Eberhard Karls Universität Tübingen
Mathematisch-Naturwissenschaftliche Fakultät

Bachelorarbeit Kognitionswissenschaft

Binokulare Bildstatistik mit virtueller Vergenz

Kevin Reich

1. September 2017

Prüfer

Prof. Dr. Hanspeter A. Mallot
Kognitive Neurowissenschaft, Fachbereich Biologie, Mathematisch-Naturwissenschaftliche Fakultät
Eberhard Karls Universität Tübingen

Betreuer

Gerrit Ecke
Kognitive Neurowissenschaft, Fachbereich Biologie
Eberhard Karls Universität Tübingen

Reich, Kevin:
Binokulare Bildstatistik mit virtueller Vergenz
Bachelorarbeit Kognitionswissenschaft
Eberhard Karls Universität Tübingen
Zeitraum: Juni 2, 2017 - September 2, 2017

Abstract

Stereoskopisches Sehen bildet die Grundlage für unser Gehirn, den Raum in seiner Tiefe wahrnehmen zu können. Stark daran beteiligt sind Simple Cells des visuellen Kortex, die als erste Neurone Informationen aus beiden Augen bekommen. Diese können spezifische Features, wie orientierte Kanten, aus dem visuellen Input extrahieren, um Korrespondenzen zwischen den Abbildungen im linken und rechten Auge zu finden. In einer Vorgängerstudie hat sich gezeigt, dass durch Training eines binokularen Sparse-Coding-Modells mit herkömmlichen, natürlichen Stereobildern Basisvektoren entstehen, deren Eigenschaften jenen von Simple Cells des V1 gleichen. Da herkömmliche Stereobilder von parallel zueinander stehenden Kameras aufgenommen werden, wurde für diese Studie ein Verfahren implementiert, das durch virtuelle Kamerarotationen, aus herkömmlichen Stereobildern neue erzeugt, in denen Vergenz vorhanden ist. Mittels dieses Verfahrens können Datenbanken mit blickzentrierten Bildern erstellt werden, die auch für zukünftige Untersuchungen hilfreich sind. Aufgrund der Vermutung, dass Bilder mit virtueller Vergenz über andere bildstatistische Eigenschaften verfügen, wurde das Verfahren auf eine Reihe herkömmlicher Stereobilder angewandt und anschließend das Modell der Vorgängerstudie mit einem Teil des neu entstandenen Datensatzes trainiert. Allerdings konnten sich dabei keine unterschiedlichen Eigenschaften der gelernten Basisvektoren zu denen aus der Vorgängerstudie feststellen lassen. Dies legt nahe, dass die bildstatistischen Eigenschaften der zugrunde liegenden Datenbanken sich nicht quantifizierbar unterscheiden.

Danksagung

An erster Stelle möchte ich meinem Betreuer Gerrit Ecke danken, der mir mit seinem wiederholten Rat und handwerklichen Geschick bei dieser Arbeit zur Seite stand. Ich möchte mich auch bei Herrn Professor Mallot bedanken, der mir die Möglichkeit geboten hat an seinem Lehrstuhl meine Arbeit abzulegen und mir einen tieferen Einblick über die Arbeits- und Herangehensweise seiner Arbeitsgruppe ermöglicht hat. Auch der ganzen Arbeitsgruppe sei gedankt für ihre Kommentare, Vorschläge und Diskussionen die auch über diese Arbeit hinaus meinen Horizont erweitert haben. Besonderer Dank gebührt noch Elke Seeser, die in sehr wenig Zeit meine Arbeit Korrektur lesen durfte. Letztlich möchte ich mich noch bei allen bedanken, die mich auf die eine oder andere Weise während dieser Arbeit unterstützt haben.

Inhaltsverzeichnis

Abbildungsverzeichnis

Stereodatensatz mit virtueller Vergenz

Training auf dem Datensatz

1 | Einleitung

Stereoskopisches Sehen ist eine fundamental wichtige Quelle an Informationen, die einem biologischen Agenten dabei helfen, sich in der Welt zurechtzufinden und sein Überleben zu sichern.
Dabei verbergen sich komplexe Mechanismen hinter der Stereopsis, denn Szenen aus der Welt werden erst einmal auf die Retina projiziert und liegen dort nur als zweidimensionale Abbildungen vor. Um aus diesen Tiefeninformationen gewinnen zu können, ist die horizontale Verschiebung der Augen von grundlegender Bedeutung. Durch sie werden Objekte aus der Welt im linken und im rechten Auge aus unterschiedlichen Blickwinkeln auf der Retina abgebildet, was dem Gehirn erlaubt wieder Tiefeninformationen zu extrahieren.
Die ersten Neuronen, die Informationen aus beiden Augen verarbeiten, sind Simple Cells des primären visuellen Kortex (Hubel & Wiesel, 1962; Barlow, Blakemore & Pettigrew, 1967). Sie bekommen Input von rezeptiven Feldern, die auf spezifische Reize wie orientierte Kanten reagieren. Somit können sie zum einen wichtige Informationen über die Welt direkt extrahieren und zum anderen kann der konstante visuelle Input durch ihre Aktivitäten auf sinnvolle Weise für die Weiterverarbeitung codiert werden.
Ein großer Teil dieser Neurone ist dabei zuständig für die Erkennung von Disparitäten – der Verschiebung des Bildes im linken und rechten Auge (Prince, Pointon, Cumming & Parker, 2002). Damit die Entdeckung von Ortsfrequenzen und Kantenorientierung gelingt, verfügen die Neurone über unterschiedliche rezeptive Felder, deren Profile orts- oder phasenverschoben sind (Maffei & Fiorentini, 1973; Fleet, Wagner & Heeger, 1995).
Um die Eigenschaften von disparitätssensitiven Simple Cells genauer untersuchen zu können, ist es notwendig, ein gutes Model für sie zu finden. Dies würde die Analyse ihrer Eigenschaften im Vergleich zu herkömmlichen invasiven Studien elementar erleichtern. Deshalb hat Zhang (2016) in einer Vorgängerstudie ein binokulares Sparse-Coding-Modell mit einem Datensatz von herkömmlichen Stereobildern trainiert. Dieses Modell lernt übergebene Stereobildpaare mittels künstlicher Neurone und ihren rezeptiven Feldern zu codieren. Das Lernen verläuft dabei selbstorganisiert und der Erfolg wird über den Unterschied von übergebenen zu rekonstruierten Bildern geprüft.

Auf den ersten Blick mag es sinnvoll scheinen, visuellen Input mit möglichst wenigen Neuronen kompakt codieren zu können und redundante Informationen zu vernachlässigen. Bei der Untersuchung des visuellen Systems von verschiedenen Säugetieren hat sich allerdings gezeigt, dass das Gehirn vielmehr auf eine Sparse-Coding-Strategie zurückgreift (Field, 1994). Dieser Ansatz erfordert eine größere Anzahl an Neurone als beim kompakten Codieren, aber nur eine geringe Anzahl soll zu jedem Zeitpunkt aktiv sein. Somit ist dieser Ansatz energieeffizient, was ihn biologisch plausibel macht.
Die Vorgabe mit möglichst wenig Neuronen Bilder codieren zu wollen führt dazu, dass sich beim Sparse-Coding rezeptive Felder bilden, die nur auf bestimmte Features reagieren. Ist dieses spezifische Feature in einem, nach dem Lernen zu codierenden Bild, nicht vorhanden, bleiben sie inaktiv. Dies hilft, das Korrespondenzproblem besser lösen zu können und ist eines der stärksten Argumente für Sparse-Coding (Field, 1994). Folgendes Beispiel kann dies verdeutlichen: Angenommen eine orientierte Kante wird auf der rechten und linken Retina abgebildet. Wenn eine bestimmte Zelle nur selten aktiv ist, die Zelle aber auf diesen Stimulus in beiden Bildern reagiert, ist es sehr wahrscheinlich, dass die Stimuli und somit die Abbildungen der orientierten Kante korrespondieren.
Daraus ergibt sich aber, dass der dem Training zugrunde liegende Datensatz von großer Bedeutung ist. Um *sparse* Codieren zu können, reagieren die Neurone nur auf spezifische Features. Je häufiger dementsprechend zum Beispiel kleine horizontale Disparitäten in dem Datensatz auftauchen, umso mehr Neurone werden sich um die Codierung und genauere Unterscheidung dieser Features kümmern. Sollten hingegen zum Beispiel vertikale Disparitäten in dem Datensatz überhaupt nicht präsent sein, so werden keine Neurone lernen diese zu codieren. Die erlernten rezeptiven Felder der künstlichen Neurone geben also Auskunft über die statistische Zusammensetzung der Bilder. Daraus kann man schließen, dass für die Entwicklung biologisch vorkommender rezeptiver Felder Datensätze benötigt werden, die den natürlichen visuellen Input simulieren.
Es gibt aber noch weitere Argumente dafür, dass Sparse-Coding im V1 betrieben wird. So führt es dazu, dass der Informationsgehalt im Verhältnis zum Rauschen im codierten Signal höher ist. Auch das Speichern und Abrufen des Inputs durch das assoziative Gedächtnis wird durch diese Strategie verbessert(Field, 1994).
Das binokulare Sparse-Coding-Modell soll auf emergente Weise rezeptive Felder erzeugen, wie sie im Laufe der Evolution entstanden sind. Auch dort sind die bedeutenden Faktoren die Genauigkeit der internen Repräsentation, ohne die wichtige Hinweise über die Umwelt verloren gehen würden und die Form der vorliegenden Codierung, ihrer Energieeffizienz und wie gut sie weiterverarbeitet werden kann.
Zhang (2016) hat gezeigt, dass dieses Modell in der Lage ist, dispa-

ritätssensitive rezeptive Felder zu erlernen, die in ihren Eigenschaften stark biologischen rezeptiven Feldern ähneln. Allerdings wurde auch ein Teil von ungewöhnlichen rezeptiven Feldern erlernt. So konnte zum Beispiel festgestellt werden, dass die Kernels mancher rezeptiven Felder über ihre Begrenzung hinauswachsen wollten. Auch wurde eine große Zahl von künstlichen Neuronen gefunden, deren linkes oder rechtes rezeptives Feld stärker ausgeprägt war.
Als mögliche Ursache wurde die Datenbank, mit der das Modell trainiert wurde, vermutet, da sie nur über herkömmliche Stereobilder verfügt, die von einer Kamera mit zwei horizontal verschobenen Sensoren erstellt wurde. Die menschlichen Augen sind hingegen fast nie parallel zueinander ausgerichtet, sondern fixieren in der Regel gemeinsame Punkte. Es wird vermutet, dass die Bildstatistik eines Datensatzes, bestehend aus Stereobildern mit Vergenz, sich von einem mit herkömmlichen Stereobildern unterscheidet. Zum Beispiel sollten mehr kleine Disparitäten in dem Datensatz auftauchen und vertikale Disparitäten eine größere Rolle spielen, was einen quantifizierbaren Einfluss auf die Zusammensetzung der Bilder haben sollte. Aus diesem Grund wird in dieser Arbeit versucht, dasselbe Modell mittels eines Stereodatensatzes zu trainieren, dessen Bildpaare durch virtuell vergierende Kameras erstellt wird. Zusätzlich soll diese Arbeit die Validität des binokularen Sparse-Coding-Modells stärken.
Da bisher keine Datensätze von Stereobildern mit Vergenzinformationen zur Verfügung stehen und auch nur wenige Verfahren zur Erzeugung von ihnen beschrieben wurden (beispielsweise Canessa, Gibaldi, Chessa, Fato, Solari & Sabatini, 2017, – wobei deren Verfahren nur Bilder mit rein virtuellem Inhalt erstellt), wird ein Verfahren implementiert, um aus herkömmlichen Stereobildern Bildpaare mit Vergenz zu erzeugen. Anschließend wird das Verfahren auf eine Reihe selbst aufgenommener Bilder angewandt.
Nach dem anschließenden Training des binokularen Sparse-Coding-Modells, werden die Eigenschaften der erlernten Basisvektoren analysiert und auf Unterschiede zur Vorgängerstudie überprüft. Da bildstatistische Unterschiede zwischen den Datenbanken vermutet werden, wird ein Unterschied in den Eigenschaften der entstehenden Basisvektoren erwartet.

2 | Stereodatensatz mit virtueller Vergenz

2.1 Aufgenommene Bilder

Für die Erstellung des Datensatzes wurden Bilder mit der ZED Stereokamera von Stereolabs aufgenommen. Um eine Vielzahl unterschiedlicher Motive aufnehmen zu können, wurde ein tragbarer Minicomputer (Abb. 2.1) konstruiert. Die Bilder der einzelnen Kameras hatten dabei eine Auflösung von 2208 x 1242 Pixeln, beziehungsweise das zusammengesetzte Bild besaß eine Auflösung von 4416 x 1242 Pixeln. Der linke Sensor ist vom rechten Sensor 12 Zentimeter entfernt. Bei kurzen Vortests hat sich herausgestellt, dass aufgenommene Bilder ab einer Distanz von weniger als 20 Zentimeter unscharf wurden. Aus diesem Grund wurde beim Aufnehmen der Bilder möglichst auf diese Mindestdistanz geachtet.

Bei den aufgenommenen Bildern wurde versucht, eine große Vielfalt von Mo-

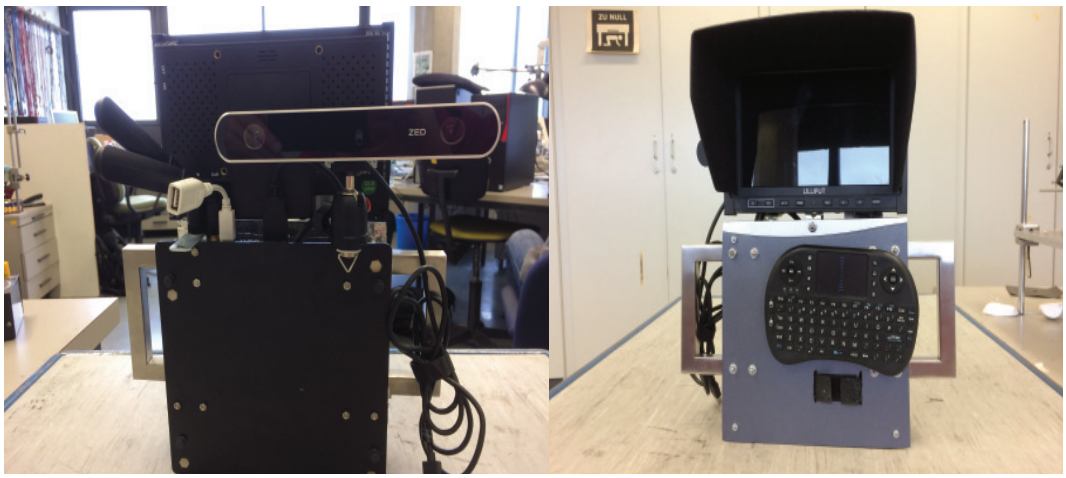

Abbildung 2.1: Stereokamera

Vorder und Rückseite des Minicomputers. Auf der Vorderseite ist die ZED Stereokamera zu sehen.

tiven abzudecken. Einige Beispielbilder sind in Abb. 2.2 zu sehen. Insgesamt wurden außerhalb von Gebäuden 73 Bilder von Lebewesen, 525 Künstliche Strukturen wie zum Beispiel Gebäude, Autos, Straßen, Skulpturen und 247 Naturaufnahmen wie Bäume, Blumen und Felder aufgenommen. Innerhalb von Gebäuden wurden 58 Bilder von Fluren und Hallen gemacht, 42 Bilder von Lebewesen und 122 Bilder von Wohnungen und Zimmern - Diese waren im Vergleich zu den Bildern von Fluren und Hallen mit vielen persönlichen Objekten gefüllt und somit sehr reich an Features. Bilder, die bezüglich der Klasse nicht eindeutig zuzuordnen waren, wurden der am besten passenden Klasse zugeordnet. So wurde eine Blume vor einer Ladenfront zur Kategorie künstlicher Strukturen außerhalb von Gebäuden gezählt, ein Vorgarten zur Kategorie Naturaufnahmen.

Abbildung 2.2: Beispielbilder

Zu sehen sind drei Bilder, die von der Stereokamera aufgenommen wurden. Dabei wurde die linke Hälfte des Bildes von der linken Kamera und die rechte Hälfte des Bildes von der rechten Kamera aufgenommen.

2.2 Theoretischer Hintergrund

Ein Ziel der Arbeit war es, ein Verfahren zu implementieren, das aus aufgenommenen Stereobildern neue erzeugt, die über Vergenz verfügen. Dafür ist ein tieferes Verständnis des menschlichen Sehapparats notwendig.
Auf der Retina liegt im Zentrum des Gelben Flecks die Fovea, in der die höchste Dichte an Sehzellen besteht. Dadurch wird an dieser Stelle die höchste Auflösung von visuellem Input erlangt. Entsprechend wird ein fixiertes Objekt, sowohl im linken als auch im rechten Auge dort abgebildet (Abb. 2.3). Dafür müssen die Sehachsen der beiden Augen auf das zu fixierende Objekt bewegt werden, was durch eine Rotation der Augen geschieht. Diese Bewegung führt dazu, dass sich die Sehachsen schneiden und wird als Vergenzbewegung bezeichnet. Die Achsen, um die rotiert werden muss, sind aus dem Listing'schen Gesetz bekannt.
Das Listing'sche Gesetz beschreibt die Orientierung der Augen im dreidimensionalen Raum (Wong, 2004): Sollte der Kopf fixiert sein, dann gibt es eine primäre Stellung der Augen, von der aus die Augen alle ihnen möglichen Orientierungen durch die Rotation um eine einzige Achse einnehmen können. Diese Achsen liegen alle in der selben Ebene, die auch als Listing'sche Ebene bekannt ist. Die Achsen sind dabei immer orthogonal zur primären Blickrichtung.

Durch den Umstand, dass unsere Augen horizontal verschoben sind und auf bestimmte Punkte vergieren können, entstehen Disparitäten. Der fixierte Punkt hat die Disparität null. Er liegt sowohl im rechten als auch im linken Auge in der Fovea. Alle Punkte, die in der selben Tiefe wie der Fixierpunkt liegen, haben ebenfalls die Disparität null. Sie liegen im sogenannten Horopter (vgl. Howarth, 2011; Siderov, Harwerth & Bedell, 1999). Punkte, die tiefer als der Fixationspunkt liegen, haben eine negative Disparität, Punkte vor dem Fixationspunkt eine positive (Verhoef, Vogels & Janssen, 2016). Darüber hinaus kann man in Abb. 2.3 auch sehen, dass alle nicht auf dem Horopter liegenden Punkte auf der rechten und der linken Retina an unterschiedlichen Orten abgebildet werden. Dieser Unterschied wird als Stereodisparität bezeichnet und hilft dem Gehirn Tiefeninformationen aus einer Szene zu gewinnen. Da unsere Kameras allerdings parallel zueinander ausgerichtet sind, können sie keinen gemeinsamen Punkt fixieren. Daher wird die Vergenzbewegungen mit den Kameras virtuell ausgeführt, um stereoskopisches Sehen genauer simulieren zu können.

