INSTITUT FÜR INFORMATIK

UND PRAKTISCHE MATHEMATIK

Struktureller Bias in neuronalen Netzen mittels Clifford-Algebren

Vladimir Banarer

Bericht Nr. 0501
Januar 2005

CHRISTIAN-ALBRECHTS-UNIVERSITÄT

KIEL

Institut für Informatik und Praktische Mathematik der
Christian-Albrechts-Universität zu Kiel
Olshausenstr. 40
D – 24098 Kiel

Struktureller Bias in neuronalen Netzen mittels Clifford-Algebren

Vladimir Banarer

Bericht Nr. 0501
Januar 2005

e-mail: vlb@ks.informatik.uni-kiel.de

Dieser Bericht gibt den Inhalt der Dissertation wieder, die der Verfasser im September 2004 bei der Technischen Fakultät der Christian-Albrechts-Universität zu Kiel eingereicht hat.
Datum der Disputation: 16. Dezember 2004.

1. Gutachter	Prof. G. Sommer (Kiel)
2. Gutachter	Prof. K. Jansen (Kiel)

Datum der mündlichen Prüfung: 16.12.2004

ZUSAMMENFASSUNG

Im Rahmen dieser Arbeit wird ein generisches Approximierungsmodell aufgestellt, das unter anderem klassische neuronale Architekturen umfaßt. Die allgemeine Rolle von a priori Wissen bei der Modellierung wird untersucht. Speziell werden Clifford-Algebren bei dem Entwurf von neuronalen Netzen als Träger struktureller Information eingesetzt. Diese Wahl wird durch die Eigenschaft von Clifford-Algebren motiviert, geometrische Entitäten sowie deren Transformationen auf eine effiziente Art darstellen bzw. berechnen zu können. Neue neuronale Architekturen, die im Vergleich zu klassischen Ansätzen höhere Effizienz aufweisen, werden entwickelt und zur Lösung von verschiedenen Aufgaben in Bildverarbeitung, Robotik und Neuroinformatik allgemein eingesetzt.

ABSTRACT

In this thesis a generic approximation model is developed, which also includes classical neural architectures. The role of a priori knowledge in such models is analyzed. In particular, Clifford algebras are used to represent structural information in neural networks. This choice is motivated by the fact that Clifford algebras can be used to efficiently represent and evaluate geometric entities, as well as certain geometric transformations. Novel neural architectures are developed and are shown to be more efficient than classical approaches. The applicability of these architectures is demonstrated on a number of different problems in image processing, robotics and neuro science.

DANKSAGUNG

Die Fertigstellung der Dissertation in der hier vorliegenden Form wäre ohne die Hilfe vieler Menschen nicht möglich gewesen. An dieser Stelle möchte ich die Gelegenheit ergreifen, mich bei allen zu bedanken.

An erster Stelle möchte ich den Dank meinem Doktorvater Prof. Dr. Gerald Sommer, dem Leiter des Lehrstuhls für Kognitive Systeme des Instituts für Informatik und Praktische Mathematik an der Christian-Albrechts-Universität zu Kiel, aussprechen. Ohne sein grosses Vertrauen, zahlreiche Diskussionen und fachliche Ratschläge hätte diese Arbeit nicht entstehen können.

Weiterhin gilt mein Dank allen Mitarbeitern des Lehrstuhls Kognitive Systeme. Insbesondere möchte ich mich bei Christian Perwass für viele fruchtbare Diskussionen, fachliche Zusammenarbeit und vor allem Freundschaft, die mir in schwierigen Zeiten eine grosse Hilfe war, bedanken. Meine Kollegen und Freunde Sven Buchholz, Oliver Granert, Martin Krause, Yohannes Kassahun, Bodo Rosenhahn und Nils Siebel trugen durch ihre moralische und fachliche Unterstützung einen großen Teil zur Entstehung dieser Dissertation bei. Ich bin auch den technischen Mitarbeitern des Lehrstuhls Henrik Schmidt und Gerd Diesner sowie unserer Sekretärin Françoise Maillard für ihre Hilfe dankbar.

Ich danke Sven Buchholz, Oliver Granert, Kristin Kowalk, Christian Perwass, Eike Staege und insbesondere Christian Staege für Probelesen und Korrekturvorschläge.

Ein weiterer Dank gilt dem DFG-Graduiertenkolleg Nr. 357 für die finanzielle Unterstützung.

Schließlich möchte ich mich bei meinen Eltern für ihr Vertrauen, ihre Unterstützung, die sie mir über die ganze Zeit auf jede erdenkliche Weise gaben, und ihre Geduld bedanken.

INHALTSVERZEICHNIS

Kapitel 1

EINLEITUNG

1.1 Motivation

Neuroinformatik stellt einen wichtigen Aspekt in der Forschung über kognitive Systeme dar. Von der biologischen Motivation ausgehend entstand auf diesem Gebiet eine inzwischen dominierende Richtung, in welcher durch eine weitgehende Abstraktion vom natürlichen Vorbild neuronale Modelle entwickelt werden, die besonders gut zur Lösung bestimmter Probleme geeignet sind. Als Funktionsapproximatoren für hochdimensionale Regressions- oder Klassifizierungsaufgaben finden sie breiten Einsatz in vielen Forschungsgebieten. Vor allem dann, wenn keine explizite oder analytische Beschreibung der gestellten Aufgabe möglich ist, lassen sich neuronal basierte Modelle besonders vorteilhaft einsetzen. Das Spektrum der möglichen Anwendungen erstreckt sich dabei von Bildverarbeitung und Robotik über Medizin bis hin zum Finanzwesen.

Der Grundgedanke eines neuronalen Ansatzes besteht darin, durch eine Vernetzung einer Vielzahl einfacher Berechnungseinheiten – der Neuronen – ein Modell zu entwickeln, das in der Lage ist, sich an eine möglichst grosse Anzahl von Gegebenheiten durch Beobachtungen flexibel anzupassen, ohne an der Güte der Approximation einzubüßen. Der Entwurf und das Training eines Netzes kann als ein Modell- und Parameteroptimierungsproblem angesehen werden. Die Qualität einer solchen Modellierung wird dabei durch die Generalisierungsfähigkeit, das Skalierungsverhalten in höherdimensionalen Räumen und die Robustheit gegenüber fehlerbehafteten Daten charakterisiert: Ein gut[1] entworfenes künstliches neuronales Netz soll diese Eigenschaften in hohem Maße aufweisen. Solche Erwartungen stellen allerdings schwer zu realisierende und teilweise nicht konsistente Ansprüche an das Modell. Insbesondere bei komplexen Problemstellungen

[1] im Sinne der gegebenen Schätzungskriterien.

bzw. hohen Anforderungen an die Universalität [2] eines Netzes steigt die Modellordnung (Anzahl von Neuronen, Verbindungsstruktur) und der damit verbundene Trainingsaufwand erheblich. Gleichzeitig verschlechtert sich die Generalisierungsfähigkeit.

Die Methodik zur Vermeidung solcher Probleme besteht darin, eine Anpassung der Modellordnung an die zu lernende Funktion durchzuführen: Das gesamte vorhandene a priori Wissen soll in das Netz einfließen. Ein sinnvoller Modellentwurf sollte sich nicht nur auf die bei der Beobachtung der momentanen Zuständen gesammelten Informationen beschränken, sondern auch weitere strukturelle Verhaltensaspekte des zu modellierenden Systems berücksichtigen.

Die allgemeinen Fragen

- wie kann ein Modell zur Approximation generisch beschrieben werden,
- wie beeinflussen strukturelle Änderung an verschiedenen Stellen das Verhalten des generischen Modells,
- welche Informationen können als a priori Wissen verwendet werden,
- auf welche Weise lässt sich a priori Wissen in einen neuronalen Modellentwurf integrieren,

stellen den Ausgangspunkt dieser Arbeit dar.

Dabei wird die Eignung von Clifford-Algebren zur Modellierung von Aufgaben in der Bildverarbeitung und Robotik untersucht. Die Wahl von Clifford-Algebren als Träger struktureller Informationen zur Repräsentation von a priori Wissen ist hier durch die Fähigkeit, Entitäten höherer Ordnung als Elemente der Algebra darzustellen, motiviert. Komplexe Zusammenhänge lassen sich auf diese Weise effizienter beschreiben.

1.2 Gliederung der Arbeit

Diese Dissertationsschrift beginnt nach der Einleitung mit der Beschreibung eines allgemeinen Konzeptes zur Modellierung von Approximierungsfunktionen in Kapitel 2. Es werden klassische Probleme wie ”curse of dimensionality” und Bias-Varianz-Dilemma diskutiert und geeignete Lösungsansätze vorgeschlagen. Weiterhin werden globale und lokale Modelle sowohl für den kontinuierlichen als

[2] Unter dem Begriff ”Universalität” wird die stark ausgeprägte Fähigkeit des Netzes, sich an die komplett unterschiedlichen Funktionen zu adaptieren, verstanden.

auch für den diskreten Fall vorgestellt. Anschließend wird eine funktionale und geometrische Interpretation für diese gegeben. Der Einfluss der Parametrisierung auf die Struktur des Modells wird detailliert beschrieben. Das Kapitel schließt mit einer Zusammenfassung über allgemeine Möglichkeiten der Modellierung mit Hilfe von a priori Wissen.

Kapitel 3 gibt einen kompakten Überblick über Clifford-Algebren. Besonderer Wert wird auf das Verständnis der geometrischen Grundlagen und der Rechenregeln gelegt. Die Ausführung beschränkt sich auf eine konsistente Erklärung von Konzepten, die im Rahmen dieser Arbeit relevant sind. Konkret liegt der Schwerpunkt auf der effizienten Beschreibung von Rotationen, starren Bewegungen und Hypersphären.

