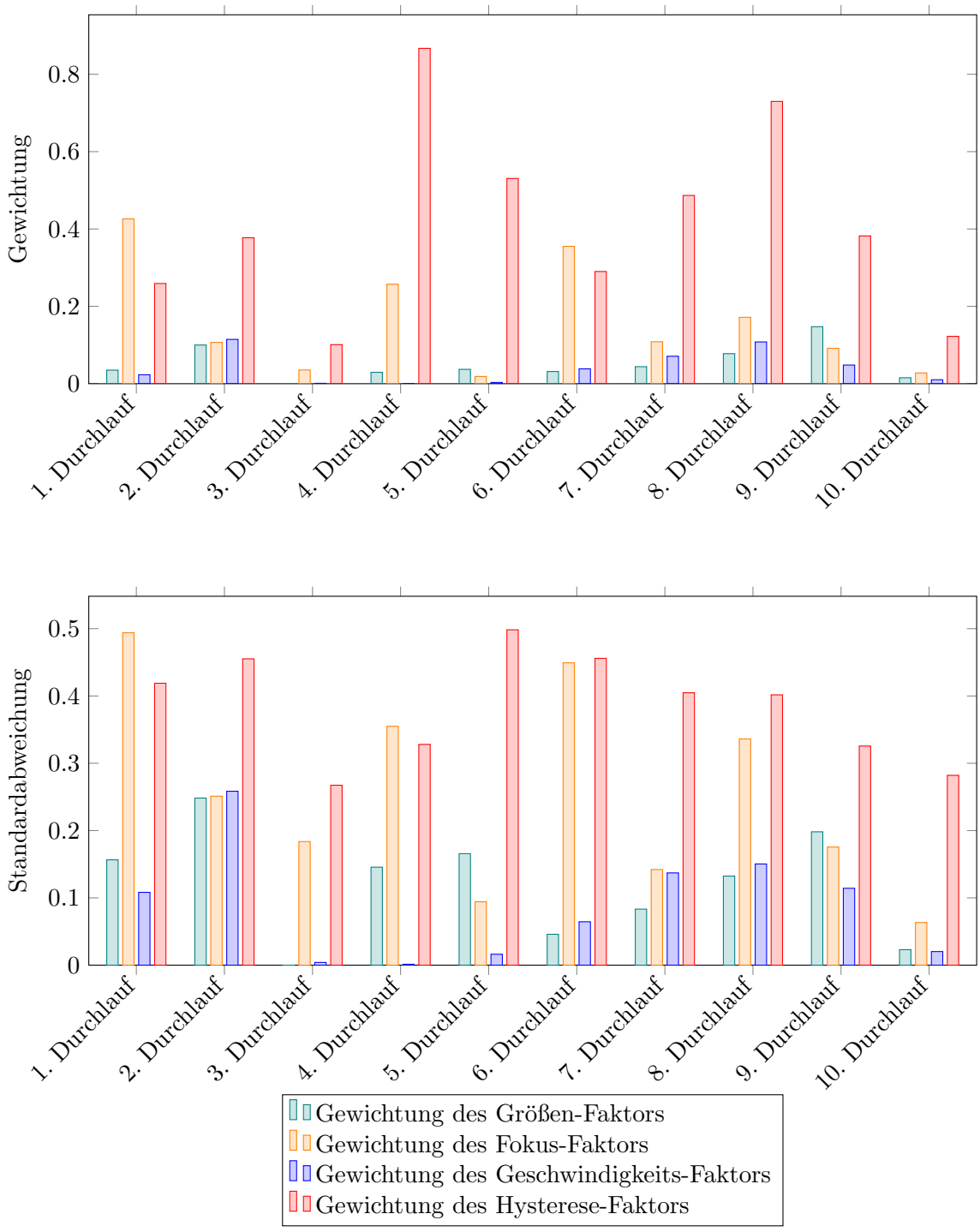


Abbildung 6: Durchschnitte und Standardabweichungen über alle betrachteten Kamerapositionen der Ausgaben des künstlichen neuronalen Netzes bei den zehn Durchläufen des NEAT-Algorithmus in $Szene_2$ (maximale Epochenanzahl 80, Populationsgröße 20)