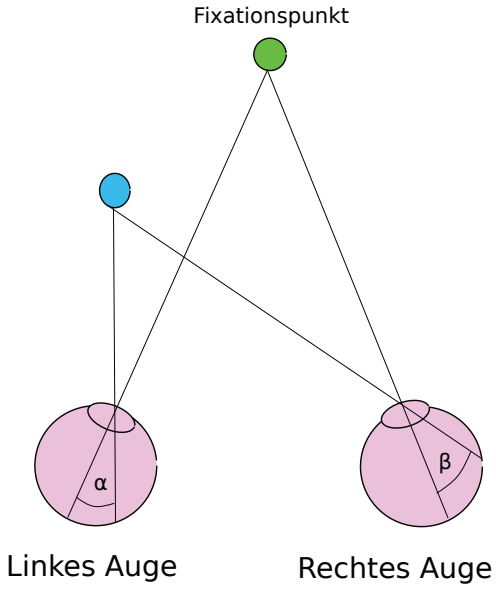

Abbildung 2.3: Vergenz

Zu sehen ist, wie die Augen das grüne Objekt fixieren. Entsprechend liegt dessen Abbildung in der Fovea. Das blaue Objekt hingegen wird im linken und im rechten Auge an unterschiedlichen Stellen abgebildet. Bei α und β handelt es sich um den Abstandswinkel zwischen der Abbildung des blauen Objekts und der Fovea (Verhoef et al., 2016). Die Disparität wird dann als der Unterschied der Winkel aus dem linken und dem rechten Auge definiert: α - β. Somit gibt sie an, wie groß die örtliche Abweichung der Abbildung eines Objektes in den beiden Augen ist. Dargestellt sind nur horizontale Disparitäten. Wenn ein Punkt außerhalb der Horizontalebene liegt, kommen auch noch vertikale Disparitäten hinzu, die ebenfalls eine wichtige Rolle für die Tiefenwahrnehmung spielen (Vienne, Plantier, Neveu & Priot, 2016). (Grafik nachgezeichnet nach Verhoef et al., 2016, Abb.1 (b))

2.2.1 Exkurs: Homogene Koordinaten und Kameramodell

Die theoretischen Hintergründe für die Erstellung des Datensatzes und somit auch der Exkurs basieren stark auf Hartley & Zisserman (2003). Der Exkurs stellt zum Großteil eine Zusammenfassung der für die weitere Arbeit relevanten Konzepte, Gleichungen und Herleitungen dar. Daher wird das Lehrbuch bei tiefer gehendem Interesse an den Hintergründen als Lektüre empfohlen.

Homogene Koordinaten

Homogene Koordinaten sind ein wichtiges Hilfsmittel in der Computergrafik, da sie unter anderem erlauben, alle affinen Transformationen durch Matrixmultiplikationen darzustellen. Sie sind Teil der projektiven Geometrie und die dazugehörigen Transformationen werden als projektive Transformationen bezeichnet. Am Beispiel einer Translation wird ihr Nutzen offensichtlich. Ein Punkt $P = (x, y)^T$ soll um den Wert a in Richtung der X-Achse und den Wert b in Richtung der Y-Achse verschoben werden.

$$\begin{pmatrix} x \\ y \end{pmatrix} \rightarrow \begin{pmatrix} x + a \\ y + b \end{pmatrix} \tag{2.1}$$

Dies kann man bewerkstelligen, indem man einen Translationsvektor $\vec{t}$ auf unseren Punkt P addiert. Da es sich bei der Translation allerdings um eine affine Transformation handelt, lässt sie sich normalerweise nicht durch eine Matrixmultiplikation ausdrücken. Dies wäre allerdings deutlich einfacher, da sich so eine beliebige Reihe von Transformationen durch eine einzige Matrix beschreiben lassen. Dafür muss man nur die Matrizen der einzelnen Transformationen in der gewünschten Reihenfolge der Transformationen, multiplizieren.
In homogenen Koordinaten hingegen ist es möglich, alle affinen Transformationen durch Matrixmultiplikationen auszudrücken, wodurch die Effizienz gesteigert wird. Alle Punkte sind dann nur noch mit der zuvor errechneten Transformationsmatrix zu multiplizieren, wodurch nicht mehr jede Transformation einzeln und nacheinander ausgeführt werden muss.
Durch folgende Abbildung kann ein Punkt in homogene Koordinaten überführt werden:

$$\mathbb{R}^2 \rightarrow \mathbb{R}^3 : \begin{pmatrix} x \\ y \end{pmatrix} \rightarrow \begin{pmatrix} \lambda x \\ \lambda y \\ \lambda \end{pmatrix} \tag{2.2}$$

Allgemeiner ist die Überführung in homogene Koordinaten eine Abbildung $\mathbb{R}^n \rightarrow \mathbb{R}^{n+1}$. Der Einfachheit halber wird hier aber erst einmal nur der zweidimensionale Ausgangsfall betrachtet.
Nach der Überführung in homogene Koordinaten ist der Punkt allerdings kein einzelner Punkt mehr. An der Abbildung kann man sehen, dass er stattdessen auf eine Ursprungsgerade abgebildet wurde. Deswegen sind auch alle skalaren Vielfachen zueinander äquivalent:
$(6, 4, 2)^T \equiv (3, 2, 1)^T \equiv \lambda(3, 2, 1)^T$
Für die inverse Abbildung gilt:

$$\mathbb{R}^3 \rightarrow \mathbb{R}^2 : \begin{pmatrix} x \\ y \\ z \end{pmatrix} \rightarrow \begin{pmatrix} \frac{x}{z} \\ \frac{y}{z} \\ z \end{pmatrix} \tag{2.3}$$

Im Folgenden werden die homogenen Koordinaten nur als Trick zur Vereinfachung von Transformationen von Punkten genutzt. Da aber oft mit eindeutig definierten Punkten gearbeitet werden soll, setzt man $\lambda = 1$, was die X- und Y-Koordinate des Ausgangspunktes unverändert lässt (siehe Abb. 2.4).
Mit den homogenen Koordinaten können wir nun unsere Translation einfach

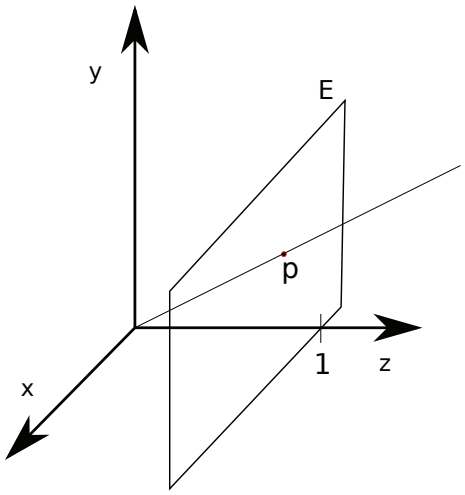

Abbildung 2.4: Punkte in homogene Koordinaten

Man kann es sich so vorstellen, dass der in unserem Beispiel zuvor zweidimensionale Raum als Ebene E in den dreidimensionalen Raum gelegt wurde. Dabei hat die Ebene E einen Z-Wert von 1 und ist parallel zur XY-Ebene. Unser Punkt in homogenen Koordinaten ist nun der Schnittpunkt der Ursprungsgeraden aus Gleichung 2.2 mit der Ebene E.

als Matrixmultiplikation ausdrücken:

$$\begin{bmatrix} 1 & 0 & a \\ 0 & 1 & b \\ 0 & 0 & 1 \end{bmatrix} * \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x+a \\ y+b \\ 1 \end{pmatrix} \tag{2.4}$$

Homogene Koordinaten haben also den Vorteil affine und lineare Transformationen als projektive Transformationen zusammenzufassen.

Das Lochkameramodell

Gerade für unser Kameramodell spielen homogene Koordinaten eine wichtige Rolle und vereinfachen viele Berechnungen. Als Modell für unsere Kamera haben wir das Lochkameramodell genutzt. Dieses beschreibt, wie Punkte $X = (x, y, z)^T$ im Raum durch das Loch der Kamera C, auch Kamerazentrum genannt, auf eine Bildebene I projiziert werden. Die Bildebene befindet sich im Abstand der Brennweite f zur Kamera und ist orthogonal zur Brennachse. Der Punkt, an dem die Brennachse die Bildebene schneidet, nennt sich Brennpunkt p oder Fokus. Für die Projektion von X auf I wird eine Gerade durch C und X gelegt und der Schnittpunkt dieser Geraden mit der Bildebene ergibt den Bildpunkt x' (siehe Abb. 2.5).

Wie man in Abb. 2.6 sehen kann, handelt es sich bei der Überführung von Kamera in Bildkoordinaten um folgende Abbildung:

$$\mathbb{R}^3 \to \mathbb{R}^2 : (x, y, z)^T \to (fx/z, fy/z)^T \tag{2.5}$$

In Abb. 2.5 wurde davon ausgegangen, dass das Kamerazentrum im Koordinatenursprung des Kamerakoordinatensystems liegt, was nicht der Fall sein muss. Für den Fall, dass der Brennpunkt $p = (p_x, p_y)^T$ verschoben ist, wird die Abbildung zu:

$$\mathbb{R}^3 \to \mathbb{R}^2 : (x, y, z)^T \to (fx/z + p_x, fy/z + p_y)^T \tag{2.6}$$

Hier ist es nun von Vorteil, die Punkte in homogenen Koordinaten vorliegen zu haben, denn dadurch lässt sich die Abbildung einfach wie folgt beschreiben:

$$f((x, y, z, 1)^T) = \begin{bmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} * \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} fx + zp_x \\ fy + zp_y \\ z \end{pmatrix} \tag{2.7}$$

Dabei wird die zentrale Matrix wie folgt benannt:

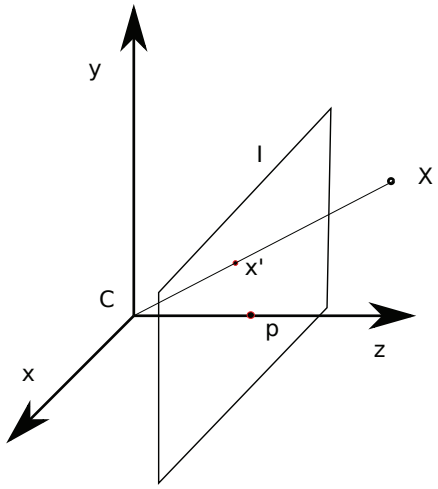

Abbildung 2.5: Variablen der Lochkamera

Hier sieht man die verschiedenen Variablen des Lochkameramodells und wie der Punkt X von Kamerakoordinaten auf den Punkt x' der Bildebene projiziert wird. Die Bildpunkte sind in rot eingezeichnet. (Grafik nach Hartley & Zisserman, 2004, Abb. 6.1)

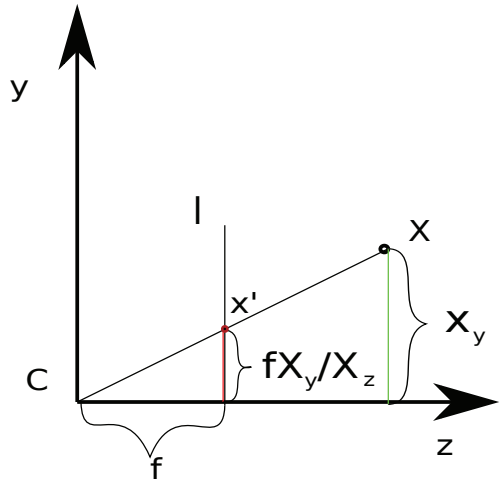

Abbildung 2.6: Berechnung des Bildpunktes in der Lochkamera

Der Punkt X wird auf den Bildpunkt x' abgebildet. Die grüne und die rote Seite sind Seiten von ähnlichen Dreiecken, da die Winkel der beiden Dreiecke die selben sind. Somit bleibt nach den Strahlensätzen das Seitenverhältnis bestehen. Da die Brennweite f bekannt ist, kann der Y-Wert unseres Bildpunktes x' einfach berechnet werden. Die Berechnung des X-Wertes erfolgt analog (Grafik nachgezeichnet nach Hartley & Zisserman, 2004, Abb. 6.1).

$$K = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.8}$$

K wird als intrinsische Matrix bezeichnet. Sie enthält die notwendigen Parameter, um Punkte aus dem Kamerakoordinatensystem in Punkte des Bildkoordinatensystems umrechnen zu können. Dies lässt sich dann einfach beschreiben als:

$$x_{image} = K[I|0]X_{cam} \tag{2.9}$$

Es handelt sich bei $[I|0]$ um eine zusammengesetzte 3x4 Matrix, bestehend aus der 3x3 Einheitsmatrix und dem Nullvektor $(0, 0, 0)^T$. Generell liegen Punkte allerdings nicht in Kamerakoordinaten sondern in Weltkoordinaten vor. Da es sich bei beiden Koordinatensystemen um dreidimensionale euklidische Koordinatensysteme handelt, kann durch Rotation und Translation das Weltkoordinatensystem einfach in das Kamerakoordinatensystem überführt werden. Die Gleichung, um direkt von Welt auf Bildkoordinaten abzubilden, sieht dann wie folgt aus:

$$x_{image} = K[R|t]X_{world} \tag{2.10}$$

Es handelt sich bei R um die 3x3 Rotationsmatrix und bei t um den Translationsvektor, die die Beziehungen zwischen Welt und Kamerakoordinaten beschreiben. Diese Projektion wird zusammengefasst und als homogene Kameraprojektionsmatrix P bezeichnet:

$$P = K[R|t] \tag{2.11}$$

All diese Parameter kann man durch das Kalibrieren der Kamera gewinnen. Roger Tsai (1987) entwickelte ein Verfahren, in dem man durch Aufnahmen von wohldefinierten und bekannten Objekten die verschiedenen Kameraparameter schätzen kann. In der Regel handelt es sich heute bei diesen Objekten um Schachbrettmuster. Von den aufgenommenen Bildern sind bestimmte Punkte in Weltkoordinaten bereits bekannt, wonach ein Matching mit den Punkten des Kamerabildes erfolgt. Dadurch kann eine möglichst optimale Konfiguration für die Kameraparameter gesucht werden, die die Punkte von Weltkoordinaten in Bildkoordinaten überführt.
Mittlerweile gibt es eine Vielzahl von Arbeiten zu dem Thema der Kamerakalibrierung und eine große Menge bereits implementierter Algorithmen, die man direkt nutzen kann. So wurden zum Beispiel Verfahren basierend auf Heikkilä und Silvén (1997) sowie Zhang (2000) in der Matlab Stereo Calibration App implementiert.
Die ZED Stereokamera wird schon mit Herstellerangaben darüber, welche Parameter sie haben sollte, geliefert. Wir verglichen diese mit den Ergebnissen

unserer manuellen Kalibrierungen.
Zum einen wurde aus der Matlab Computer Vision Toolbox die Stereo Calibration App genutzt, welche eine Reihe von Aufnahmen eines wohldefinierten Schachbrettmusters benötigt, um die Kameraparameter zu schätzen, und zum anderen das von Stereolabs angebotene ZED Calibration Tool. Dieses zeigt ein Schachbrettmuster auf einem Bildschirm an, auf welches die ZED Stereokamera aus verschiedenen Distanzen und Winkeln schauen muss.
Da die Parameter des ZED Calibration Tools näher an den Herstellerangaben lagen, wurden diese für weitere Berechnungen verwendet. Zuvor wurde die Abschätzung getroffen, dass der Fehler bei der Fertigung der Kamera größer ist als jener des ZED Calibration Tools. Deshalb wurden nicht die Angaben des Herstellers direkt, sondern die Parameter der Kalibrierung genutzt.

2.2.2 Bildtransformation

Feature Matching

Die beiden Kameras der ZED Stereokamera stehen, wie bereits erwähnt, parallel zueinander. Um nun Vergenzbewegungen der Kameras auf einen Fixpunkt simulieren zu können, muss zuerst Feature Matching in den Stereobildern betrieben werden, um korrespondierende Punkte zwischen dem linken und dem rechten Bild zu finden.
Dafür werden im ersten Schritt Features in den beiden Bildern identifiziert. Zum Detektieren und Beschreiben von Features wurden die in Matlab integrierten Funktionen zum Matching von SURF Featurs genutzt.
SURF wurde von Bay, Tutyelaars, Ess & Van Gool (2008) entwickelt und entdeckt mithilfe eines Hesse-Blob-Detektors verschiedene Punkte, die von Interesse sind. Danach werden diese Punkte mit einem Deskriptor versehen werden, der sie möglichst auf robuste Weise beschreibt und daher unabhängig von zum Beispiel Skalierung sein sollte. Diese Deskriptoren sind dann im letzten Schritt essentiell für das Matching der Features.
Beim endgültigen Feature Matching werden die Deskriptoren, die man in den beiden Bildern gewonnen hat, verglichen. So konnten m Paare von korrespondierenden Punkten $(l_1, ..., l_m)$, $(r_1, ..., r_m)$ gefunden werden.
Anschließend wird in Matlab die Fundamentalmatrix geschätzt. Sie beschreibt die Beziehung zwischen zwei Bildern, auf denen die selbe Szene zu sehen ist, und macht Aussagen darüber, an welchen Stellen in den Bildern bestimmte Features aus der Szene auftreten können. Entsprechend können mit ihr falsche Matches ausgeschlossen werden. Die Fundamentalmatrix wurde mithilfe des RANSAC Algorithmus berechnet (Luong & Faugeras, 1996).

Virtuelle Kamerarotation

Nachdem korrespondierende Punkte in den beiden Bildern identifiziert worden sind, wurde die Verzeichnung aus den Bildern und auch aus den korrespondierende Punkten herausgerechnet. Dies dient zum einen dazu, das Netz mit möglichst kameraunabhängigen Bildern trainieren zu können und zum anderen ist es auch wichtig für unser Modell der Lochkamera. Denn eine verzerrende Linse führt dazu, dass es keine Geraden mehr gibt, die durch X_{world}, I und C gehen.
Bei der Kamerakalibrierung wird auch die Linsenverzerrung mitbestimmt, wodurch diese einfach wieder herausgerechnet werden kann. Entsprechend kann mit dem Lochkameramodell gearbeitet werden.