Im 4. Kapitel werden häufig verwendete neuronale Architekturen als Realisierungen des in Kapitel 2 eingeführten allgemeinen Modells zur Approximation besprochen. Das Kapitel beginnt mit der Einführung des generischen Neurons, diskutiert die Eigenschaften von globalen Netzen (MLP) und schließt mit der Beschreibung von neuronalen Modellen mit lokalen rezeptiven Feldern (RBF und DCS). Zu jeder der vorgestellten Architekturen wird eine funktionale und geometrische Interpretation gegeben.

Die Eigenschaft einer Clifford-Algebra, Euklidische Hypersphären in einem höherdimensionalen Raum auf eine lineare Weise beschreiben zu können, wird in Kapitel 5 erstmalig ausgenutzt, um ein klassisches, neuronales Modell zu erweitern. Dabei wird demonstriert, auf welche Weise eine spezielle Einbettung von Daten die MLP-Architektur um die Funktionalität bereichert, ursprünglich nichtlineare kugelförmige Trennungsflächen effizient zu berechnen. Die Leistungsfähigkeit des so verbesserten Modells als Klassifikator wird sowohl anhand einiger künstlich generierter Daten als auch anhand echter Messungen getestet.

Kapitel 6 behandelt das aus dem Bereich der Bildverarbeitung stammende Problem der Segmentierung eines optischen Flusses nach projizierten Bewegungen mehrerer 3D-Objekte. Unter der Annahme, dass nur starre Objekte den Fluss induzieren, werden mehrere Hypothesen bezüglich der Art von projizierten Bewegungen aufgestellt. Für diese werden dann neue neuronale Algorithmen entworfen. Die Möglichkeit, in einer speziellen Clifford-Algebra allgemeine Rotationen effizient berechnen zu können, erlaubt dabei, ein auf der DCS-Architektur basierendes Modell zu entwerfen, das im Vergleich mit anderen untersuchten Modellen eine bessere Leistung aufweist. Experimente mit künstlichen und echten Bildfolgen schließen das Kapitel ab.

Ein Algorithmus zur Steuerung der Bewegungen eines Roboterarmes ist in Kapitel 7 vorgestellt. Dem neuen Algorithmus liegt ein DCS-Netz zugrunde. Für das

effiziente Training wird eine Metrik entwickelt, die auf natürliche Weise die Topologie des Konfigurationsraumes des Armes berücksichtigt. Die Repräsentation von Steuerungsdaten als Rotoren einer Clifford-Algebra vereinfacht durch eine geometrische Interpretation den Entwurf der geeigneten Metrik. Gleichzeitig verringert sie die Berechnungskomplexität.

Jedes Kapitel schließt mit einer Zusammenfassung der wichtigsten Ergebnisse. Aus dem Text wird ersichtlich, welche Definitionen, Modellentwürfe und Algorithmen die Originalbeiträge dieser Arbeit[3] sind und welche aus der Literatur stammen.

Die Dissertationsschrift endet im 8. Kapitel mit einer Diskussion.

[3] Mit dem Begriff ”diese Arbeit” sind die vorliegende Dissertationsschrift sowie die Publikationen [4, 5, 80] gemeint.

Kapitel 2

ALLGEMEINE ASPEKTE DER MODELLIERUNG

Viele Abläufe in der Natur lassen sich über so genannte Prozesse als mathematische Modelle beschreiben. Diese Modelle können ganz allgemein als Abbildungen zwischen verschiedenen Räumen interpretiert werden. Sie werden als Funktionen formalisiert.

Eine der wichtigsten Aufgaben der Neuroinformatik besteht darin, adäquate mathematische Modelle zu finden und diese durch Beobachtungen möglichst gut an die betrachtete Umgebung anzupassen.

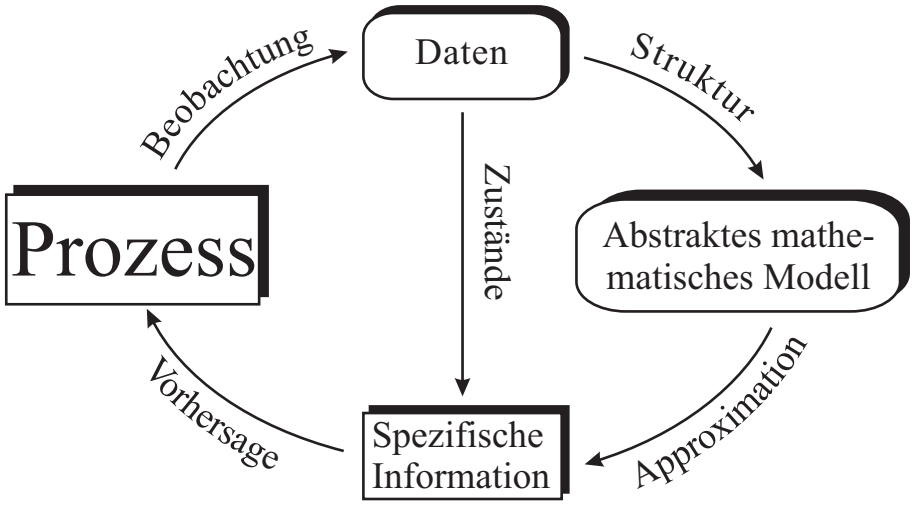


Abbildung 2.1: Durch die Beobachtung von Struktur und Zuständen eines Prozesses versucht man, ein plausibles mathematisches Modell aufzustellen und eine gute Approximation zu erreichen.

2.1 Modellierung

Ein gut an die gestellte Aufgabe angepasstes Modell explizit aufzustellen, stellt oft ein schwieriges Problem dar. Daher werden in solchen Fällen Annahmen über die Art der zu approximierenden Funktionen gemacht und durch Beobachtungen Informationen gesammelt. Diese werden genutzt, um die Modellierung möglichst plausibel zu gestalten. Die Beobachtungen können in Form von Zuständen (”Schnappschüsse”, Beispieldaten/Trainingsdaten) und Dynamik bzw. Struktur (Gesetzmäßigkeiten, Verhaltensweise, Verlauf, Funktionenklasse) vorliegen.

Sind lediglich Zustände beobachtbar, ist die gesuchte Abbildung als eine (diskrete) Abtastung des gesamten Definitionsbereiches modellierbar. Eng verwandt damit ist das so genannte ”curse of dimensionality” Problem [10]. Dabei handelt es sich um die Tatsache, dass sich das Hypervolumen des Eingaberaumes als eine Funktion der Dimension exponentiell verhält: Das Erhöhen der Dimension des Eingaberaumes führt dazu, dass man eine exponentiell wachsende Anzahl von Zuständen braucht, um die Abtastung des gesamten Raumes mit gleicher Dichte zu realisieren. Für hochdimensionale Räume werden also potentiell unendlich viele Zustände für eine gute Approximation gebraucht.

Ist nur die Struktur bekannt, so kann die gesuchte Funktion nur qualitativ aber nicht quantitativ ermittelt werden.

Eine gute Approximation erfordert sowohl die Kenntnis der Zustände als auch der Dynamik bzw. Struktur im ausreichenden Maß. Dabei werden im Idealfall gerade so viel Zustände, wie zu der Ausschöpfung aller Freiheitsgrade der gewählten Funktion[1] notwendig sind, gebraucht. Da aber in der Praxis gemessene Daten nur in einer verrauschten Form vorliegen, und die Prozessdynamik meistens sich nur schwer beobachten lässt, versucht man, die Approximation durch das Erhöhen der Anzahl der Beobachtungen und/oder durch die Verwendung einer geeigneteren Funktionenklasse zu stabilisieren. Die Suche nach solch einer Funktionenklasse stellt eine sehr schwierige Aufgabe dar. Allerdings kann der Suchraum unter der Berücksichtigung des a priori Wissens über die zu modellierende Funktion erheblich eingeschränkt werden.

2.2 Globale Approximation

Das Problem der Approximation kann folgendermaßen formalisiert werden:

[1] Diese Funktion wird anhand des a priori Wissens über die Prozessdynamik gewählt.

Sei $\mathbf{C}$ eine Abbildung vom Definitionsbereich $\mathbb{D} \subseteq \mathbb{R}^n$ zum Wertebereich $\mathbb{W} \subseteq \mathbb{R}^m$:

$$\begin{aligned} \mathbf{C} : \mathbb{D} &\longrightarrow \mathbb{W} \\ \mathbf{x} &\longmapsto \mathbf{y} \end{aligned} \tag{2.1}$$

mit $\mathbf{x} \in \mathbb{D}$ und $\mathbf{y} \in \mathbb{W}$.

Weiterhin sei $\mathbf{V}$ eine Funktion von $\mathbb{R}^n$ nach $\mathbb{R}^m$ aus einer gewählten Klasse der Funktionen[2] $\mathcal{V}^k$, die durch k Parameter beschrieben wird:

$$\begin{aligned} \mathbf{V} : \mathbb{R}^n \times \mathbb{R}^k &\longrightarrow \mathbb{R}^m \\ (\mathbf{x}, \mathbf{p}) &\longmapsto \mathbf{y}. \end{aligned} \tag{2.2}$$

Dabei sei $\mathbf{x} = (x_1 \dots x_n) \in \mathbb{R}^n$ ein Eingabevektor, $\mathbf{p} = (p_1 \dots p_k) \in \mathbb{R}^k$ der Parametervektor der Funktion $\mathbf{V}$ und $\mathbf{y} = (y_1 \dots y_m) \in \mathbb{R}^m$ der Ausgabevektor.