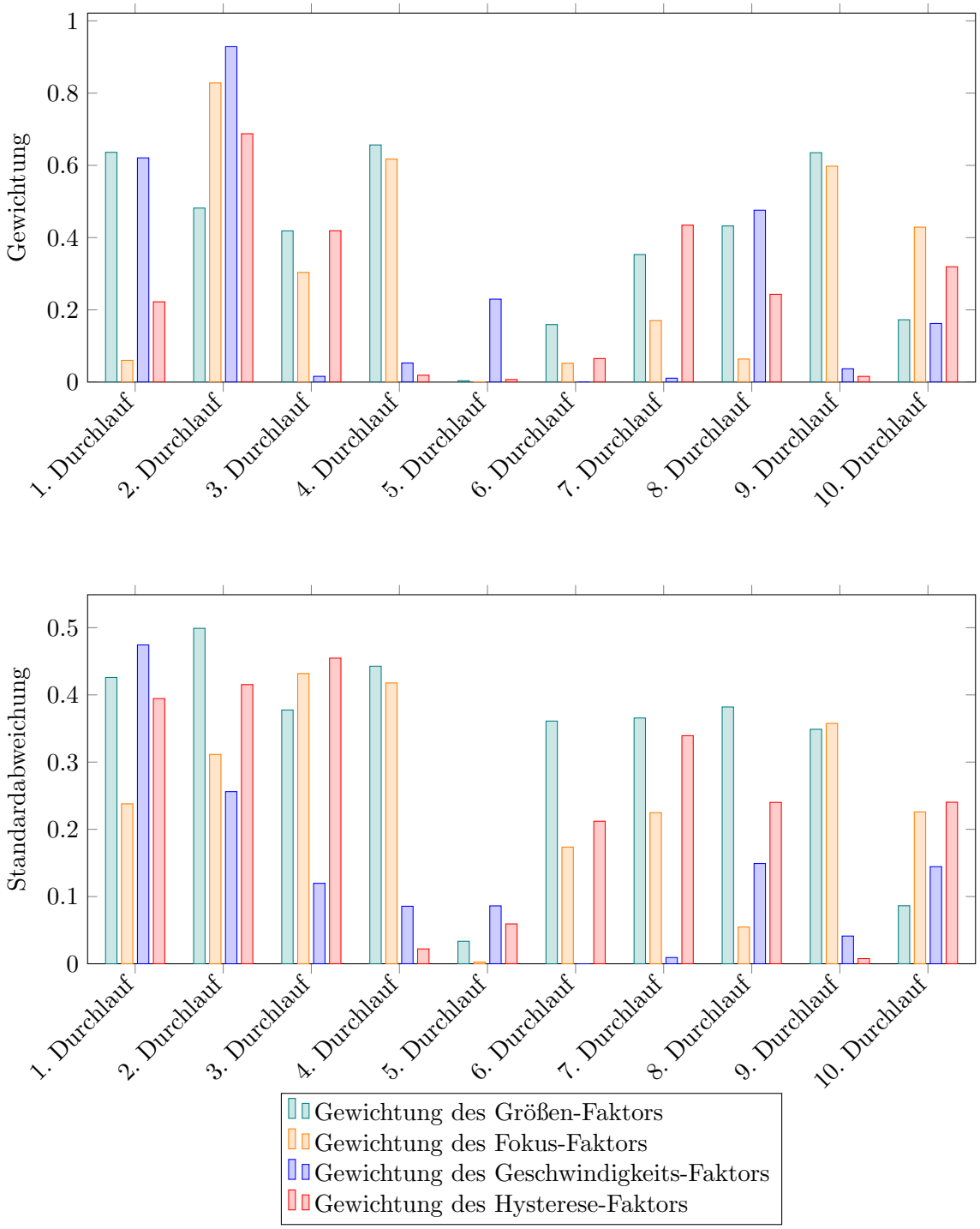


Abbildung 7: Durchschnitte und Standardabweichungen über alle betrachteten Kamerapositionen der Ausgaben des künstlichen neuronalen Netzes bei den zehn Durchläufen des NEAT-Algorithmus in $Szene_3$ (maximale Epochenanzahl 80, Populationsgröße 20)