Abbildung 2.7: Beispiel Feature Matching

Hier sieht man, wie zwischen dem Bild der linken und der rechten Kamera aus Abb. 2.2C sechs korrespondierende Punkte gefunden wurden.

Für zwei Kameras, die sich durch eine reine Rotation unterscheiden, sind die beiden Kamerakoordinatensysteme durch die selbe Rotation ineinander überführbar und werden durch R beschrieben. Entsprechend ist der Translatiosnvektor $t = (0, 0, 0)^T$. Dies ähnelt der Beziehung des Weltkoordinatensystems zum Kamerakoordinatensystem, nur dass es sich hier um zwei Kamerakoordinatensysteme handelt, die zueinander verdreht sind. Somit lässt sich Gleichung 2.10 hier anwenden (Herleitung nach Hartley & Zisserman, 2003):

$$x_{imageRotated} = K[R|t]x_{image} = K[R|0]x_{image} \tag{2.12}$$

Die Rotationsmatrix R kann aus der 3x4 Matrix herausgezogen werden:

$$x_{imageRotated} = K[R|0]x_{image} = KR[I|0]x_{image} \tag{2.13}$$

Und durch das Einfügen einer impliziten Einheitsmatrix erhält man:

$$x_{imageRotated} = KRI[I|0]x_{image} = KRK^{-1}K[I|0]x_{image} \tag{2.14}$$

In diese Gleichung muss nur noch Gleichung 2.9 eingesetzt werden, um die Formel für die virtuelle Rotation einer Kamera zu erhalten.

$$x_{imageRotated} = KRK^{-1}x_{image} \tag{2.15}$$

Die Gleichung kann wie folgt verbildlicht werden: Zuerst wird der Bildpunkt x_{image} von homogenen Bildkoordinaten in nicht homogene Kamerakoordinaten überführt, dann wird die Kamera virtuell um R gedreht, wodurch ein gedrehtes Kamerakoordinatensystem entsteht. Danach wird der Punkt wieder auf die rotierte Bildebene projiziert und liegt wieder in homogenen Koordinaten vor.

Rotationsmatrix

Da die intrinsische Matrix K bereits bekannt ist, wird nur noch die Rotationsmatrix R benötigt
Nach dem Listing'schen Gesetz muss die Rotation um eine Achse u erfolgen, die orthogonal zur Richtung der Blickbewegung liegt. Dies wird im Folgenden am Beispiel einer einzelnen Kamera gezeigt und funktioniert für die andere Kamera analog. Im Ausgangszustand ist der Brennpunkt p die Mitte unseres Bildes und würde somit in der gedachten Fovea unserer Kamera liegen. Für die Vergenzbewegung geht die primäre Blickbewegung entsprechend von dem Brennpunkt zu einem der Matching Feature Points l_n und lässt sich somit als Vektor beschreiben (vergleiche auch Abb. 2.8):

$$\vec{(}v) = \begin{pmatrix} l_{n_1} - p_1 \\ l_{n_2} - p_2 \end{pmatrix} \tag{2.16}$$

Der dazu orthogonale Vektor lässt sich durch eine Drehung um 90 Grad berechnen:

$$\vec{(}u) = \begin{pmatrix} l_{n_1} - p_1 \\ l_{n_2} - p_2 \end{pmatrix} \begin{bmatrix} cos\Theta & -sin\Theta \\ sin\Theta & cos\Theta \end{bmatrix} \tag{2.17}$$

Hier gilt $\Theta = 90$ Grad. v liefert uns dann die Orientierung für unsere Drehachse. Darüber hinaus ist aus dem Listing'schen Gesetz bekannt, dass die Drehachse durch das Kamerazentrum gehen und in der Listing'schen Ebene liegen muss. Diese ist parallel zur Bildebene, woraus für die Rotationsachse u $z = 0$ folgt. Somit ergibt sich:

$$u = \begin{pmatrix} -v_2 \\ v_1 \\ 0 \end{pmatrix} \tag{2.18}$$

Cole (2015) beschreibt, dass für eine Rotation um eine arbiträre, durch den Ursprung gehende Achse u folgende fünf Schritte durchzuführen sind:

- Rotiere u in die X-Z-Ebene und den Punkt x relativ zu u
- Rotiere u so, dass u auf der Z-Achse zu liegen kommt und rotiere x relativ dazu
- Rotiere x um den Rotationswinkel Θ um die Z-Achse
- Rotiere u und x' zurück in die X-Z-Ebene, indem die zuvor durchgeführte Drehung von u umgekehrt wird
- Rotiere u und x zurück so, dass die Drehachse u an ihrer ursprünglichen Position liegt

Dies ist alles durch Rotationen um die X-, Y- und Z-Achse zu bewerkstelligen. Die Matrix, die aus der Multiplikation dieser Operationen entsteht, sieht wie folgt aus:

$$R = \begin{bmatrix} cos\Theta+u_x^2(1-cos\Theta) & u_xu_y(1-cos\Theta)-u_zsin\theta & u_xu_z(1-cos\Theta)+u_ysind\Theta \\ u_yu_x(1-cos\Theta)+u_zsin\Theta & cos\Theta+u_y^2(1-cos\Theta) & u_yu_z(1-cos\Theta)-u_xsind\Theta \\ u_zu_x(1-cos\Theta)-u_ysind\Theta & u_zu_y(1-cos\Theta)+u_xsin\Theta & cos\Theta+u_z^2(1-cos\Theta) \end{bmatrix} \tag{2.19}$$

Hierbei wird davon ausgegangen, dass die Rotationsachse u als Einheitsvektor vorliegt, da sich damit die Matrix kompakter darstellen lässt.
Da bei unserer Kamerarotation die Z-Komponente von u null ist, vereinfacht sich die Matrix zu:

$$R = \begin{bmatrix} cos\Theta + u_x^2(1-cos\Theta) & u_xu_y(1-cos\Theta) & u_ysind\Theta \\ u_yu_x(1-cos\Theta) & cos\Theta + u_y^2(1-cos\Theta) & -u_xsind\Theta \\ -u_ysind\Theta & u_xsin\Theta & cos\Theta \end{bmatrix} \tag{2.20}$$

Θ ist hierbei der Rotationswinkel, um den die Kamera um die Drehachse gedreht werden muss, damit der Punkt l_n der neue Brennpunkt ist. Für Θ gilt (siehe auch Abb.2.8):

$$tan(\Theta) = \frac{||\vec{v}||}{f} \tag{2.21}$$

Da Θ und u_l sowie u_r bekannt sind, kann R_l für die Rotation der linken und R_r für die Rotation der rechten Kamera bestimmt werden.
Somit ist die Homographie aus 2.15 bekannt und könnte auf die Bilder angewendet werden.
Um bei der Interpolation aber möglichst wenig Informationen zu verlieren, wurde vorher durch Herunterskalieren der Bilder eine Subpixel Auflösung erzielt. Somit lautet die endgültige Gleichung für unsere Transformation:

$$x_{imageRotated} = SKRK^{-1}x_{image} \tag{2.22}$$

Dabei stellt S die Skalierungsmatrix dar:

$$S = \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.23}$$

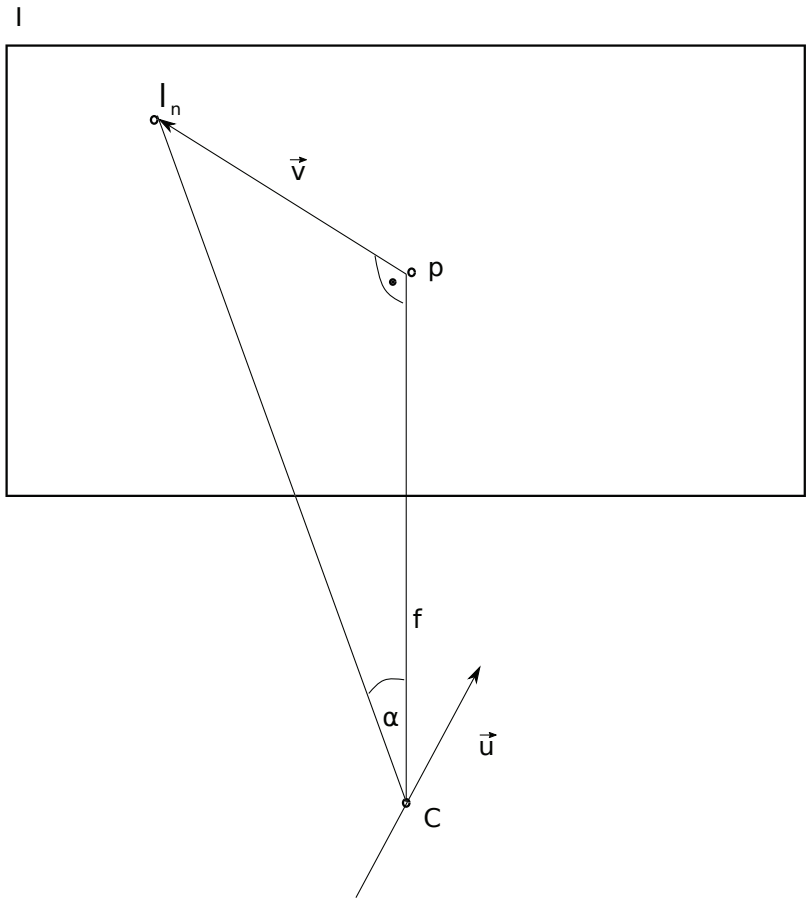

Abbildung 2.8: Berechnung des Rotationswinkels

Der Vektor $\vec{v}$ beschreibt die Richtung der primären Blickbewegung, wenn man vom Brennpunkt ausgehend auf den Punkt l_n schauen möchte. Zu diesem ist der Richtungsvektor $\vec{u}$ orthogonal. Er gibt die Richtung unserer Rotationsachse u an. Der Rotationswinkel Θ ist dabei gleich dem Winkel α des eingezeichneten Dreiecks.

2.3 Erstellung des Datensatzes

Die erläuterte Vorgehensweise wurde in Matlab implementiert und ausgeführt. Für die transformierten Bilder ergeben sich allerdings ein paar Probleme. Zum einen sind die beiden Bilder mit hoher Wahrscheinlichkeit unterschiedlich groß. Dies ergibt sich dadurch, dass die rechte und die linke Kamera um verschieden große Winkel gedreht werden mussten, um auf die korrespondierenden SURF Features zu schauen.

Abbildung 2.9: Beispiel für Bilder nach virtueller Kamerarotation

In diesen beiden Bildern wurden die jeweiligen Kameras virtuell auf einen neuen Fixationspunkt gedreht, welcher durch das blaue x markiert wurde. Dabei ist zu erkennen, dass sich das Bild verzerrt. Dies geschieht aufgrund einer Veränderung der Z-Werte bei der Rotation, wodurch manche Bereiche näher kommen und andere weiter in die Tiefe gehen. Das blaue x befindet sich allerdings nicht in der Mitte des endgültigen Bildes, dies geschieht dadurch, dass das Bild neu referenziert wird bei der Transformation. Wäre visuelle Information über das Umfeld des neuen Fixationspunktes verfügbar, so würde dieser nach der Transformation wieder in der Mitte des Bildes liegen. So hingegen kommen nur schwarze Bereiche in der Richtung des Fixationspunktes hinzu, welche beim neu Referenzieren herausfallen. Dadurch funktioniert die Transformation wie ein Shift-Objektiv.

Als Drittes kommt hinzu, dass die Bilder nach der Transformation über schwarze Keile verfügen, die keinerlei Information enthalten und in dieser Form auch nicht in natürlichen Szenen auftreten.
Das letzte Problem ist, dass die Feature Points, auf die vergiert wurde, nicht im Zentrum des Bildes liegen.
Durch Ausschneiden um den neuen Fixationspunkt werden die meisten dieser Probleme gelöst. Der gewählte Bereich hatte dabei eine Größe von 568x360 Pixeln. Anschließend musste noch darauf geachtet werden, dass Bildausschnitte, die über schwarze Bereiche verfügen, nicht für den Datensatz geeignet sind.

In dem Algorithmus wurde diese Selektion wie folgt implementiert: Da die Pixelwerte als uint8 codiert waren, wurden alle Pixel, die über einen Pixelwert von null verfügten, vor der Transformation auf eins gesetzt. Dadurch konnte sichergestellt werden, dass alle Nullen in den entstandenen Bildausschnitten künstlicher Natur sein mussten. Somit konnten alle Bildausschnitte mit künstlichen schwarzen Bereichen aussortiert werden.
Allerdings gab es darüber hinaus noch weitere Einschränkungen. So hat Stahl (1999) entdeckt, dass im Schnitt ab einem Vergenzwinkel von 20 Grad der Mensch von einer Augenbewegung zu einer Kopfbewegung übergeht. Entsprechend wurden künstlichen Vergenzwinkel von mehr als 20 Grad ebenfalls ausgeschlossen.
Eine weitere Einschränkung unserer Auswahl kommt daher, dass Matching Features oft sehr nahe beieinander liegen können (Siehe die oberen beiden Matching Features in Abb. 2.7). Um zu vermeiden, dass die entstehenden Bildpaare sich zu ähnlich sind, wurden alle Matching Features in einem Radius von 30 Pixeln zu einem Feature, auf welches vergiert wurde, gelöscht.
Unter Berücksichtigung dieser Selektionskriterien wurden dann aus Bildern Paare von schwarz-weißen Bildausschnitten erstellt, wobei zuerst auf die Matching Features mit der größten Featurestärke vergiert wurde.

In Abb. 2.10 kann man Paare von Bildausschnitten sehen, die durch den Algorithmus erzeugt wurden. So wurden aus den ursprünglich 1067 Bildern insgesamt 9075 Bilder für den Datensatz gewonnen.

Abbildung 2.10: Zwei durch Kamerarotation entstandene Bildpaare

Hier sind Paare von Bildern zu sehen, die durch virtuelle Rotationen der Kameras erzeugt wurden. Dabei liegen die endgültigen Bilder in schwarz-weiß vor, da für meine Untersuchung von Disparität Farbe keine große Rolle spielt.

2.4 Validierung des Datensatzes

2.4.1 Untersuchung 1

Bevor der Datensatz endgültig erstellt wurde, war es noch wichtig, die Korrektheit des entwickelten Algorithmus zu überprüfen.
Der einfachste Weg, um dies sicherzustellen, war es, Bilder von einer tatsächlichen Kamerarotation mit Bildern der virtuellen zu vergleichen. Dafür wurde mit einer Black Fly S von Point Grey eine Hausfassade unter verschiedenen Rotationswinkeln aufgenommen.
Zum Messen der Rotationswinkel und zur Sicherstellung, dass es sich um eine reine Rotation handelt, wurde eine Konstruktion gebaut, die ähnlich einer Kardanischen Aufhängung über mehrere Freiheitsgrade für die Rotation verfügt (Abb. 2.11).
Die Rotationswinkel reichten dabei von -20 bis +20 Grad um die Y-Achse und von -15 bis +10 Grad um die X-Achse. Die Winkel wurden relativ zu einem als zentral deklarierten Bild gemessen. Insgesamt wurden so 26 Bilder aufgenommen.
Die Kamerakalibrierung erfolgte hierfür mit der Camera Calibration App aus der Matlab Computer Vision Toolbox.
Da bei der mechanischen Rotation nur Winkel gemessen wurden, musste der Punkt, auf den die Kamera virtuell gedreht werden sollte, erst berechnet werden.
Wie man in Abb. 2.12 sehen kann, berechnet man mit dem Winkelsummensatz und dem Sinussatz den Unterschied der X-Komponente vom aktuellen Brennpunkt und dem neuen Brennpunkt, auf den mit der Kamera vergiert werden soll:

$$a = sin(\alpha) * \frac{f}{sin(\gamma)} \tag{2.24}$$

Für die Y-Komponente des neuen Brennpunktes verläuft die Berechnung nahezu analog.
Um die Bilder vergleichen zu können, wurden die Brennpunkte der Bilder aus der mechanischen Rotation mit denen der korrespondierenden virtuellen Rotation übereinander gelegt. So konnte anschließend Feature Matching betrieben werden. Da die Bilder unterschiedlich groß sind, musste um sie herum noch ein schwarzer Rand eingefügt werden, um sie auf die gleiche Größe zu bringen.
Die Abweichung der SURF Features kann man in den Histogrammen (Abb. 2.13) sehen.
Während die Abweichungen in Abb. 2.13 durch Messfehler erklärbar sind, sieht man in Abb. 2.14 Abweichungen, die an eine Rotation erinnern. Aus diesem Grund ist ein systematischer Fehler entweder in der virtuellen oder in der realen Rotation nicht auszuschließen.

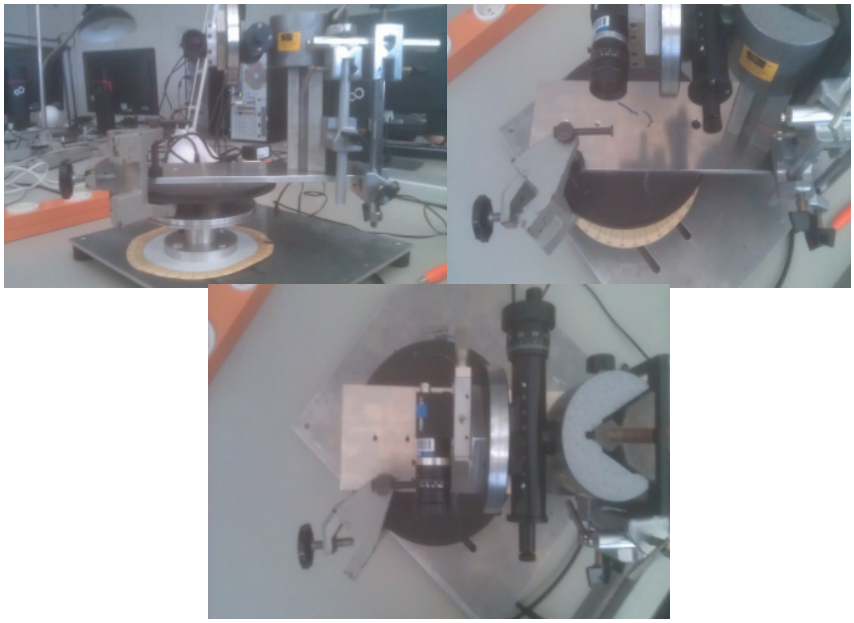

Abbildung 2.11: Kardanische Aufhängung

Die Konstruktion hat vergleichbar mit einer Kardanischen Aufhängung zwei Freiheitsgrade bei der Rotation. Sie steht auf einem Drehteller, wodurch frei um die Y-Achse gedreht werden kann. Darüber hinaus ist die Kamera auf einem drehbaren Kreiswinkelmesser über dem Zentrum des Drehtellers angebracht. So ist es möglich, die Kamera um die X- und Y-Achse zu drehen, ohne eine spürbare Translation zu erhalten.