Die Güte der Approximation der Abbildung $\mathbf{C}$ durch die Funktion $\mathbf{V}$ kann durch ein quadratisches Fehlerfunktional F ausgedrückt werden:

$$F = \int_{\mathbb{D}} \left(\mathbf{V}(\mathbf{x}, \mathbf{p}) - \mathbf{C}(\mathbf{x})\right)^2 \mathrm{d}\mathbf{x}. \tag{2.3}$$

Für den diskreten Fall, das heißt wenn die Abbildung $\mathbf{C}$ durch eine endliche Anzahl N von Zuständen $\{(\mathbf{x}_i, \mathbf{y}_i = \mathbf{C}(\mathbf{x}_i))\}_{i=1\dots N}$ gegeben ist, ergibt sich der quadratische Fehler zu:

$$\begin{aligned} F &= \int_{\mathbb{D}} \sum_{i=1}^{N} \left(\mathbf{V}(\mathbf{x}, \mathbf{p}) - \mathbf{C}(\mathbf{x})\right)^2 \delta(\mathbf{x} - \mathbf{x}_i)\mathrm{d}\mathbf{x} \\ &= \sum_{i=1}^{N} \int_{\mathbb{D}} \left(\mathbf{V}(\mathbf{x}, \mathbf{p}) - \mathbf{C}(\mathbf{x})\right)^2 \delta(\mathbf{x} - \mathbf{x}_i)\mathrm{d}\mathbf{x} \\ &= \sum_{i=1}^{N} \left(\mathbf{V}(\mathbf{x}_i, \mathbf{p}) - \mathbf{C}(\mathbf{x}_i)\right)^2 \\ &= \sum_{i=1}^{N} \left(\mathbf{V}(\mathbf{x}_i, \mathbf{p}) - \mathbf{y}_i\right)^2. \end{aligned} \tag{2.4}$$

[2] Unter einer Funktionenklasse $\mathcal{V}^k$ wird die Menge aller Funktionen mit k Parametern, die eine gemeinsame Verhaltensweise aufweisen, verstanden. Zum Beispiel bildet die Menge aller Polynome von Grad $k-1$ eine Funktionenklasse $\mathcal{V}^k_{pol} = \left\{\sum_{i=0}^{k-1} a_i x^i\right\}$ mit k Parametern. Jede Funktion $\mathbf{V} \in \mathcal{V}^k_{pol}$ lässt sich als $\mathbf{V}(x, \mathbf{p}) = \sum_{i=0}^{k-1} p_i x^i$ aufschreiben.

Die Approximation einer Funktion kann als eine Minimierung der Fehlerfunktion F bezüglich der Parameter $\mathbf{p}$ der Funktion $\mathbf{V}$ formuliert werden:

$$\min_{\mathbf{p}} \int_{\mathbb{D}} \left(\mathbf{V}(\mathbf{x},\mathbf{p}) - \mathbf{C}(\mathbf{x})\right)^2 \mathrm{d}\mathbf{x}, \tag{2.5}$$

oder für den diskreten Fall

$$\min_{\mathbf{p}} \sum_{i=1}^{N} \left(\mathbf{V}(\mathbf{x}_i,\mathbf{p}) - \mathbf{y}_i\right)^2. \tag{2.6}$$

2.3 Bias-Varianz-Dilemma

In der Regel sind beobachtete Daten verrauscht, so dass anstatt von Paaren $(\mathbf{x}, \mathbf{C}(\mathbf{x}))$ die Muster $(\mathbf{x}, \mathbf{C}(\mathbf{x}) + \xi)$ mit dem Rauschen ξ vorliegen. Der Erwartungswert E des quadratischen Fehlers F wird berechnet und aufgeteilt in Terme, die sich auf das Rauschen ξ beziehen:

$$\begin{aligned} &E\left[\left(\mathbf{V}(\mathbf{x},\mathbf{p}) - (\mathbf{C}(\mathbf{x}) + \xi)\right)^2\right] = \\ &\underbrace{\left(E\left[(\mathbf{C}(\mathbf{x}) + \xi)\right] - \mathbf{V}(\mathbf{x},\mathbf{p})\right)^2}_{\text{Systematischer Fehler}} + \underbrace{E\left[\left((\mathbf{C}(\mathbf{x}) + \xi) - E\left[\mathbf{C}(\mathbf{x}+\xi)\right]\right)^2\right]}_{\text{Unsystematischer Fehler}}. \end{aligned} \tag{2.7}$$

Der systematische Fehler ist ein Maß für die Güte einer gewählten Approximierungsfunktion und steht in einer direkten Verbindung mit der beobachteten Dynamik. Durch die sinnvolle Wahl der Funktionenklasse bei der Modellierung sowie die geeignete Minimierung bezüglich der Parameter der Approximierungsfunktion lässt sich dieser Fehler verkleinern. Der unsystematische Fehler liegt an Meßungenauigkeiten bei der Beobachtung der Zustände und kann üblicherweise nicht komplett eliminiert werden.

Im Bereich der Neuroinformatik ist eine Umformung der Gleichung (2.7) als so genanntes Bias-Varianz-Dilemma bekannt [36]. Man teilt dabei den Erwartungswert E_Z des quadratischen Fehlers F über der Menge der Trainingsdaten $Z = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1\ldots N}$ in Bias- und Varianz-Terme auf, die sich auf Erwartungswerte von $\mathbf{V}(\mathbf{x},\mathbf{p})$ beziehen:

$$\begin{aligned} &E_Z\left[\left(\mathbf{V}(\mathbf{x},\mathbf{p}) - \mathbf{y}\right)^2\right] = \\ &\underbrace{\left(E_Z\left[\mathbf{V}(\mathbf{x},\mathbf{p})\right] - \mathbf{y}\right)^2}_{\text{Bias}} + \underbrace{E_Z\left[\left(\mathbf{V}(\mathbf{x},\mathbf{p}) - E_Z\left[\mathbf{V}(\mathbf{x},\mathbf{p})\right]\right)^2\right]}_{\text{Varianz}}. \end{aligned} \tag{2.8}$$

Die Wahl eines Modells von genügend hoher Ordnung erlaubt eine optimale Approximation von Daten auf der gegebenen Menge Z. Die Generalisierungsfähigkeit[3] ist dabei nicht notwendigerweise zufriedenstellend, da die Approximierungsfunktion den verrauschten Daten folgt und nicht die gesuchte Dynamik auf dem gesamten Raum wiedergeben kann. Der Bias-Term wird in diesem Fall zu Null tendieren, wogegen der Varianz-Term gross wird. Falls die Funktionenklasse ungünstig gewählt worden ist, kann weder Dynamik noch die Minimierung des quadratischen Fehlers auf den bekannten Daten gewährleistet werden.

2.4 Lokale Approximation

Das Ziel einer Modellierung besteht darin, durch Beobachtung von Dynamik und Zuständen eine passende Balance zwischen Bias und Varianz zu finden.

2.4.1 Generische Modelle

Falls die Abbildung $\mathbf{C}$ ein schwer vorhersehbares Verhalten aufweist, so werden im Allgemeinen Funktionenklassen mit hoher Komplexität, beziehungsweise mit einer hohen Anzahl von Parametern k für eine Approximation gebraucht. Die Minimierung kann dadurch numerisch instabil werden. Um dem entgegenzuwirken, können anstatt einer Funktion $\mathbf{V} \in \mathcal{V}^k$ eine Menge von einfacheren (Basis-)Funktionen $\{\mathbf{V}_j \in \mathcal{V}^{k_j}\}_{j=1\ldots K}$ eingesetzt werden. Die Approximierungsfunktion $\mathbf{V}$ kann dann als Überlappung von Basisfunktionen $\mathbf{V}_j$ mit den dazu gehörigen Wichtungsfunktionen $\mathbf{L}_j$ beschrieben werden:

$$\mathbf{V}(\mathbf{x}) = \sum_{j=1}^{K} \mathbf{L}_j(\mathbf{x}, \mathbf{q}_j)\mathbf{V}_j(\mathbf{x}, \mathbf{p}_j) \tag{2.9}$$

mit

$$\begin{aligned} \mathbf{V}_j : \mathbb{R}^n \times \mathbb{R}^{|\mathbf{p}_j|} &\longrightarrow \mathbb{R}^m & \qquad \mathbf{L}_j : \mathbb{R}^n \times \mathbb{R}^{|\mathbf{q}_j|} &\longrightarrow \mathbb{R} \\ (\mathbf{x}, \mathbf{p}_j) &\longmapsto \mathbf{y} & (\mathbf{x}, \mathbf{q}_j) &\longmapsto a, \end{aligned}$$

wobei $\mathbf{q}_j = (q_{j1} \ldots q_{jl}) \in \mathbb{R}^{l=|\mathbf{q}_j|}$ der Parametervektor der Wichtungsfunktion $\mathbf{L}_j$ ist. Dies wird in Abbildung 2.2 beispielhaft visualisiert.

[3] Unter dem Begriff "Generalisierungsfähigkeit" werden die Interpolation- und Extrapolationeigensaften des Modells verstanden.

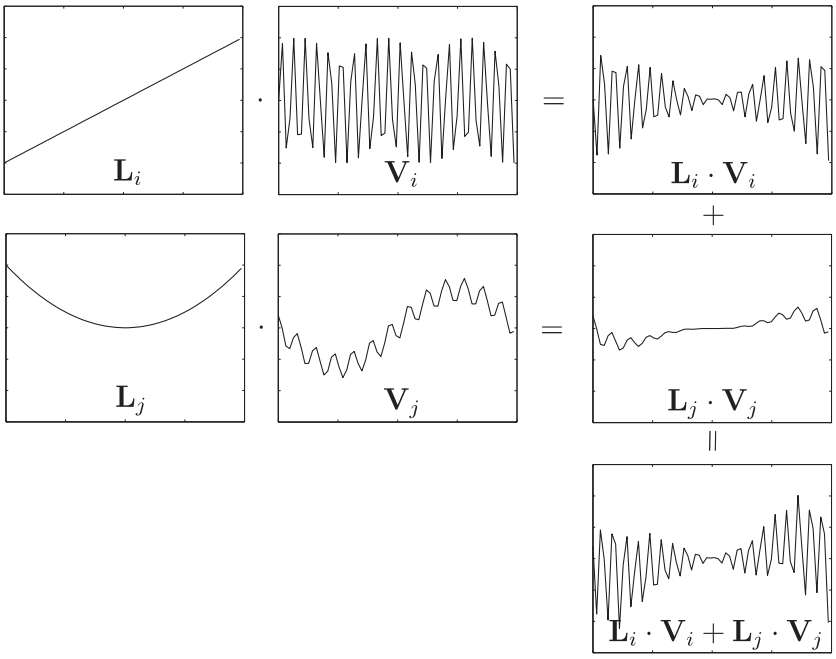


Abbildung 2.2: Die Approximierungsfunktion kann als Überlappung von gewichteten Basisfunktionen $\mathbf{V}_i$ und $\mathbf{V}_j$ beschrieben werden.

Die Approximation kann in diesem Fall als eine Minimierungsaufgabe bezüglich der Anzahl der Basisfunktionen und der Anzahl der zugehörigen Wichtungsfunktionen K sowie der Parameter $\mathbf{p}_j$ und $\mathbf{q}_j$ formuliert werden:

$$\min_{K,\{\mathbf{p}_j\},\{\mathbf{q}_j\}} \sum_{j=1}^{K} \int_{\mathbb{D}} \mathbf{L}^2(\mathbf{x}, \mathbf{q}_j) \left(\mathbf{C}(\mathbf{x}) - \mathbf{V}_j(\mathbf{x}, \mathbf{p}_j)\right)^2 d\mathbf{x}. \tag{2.10}$$

Für eine diskrete Approximation ergibt sich analog:

$$\min_{K,\{\mathbf{p}_j\},\{\mathbf{q}_j\}} \sum_{i=1}^{N} \sum_{j=1}^{K} \mathbf{L}^2(\mathbf{x}_i, \mathbf{q}_j) \left(\mathbf{y}_i - \mathbf{V}_j(\mathbf{x}_i, \mathbf{p}_j)\right)^2. \tag{2.11}$$

Die Gleichungen (2.10) und (2.11) beschreiben die Approximation als eine Minimierungsaufgabe bezüglich einer quadratischen Fehlerfunktion in allgemeiner Form. Die Wahl der Basis- sowie der Wichtungsfunktionen beeinflusst sowohl die Gestalt der Approximierungsfunktion als auch die Qualität der Minimierung. Die allgemeine Form für die Minimierung (2.10) bzw. (2.11) bietet im Vergleich zu (2.5) und (2.6) mehr Eingriffsmöglichkeiten zur Verbesserung der Approximationsergebnisse. Die Aufteilung der Approximierungsfunktion in Basis- und

Wichtungsfunktionen erlaubt, vorhandenes a priori Wissen gezielt an verschiedene Stellen in die Fehlerfunktion einzusetzen, um damit die Minimierungsaufgabe zu vereinfachen.

So entspricht zum Beispiel die allgemeine Form (2.9) der Approximierungsfunktion $\mathbf{V}$ einer Linearkombination der Basisfunktionen falls die Wichtungsfunktionen als Konstanten gewählt werden:

$$\mathbf{V} = \sum_{j=1}^{K} a_j \mathbf{V}_j(\mathbf{x}, \mathbf{p}_j). \tag{2.12}$$

Eine Minimierung dieser Funktion kann vereinfacht werden, wenn sie stückweise lokal durchgeführt wird. Dies lässt sich dadurch realisieren, dass man die Wichtungsfunktionen $\mathbf{L}_j$ so wählt, dass sie disjunkte lokale Bereiche $\mathcal{L}_j$ im Eingaberaum $\mathbb{D}$ beschreiben:

$$\begin{aligned} \mathbf{L}_j : \mathbb{R}^n \times \mathbb{R}^{|\mathbf{q}_j|} &\longrightarrow \{0,1\} \\ (\mathbf{x}, \mathbf{q}_j) &\longmapsto \begin{cases} 1 & \text{falls } \mathbf{x} \in \mathcal{L}_j \\ 0 & \text{sonst} \end{cases} \\ \mathcal{L}_j \cap \mathcal{L}_i = \emptyset \quad \forall i, j \in \{1 \ldots K\}. & \end{aligned} \tag{2.13}$$

Diese Art der Wichtungsfunktion wird Lokalisator-Funktion[4] genannt.

Die Lokalisator-Funktionen teilen den Eingaberaum in mehrere disjunkte Bereiche. Die Gestalt dieser Bereiche wird durch die Funktionenklasse, zu der die Lokalisator-Funktion $\mathbf{L}_j$ gehört, sowie die Parameter $\mathbf{q}_j$ bestimmt[5]. In jedem Bereich wirkt nur eine Basisfunktion. Dieses Konzept erlaubt, eine Minimierung stückweise in jedem Bereich durchzuführen. Dabei entspricht die zu minimierende Fehlerfunktion einer einfacheren Form wie in den Gleichungen (2.5) und (2.6). Eine solche Vorgehensweise ist beispielhaft in in der Abbildung 2.3 visualisiert.

Möchte man die Bereiche unscharf bzw. leicht überlappend definieren, so nehmen die Lokalisator-Funktionen folgende Gestalt an:

$$\begin{aligned} \mathbf{L}_j : \mathbb{R}^n \times \mathbb{R}^{|\mathbf{q}_j|} &\longrightarrow [0,1] \\ (\mathbf{x}, \mathbf{q}_j) &\longmapsto \begin{cases} \in (0,1] & \text{falls } \mathbf{x} \in \mathcal{L}_j, \\ 0 & \text{sonst.} \end{cases} \end{aligned} \tag{2.14}$$

[4] Eine Lokalisator-Funktion wird auch Fenster- bzw. Apertur-Funktion genannt.

[5] Die Lokalisator-Funktionen können zum Beispiel als Ellipsoide, Hypersphären oder auch als Funktionen auf nicht zusammenhängenden Bereichen definiert sein.

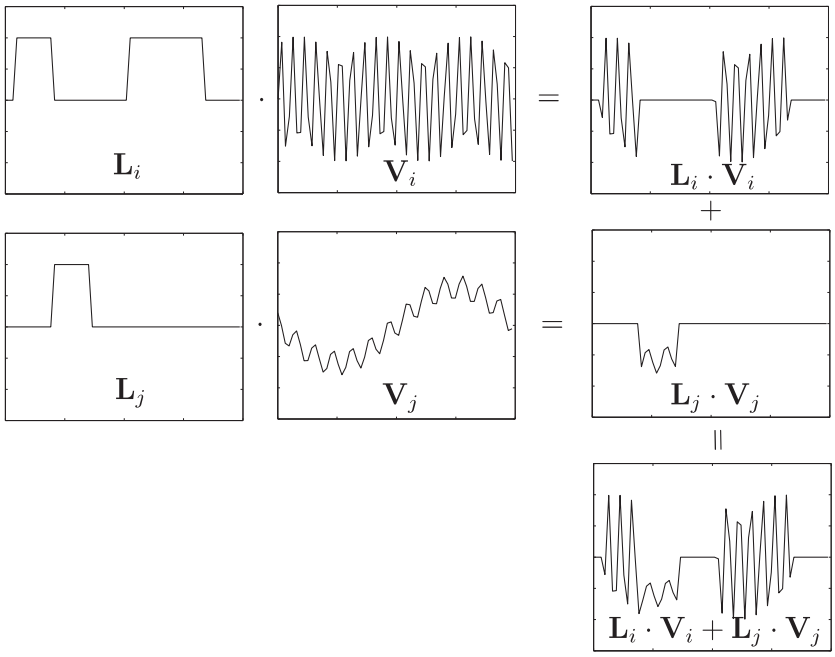


Abbildung 2.3: Lokalisator-Funktionen $\mathbf{L}_i$ und $\mathbf{L}_j$ lassen die zugehörigen globalen Basisfunktionen $\mathbf{V}_i$ und $\mathbf{V}_j$ nur lokal wirken.

Mit dem a priori Wissen über die Basisfunktionen sowie die entsprechenden Lokalisator-Funktionen – wie zum Beispiel in den Gleichungen (2.13) oder (2.14) beschrieben – kann die Minimierung von (2.10) bzw. (2.11) gesteuert werden.