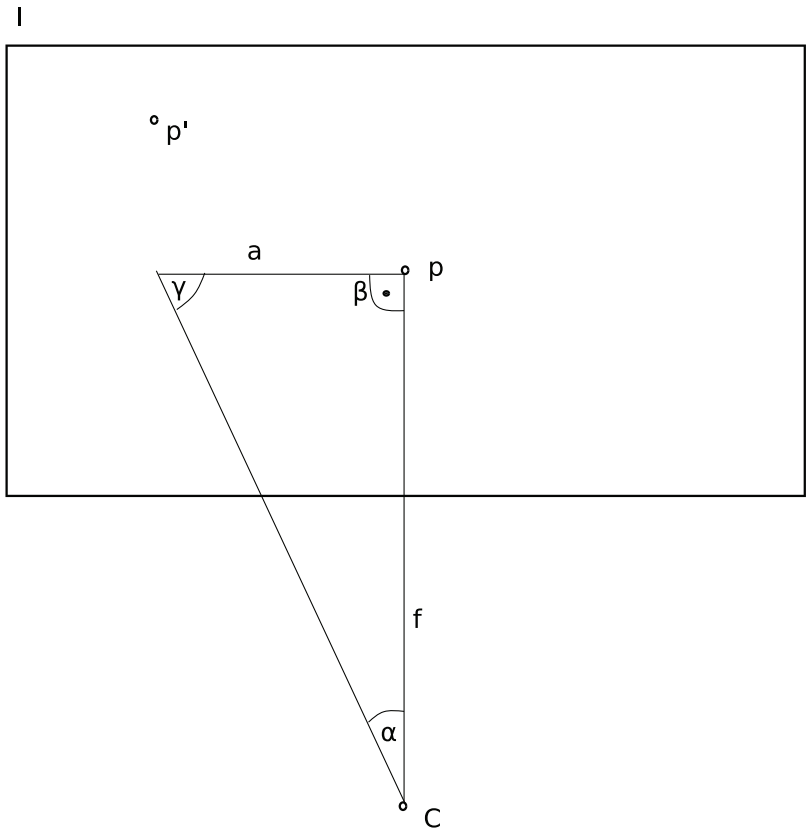

Abbildung 2.12: Berechnung des neuen Brennpunktes

Bei dem Winkel α handelt es sich um den Rotationswinkel um die Y-Achse, die Brennweite f ist bekannt und der Winkel zwischen Brennachse und Bildebene muss 90 Grad betragen. Damit lassen sich alle Winkel und somit auch a und der X-Wert des neuen Brennpunktes berechnen (Gleichung 2.24).

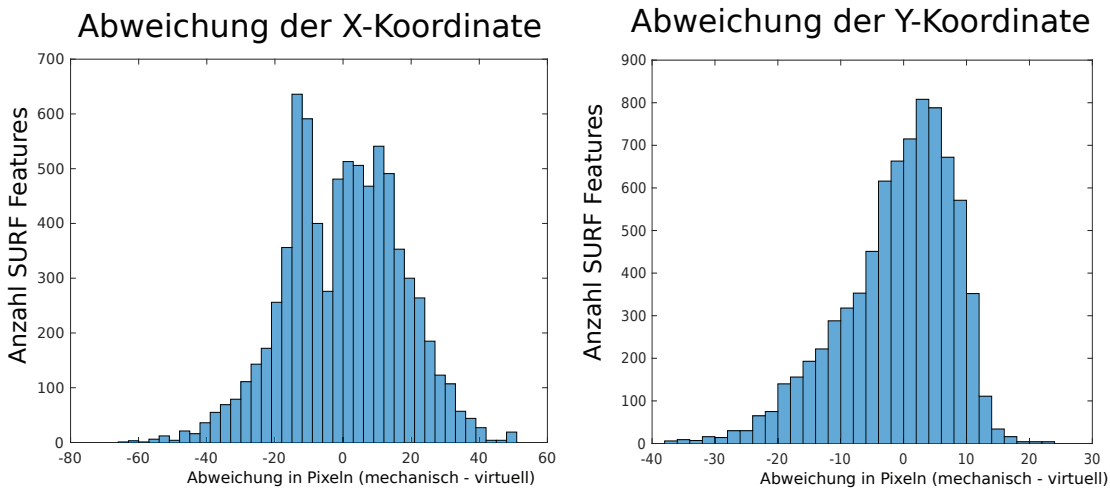

Abbildung 2.13: Unterschied von mechanischer zu virtueller Rotation

Hier wurde die Abweichung der korrespondierenden SURF Features von mechanisch und virtuell rotierten Bildern auf der X-Achse aufgetragen. Auf der Y-Achse sieht man die Anzahl der SURF-Features, wobei die SURF-Features aus allen korrespondierenden Bildpaaren (virtuelle und mechanische Drehung um den selben Winkel) berücksichtigt wurden. Wenn die mechanische Rotation der Kamera exakt der virtuellen entsprochen hätte, wären die Abstände genau null. Die Abstände sind in Pixeln angegeben, wobei ein Pixel einer Abweichung von 0.048 Grad entspricht. Man kann annehmen, dass Messfehler von bis zu einem Grad auftauchen können.

2.4.2 Untersuchung 2

Um diese Abweichung genauer zu untersuchen, wurden mittels OpenCV die optimalen Homographien zwischen dem zentralen Bild und den real rotierten Bildern berechnet. Anschließend wurde das zentrale Bild entsprechend transformiert.
Somit gab es 3 verschiedene Sets an Daten für diese Untersuchungen: die Bilder der mechanisch durchgeführten Rotationen, die Bilder der virtuellen Rotation durch meinen Algorithmus und Bilder der Homographien aus OpenCV.
Wie man sehen kann, handelt es sich bei letzteren um eine hinreichend gute Beschreibung unserer mechanisch ausgeführten Rotationen (Abb. 2.15 und Abb. 2.16).
Da die mittels der Homographien transformierten Bilder sehr gut zu den aufgenommenen Bildern passen, kann aus ihnen die tatsächlich, mechanisch ausgeführten Rotationen extrahiert werden. Mittels einer in OpenCV implementierten Funktion wurden die Rotationsmatritzen nach dem von Malis und Vargas (2007) beschriebenen Verfahren aus den Homographien berechnet.
Das Verfahren gibt zwei mögliche Rotationsmatrizen für jede Homographie aus. Es wurde jene ausgewählt, die näher an der theoretisch idealen Rotation lag. Wenn beispielsweise eine mechanische Rotation von 5 Grad um die X-Achse und 10 Grad um die Y-Achse gemessen worden war, dann wurde jene Rotationmatrix ausgewählt, deren Rotationen um die entsprechende Achse näher an den idealen Werten lagen. Die idealen Werte unterscheiden sich von den tatsächlichen Werten der Rotation durch den Messfehler, welcher tendenziell klein sein sollte.
In Abb. 2.17 kann man den Unterschied der tatsächlichen Rotationswinkel aus der mechanischen Rotation und den idealen, dabei gemessenen Rotationswinkeln sehen. Besonders auffällig hierbei ist, dass eine Rotation um die Z-Achse vorliegt, was nahelegt, dass die Achsen unseres Kamerakoordinatensystems verschoben waren.
Wenn man den Unterschied der Gesamtrotation um die arbiträren Achsen betrachtet, sieht man, dass der Unterschied sehr gering ist. Hier weicht die mechanisch ausgeführte Drehung kaum von der Drehung ab, die idealerweise durchgeführt worden wäre. Dieser kleine Unterschied liegt noch im Rahmen von möglichen Messfehlern.
Bemerkenswert ist hierbei noch die Tatsache, dass die Messfehler nicht um null verteilt sind. Dies kann durch eine systematische Unterschätzung der Größe von Rotationswinkeln beim Ausführen der mechanischen Rotation erklärt werden.
Eine alternative Erklärung ist in der Auswahl der Rotationsmatrizen zu

Abbildung 2.14: Beispiel für den Unterschied von mechanischer zu virtueller Rotation

Hier ist in rot das Bild aus der mechanischen Rotation zu sehen. In blau sieht man das Bild aus der virtuellen Rotation, das mein Algorithmus erzeugt hat. Als Ausgang wurde dafür das als zentral deklarierte Bild genommen. Dieses wurde um die aufgezeichneten Winkel virtuell gedreht, um die mechanische Drehung zu simulieren. Sollten die mechanische und die virtuelle Rotation identisch sein, würden die eingezeichneten SURF-Features sich genau überlagern. Dies ist nicht der Fall. Auch wenn der Fehler klein ist, weichen sie in einem potenziell systematischen Muster voneinander ab. Dieser systematische Fehler könnte entweder in der virtuellen oder in der mechanischen Rotation seinen Ursprung haben.

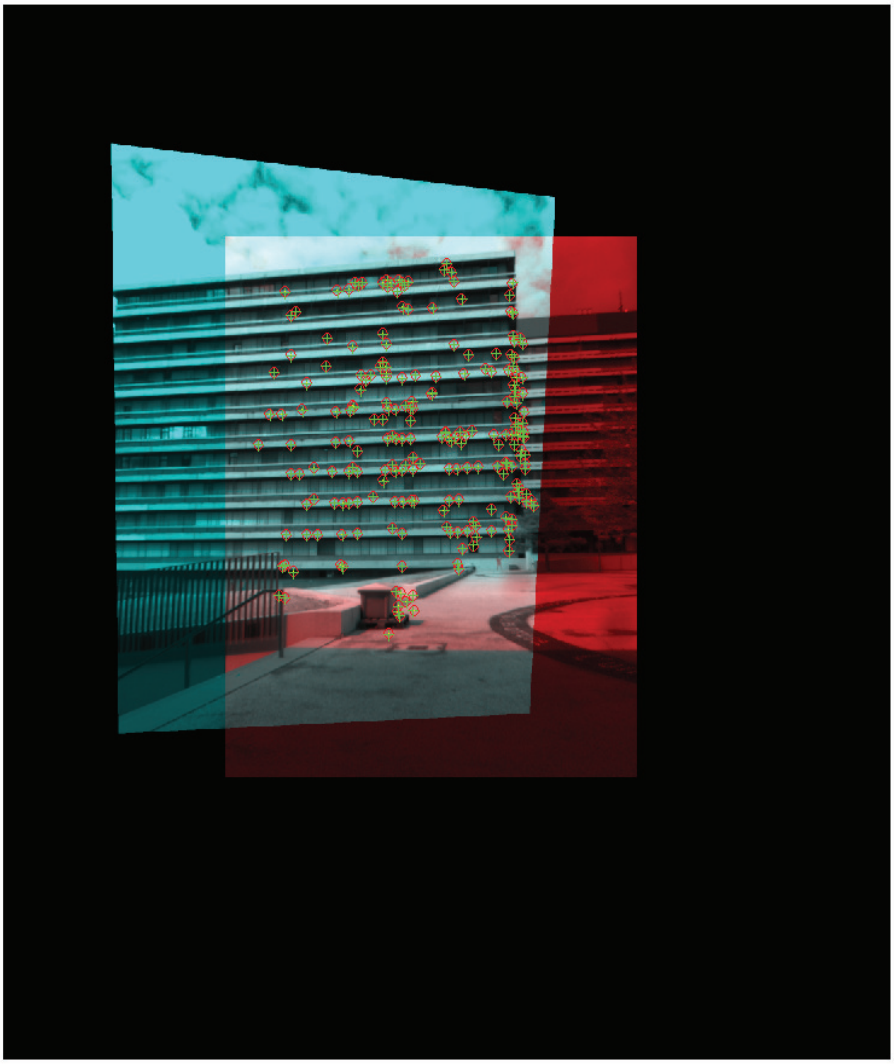

Abbildung 2.15: Beispiel eines durch OpenCV transformierten Bildes

Hier ist in rot ein durch mechanische Rotation entstandenes Bild zu sehen. Es wurde durch das blaue Bild überlagert. Dieses wurde durch eine mittels OpenCV berechnete virtuelle Transformation des zentralen Bildes erstellt. Es ist zu sehen, dass die einzelnen SURF Features des tatsächlich rotierten Bildes kaum von jenen des durch OpenCV berechneten Bildes abweichen.

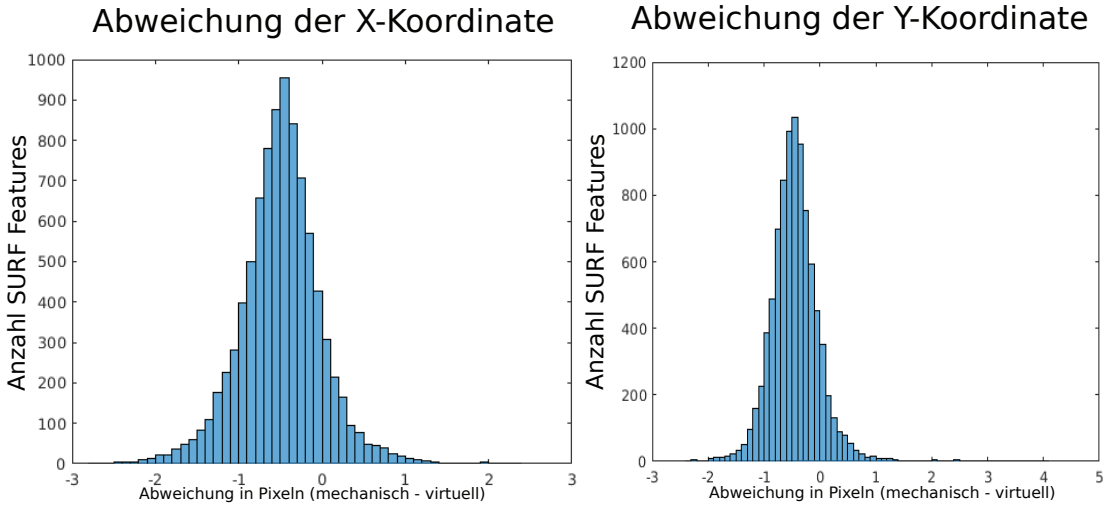

Abbildung 2.16: Unterschied der OpenCV Transformation und der mechanischen Rotation

Hier wurde die Abweichung der korrespondierenden SURF Features von mechanisch rotierten und mittels OpenCV transformierten Bildern auf der X-Achse aufgetragen. Auf der Y-Achse sieht man die Anzahl der SURF-Features, wobei die SURF-Features aus allen Einzelrotationen (für jedes aufgenommene Bild war eine Kamerarotation notwendig) berücksichtigt wurden. In dem linken Histogramm sind die Abweichungen der X-Werte der Matching SURF Features und im Rechten die Abweichung ihrer Y-Koordinaten zu sehen. Da die Abweichungen äußerst gering sind, kann davon ausgegangen werden, dass die Homographie die mechanisch ausgeführte Rotation hinreichend genau beschreibt.

finden. So liefert das Verfahren von Malis und Vargas (2007) immer zwei mögliche Rotationsmatrizen. Bei genauerer Untersuchung ließ sich feststellen, dass in den meisten Fällen eine der beiden einen größeren Drehwinkel als die ideale Rotation und eine einen kleineren Drehwinkel als die ideale Rotation hatte. Es wurde offensichtlich, dass die Rotationsmatrix mit dem kleineren Θ immer näher an der idealen Rotation lag. Diese muss allerdings nicht zwangsläufig die tatsächlich ausgeführte Rotation am besten beschreiben. So könnte durch die korrekte Auswahl der Rotationsmatrizen ein um null verteilter Messfehler gefunden werden. Die korrekte Auswahl für die beste Beschreibung der mechanisch Ausgeführten Rotationen ist im Nachhinein allerdings schwer festzustellen.

Wichtig ist hierbei vor allem, dass die systematische Abweichung in der mechanischen Rotation und ihrer Abweichung zur idealen Rotation zu finden ist. Somit handelt es sich bei dem zu beobachtenden Unterschied von Bildern aus der mechanischer Rotation und den von meinen Algorithmus erzeugten Bildern um kleine Messfehler und systematisch verschobene Kameraachsen bei der mechanischen Drehung.

2.4.3 Untersuchung 3

Wie in Abb. 2.10 B zu sehen ist, kann es zu einem erstaunlichen Größenunterschied bei der Abbildung von Objekten durch die Rotation der linken und der rechten Kamera kommen.

Um diesen Effekt zu untersuchen, wurde eine Skala ausgedruckt und viermal fotografiert. Die Kamera war dabei auf einer Metallschiene angebracht. Nun wurde ein Bild an Position 1 aufgenommen, auf dem die Skala gerade noch am rechten Bildrand zu sehen war. Dann wurde die Kamera auf der Metallschiene um 27 Zentimeter nach rechts verschoben und ein weiteres Bild wurde an Position 2 aufgenommen (Abb. 2.18). Anschließend wurde an den selben Positionen jeweils ein weiteres Bild aufgenommen, bei denen die Kameras durch eine Drehung um die Y-Achse auf die Skala ausgerichtet wurden. Dabei wurden die Rotationswinkel gemessen, um die Kameras virtuell analog drehen zu können.

Vor der Messung wurde die Radialverzeichnung aus allen Bildern entfernt. Hierbei ergab sich, dass die Skala an Position 1 eine Länge von 193 Pixeln vor und von 179 Pixeln nach der mechanischen Rotation aufwies. Nach der virtuellen Rotation des ersten Bildes von Position 1 wurde eine Länge der Skala von 180 Pixeln gemessen.