2.4.2 Spezialfälle

Falls eine Abbildung $\mathbf{C}$ nur durch die Zustände $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1\ldots N}$ gegeben ist, und die Dynamik von $\mathbf{C}$ sich schlecht beschreiben lässt, so kann man durch die Beschränkung der Größe von lokalen Bereichen $\mathcal{L}_j$ auf einen Punkt und das Setzen der Anzahl der Bereiche auf N die punktweise Approximation exakt bestimmen. Die lokalen Bereiche entsprechen in diesem Fall den Punkten $\mathbf{x}_i$ im Eingaberaum:

$$\mathbf{L}_i(\mathbf{x}, \mathbf{q}_i) = \begin{cases} 1 & \text{falls } \mathbf{x} = \mathbf{x}_i \\ 0 & \text{sonst} \end{cases} \quad \forall \mathbf{x} \in \mathbb{D}, \forall i = 1 \ldots N. \tag{2.15}$$

Die Minimierungsaufgabe besteht jetzt darin, die Parameter $\mathbf{p}_i$ der Basisfunktionen $\mathbf{V}_i$ zu bestimmen, die aus einer Funktionenklasse $\mathcal{V}^k_{const}$ stammen, welche die

Menge aller konstanten Funktionen von $\mathbb{D}$ nach $\mathbb{W}$ beschreibt. Diese Minimierungsaufgabe ist einfach lösbar, da die Anzahl und die Parameter der Lokalisator-Funktion und die Funktionenklasse der Basisfunktionen festgelegt sind und sich die restlichen gesuchten Parameter direkt aus der Menge der Beispieldaten berechnen lassen:

$$\mathbf{V}_i(\mathbf{x}, \mathbf{p}_i) = \mathbf{y}_i \quad \forall \mathbf{x} \in \mathbb{D}, \forall i = 1 \ldots N. \tag{2.16}$$

Die Approximierungsfunktion entspricht in diesem Fall einer Look-Up-Tabelle. Die Daten werden auswendig gelernt, allerdings ist dabei weder eine Interpolation noch eine Extrapolation möglich. In diesem Fall zeigt sich das "curse of dimensionality" Problem besonders stark: Die berechnete Approximierungsfunktion besitzt keine Generalisierungsfähigkeit.

Wenn die Dynamik von $\mathbf{C}$ im Lokalen bekannt ist – das heißt, wenn die Funktionenklassen für die Basisfunktionen gut beschreibbar sind – so erhöht sich die Komplexität der Minimierungsaufgabe im Vergleich zum oben beschriebenen Fall. Man sucht nicht nur die Parameter $\mathbf{p}_j$ für die Basisfunktionen $\mathbf{V}_j$, sondern auch die Parameter $\mathbf{q}_j$ der Lokalisator-Funktionen $\mathbf{L}_j$ sowie die Anzahl der lokalen Bereiche K. Diese komplexe Minimierungsaufgabe kann nur dann erfolgreich gelöst werden, wenn sowohl das a priori Wissen in Form von Funktionenklassen, welche die Dynamik des Systems beschreiben, als auch Beispieldaten in ausreichender Menge vorhanden sind. Weiterhin ist die Wahl eines an die gestellte Aufgabe angepassten Minimierungsverfahren von grosser Wichtigkeit. Der Vorteil bei diesem Vorgehen besteht darin, dass im Fall der erfolgreichen Approximation die Anzahl der lokalen Bereiche insbesondere im Vergleich zu Look-Up-Tabellen klein ist, und die berechnete Approximierungsfunktion generalisierungsfähig ist. Die Generalisierungsfähigkeit ist aber meistens auf die Teile des Eingaberaumes beschränkt, die mit den lokalen Bereichen $\mathcal{L}_j$ abgedeckt sind. Mit anderen Worten, die Approximierungsfunktion besitzt eine gute Interpolationseigenschaft aber keine gute Extrapolationseigenschaft. Viele Typen von neuronalen Netzen basieren auf einer solchen Art der Approximation. Dazu gehören lokale Netzwerkarchitekturen wie zum Beispiel RBF-Netze, DCS, SVM [10, 12, 105].

Falls sich die Dynamik von $\mathbf{C}$ leicht auf dem gesamten Eingaberaum beobachten lässt, ist es möglich, nur einen lokalen Bereich zu verwenden, der auf den gesamten Raum ausgedehnt wird. Dementsprechend wird nur eine Basisfunktion eingesetzt. Die allgemeine Minimierungsaufgabe (2.10) reduziert sich auf die Minimierungsaufgabe (2.5). Diese Aufgabe lässt sich gut lösen, wenn die gewählte Funktionenklasse $\mathcal{V}^k$ "einfach" ist und die gesuchte Abbildung $\mathbf{C}$ sich damit gut beschreiben lässt. In diesem Fall wirkt die Approximierungsfunktion global, und kann sowohl gut interpolieren als auch gut extrapolieren. Im Bereich der Neuroinformatik entspricht dieses Konzept solchen global wirkenden Architekturen wie

zum Beispiel SLP und MLP [68, 91].

2.5 Zugang mit der Kern-Funktion

Da die Basisfunktion $\mathbf{V}$ die kleinste Approximierungseinheit darstellt, ist es wichtig zu verstehen, wie sich diese am besten ausdrücken lässt. Eine duale Darstellung desselben Problems entweder als eine Klassifizierungsaufgabe oder Approximierungsaufgabe kann sich dabei vorteilhaft auf den Entwurf des Modells auswirken.

Für eine gegebene Funktion W

$$\begin{aligned} W : \mathbb{R}^m &\longrightarrow \mathbb{R}^l \\ \mathbf{x} &\longmapsto \mathbf{y}. \end{aligned}$$

beschreibt $\mathrm{Kern}(W)$ die Menge aller Punkte aus dem Eingaberaum, für welche die Funktion den Nullwert annimmt:

$$\mathrm{Kern}(W) = \left\{ \mathbf{x} \in \mathbb{R}^m | W(\mathbf{x}) = 0 \in \mathbb{R}^l \right\}.$$

$\mathrm{Kern}(W)$ beschreibt einen Unterraum in $\mathbb{R}^m$. Geometrisch kann dieser als eine Hyperfläche, die den Raum in zwei Teile trennt, interpretiert werden.

Die Kernfunktion von W wird als eine Funktion $\mathbf{V}$ definiert, die als Wertebereich den Kern von W hat:

$$\mathbf{V} : \mathbb{R}^n \rightarrow \mathrm{Kern}(W) \subseteq \mathbb{R}^m,$$

oder anders formuliert

$$W(\mathbf{V}(\mathbf{x})) = \mathbf{0} \in \mathbb{R}^l \quad \forall \mathbf{x} \in \mathbb{R}^n.$$

Die Funktion W wird eine umgekehrte Kernfunktion zu $\mathbf{V}$ genannt.

Die Approximation einer Funktion in einem Raum kann durch die Verwendung einer entsprechenden umgekehrten Kernfunktion als eine Suche nach einer Trennfläche bzw. als eine Klassifizierungsaufgabe in einem anderem Raum formuliert werden.

Die einfachste Form einer Trennung in einem n-dimensionalen Raum ist ein $(n-1)$-dimensionaler Unterraum, der durch eine Hyperebene definiert wird. Eine Trennebene kann auf eine lineare Weise beschrieben werden. Eine Klassifizierung kann in einer einfachen Form als eine lineare Minimierungsaufgabe gestellt werden.

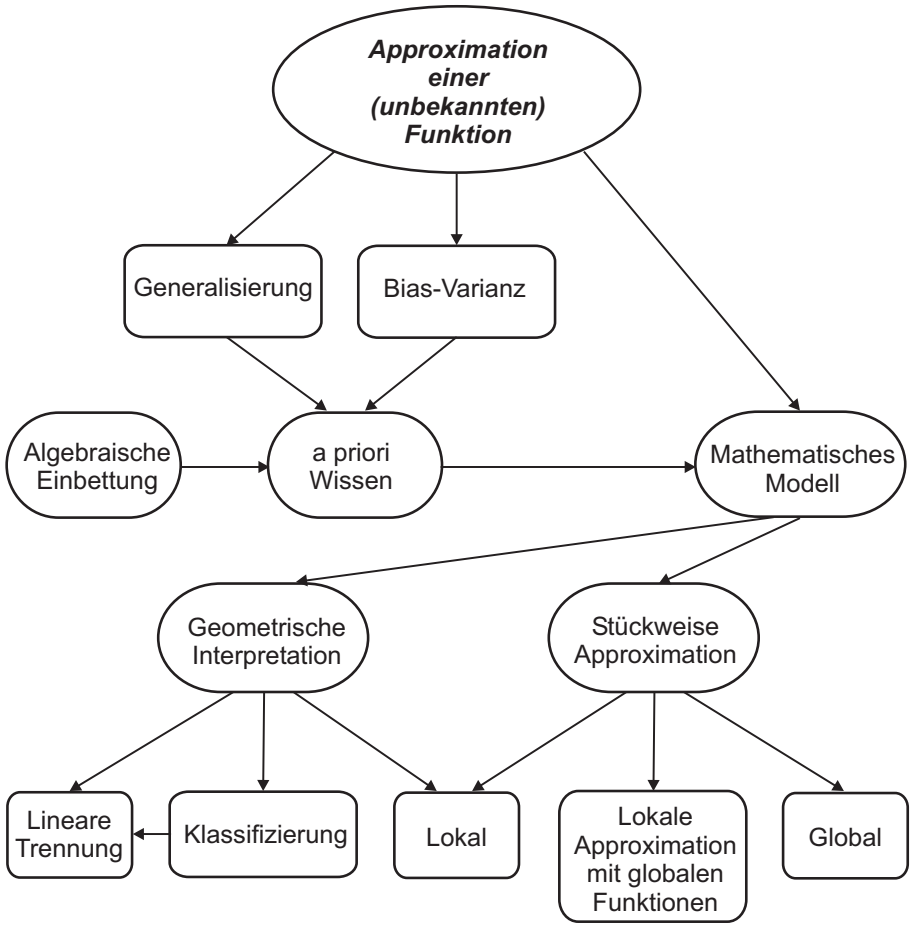


Abbildung 2.4: Approximation einer Funktion – Zusammenfassung.

2.6 Zusammenfassung

Eine lineare Trennung bietet in vielen Fällen nur eine suboptimale Lösung. Es sind komplexere Trennflächen erforderlich, um eine gute Approximation zu gewährleisten. Das Erhöhen der Komplexität von der Trennfunktion wirkt unmittelbar auf die Berechnungskomplexität, numerische Stabilität und die Generalisierungsfähigkeit der Approximation. Oft werden neue aufwendigere Algorithmen notwendig, um diesem Problem entgegenwirken zu können [37, 60]. Eine Alternative besteht darin, die Approximierungsaufgabe in einen anderen Raum zu übertragen, um die Komplexität zu reduzieren und nach Möglichkeit die Aufgabe zu linearisieren. Dies lässt sich in einigen Fällen durch die Wahl einer geigneten algebraischen Einbettung bewerkstelligen. Ein so umformuliertes Problem lässt sich dann gut mit Standardverfahren behandeln.

Das Konzept der algebraischen Einbettung lässt sich auf die Approximation der gesamten Funktion wie in den Gleichungen (2.10) und (2.11) übertragen. Je nach Aufgabe kann es erforderlich sein, die einzelnen Basisfunktionen in den selben oder auch in andere Räume auf eine spezielle Weise einzubetten.

Eine wichtige Rolle spielt auch Methode der Berechnung des Approximierungsfehlers. Sie hängt mit der Art der Datenrepräsentation zusammen. Vor allem übt die Struktur des Raumes, in dem die Daten dargestellt sind, allgemein bzw. die Metrik in diesem Raum speziell einen starken Einfluss auf die Qualität der Approximation aus. Die Wahl eines Distanzmaßes, das die Struktur des Raumes berücksichtigt, in dem das Problem formuliert wurde, ist daher von grosser Bedeutung. Abbildung 2.4 fasst die verschiedenen, allgemeinen Möglichkeiten zur Approximation einer unbekannten Funktion, die im Rahmen dieser Arbeit von Bedeutung sind, in einem Flussdiagramm zusammen.

Kapitel 3

CLIFFORD-ALGEBRA - ÜBERBLICK

Clifford-Algebra hat in den letzten Jahren in den Bereichen Bildverarbeitung und Robotik immer mehr an Bedeutung gewonnen. Besonders wichtig für Anwendungen in diesen Bereichen sind die Möglichkeiten, geometrische Entitäten wie Punkte, Geraden und Ebenen sowie Relationen zwischen ihnen als Elemente dieser Algebren auf kompakte Weise zu beschreiben. Die Einbettung von Euklidischen Räumen in Algebren über konforme Räume erlaubt weitere geometrische Strukturen wie Hypersphären als Nullräume von Elementen der Algebra linear auszudrücken. Auch die Wirkung von speziellen Transformationen wie Rotationen und Translationen kann durch die Anwendung des Versorproduktes auf verschiedene geometrische Entitäten auf die gleiche Weise berechnet werden. Weiterhin lassen sich einige Transformationen wie zum Beispiel allgemeine Rotationen bzw. starre Bewegungen durch eine geeignete Einbettung in höherdimensionale Algebren linearisieren.

Die Suche nach geeigneten geometrischen Interpretation für die Funktionalität von neuronalen Netzen sowie nach leicht berechenbaren Modellen zur Approximation macht die Clifford-Algebra auch für den Bereich der Neuroinformatik interessant.

Dieses Kapitel führt die Grundlagen der Clifford-Algebra ein. Es werden die Möglichkeiten dieser Algebra zur Modellierung von Entitäten und Transformationen in Euklidischen und konformen Räumen gezeigt. Der Schwerpunkt liegt dabei auf der Darstellung von Hyperebenen und Hypersphären sowie von allgemeinen Rotationen in konformen Räumen. Diese Aspekte der Algebra sind besonders wichtig im Kontext dieser Arbeit. Ein umfassender Einblick in weitere Forschungsbereiche, wo die Algebra erfolgreich eingesetzt wird, wird beispielhaft in [20, 41, 47, 59, 62, 83, 86, 98, 99] geboten.

3.1 Grundlegende Definitionen

In diesem Abschnitt wird nicht versucht, eine vollständige Einführung in die Clifford-Algebra zu geben. Es wird lediglich ein kompakter, im Rahmen dieser Arbeit hinreichend konsistenter Überblick angeboten.

Eine n-dimensionale Clifford-Algebra $\mathcal{G}_{p,q,r}$ ($p, q, r \in \mathbb{N}_0$ mit $n = p + q + r$) wird als ein geordneter 2^n-dimensionaler linearer Raum mit

- einer Struktur über die Unterräume verschiedener Dimensionalität, den so genannten Blades,
- einer Addition,
- einer Skalarmultiplikation,
- einer nichtkommutativen Multiplikation, die bezüglich Addition assoziativ und distributiv ist:
  1. $\forall \mathbf{A}, \mathbf{B}, \mathbf{C} \in \mathcal{G}_{p,q,r}: \quad (\mathbf{AB})\mathbf{C} = \mathbf{A}(\mathbf{BC})$,
  2. $\forall \mathbf{A}, \mathbf{B}, \mathbf{C} \in \mathcal{G}_{p,q,r}: \quad \mathbf{A}(\mathbf{B} + \mathbf{C}) = \mathbf{AB} + \mathbf{AC}$,
  3. $\forall \mathbf{A}, \mathbf{B} \in \mathcal{G}_{p,q,r}; \alpha \in \mathbb{R}: \quad (\alpha\mathbf{A})\mathbf{B} = \mathbf{A}(\alpha\mathbf{B}) = \alpha(\mathbf{AB})$,

  dem so genannten geometrischen oder Clifford-Produkt,

definiert. Im Gegensatz zu Vektorräumen können Elemente einer solchen Algebra auch Entitäten höherer Ordnung repräsentieren. Sie werden durch die lineare Kombination von Blades verschiedener Dimensionalität gebildet, und heißen Multivektoren. Multivektoren, die nur aus Blades gleichen Grades bestehen heißen homogen.

Das Clifford-Produkt von zwei Vektoren[1] $\mathbf{A}$ und $\mathbf{B}$ kann als die Summe ihrer symmetrischen und antisymmetrischen Komponenten geschrieben werden, dem inneren ($\cdot$) und äußeren ($\wedge$) Produkt:

$$\mathbf{AB} = \mathbf{A} \wedge \mathbf{B} + \mathbf{A} \cdot \mathbf{B}.$$

Weiterhin wird folgende Beziehung zwischen drei beliebigen Vektoren $\mathbf{A}$,$\mathbf{B}$ und $\mathbf{C}$ einer Clifford-Algebra erfüllt:

$$\mathbf{A} \cdot (\mathbf{B} \wedge \mathbf{C}) = (\mathbf{A} \cdot \mathbf{B})\mathbf{C} - (\mathbf{A} \cdot \mathbf{C})\mathbf{B}.$$

[1] Homogene Multivektoren des ersten Grades werden weiterhin Vektoren genannt.

Die Algebra $\mathcal{G}_{p,q,r}$ wird durch die Anwendung des Clifford-Produktes auf den Vektorraum $\mathbb{R}^{p,q,r}$ mit der Signatur (p,q,r) auf folgende Weise gebildet: Für zwei orthonormale Basisvektoren e_i und e_j aus $\mathbb{R}^{p,q,r}$ ist das Clifford-Produkt als

$$\mathsf{e}_i\mathsf{e}_j = \begin{cases} 1, & i = j \in \{1, \cdots, p\} \\ -1, & i = j \in \{p+1, \cdots, p+q\} \\ 0, & i = j \in \{p+q+1, \cdots, n\} \\ \mathsf{e}_{ij} = \mathsf{e}_i \wedge \mathsf{e}_j = -\mathsf{e}_j \wedge \mathsf{e}_i \in \mathcal{G}_{p,q,r}, & i \neq j \end{cases} \tag{3.1}$$

definiert. Das Produkt zweier orthogonaler Basisvektoren bildet dabei eine höherdimensionale Entität, die als der von diesen beiden Vektoren aufgespannte Unterraum interpretiert werden kann. Generell kann jeder Multivektor $\mathbf{A}$ einer n-dimensionalen Clifford-Algebra als die Summe von Elementen verschiedener Grade ausgedrückt werden:

$$\mathbf{A} = \sum_{i=0}^{n} \langle \mathbf{A} \rangle_i, \tag{3.2}$$

wobei $\langle . \rangle_i$ die Projektion des Multivektors auf die Entitäten vom Grad i angibt. Ein homogener Multivektor $\mathbf{A}$ vom Grad i wird auch mit $\mathbf{A}_{\langle i \rangle}$ bezeichnet.

Eine Clifford-Algebra $\mathcal{G}_{p,q,r}$ über den Vektorraum $\mathbb{R}^{p,q,r}$ heißt universell, falls $\mathbb{R}^{p,q,r}$ nicht degeneriert[2] ist, und die Dimension der Algebra 2^{p+q+r} ist. Eine Veränderung der Basis einer universellen Algebra ändert die Struktur der Algebra nicht (bis auf Isomorphismus). In dieser Arbeit werden nur solche universelle Algebren betrachtet. Es gelten dabei folgende Abkürzungen: $\mathcal{G}_{p,q} \equiv \mathcal{G}_{p,q,0}$ und $\mathcal{G}_p \equiv \mathcal{G}_{p,0,0}$.