An Position 2 hatte die Skala vor der mechanischen Rotation eine Länge von 200 Pixeln und nach der Rotation eine von 197 Pixeln, wobei auch hier die Länge der Skala der virtuellen Rotation mit einer Länge von 198 Pixeln um einen Pixel abweicht. Bei der Abweichung kann es sich um einen

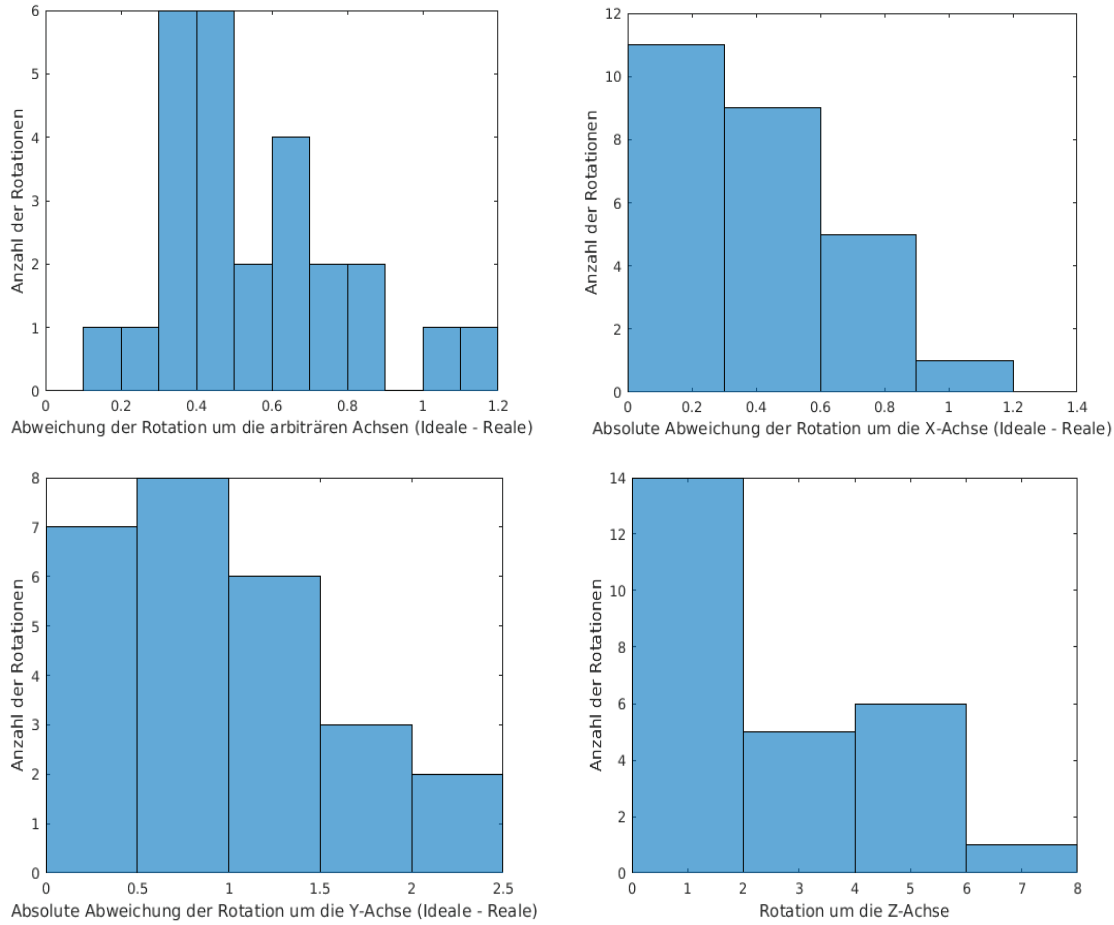

Abbildung 2.17: Abweichung der Rotationswinkel von idealer und mechanischer Rotation

In den Histogrammen sind die Abweichungen der idealen Rotation von der tatsächlichen Rotation zu sehen. Auf der X-Achse sind die Unterschiede in Grad aufgetragen, auf der Y-Achse die Anzahl der Rotationen (für jedes aufgenommene Bild wurde eine Kamerarotation durchgeführt). Vor allem ist auffällig, dass Rotationen um die Z-Achse vorliegen, da unsere Kardanische Aufhängung über keinen Freiheitsgrad bezüglich dieser Achse verfügt. Wenn die real ausgeführten Rotationen allerdings nicht perfekt um eine der Kameraachsen erfolgt sind, kann es dazu kommen, dass sich auch Drehungen um die Z-Achse ergeben. Dieses Ergebnis lässt sich am besten dadurch erklären, dass die Kamera leicht schief montiert wurde und somit die Achsen des Kamerakoordinatensystems verschoben waren. Deshalb wurde nicht exakt um die X- bzw. Y-Achse des Kamerakoordinatensystems gedreht. Die Abweichung der Rotationen um die arbiträren Achsen sind viel kleiner und in einem durch Messfehler zu erklärenden Rahmen.

Abbildung 2.18: Größenunterschied von Objekten bei der Rotation

Um den Größenunterschied von Objekten in der linken und rechten Kamera nach der virtuellen Rotation zu untersuchen, wurden eine Reihe von Bildern einer Skala aufgenommen. In der obereren Reihe sind die Bilder zu sehen, wie sie vor der mechanischen Rotation aufgenommen wurden. In der mittleren Reihe sieht man die Bilder nach der mechanischen Rotation. In der unteren Reihe sind die Bilder zu sehen, in denen die Kamera virtuell um den selben Winkel rotiert wurde. Gemessen wurde der Größenunterschied der Skala von null bis eins, wobei vor der Messung die Radialverzeichnung aus allen Bildern entfernt wurde. Der Größenunterschied der Skala nach mechanischer und nach virtueller Rotation betrug einen Pixel.

Effekt der Interpolation handeln. Gemessen wurde der Abstand mit dem Abstandsmesser-Tool des Bildbearbeitungsprogrammes GIMP.
Alles in allem legt das Ergebnis nahe, dass Größenunterschiede, wie beispielsweise in Abb. 2.10B gezeigt, durch die Rotation von Kameras vorkommen. Darüber hinaus untermauert dies auch, dass der Algorithmus auf korrekte Weise Bilder rotiert.

2.5 Diskussion

Ziel war es, ein Verfahren zu entwickeln, welches Kameras virtuell auf bestimmte Punkte in herkömmlichen Stereobildern vergieren lässt und dadurch neue Stereobildpaare mit Vergenz erzeugt. Dies ist gelungen und die Korrektheit des Verfahrens ist überprüft worden. Eine exaktere Messung unter Laborbedingungen wäre dafür wünschenswert gewesen, aber auch so hat sich die Korrektheit hinreichend herauskristallisiert. Dennoch gibt es Verbesserungsmöglichkeiten für die Qualität der entstandenen Bilder.
Zum einen wurde zuerst die Radialverzeichnung aus dem Bild herausgerechnet und in einem weiteren Schritt die Homographie auf das Bild angewendet. Dies führt dazu, dass man zweimal interpolieren muss und hat somit einen Verlust von Informationen zur Folge. Durch eine zukünftige Anpassung des Algorithmus sollte es möglich sein, die virtuelle Rotation der Kamera direkt nach dem Entfernen der Radialverzeichnung durchzuführen, um somit nur einmal interpolieren zu müssen.
Zusätzlich gehen Informationen dadurch verloren, dass auf alle Pixelwerte, die im Bild null sind, eins addiert wird, um unerwünschte Bilder zu selektieren. Hier könnte es durchaus noch von Vorteil sein, eine andere Methode zu implementieren, die ohne Informationsverlust dasselbe gewährleistet.
Letztlich lässt sich durch Justieren der Parameter noch Feintuning betreiben, was den entstehenden Datensatz angeht. So könnte der Radius, in dem Matching Features gelöscht werden, vergrößert werden, um so sich weniger ähnelnde Bilder zu erzeugen. Eine weitere Möglichkeit wäre die Detektion von SURF Features liberaler zu gestalten, um auf diese Weise mehr Bilder zu erhalten. Es sollte allerdings besonders darauf geachtet werden, dass beim Feature Matching die Parameter strikt genug sind, um keine falschen Matches zu generieren.
Auch für die Ausgansgdatenbank gibt es noch Verbesserungsmöglichkeiten. Um stereoskopisches Sehen genauer simulieren zu können, macht es Sinn, eine Stereokamera zu benutzen, deren Sensoren in einem ähnlichen Abstand wie die Augen von Säugetieren zueinander verschoben sind.
Je nach Untersuchung mag es auch sinnvoll sein, eine Kamera zu verwenden, die Bilder von Objekten aus sehr geringer Distanz aufnehmen kann, oder nur Bilder mit gewisser Tiefe zu wählen, die den direkt mit Händen zu manipulierenden Raum abdecken. In der Zukunft könnten solche Datenbanken dann für eine Vielzahl von Untersuchungen zu zielorientiertem Handeln, Planen und der Aufmerksamkeitssteuerung eingesetzt werden (Canessa et al., 2017). Auch für die Untersuchung von sensomotorischer Transformation könnte eine mittels des Verfahrens erstellte Datenbank helfen (vgl. Verhoef et al., 2016).

3 | Training auf dem Datensatz

3.1 Das binokulare Sparse-Coding-Modell

3.1.1 Visuelle rezeptive Felder

Bei visuellen rezeptiven Feldern handelt es sich um Bereiche der Retina, welche die Feuerrate von nachgeschalteten Neuronen maßgeblich beeinflussen. Wie in der Vorgängerstudie war vor allem das Modellieren von Simple Cells des primären visuellen Kortex von Interesse, die sensitiv für Disparitätsunterschiede sind. Der Begriff der Simple Cells wurde von Hubel & Wiesel (1962) eingeführt, die den primären und sekundären visuellen Kortex von Katzen untersucht haben.
Simple Cells zeichnen sich dadurch aus, dass sie klar definierte Regionen in ihren rezeptiven Feldern besitzen, die zu einer Erregung oder einer Inhibition der Neuronenaktivität führen (Abb. 3.1). Hubel & Wiesel(1962) beschreiben, dass Simple Cells nur dann antworten, wenn ein Stimulus an der richtigen Stelle in ihrem rezeptiven Feld auftaucht und über die richtige Orientierung verfügt. Ihre rezeptiven Felder werden als lokalisiert, orientierungsselektierend und Bandpass bezeichnet (vgl. Bear, Mark, Connors & Paradiso, 2009).
Um Disparitäten detektieren zu können, verfügen disparitätssensitive Simple Cells über ein rezeptives Feld im linken und eines im rechten Auge. Diese können identisch sein, was bedeutet, dass von dieser Zelle eine Disparität von null bevorzugt wird. Sind die rezeptiven Felder entweder orts- oder phasenverschoben, dann bevorzugen sie eine Disparität ungleich null und reagieren auf Stimuli, die in den Augen ebenfalls verschoben auf ihre rezeptiven Felder fallen (vgl. zum Beispiel Anzai, Ohzawa & Freeman, 1999; Fleet, Wagner & Heeger, 1996).
Ortsverschiebung kann man sich leicht vorstellen. Disparität, die durch eine horizontale Verschiebung des Bildes im linken und rechten Auge entsteht,

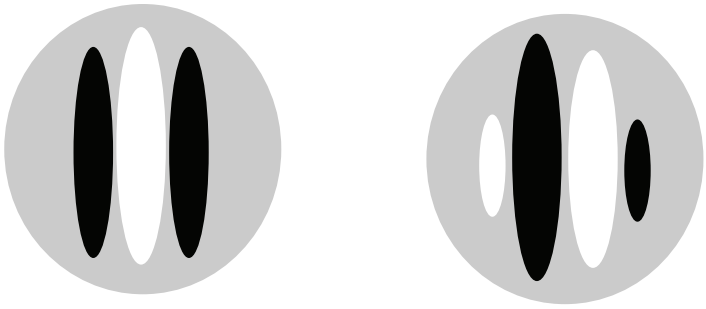

Abbildung 3.1: Beispiel für rezeptive Felder

Dargestellt sind zwei phasenverschobene rezeptive Felder. Sollte ein Lichtstimulus in den weißen Bereich fallen, führt dies zu einer Erregung der Zelle. Diese Bereiche werden auch als ON-Regionen bezeichnet. Fällt ein Lichtstimulus hingegen in den schwarzen Bereich, führt dies zu einer Inhibition der Aktivität. Diese Bereiche des rezeptiven Feldes werden OFF-Regionen genannt. Es wird allgemein davon ausgegangen, dass über das gesamte rezeptive Feld summiert wird. So erzeugt ein Stimulus, der gleichermaßen in OFF wie in ON Regionen fällt, keine Aktivität. Durch die Phasenverschiebung würde eine Simple Cell im V1, die mit beiden rezeptiven Feldern verschaltet ist, dann maximal reagieren, wenn ein Stimulus im rechten Feld nach rechts verschoben auf die ON Region trifft. Bei der Ortsverschiebung hingegen wird das ganze rezeptive Feld verschoben.

kann durch ebenfalls horizontal verschobene rezeptive Felder in den beiden Augen detektiert werden (Tsao, Conway & Livingston, 2003). Der andere Typus sind phasenverschobene rezeptive Felder.
Man kann die Profile von rezeptiven Feldern gut durch Gaborfunktionen modellieren (Jones & Palmer, 1987). Eine Gaborfunktion ist das Produkt einer Gaußfunktion und einer Sinuswelle. Die Phase eines rezeptiven Feldes bezieht sich dabei auf die Phase der Sinuswelle und gibt die Abfolge von ON und OFF Regionen an, die somit verschoben werden können (Tsao, Conway & Livingston, 2003).
Durch eine Verschiebung des Zentrums der Gaußfunktion kann auch die Ortsverschiebung durch Gaborfunktionen modelliert werden.

3.1.2 Sparse-Coding

In der von Olshausen & Field (1996) vorgeschlagenen Herangehensweise für das Finden von biologisch plausiblen rezeptiven Feldern wird nach einem Code für natürliche Bilder gesucht, der *sparse* ist. Unter natürlichen Bildern versteht man die Untermenge von Bildern, die natürlich vorkommende Szenen enthalten. Im Vergleich dazu wäre ein Bild mit zufälligen Grauwerten pro Pixel höchstwahrscheinlich nicht in dieser Menge enthalten und somit für das Modellieren von biologischen Feature-Detektoren uninteressant.
Olshausen & Field (1996) beginnen mit der Annahme, dass natürliche Bilder und somit sämtlicher visueller Input in eine Reihe von grundlegenden Features, wie orientierte Kanten und Ecken, zerlegt werden kann. Da die Zahl der grundlegenden Features in einem einzelnen Bild endlich ist, kann das Bild durch die Kombination von endlich vielen Features rekonstruiert werden. Dies lässt sich wie folgt beschreiben:

$$I = \sum_i \Phi_i z_i \tag{3.1}$$

Bei den Φ_i handelt es sich um einen Satz von Basisvektoren, welche die grundlegenden Features repräsentieren. Ziel ist es, einen Satz von Basisvektoren zu erhalten, der in der Lage ist, die gesamte Menge von natürlichen Bildern zu codieren. Ihre Zusammensetzung hängt stark von den statistischen Auftretungswahrscheinlichkeiten verschiedener Features in den Bildern ab.
Bei z_i handelt es sich um Featuremaps (Schultz et al., 2014), welche von Bild zu Bild unterschiedlich sind. Sie enthalten Informationen über die Aktivität von Basisvektoren an den verschiedenen Stellen.
Das Ziel beim Sparse-Coding ist zum einen, jedes einzelne Bild mit einer möglichst kleinen Menge an aktiven Basisvektoren codieren zu können, ohne dabei Informationen zu verlieren (Field, 1994). Dies geht damit einher, dass jeder Basisvektor möglichst selten aktiv sein sollte. Zum anderen ist

es darüber hinaus noch wünschenswert, dass für die Codierung der Menge aller natürlichen Bilder jeder Basisvektor in etwa gleich oft benötigt wird. Ansonsten hat man im Extremfall Basisvektoren dabei, die nur für die Codierung eines einzelnen Bildes gebraucht werden.
Für diese Herangehensweise haben Olshausen & Field (1996) ein Verhältnis von Sparsity zu Informationsgehalt aufgestellt:

$$E = -[\textit{Information erhalten}] - \lambda[\textit{Sparsity}] \tag{3.2}$$

Hieran zeigt sich, dass der Wunsch, Bilder möglichst genau codieren zu können, in einem Wettstreit mit dem Ziel einer hohen Sparsity im Code liegt. Wenn diese beiden Ansprüche korrekt ausbalanciert werden, so ist man in der Lage, über das Minimieren der Energiefunktion E einen Satz von Basisvektoren zu erlernen, die den rezeptiven Feldern im primären visuellen Kortex stark ähneln (Olshausen & Field, 1996).
Um die Güte der erlernten Basisvektoren sicherzustellen, wird im überbesetzten Fall nach ihnen gesucht. Das bedeutet, dass mehr Basisvektoren als Dimensionen des Input-Bildes benötigt werden. In diesem Fall wird es wahrscheinlicher, dass eine gute Repräsentation für das Bild gefunden wird, die gleichzeitig sparse ist (Schultz et al., 2014). Der Nachteil hierbei ist natürlich, dass je größer das Input-Bild wird, umso mehr Basisvektoren benötigt werden.
Um dieses Problem zu umgehen, kann das von Zeiler, Krishna, Taylor & Fergus (2010) beschriebene *Deconvolutional Neural Network* genutzt werden. Da die einzelnen Ausschnitte eines natürlichen Bildes statistisch invariant von ihrer Position auf dem Bild sind, kann das Bild gefaltet und durch Faltungskerne codiert werden (Zeiler et al., 2010; Zhang, 2016). Auf den Featuremaps werden die Aktivierungskoeffizienten der Faltungskerne an den mit Bildausschnitten korrespondierenden Stellen gespeichert, um die räumliche Struktur des Bildes aufrecht zu erhalten. So ist es möglich, eine große Überbesetzung mit einer geringen Anzahl an Basisvektoren zu erreichen. Bei den rezeptiven Feldern von Basisvektoren handelt es sich dann um Faltungskerne.