Eine universelle Clifford-Algebra, die mit einer geometrischen Interpretation versehen ist, wird auch geometrische Algebra genannt. So gilt zum Beispiel für zwei beliebige Vektoren einer Algebra $\mathbf{A}$ und $\mathbf{B}$, falls diese zwei Richtungen A und B in einem Euklidischen Raum repräsentieren:

$$\mathbf{A} \wedge \mathbf{B} = 0 \Longleftrightarrow A || B$$

und

$$\mathbf{A} \cdot \mathbf{B} = 0 \Longleftrightarrow A \perp B.$$

Die Bezeichnung geometrische Algebra wird im Bereich Robotik und Bildverarbeitung von vielen Autoren bevorzugt benutzt [28, 29, 46, 48, 89, 90, 100].

Generell ist eine n-dimensionale Clifford-Algebra $\mathcal{G}_n$ durch 2^n Basiselemente bzw. Basisblades $\{\mathsf{E}_i\}_{2^n}$ gegeben. Für je zwei Basisblades $\mathsf{E}_{\langle r \rangle}$ und $\mathsf{E}_{\langle s \rangle}$ wird das

[2] Eine Clifford-Algebra $\mathcal{G}_{p,q,r}$ heißt degeneriert falls $r > 0$ ist.

geometrische Produkt als

$$\mathsf{E}_{\langle r\rangle}\mathsf{E}_{\langle s\rangle} = \langle \mathsf{E}_{\langle r\rangle}\mathsf{E}_{\langle s\rangle}\rangle_{r+s-2m} \tag{3.3}$$

gebildet. Die Zahl m gibt dabei die gemeinsame Anzahl von Basisvektoren an, die zur Bildung beider Basisblades beigetragen haben. Das geometrische Produkt zweier beliebiger Blades $\mathbf{A}_{\langle k\rangle}, \mathbf{B}_{\langle l\rangle} \in \mathcal{G}_n$ entsteht aus linearer Kombination von Basisblades und lässt sich als

$$\begin{aligned}\mathbf{A}_{\langle k\rangle}\mathbf{B}_{\langle l\rangle} &= \sum_{i=0}^{n}\langle \mathbf{A}_{\langle k\rangle}\mathbf{B}_{\langle l\rangle}\rangle_i \\ &= \langle \mathbf{A}_{\langle k\rangle}\mathbf{B}_{\langle l\rangle}\rangle_{|k-l|} + \langle \mathbf{A}_{\langle k\rangle}\mathbf{B}_{\langle l\rangle}\rangle_{|k-l|+2} + \cdots \\ &+ \langle \mathbf{A}_{\langle k\rangle}\mathbf{B}_{\langle l\rangle}\rangle_{k+l-2} + \langle \mathbf{A}_{\langle k\rangle}\mathbf{B}_{\langle l\rangle}\rangle_{k+l}.\end{aligned} \tag{3.4}$$

berechnen. Das Element des kleinsten Grades ($|k-l|$) entspricht dabei dem inneren Produkt. Das äußere Produkt wird durch das Element des höchsten Grades ($k+l$) gebildet. Für zwei beliebig gewählte Blades sind nicht alle Elemente in der Summe notwendigerweise enthalten.

Im Rahmen dieser Arbeit (Kapitel 5 und 6) ist eine adäquate Beschreibung für eine effiziente neuronale Modellierung erforderlich. Die Tensorrechnung bietet sich für diese Zwecke wegen der Möglichkeit der leistungsfähigen Implementierung an. Dabei lässt sich die Struktur von $\mathcal{G}_n$ als $2^n \times 2^n \times 2^n$ Tensor G ausdrücken. Der Tensor wird durch eine und damit jede Orthonormalbasis $\{\mathsf{e}_i\}_n$ des zugrundeliegenden Vektorraumes induziert. Die Elemente g_{ijk} von G mit $g_{ijk} \in \{-1, 0, 1\}$ und $i, j, k \in \{1, \cdots, 2^n\}$ beschreiben die Relation zwischen verschiedenen Basiselementen E_j bezüglich des geometrischen Produktes:

$$\mathsf{E}_i\mathsf{E}_j = \sum_{k=1}^{2^n} g_{ijk}\mathsf{E}_k. \tag{3.5}$$

Die Elemente g_{ijk} des Tensors G werden gemäß den Multiplikationsregeln (3.1) berechnet. In universellen Algebren gilt:

$$\forall i, j \in \{1, \cdots, 2^n\} \quad \exists^1 k \in \{1, \cdots, 2^n\} : \qquad g_{ijk} \neq 0.$$

Jeder Multivektor $\mathbf{A} \in \mathcal{G}_n$ lässt sich als lineare Kombination von Basiselementen E_i ausdrücken:

$$\mathbf{A} = \sum_{i=1}^{2^n} a_i\mathsf{E}_i, \quad a_i \in \mathbb{R}. \tag{3.6}$$

Das geometrische Produkt einer n-dimensionalen Algebra kann bei einer geeigneten Darstellung in $\mathbb{R}^{2^n}$ mit Hilfe des Tensors G berechnet werden. Repräsentieren die Vektoren

$$\mathbf{a} = (a_1, \cdots, a_{2^n})^T \in \mathbb{R}^{2^n} \quad \text{und} \quad \mathbf{b} = (b_1, \cdots, b_{2^n})^T \in \mathbb{R}^{2^n}$$

die Multivektoren

$$\mathbf{A} = \sum_{i=1}^{2^n} a_i \mathsf{E}_i \in \mathcal{G}_n \quad \text{und} \quad \mathbf{B} = \sum_{j=1}^{2^n} b_j \mathsf{E}_j \in \mathcal{G}_n,$$

so entspricht $\mathbf{c} = (c_1, \cdots, c_{2^n})^T \in \mathbb{R}^{2^n}$ mit

$$c_k = \sum_{i=1,j=1}^{2^n} g_{ijk} a_i b_j, \quad k = 1 \cdots 2^n \tag{3.7}$$

dem Ergebnis

$$\mathbf{C} = \mathbf{AB} = \sum_{i=k}^{2^n} c_k \mathsf{E}_k \in \mathcal{G}_n. \tag{3.8}$$

des geometrischen Produktes zwischen den Multivektoren $\mathbf{A}$ und $\mathbf{B}$.

3.2 Euklidische Räume

Um einen n-dimensionalen Euklidischen Raum $\mathbb{E}^n$ zu beschreiben, wird typischerweise eine geometrische Algebra mit positiver Signatur $\mathcal{G}_n$ ($n = p$) eingesetzt. In Bildverarbeitung und Robotik spielen 2D- und 3D-Räume eine besonders wichtige Rolle, da in diesen Räumen die Beziehungen zwischen den 3D-Objekten und ihren 2D-Repräsentationen in der Bildebene modelliert werden. Da die Euklidische Einbettung für alle Dimensionen auf eine ähnliche Weise geschieht, wird diese im Folgenden am Beispiel des Euklidischen 3D-Raumes $\mathbb{E}^3$ verdeutlicht.

3.2.1 Darstellung von Entitäten in $\mathcal{G}_3$

Seien $\{\mathsf{e}_1, \mathsf{e}_2, \mathsf{e}_3\}$ drei orthonormale Basisvektoren des Euklidischen Vektorraumes $\mathbb{E}^3$. Die induzierte geometrische Algebra $\mathcal{G}_3$ hat $2^3 = 8$ Basiselemente:

- 1: Skalar,
- $\mathsf{e}_1, \mathsf{e}_2, \mathsf{e}_3$: 3 Vektoren – Einheitsvektoren oder Richtungen,
- $\mathsf{e}_{23}, \mathsf{e}_{31}, \mathsf{e}_{12}$: 3 Bivektoren – die Einheitsebenen,
- $\mathsf{e}_{123} \equiv \mathsf{I}$: Einheitspseudoskalar – Einheitsvolumen.

Verschiedene geometrische Objekte wie Punkte, Geraden und Ebenen können in dieser Algebra repräsentiert werden. Ein Punkt x mit den Euklidischen Koordinaten $(x_1, x_2, x_3) \in \mathbb{E}^3$ kann als eine lineare Kombination von Basisvektoren beschrieben werden:

$$\mathbf{x} = x_1\mathsf{e}_1 + x_2\mathsf{e}_2 + x_3\mathsf{e}_3. \tag{3.9}$$

Eine Gerade l[3] wird durch die Richtung r und das Moment m gegeben:

$$\mathbf{l} = \mathbf{r} + \mathbf{m}. \tag{3.10}$$

In dieser Darstellung ist die Richtung r mit dem Vektor $\mathbf{r}$ aus der Algebra kodiert. Das Moment $\mathbf{m}$ lässt sich als äußeres Produkt zwischen einem Punkt $\mathbf{y}$ auf der Geraden $\mathbf{l}$ und der Richtung $\mathbf{r}$ ausdrücken:

$$\mathbf{m} = \mathbf{y} \wedge \mathbf{r} \tag{3.11}$$

und wird damit in $\mathcal{G}_3$ als Bivektor repräsentiert. Gleichung (3.10) stellt eine Repräsentation von Geraden in Plückerkoordinaten dar.