3.1.3 Modell

In dem Sparse-Coding-Modell von Schultz et al.(2014) werden $M \times N$ Pixel große Bilder I durch einen Satz von überbesetzten Basisvektoren Φ_j und den dazugehörigen Feature Maps z_j codiert. Bei den Φ_j handelt es sich um einen Satz von j Faltungskernen.

$$I \approx \sum_{j=1}^{J} \Phi_j * z_j \tag{3.3}$$

Mit dieser Formel können Bilder durch verschobene Faltungskerne, die mit den Aktivierungskoeffizienten der korrespondierenden Stellen gewichtet sind, rekonstruiert werden. Die Verschiebung des Faltungskerns wird dabei als Stride S bezeichnet. Ein Stride von eins beschreibt eine Verschiebung des Faltungskerns um einen Pixel. Daraus folgt, dass die Größe der Featuremaps gleich $1/S^2$ ist.
Für die Suche nach einem Sparse-Code wird folgende Gleichung basierend auf Gleichung 3.2 und Gleichung 3.3 genutzt.(Schultz et al., 2014):

$$E = \frac{1}{2}\underbrace{\|I - \sum_{j=1}^{J} \Phi_j * z_j\|_2^2}_{Residuum} + \underbrace{C(z)}_{Sparsity} \tag{3.4}$$

Der Rekonstruktionsfehler wird als L_2-Norm des Unterschieds von dem übergebenen Bild und dessen Rekonstruktion berechnet. Die Sparsity wird durch folgenden Term forciert:

$$C(z) = \sum_{j=1}^{J} C_\lambda(Z_n), \text{ mit } C_\lambda(a) = \begin{cases} \lambda \text{ falls } |a| \geq \lambda \\ 0 \text{ sonst} \end{cases} \tag{3.5}$$

Hier wird die rechenaufwändige L_0-Norm durch die L_1-Norm angenähert und Aktivitätsmuster der künstlichen Neurone, die nicht sparse sind, bestraft.
Da in einem Deconvolutional Neural Network die Akvititäten von Neuronen nur durch benachbarte Neurone beeinflusst werden, haben Schultz et al.(2014) Locally Competetive Algorithms (LCAs) wie von Rozel, Johnson, Baraniuk & Olshausen(2008) beschrieben genutzt. Dadurch haben in ihrem *Deconvolutional LCA Network* Neurone *Leaky-Eigenschaften* und darüber hinaus kann laterale Inhibition zwischen den künstlichen Neuronen simuliert werden. Ihr Modell wird dann durch folgende Gleichungen beschrieben:

$$r_j = I_j - (\Phi z)_j \tag{3.6}$$

$$\frac{du_k}{dt} = -u_k + z_k + (\Phi^T r)_k \tag{3.7}$$

$$z_k = C(u_k) \tag{3.8}$$

r_j beschreibt dabei die Residuumsschicht, die den Fehler des rekonstruierten Bildes bezüglich des momentanen Bildinputs enthält. Sie korrespondiert mit dem Residuum aus Gleichung 3.4. u_k beschreibt den internen Zustand eines korrespondierenden Neurons z_k aus der V1-Schicht. Somit gibt Gleichung 3.7 die Veränderung der Neuronenaktivität über die Zeit an. Gleichung 3.8 nutzt wieder die Schwellenwertfunktion aus Gleichung 3.5.
Der Basisvektor Φ_n wird dann durch Gradientenabstieg von Gleichung 3.4

gelernt:

$$\Delta\Phi_n(q) = \delta \sum_{p=1}^{P} \left(\overbrace{I(p) - \sum_{q} \Phi_n(q) z_n(p-q)}^{\text{Residuum von Bildpunkt p}} \right) z_n(p-q) \tag{3.9}$$

Aufgrund der Faltung berechnet sich die Änderung eines Faltungskerns Φ_n an der Stelle q durch den Einfluss dieser Stelle auf den Fehler der Rekonstruktion der einzelnen Bildpunkte p.
Lundquist et al.(2016) haben das Konzept des Sparse-Codings zu einem binokularen Modell erweitert. Die Energiefunktion sieht dafür wie folgt aus:

$$E = \frac{1}{2} || \underbrace{I_L - \sum_{j=1}^{J} \Phi_{L_j} * z_j}_{Linkes\ Residuum} ||_2^2 + \frac{1}{2} || \underbrace{I_R - \sum_{j=1}^{J} \Phi_{R_j} * z_j}_{Rechtes\ Residuum} ||_2^2 + \underbrace{C(z)}_{Sparsity} \tag{3.10}$$

Bei Φ_{L_j} und Φ_{R_j} handelt es sich um das linke und das rechte rezeptive Feld eines künstlichen Neurons z_j aus der V1-Schicht z.
Yue Zhang(2016) hat das Deconvolutional LCA Network mit dem binokularen Sparse-Coding-Modell kombiniert (Abb. 3.2).

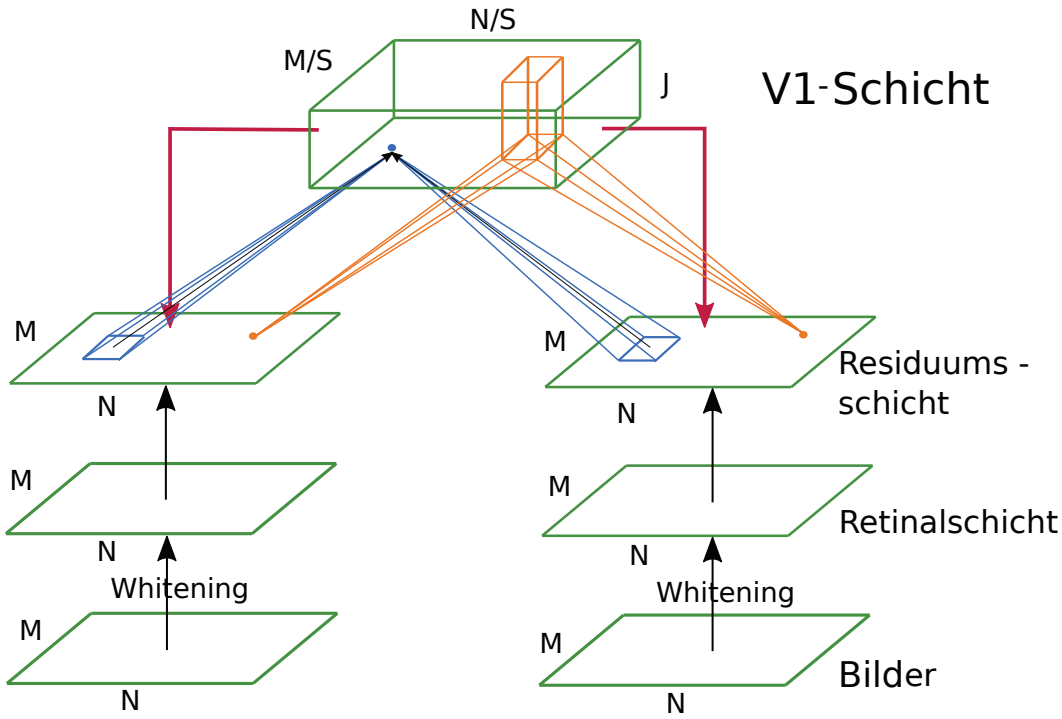

Abbildung 3.2: Binokulares Sparse-Coding-Model

Hier ist der Aufbau des binokularen Sparse-Coding-Modells zu sehen. Es bekommt ein linkes und ein rechtes Bild als Eingabe. In der Residuumsschicht wird der Unterschied der Eingabe (schwarze Pfeile von unten kommend) und der Rekonstruktion (rote Pfeile) berechnet. Die V1-Schicht enthält die Featuremaps der J Basisvektoren. In blau sieht man die rezeptiven Felder eines künstlichen Neurons eingezeichnet. In orange sieht man die künstlichen Neurone in der V1-Schicht, deren rezeptive Felder durch die korrespondierenden, orangefarbenen Punkte beeinflusst werden (Grafik nach Yue Zhang, 2016).

3.2 Methode

3.2.1 Training

Das Modell wurde mittels der Petavision Toolbox implementiert. Dabei handelt es sich um eine Toolbox für das Simulieren von neuronalen Aktivitäten durch künstliche neuronale Netze. Aus Zeitgründen konnte das Netzwerk nur mit einem Teil des oben beschriebenen Datensatzes trainiert werden. Es wurden 256 Basisvektoren mit einer Größe von 64x32 Pixeln gelernt und ein Stride von $S = 4$ genutzt. Da mit einem rektifizierten Modell gearbeitet wurde, ergibt sich eine Überbesetzung der Basis von $\frac{256}{2 \cdot 4^2} = 8$.
Um das Training schneller zu gestalten, wurden gleichzeitig mehrere Modelle trainiert und in regelmäßigen Intervallen die besten Gewichte als Ausgangspunkt für das weitere Training aller Modelle genutzt. Zwischen den Modellen wurden die Lernrate und der Schwellenwert für die Sparsity variiert.

3.2.2 Image-Shift-Test

Als nächstes wurden die entstandenen rezeptiven Felder mittels eines von Zhang (2016) entworfenen Image-Shift-Tests auf ihre Aktivitäten bezüglich verschiedener horizontaler und vertikaler Disparitäten untersucht.
In diesem Test wurden die Gewichte konstant gehalten und das Model musste 425 zueinander verschobene natürliche Bilder codieren. Dabei wurde auf der linken Seite immer das selbe Bild übergeben. Auf der rechten Seite wurde dieses Bild mit einer horizontalen Verschiebung von -6 bis 6 Pixeln und einer vertikalen Verschiebung von -4 bis 4 Pixeln präsentiert.

3.3 Ergebnisse

Für die Auswertung der erlernten rezeptiven Felder wurden von Zhang (2016) entwickelte Verfahren verwendet.
In Abb. 7.1 sind alle 256 gelernten rezeptiven Felder zu sehen und in Abb. 3.3 die 42 aktivsten rezeptiven Felder. Wie in Abb. 3.4 deutlich wird, konnte eine Reihe von verschiedenen, rezeptiven Feldern gefunden werden.
Um die Eigenschaften der erlernten, rezeptiven Felder untersuchen zu können,

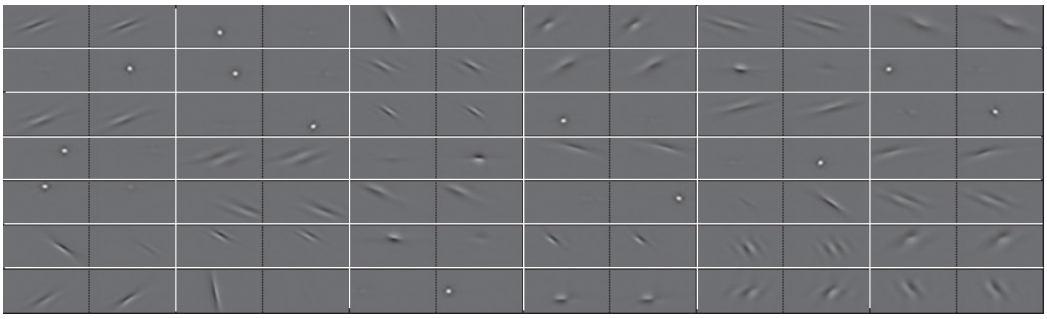

Abbildung 3.3: Die 42 aktivsten Basisvektoren

Hier sind die Profile der rezeptiven Felder von den 42 aktivsten Basisvektoren zu sehen. Das linke und das rechte rezeptive Feld eines Basisvektors ist durch eine dünne schwarze Linie getrennt worden. Eine weiße Linie zeigt den Übergang zu den rezeptiven Feldern des nächsten Basisvektors an.

wurden ihnen zweidimensionale Gaborfunktionen mit Gauß-Enveloppe angepasst. Dadurch lassen sich Phase, Ort, Wellenlänge und Orientierung der rezeptiven Felder mathematisch beschreiben und vergleichen. 50 rezeptive Felder wurden aufgrund schlechter *Fits* ausgeschlossen.
Abb. 3.5 A und C zeigen, dass die Wellenlänge der linken und rechten rezeptiven Felder, sowie deren Orientierung, stark miteinander korrelieren. In Abbildung 3.5 B und D ist zu sehen, dass über 90 % der Abweichungen von Wellenlänge und Orientierung weniger als zwei Pixel, beziehungsweise 5 Grad betragen.
Wenn man dies mit den Ergebnissen aus der Vorgängerstudie vergleicht, sieht man, dass die Eigenschaften von rezeptiven Feldern in den beiden Studien kaum voneinander abweichen. Interessanterweise ist in beiden Studien zu finden, dass der Orientierungsunterschied vom linken und rechten rezeptiven Feld nicht um null verteilt ist. Vielmehr ist die Verteilung leicht nach rechts verschoben.
In der Abbildung 3.6 B zeigt sich, dass sowohl phasen- als auch ortsverschobene rezeptive Felder gelernt wurden. Die Eigenschaften der rezeptiven Felder weichen auch hier nicht von jenen aus der Vorgängerstudie ab.

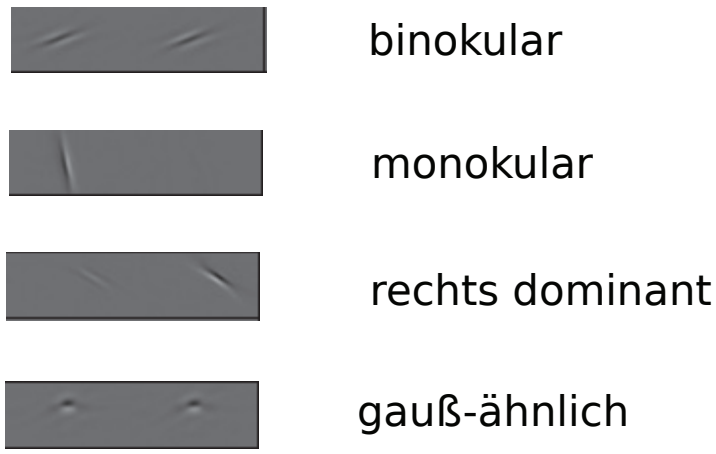

Abbildung 3.4: Verschiedene Arten von rezeptiven Feldern

Hier sind die Profile der rezeptiven Felder von verschiedenen Typen an Basisvektoren abgebildet. Binokulare Basisvektoren verfügen über rezeptive Felder, deren linkes und rechtes Profil gleichmäßig stark ausgeprägt ist. Bei monokularen ist nur eines der rezeptiven Felder ausgeprägt. Bei einem okulär-dominanten Basisvektor ist das eine rezeptive Feld stärker ausgeprägt als das andere. Gauß-ähnliche Basisvektoren haben rezeptive Felder, die sich besser durch Gauß- als durch Gaborfunktionen beschreiben lassen (Zhang, 2016).

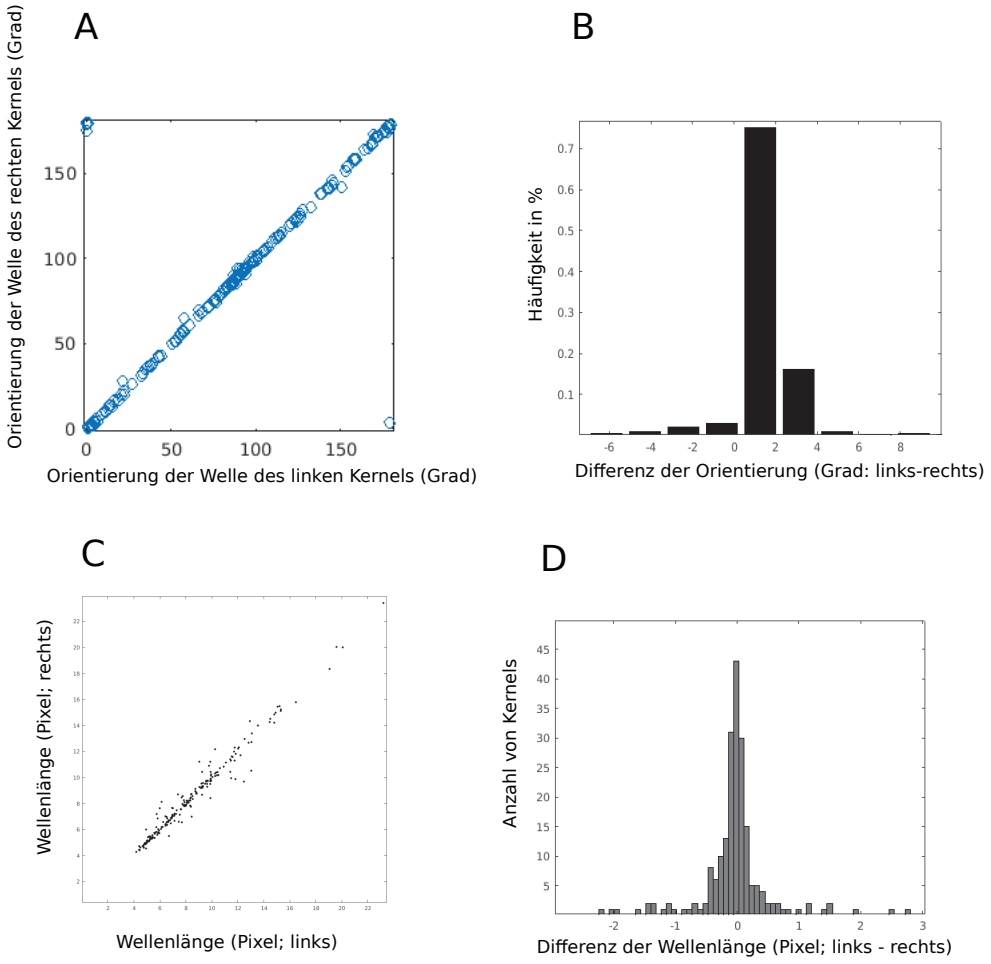

Abbildung 3.5: Eigenschaften der rezeptiven Felder von Basisvektoren I

A ist ein Scatterplot, der die Korrelation der Orientierung von linkem und rechtem rezeptiven Feld eines Basisvektors angibt. Es kann gesehen werden, dass diese stark korrelieren. B zeigt den Unterschied der Orientierung von linkem und rechtem rezeptiven Feld in Grad an. In C sieht man einen Scatterplot der zeigt, dass die Wellenlängen der linken und rechten rezeptiven Felder korrelieren. In D ist der Unterschied der Wellenlänge von linken und rechtem rezeptiven Feld abgebildet. Diese Korrelationen und Verteilungen wurden in der nahezu selben Form in der Vorgängerstudie gefunden (vgl. Zhang, 2016).

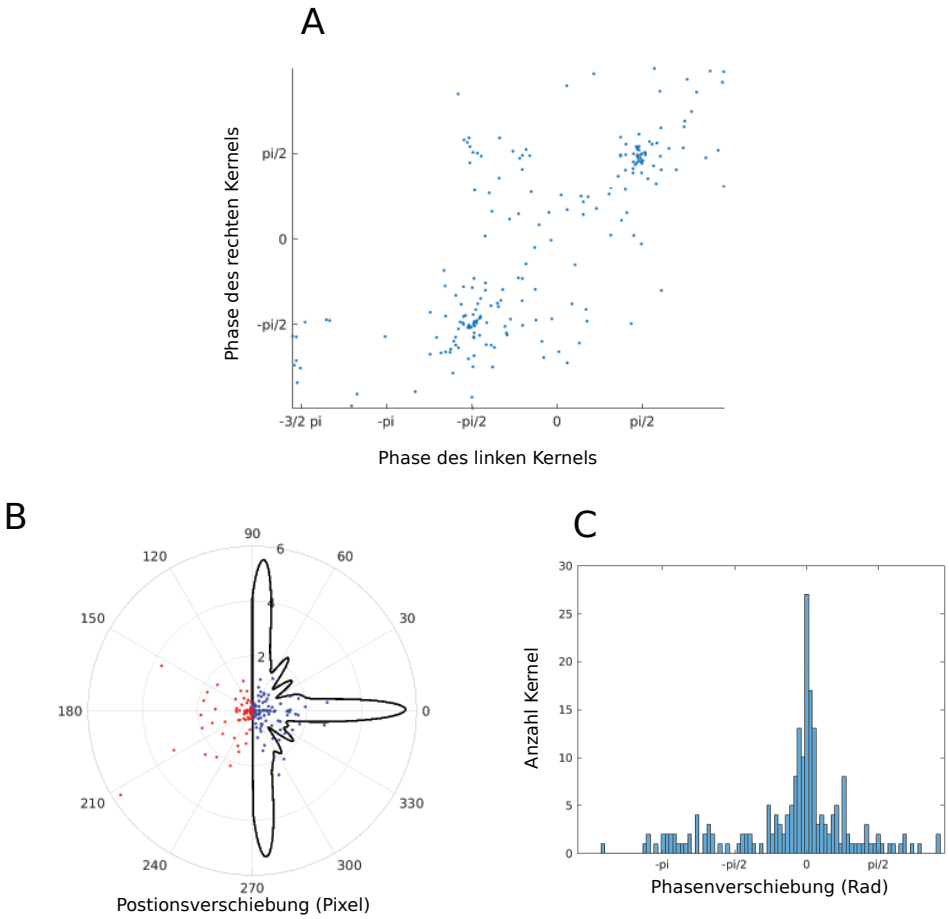

Abbildung 3.6: Eigenschaften der rezeptiven Felder von Basisvektoren II

A ist ein Scatterplot, in dem die Phase des rechten rezeptiven Feldes gegen die Phase des linken rezeptiven Feldes aufgetragen wurde. Die Punkte häufen sich bei$(-\frac{\pi}{2}, -\frac{\pi}{2})$ und $(\frac{\pi}{2}, \frac{\pi}{2})$. Dies legt nahe, dass die meisten Basisvektoren über ungerade rezeptive Felder verfügen (Sinus). Bei B handelt es sich um einen Polarplot, in dem der Positionsunterschied vom linken und vom rechten rezeptiven Feld in Orientierungsrichtung der rezeptiven Felder aufgetragen wurde. Rezeptive Felder mit einer vertikalen Orientierung hatten dabei 0 Grad und horizontale hatten 90 beziehungsweise 270 Grad Orientierung. Die roten Punkte zeigen negative, die blauen positive Positionsverschiebungen an. Die schwarze Linie zeigt die Dichteverteilung der Vektoren. Man kann sehen, dass mehr vertikal orientierte rezeptive Felder eine horizontale Verschiebung aufweisen als horizontale Vektoren vertikale Verschiebungen. In der Abbildung C zeigen sich die Phasenverschiebungen von linken und rechten Kernels. Der größte Teil befindet sich dabei im Bereich von $-\frac{\pi}{2}$ zu $\frac{\pi}{2}$. Auch die Eigenschaften dieser rezeptiven Felder weichen nicht elementar von denen aus der Vorgängerstudie ab (vgl. Zhang, 2016)

In Abbildung 3.7 sind die Tuningkurven von verschiedenen Basisvektoren zu sehen. Dabei zeigt sich, dass wie in der Vorgängerstudie Basisvektoren mit Tuningkurven zu finden sind, die denen von Affenneuronen gleichen (Poggio, Gonzales & Krause, 1988; Zhang, 2016).
In Abbildung 3.8 ist die L_1-, L_2- und die Energienorm E für den Image-Shift-Test zu sehen. Die L_2- und die Energienorm sind konstant niedrig, während die L_1-Norm in der Mitte, bei den Bildpaaren mit geringerer Disparität, einsinkt.
In Abbildung 3.9 A sind die bevorzugten Disparitäten einzelner Basisvektoren zu sehen. Dabei ist festzustellen, dass besonders viele für Disparitäten nahe null getunt sind. In 3.9 B sind die Aktivitäten aller Neurone für die verschiedenen Disparitäten aufsummiert worden. Hier sind weniger Basisvektoren für horizontalen Disparitäten und Disparitäten nahe null aktiv. Diese Muster haben sich in der selben Weise in der Vorgängerstudie finden lassen (vgl. Zhang, 2016).

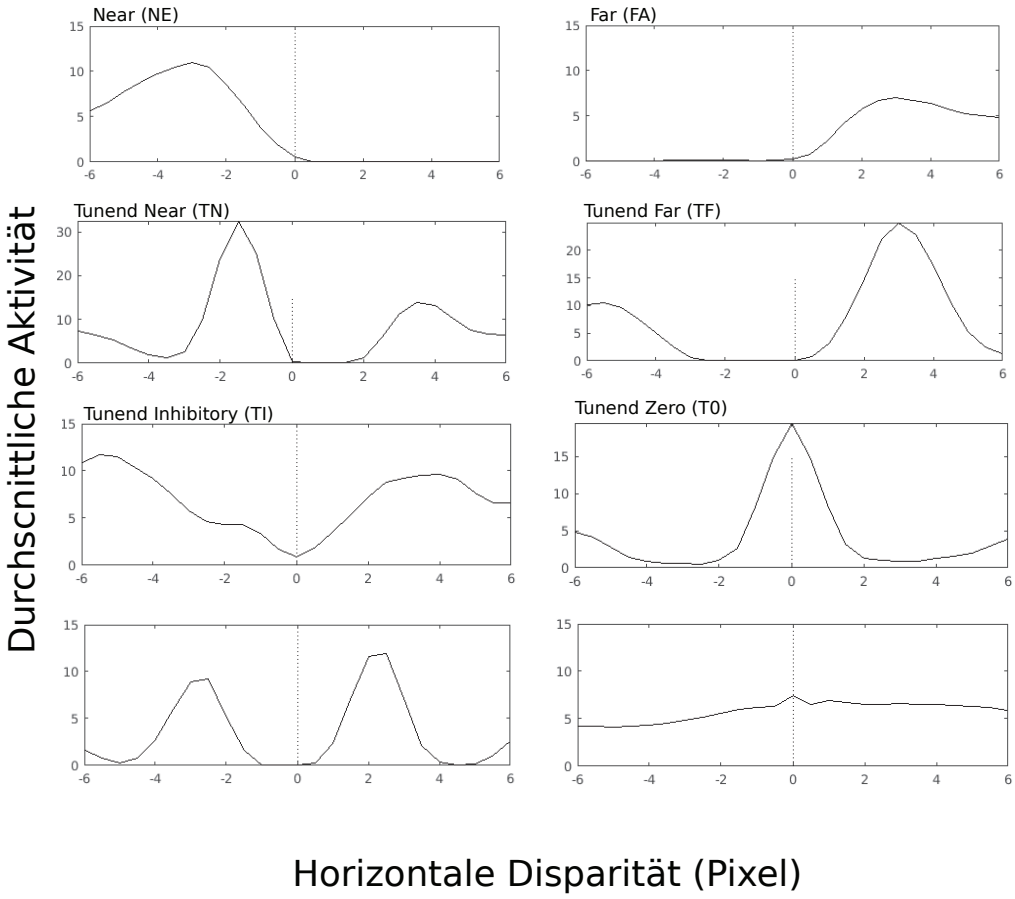

Abbildung 3.7: Tuningkurven der Basisvektoren

Hier ist zu sehen, dass Tuningkurven für jede der von Poggio et al. (1988) beschriebenen sechs Kategorien gefunden wurden. Basisvektoren aus der Kategorie (NE) reagieren auf Stimuli, die näher als der Fixationspunkt sind, während jene aus der Kategorie (FA) auf Stimuli, die weiter entfernt als der Fixationspunkt sind, reagieren. Neurone aus den Kategorien (TF) und (TN) reagieren bevorzugt auf spezifische Disparitäten vor oder hinter dem Fixationspunkt. Basisvektoren aus der Kategorie (TI) reagieren bevorzugt auf Disparitäten größer als null, Basisvektoren in der Kategorie (T0) bevorzugt auf Disparitäten gleich oder nahe null. In der untersten Reihe sind zwei Beispiele von Tuningkurven zu sehen, die sich nicht eindeutig in eine der Kategorien einordnen lassen.

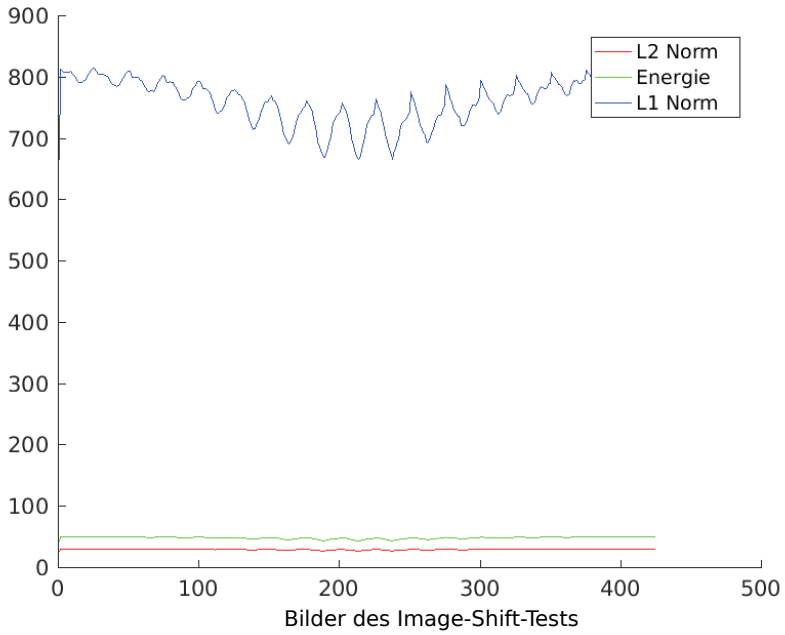

Abbildung 3.8: Fehlerkurve für den Image-Shift-Test

Hier wurden für jedes Bild aus dem Image-Shift-Test die verschiedenen Normen aufgetragen. Zum einen ist die L_1-Norm zu sehen, die anzeigt, wie viele Basisvektoren aktiv waren für die Rekonstruktion eines Bildes (vgl. Gleichung 3.5). Die L_2-Norm zeigt den noch existierenden Unterschied von rekonstruiertem zu übergebenem Bild an (vlg. Gleichung 3.10). Die Energienorm berechnet sich aus der Gewichtung der beiden. Die verschiedenen Normen wurden für die letzte Repräsentation jedes der 425 Bilder aufgezeichnet. Dabei ist auffällig, dass die L_1-Norm in der Mitte niedriger ist. Die Energie- und L_2-Norm hingegen bleiben konstant niedrig. Die Bilder wurden der Reihe nach von den stärksten negativen Verschiebungen zu den stärksten positiven hin präsentiert. Das 213. Bildpaar hatte dabei die Disparität null (identische Bilder).

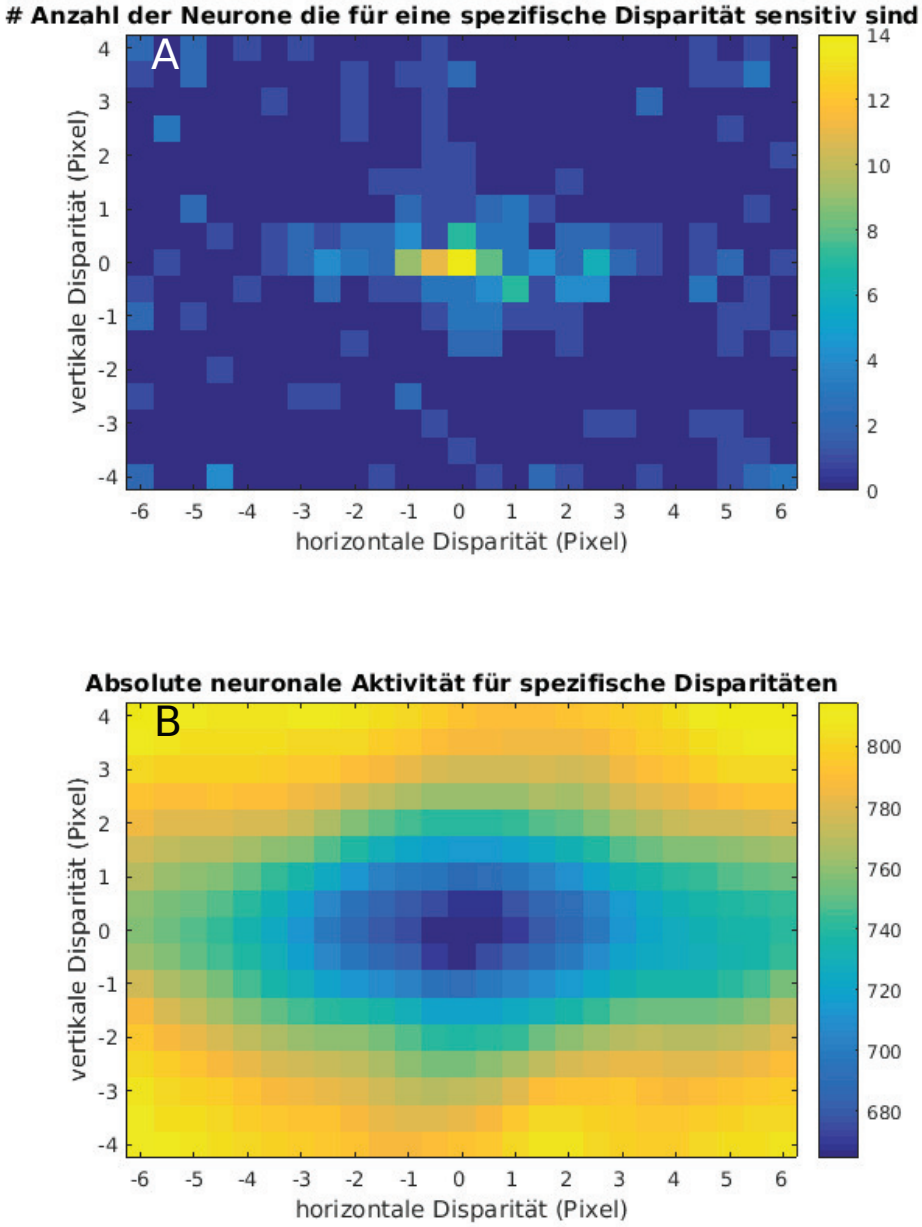

Abbildung 3.9: Ergebnisse des Image-Shift-Tests

In A wurden für die verschiedenen Disparitäten die Zahl jener Basisvektoren aufsummiert, die für diese Disparität am aktivsten waren (für sie getunt sind). Dabei ist eine Ansammlung von Basisvektoren bei Disparitäten nahe null zu beobachten. In B wurden die Aktivitäten der Neurone für alle Disparitäten aufsummiert. Besonders viele Neurone waren dabei für vertikale Disparitäten aktiv. Bei horizontalen Disparitäten und jenen nahe null konnten die Bilder besser sparse codiert werden.

3.4 Diskussion

Ziel war es, beim Training des binokularen Sparsecodingmodells zum einen einen Satz Basisvektoren zu erlernen, der Disparitäten codiert. Zum anderen war der Unterschied zwischen rezeptiven Feldern, die auf einem Stereodatensatz mit virtueller Vergenz gelernt wurden, und jenen aus der Vorgängerstudie von Interesse. Im Falle einer Abweichung sollten sie noch auf ihre biologische Plausibilität geprüft werden.
Wie man in 7.1 sehen kann, hat das binokulare Sparsecodingmodell rezeptive Felder gelernt, die sowohl biologischen als auch jenen aus der Vorgängerstudie ähnlich sehen (Zhang, 2016; Schultz et al., 2014; Lundquist et al., 2016).
Allerdings gibt es noch eine Reihe von rezeptiven Feldern, die verschwommen wirken. Eine mögliche Erklärung für diese ist, dass die Lernzeit bisher noch nicht ausgereicht hat, damit sich die rezeptiven Felder richtig ausdifferenzieren konnten. Die unterste Reihe der Tunningkurven aus Abb. 3.7 stützt diese These. Es scheint so, als sollte der rechte Basisvektor für Disparitäten nahe null getunt sein, nur haben sich seine rezeptiven Felder bisher nicht eindeutig herausbilden können. Die Tuningkurve in der untersten Reihe auf der linken Seite könnte sich durch weiteres Training zu einer inhibitorisch getunten ausdifferenzieren.
Eine alternative Erklärung ist, dass die Anzahl an Basisvektoren nicht ausgereicht hat. Die Profile mancher der rezeptiven Felder könnten deshalb diffus sein, da sich dort mehrere Kernels zu bilden versuchen. Betrachtet man die linke Tuningkurve aus der untersten Reihe von Abb. 3.7, sieht man, dass der Basisvektor für eine Disparität größer null und für eine Disparität kleiner null getunt ist (TN & TF). Bei einer größeren Anzahl an Basisvektoren könnte es gut sein, dass sich zwei verschiedene Basisvektoren bilden, bei dem der eine für eine Disparität kleiner null und der andere für eine größer null getunt ist. Auf selbe Weise ist es möglich, dass die okular-dominanten rezeptiven Felder sich in ein monokulares und ein binokulares rezeptives Feld aufteilen.
Eine weitere Auffälligkeit sind vertikal orientierte Basisvektoren, deren Profil oben und unten abgeschnitten wirkt. Eine mögliche Erklärung hierfür ist, dass sie vertikale Kanten detektieren, die größer sind, als ihr rezeptives Feld es zulässt. Nienborg, Bridge, Parker & Cumming (2004) halten die Größe von rezeptiven Feldern für eine wichtige Eigenschaft für die Detektion von Disparitäten. Entsprechend könnte das Lernen von größeren rezeptiven Feldern zu besseren Ergebnissen führen und diese Auffälligkeit beseitigen.
Wie in der Vorgängerstudie sind auch wieder eine Reihe monokularer Kernels gelernt worden. Diese sind aller Wahrscheinlichkeit nach notwendig für das Codieren von Features, die aufgrund der horizontalen Verschiebung der Kameras nur auf einem der Bilder zu sehen sind.
Ein wichtiges Maß dafür, dass die Codierung von Stereobildern gelernt wurde, findet sich in Abbildung 3.5 A. Die Korrelation der Orientierung von linkem

und rechtem rezeptivem Feldprofil legt nahe, dass Features mit der selben Orientierung im linken und rechten Bild detektiert werden. Die Korrelation der Wellenlänge (Abb. 3.5 C) stärkt dabei die Vermutung, dass es sich um das selbe Feature handelt.
Ringach (2002) hat auch gezeigt, dass Neurone, die für räumliche Frequenzen und Orientierung getunt sind, ungerade Symmetrie besitzen, wie sie auch in 3.6 A zu finden ist.
Wie in 3.1.1 beschrieben sind phasen- und ortsverschobene rezeptive Felder bedeutend für die Detektion von Disparitäten. In Abb. 3.6 B und C sieht man, dass beides vorgefunden wurde. Mit Abb. 3.8 hinzu kann insgesamt davon ausgegangen werden, dass erfolgreich gelernt wurde Disparitäten sparse zu codieren.
Vergleicht man die Abbildungen 3.5 und 3.6 mit jenen aus der Vorgängerstudie, so gibt es hier keine auffälligen Abweichungen. Entsprechend sollten auch die Ergebnisse des Image-Shift-Tests nicht voneinander abweichen.
Anhand von Abb. 3.9 B und Abb. 3.8 sieht man, dass die Repräsentation von Disparitäten nahe der Nulldisparität weniger Basisvektoren benötigt als für extremere Disparitäten. Dieser Befund tauchte auch in der Vorgängerstudie auf und wurde von Zhang (2016) über die Bildstatistik erklärt. Da extreme Disparitäten selten in natürlichen Bildern auftreten, lernt das Netz weniger Basisvektoren, die spezifisch für diese Disparitäten getunt sind. Entsprechend werden für das Codieren dieser großen Disparitäten mehr Basisvektoren benötigt. Da Disparitäten nahe null hingegen häufig vorkommen, ist es wahrscheinlicher, dass es Basisvektoren mit der exakt benötigten Verschiebung gibt. Gleichermaßen legt das Ergebnis nahe, dass horizontale Disparitäten häufiger vorkommen und wichtiger sind als Vertikale.
Vergleicht man diese Ergebnisse mit der Vorgängerstudie, dann ist festzustellen, dass die Eigenschaften der Basisvektoren nicht von jenen aus der Vorgängerstudie abweichen, obwohl Vergenz in den Stereobildern enthalten war. Dies deutet daraufhin, dass die statistische Häufigkeit spezifischer Features in beiden Datensätzen nahezu gleich ist. Allerdings waren die Bildausschnitte auch größer gewählt als in der Vorgängerstudie, wodurch wieder größere Disparitäten enthalten waren.
Da kein Unterschied der Basisvektoren festzustellen war, ist eine Prüfung auf ihre biologische Plausibilität nicht notwendig, da noch die Prüfung aus der Vorgängerstudie Bestand hat.

4 | Zusammenfassung

Das Ziel dieser Arbeit war es zum einen, ein Verfahren zu implementieren, das durch virtuelle Kamerarotation aus herkömmlichen Stereobildern einen Satz von Bildpaaren mit Vergenz erstellt. Die Korrektheit dieses Verfahrens wurde gezeigt und auf einen Satz von tausend selbst erstellten Bildern angewandt. Dabei wurden etwa neuntausend neue Stereobildpaare erstellt.
Anschließend wurde mit dem Datensatz das binokulare Sparse-Coding-Modell trainiert und die Ergebnisse auf Unterschiede zur Vorgängerstudie hin untersucht. Dabei ließen sich keine feststellen.
Da die Basisvektoren bei diesem Ansatz lernen, wenige grundlegende Features zu repräsentieren, sind ihre Eigenschaften stark von den in der Datenbank vorkommenden Features abhängig. Entsprechend geben die erlernten Basisvektoren Auskunft über die statistischen Eigenschaften der Bilder. Sind sehr viele vertikale Disparitäten vorhanden und wichtig für die Rekonstruktion eines Bildes, dann werden sich viele Basisvektoren darauf spezialisieren, diese zu codieren. Das heißt es muss genau darauf geachtet werden, mit welchen Mitteln die Datenbank erstellt wurde und über welche Art von Bildern sie verfügt. Aus diesem Grund wird eine bessere Herangehensweise vorgeschlagen, um spezifisch den Einfluss der Vergenz zu untersuchen. Das Modell sollte zweimal mit den selben Bildausschnitten trainieren, einmal vor und einmal nachdem die Kameras virtuell gedreht wurden. Somit lässt sich quantifizieren, welchen spezifischen Einfluss die Vergenz auf die statistischen Eigenschaften der Bildmenge hat.
Darüber hinaus wurde der Vergleich zwischen den Basisvektoren dieser Studie und der vorhergehenden aus Zeitgründen nur oberflächlich und nicht mit adäquaten statistischen Methoden durchgeführt. Im Endeffekt wurden nur die verschiedenen Verteilungen und Plots auf schematische Ähnlichkeit geprüft. Für die eben vorgeschlagene Untersuchung wäre es sinnvoll, die Auswertung mit statistischen Tests durchzuführen und daraus ein Maß zu entwickeln, das qualifizierte Auskünfte über die Ähnlichkeit von Sätzen an Basisvektoren machen kann.
Zusätzlich gibt es eine große Anzahl an Parametern sowohl im Datensatz als auch im Modell, die aus Zeitgründen nicht variiert wurden. So könnten größere rezeptive Felder, kleinere Bildausschnitte, ein kleinerer Stride, ein Feintuning

der Lernrate und Sparsity sowie ausreichend Zeit zum Lernen noch zu anderen Ergebnissen führen.
Nichtsdestotrotz hat sich gezeigt, dass das Modell auch mit weniger Basisvektoren und auf einem anderen Datensatz in der Lage ist, biologisch plausible rezeptive Felder zu erlernen.
Unabhängig davon können mittels des hier vorgestellten Algorithmus erstellte Datensätze noch für eine Reihe weiterer Untersuchungen genutzt werden. So ist der Vorteil dieses Verfahrens, dass es Bilder erzeugt, die wie beim Vergieren blickzentriert wurden. Eine solche Datenbank kann somit zur Untersuchung von zielgerichtetem Handeln und Aufmerksamkeit dienen (Canessa et al., 2017).
Eine mögliche Herangehensweise, um Vergenz für das binokulare Sparse-Coding-Modell notwendig zu machen, wäre die Konstruktion eines Roboters mit Kameras, die sich unabhängig voneinander bewegen können. Der visuelle Input des Roboters müsste dann mit dem binokularen Sparsecodingmodell so codiert werden, dass die Retinalschicht über eine unterschiedlich gute Auflösung, ähnlich der Retina, verfügt. Der codierte, visuelle Input sollte für eine weitere Aufgabe notwendig sein, deren Erfüllung bewertet werden kann. Hierdurch sollten künstliche Neurone entstehen, die mit Vergenz arbeiten und die Vergenzbewegungen der Kamera kontrollieren können.
Allerdings ist Erkelens (2001) darauf gestoßen, dass die Verarbeitung von Disparität, die Vergenzbewegungen steuert, an einem anderen Ort stattfindet als die Gewinnung von Tiefeninformation aus Disparität.
Wenn dies der Fall ist, sollten die Neurone der V1-Schicht des binokularen Sparse-Coding-Modells unabhängig von Vergenz gelernt werden und eine weitere Schicht für die Kontrolle der Vergenz notwendig machen. Durch einen solchen Versuch könnte der Befund von Erkelens gestärkt werden.
Auf jeden Fall sind das Modell und der Datensatz in der Lage zukünftig tiefere Einblicke in die Funktionsweise des Gehirns zu liefern.

5 | Quellen

Anzai, A., Ohzawa, I., & Freeman, R. D. (1999). Neural mechanisms for encoding binocular disparity: receptive field position versus phase. *Journal of Neurophysiology*, 82(2), 874-890.

Barlow, H. B., Blakemore, C., & Pettigrew, J. D. (1967). The neural mechanism of binocular depth discrimination. *The Journal of physiology*, 193(2), 327-342.

Baya, H., Essa, A., Tuytelaarsb, T., & Van Gool, L. (2008). Speeded-up robust features (SURF). *Computer vision and image understanding*, 110(3), 346-359.

Bear, Mark F., Barry W. Connors, and Michael A. Paradiso. Neurowissenschaften."*Ein grundlegendes Lehrbuch für Biologie, Medizin und Psychologie* 3(2009).

Canessa, A., Gibaldi, A., Chessa, M., Fato, M., Solari, F., & Sabatini, S. P. (2017). A dataset of stereoscopic images and ground-truth disparity mimicking human fixations in peripersonal space. *Scientific Data*, 4.

Cole, I. R. (2015) Modelling CPV (Doctoral Dissertation). Retrieved from https://dspace.lboro.ac.uk/2134/18050 , Loughborough University Institutional Repository

Erkelens, C. J. (2001). Organisation of signals involved in binocular perception and vergence control. *Vision research*, 41(25), 3497-3503.

Field, D. J. (1994). What is the goal of sensory coding?. *Neural computation*, 6(4), 559-601.

Fleet, D. J., Wagner, H., & Heeger, D. J. (1996). Neural encoding of binocular disparity: energy models, position shifts and phase shifts. *Vision research*, 36(12), 1839-1857.

Hartley, R., & Zisserman, A. (2003). *Multiple view geometry in computer vision*. Cambridge university press.

Heikkila, J., & Silven, O. (1997, June). A four-step camera calibration procedure with implicit image correction. In Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 *IEEE Computer Society Conference on* (pp. 1106-1112). IEEE.

Howarth, P. A. (2011). The geometric horopter. *Vision research*, 51(4), 397-399.

Hubel, D. H., & Wiesel, T. N. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160(1), 106-154.

Jones, J. P., & Palmer, L. A. (1987). An evaluation of the two-dimensional Gabor filter model of simple receptive fields in cat striate cortex. *Journal of neurophysiology*, 58(6), 1233-1258.

Lundquist, S. Y., Paiton, D. M., Schultz, P. F., & Kenyon, G. T. (2016, March). Sparse encoding of binocular images for depth inference. In Image Analysis and Interpretation (SSIAI), 2016 *IEEE Southwest Symposium on* (pp. 121-124). IEEE.

Luong, Q. T., & Faugeras, O. D. (1996). The fundamental matrix: Theory, algorithms, and stability analysis. International journal of computer vision, 17(1), 43-75.

Maffei, L., & Fiorentini, A. (1973). The visual cortex as a spatial frequency analyser. *Vision research*, 13(7), 1255-1267.

Malis, E., & Vargas, M. (2007). Deeper understanding of the homography decomposition for vision-based control (Doctoral dissertation, INRIA).

Nienborg, H., Bridge, H., Parker, A. J., & Cumming, B. G. (2004). Receptive field size in V1 neurons limits acuity for perceiving disparity modulation. *Journal of Neuroscience*, 24(9), 2065-2076.

Olshausen, B. A., & Field, D. J. (1996). Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583), 607.

Poggio, G. F., Gonzalez, F., & Krause, F. (1988). Stereoscopic mechanisms in monkey visual cortex: binocular correlation and disparity selectivity. *Journal of Neuroscience*, 8(12), 4531-4550.

Prince, S. J. D., Pointon, A. D., Cumming, B. G., & Parker, A. J. (2002). Quantitative analysis of the responses of V1 neurons to horizontal disparity in dynamic random-dot stereograms. *Journal of Neurophysiology*, 87(1), 191-208.

Ringach, D. L. (2002). Spatial structure and symmetry of simple-cell receptive fields in macaque primary visual cortex. *Journal of neurophysiology*, 88(1), 455-463.

Rozell, C. J., Johnson, D. H., Baraniuk, R. G., & Olshausen, B. A. (2008). Sparse coding via thresholding and local competition in neural circuits. *Neural computation*, 20(10), 2526-2563.

Schultz, P. F., Paiton, D. M., Lu, W., & Kenyon, G. T. (2014). Replicating kernels with a short stride allows sparse reconstructions with fewer independent kernels. *arXiv preprint arXiv:1406.4205*.

Siderov, J., Harwerth, R. S., & Bedell, H. E. (1999). Stereopsis, cyclovergence and the backwards tilt of the vertical horopter. *Vision research*, 39(7), 1347-1357.

Tsai, R. (1987). A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses. *IEEE Journal on Robotics and Automation*, 3(4), 323-344.

Tsao, D. Y., Conway, B. R., & Livingstone, M. S. (2003). Receptive fields of disparity-tuned simple cells in macaque V1. *Neuron*, 38(1), 103-114.

Stahl, J. S. (1999). Amplitude of human head movements associated with horizontal saccades. *Experimental brain research*, 126(1), 41-54.

Verhoef, B. E., Vogels, R., & Janssen, P. (2016). Binocular depth processing in the ventral visual pathway. *Phil. Trans. R. Soc. B*, 371(1697), 20150259.

Vienne, C., Plantier, J., Neveu, P., & Priot, A. E. (2016). The Role of Vertical Disparity in Distance and Depth Perception as Revealed by Different Stereo-Camera Configurations. *i-Perception*, 7(6), 2041669516681308.

Wong, A. M. (2004). Listing's law: clinical significance and implications for neural control. *Survey of ophthalmology*, 49(6), 563-575.

Zhang, Y. (2016). Quantitative analysis of the responses of a sparse coding model of binocular simple cells in V1 to binocular disparity in natural images. (Labrotation Report; not published).

Zhang, Z. (2000). A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence*, 22(11), 1330-1334.

Zeiler, M. D., Krishnan, D., Taylor, G. W., & Fergus, R. (2010, June). Deconvolutional networks. In Computer Vision and Pattern Recognition (CVPR), 2010 *IEEE Conference on* (pp. 2528-2535). IEEE.

6 | Appendix

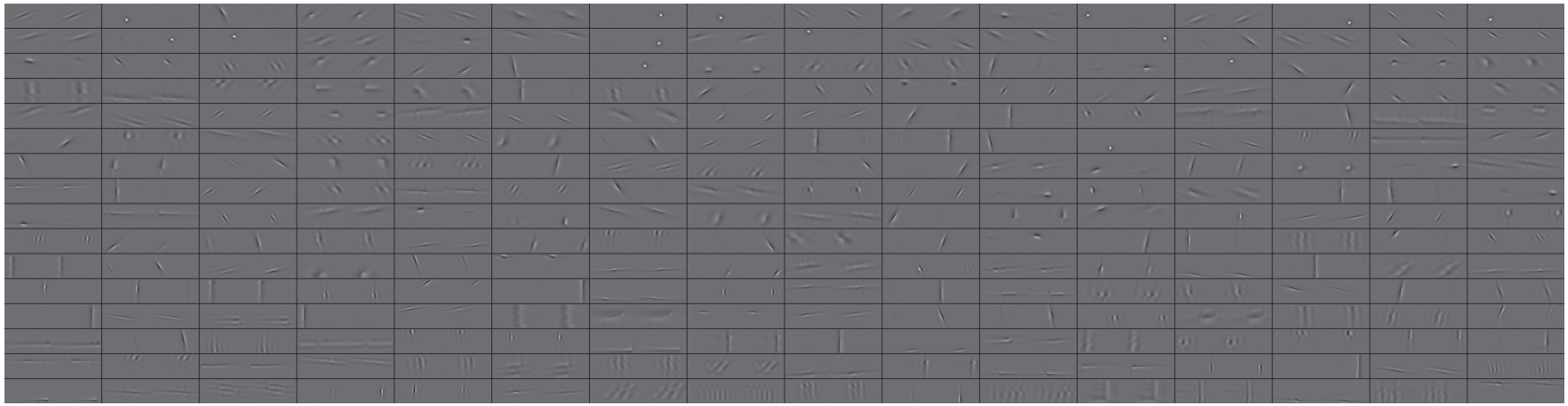

Abbildung 6.1: Darstellung aller gelernten Basisvektoren

Hier sind alle 256 rezeptiven Felder zu sehen, die vom binokularen Sparse-Coding Modell gelernt wurden. Sie sind nach der Häufigkeit ihrer Aktivität in absteigender Reihenfolge sortiert. In der untersten Reihe sieht man einige untypische Profile von rezeptiven Feldern. Diese könnten aufgrund von zu wenig Basisvektoren, zu kurzer Lernzeit oder nicht optimal eingestellten Parametern gelernt worden sein.

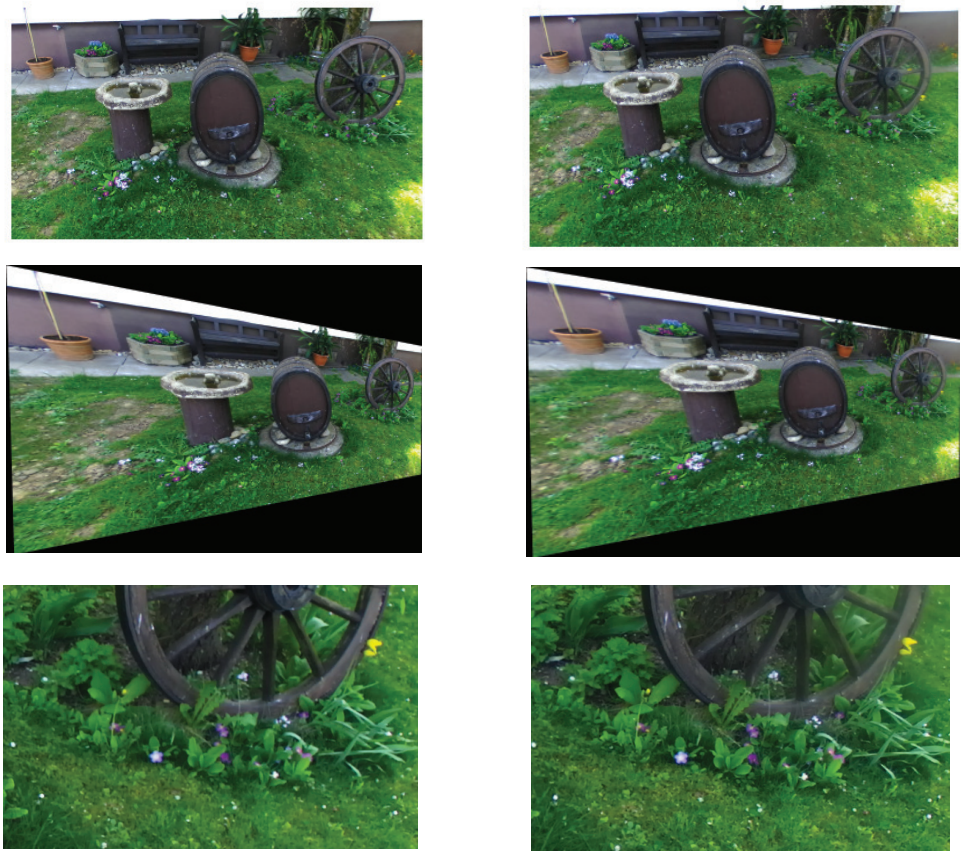

Abbildung 6.2: Beispielhafte Bildtransformation

In der obersten Reihe sieht man die Bilder aus denen Verzeichnung herausgerechnet wurde. In der zweiten wurden die Kameras rotiert. Im untersten sind die entstandenen Bildausschnitte daraus zu sehen.

Selbständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Bachelorarbeit selbständig und nur mit den angegebenen Hilfsmitteln angefertigt habe und dass alle Stellen, die dem Wortlaut oder dem Sinne nach anderen Werken entnommen sind, durch Angaben von Quellen als Entlehnung kenntlich gemacht worden sind. Diese Bachelorarbeit wurde in gleicher oder ähnlicher Form in keinem anderen Studiengang als Prüfungsleistung vorgelegt.

Ort, Datum Unterschrift