Das äußere Produkt zwischen einem Punkt $\mathbf{x}$ und einer Gerade $\mathbf{l}$ ist genau dann Null, wenn der Punkt auf der Geraden liegt. Da eine Gerade durch die Menge aller darauf liegenden Punkte gegeben wird, beschreibt der Nullraum bezüglich des äußeren Produktes des Multivektors $\mathbf{l}$ die Gerade als geometrische Entität im Euklidischen Raum:

$$x \in l \iff \mathbf{x} \wedge \mathbf{l} = 0. \tag{3.12}$$

Auf eine ähnliche Weise werden Ebenen in $\mathcal{G}_3$ modelliert:

$$\mathbf{p} = \mathbf{mn} + d\mathsf{I}, \tag{3.13}$$

wobei $\mathbf{p}$ die multivektorielle Darstellung der Ebene p ist, d den Abstand der Ebene zum Ursprung beschreibt, und der Einheitsbivektor $\mathbf{n}$ die Orientierung der Ebene angibt. Der Nullraum (in Bezug auf das äußere Produkt) von $\mathbf{p}$ beschreibt eine Ebene im Euklidischen Raum:

$$x \in p \iff \mathbf{x} \wedge \mathbf{p} = 0. \tag{3.14}$$

[3] In dieser Arbeit wird mit l die eigentliche geometrische Entität und mit $\mathbf{l}$ die entsprechende Repräsentation in $\mathcal{G}_n$ bezeichnet.

3.2.2 Rotationen und Translationen in $\mathcal{G}_3$

Betrachtet man die Beziehung zwischen Vektoren und Bivektoren in $\mathcal{G}_3$ gemäß den Multiplikationsregeln (3.1), so lässt sich folgender Zusammenhang feststellen:

$$\begin{aligned} \mathsf{e}_{ij}\mathsf{e}_i = -\mathsf{e}_j, \qquad & \mathsf{e}_{ij}\mathsf{e}_j = \mathsf{e}_i, \\ \mathsf{e}_{ij}\mathsf{e}_k = \mathsf{e}_{ijk}. & \end{aligned} \tag{3.15}$$

Vom geometrischen Gesichtspunkt aus bedeutet dies, dass ein Einheitsbivektor einen Einheitsvektor um $\pi/2$ in der Ebene des Einheitsbivektors rotiert, oder ein Volumen bildet, falls der Vektor e_k senkrecht zur Ebene e_{ij} ist.

Im Allgemeinen kann jede Rotation von Punkten im Euklidischen Raum als eine lineare Kombination von Bivektoren und einem Skalar (gerade Elemente von $\mathcal{G}_3$) repräsentiert werden. Jeder Multivektor R vom geraden Grad mit $R\widetilde{R} = 1 \in \mathcal{G}_3$

$$\begin{aligned} R &= r_0 + r_1\mathsf{e}_{23} + r_2\mathsf{e}_{31} + r_3\mathsf{e}_{12} \\ \widetilde{R} &= r_0 - r_1\mathsf{e}_{23} - r_2\mathsf{e}_{31} - r_3\mathsf{e}_{12} \end{aligned} \tag{3.16}$$

kann zu Berechnung von Rotationen benutzt werden, wobei $\widetilde{R}$ die Reverse von R ist. Ein solcher Multivektor heißt Rotor.

In der Euler-Darstellung[4] lässt sich ein Rotor folgendermaßen beschreiben:

$$R = \exp\left(-\frac{\theta}{2}\mathbf{n}\right) = \cos\left(-\frac{\theta}{2}\right) - \mathbf{n}\sin\left(-\frac{\theta}{2}\right). \tag{3.17}$$

θ gibt dabei den Rotationswinkel an, und $\mathbf{n}$ ist die Bivektor-Darstellung der Rotationsebene. Gleichung (3.17) lässt Rückschlüsse darauf ziehen, wie die Elemente des Multivektors in (3.16) geometrisch zu interpretieren sind.

Die Rotation $\mathcal{R}$ eines Punktes $\mathbf{x}$ um den Ursprung wird mit Hilfe eines Versorproduktes in $\mathcal{G}_3$ realisiert:

$$\mathbf{y} = \mathcal{R}(\mathbf{x}) = R\mathbf{x}\widetilde{R}. \tag{3.18}$$

Die Menge der Rotationen in $\mathcal{G}_3$ bildet eine multiplikativ abgeschlossene Gruppe $SO(3)$: Das geometrische Produkt zweier Rotoren ist wieder ein Rotor. Diese Gruppe lässt sich als die Menge aller Multivektoren der geraden Unteralgebra $\mathcal{G}_3^+$ der Algebra $\mathcal{G}_3$ beschreiben, deren geometrisches Produkt mit den entsprechenden Reversen eins ist:

[4] nicht zu verwechseln mit Euler-Winkel-Darstellung, einer Parametrisierung über die Rotationswinkel der einzelnen Koordinatenachsen des Raumes.

$$\{R \in \mathcal{G}_3^+ | R\widetilde{R} = 1\}.$$

Rotoren lassen sich nicht nur auf Punkte, sondern auch auf andere geometrische Entitäten auf dieselbe Weise anwenden. Weiterhin wirken Rotoren linear auf alle Elemente von $\mathcal{G}_3$ und linearisieren darüber hinaus Rotationen in Räumen jeder Dimension.

Translationen $\mathcal{T}$ bilden eine additiv abgeschlossene Gruppe. Eine Translation wird als ein Verschiebungsvektor $\mathbf{t}$ in $\mathcal{G}_3$ kodiert:

$$\mathbf{y} = \mathcal{T}(\mathbf{x}) = \mathbf{x} + \mathbf{t}. \tag{3.19}$$

Im Gegensatz zu Rotationen sind Translationen in $\mathcal{G}_3$ nichtlineare Operationen.

Die sich stark unterscheidenden Darstellungsarten von Rotationen und Translationen in $\mathcal{G}_3$ (additiv/multiplikativ, linear/nichtlinear) bieten ungünstige Voraussetzungen zur effizienten Repräsentation der Gruppe von starren Bewegungen als Kombination von Rotationen und Translationen. Durch eine algebraische Einbettung in höherdimensionale Räume ist es aber möglich, die Gruppe der starren Bewegungen zu linearisieren. Im Rahmen dieser Arbeit wird insbesondere die konforme Einbettung benutzt.

3.3 Konforme Räume

Ein n-dimensionaler konformer Raum $\mathbb{K}^n$ wird über den n-dimensionalen Euklidischen Raum $\mathbb{E}^n$ modelliert, indem man eine zusätzliche Komponente hinzufügt, und wird durch $\mathbb{R}^{n+1}$ repräsentiert. Der Basisvektor für diese zusätzliche Dimension wird im Rahmen dieser Arbeit mit $\mathsf{e}_+ \equiv \mathsf{e}_{n+1}$ bezeichnet.

Die Einbettung von $\mathbb{E}^n$ in $\mathbb{K}^n$ wird auf eine nicht lineare Weise mittels stereographischer Projektion $\mathcal{K}$ durchgeführt:

$$\begin{aligned} \mathcal{K} : \mathbb{E}^n &\longrightarrow \mathbb{K}^n \equiv \mathbb{R}^{n+1} \\ \mathbf{x} &\longmapsto \frac{2}{\mathbf{x}^2+1}\mathbf{x} + \frac{\mathbf{x}^2-1}{\mathbf{x}^2+1}\mathsf{e}_+, \end{aligned} \tag{3.20}$$

wobei

$$\mathbf{x}^2 = \mathbf{x} \cdot \mathbf{x} \in \mathbb{R}_{\geq 0}$$

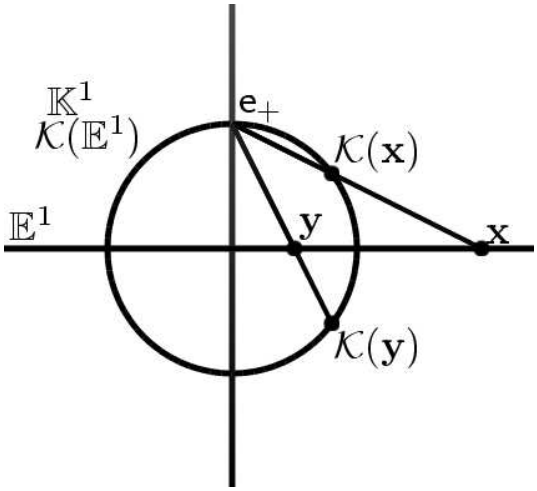


Abbildung 3.1: Die stereographische Projektion der Punkte $\mathbf{x}$ und $\mathbf{y}$ aus $\mathbb{E}^1$ auf den Einheitskreis in $\mathbb{K}^1$.

der quadrierten Länge des Euklidischen Vektors $\mathbf{x}$ entspricht.

Die Transformation $\mathcal{K}$ bildet die Punkte des n-dimensionalen Euklidischen Raumes $\mathbb{E}^n$ auf die Einheitshypersphäre im $(n+1)$-dimensionalen Raum $\mathbb{K}^n$ ab. Die Abbildung 3.1 illustriert die Einbettungsvorschrift für den eindimensionalen Fall. Man kann erkennen, dass der nicht eingeschränkte Euklidische Raum auf den (eingeschränkten) Kreis projiziert wird. Insbesondere werden die Punkte im Unendlichen $-\infty$ und $+\infty$ aus $\mathbb{E}^1$ auf den Punkt e_+ auf dem Einheitskreis in $\mathbb{K}^1$ abgebildet. In Abbildung 3.2 wird veranschaulicht, dass sowohl Kreise als auch Geraden aus $\mathbb{E}^2$ auf der Einheitshypersphäre in $\mathbb{K}^2$ als Kreise dargestellt werden. Insbesondere werden die Geraden auf die Kreise abgebildet, die e_+ enthalten.

Die Rücktransformation $\mathcal{K}^{-1}$ ist nur für die Punkte $\mathbf{x} \in \mathbb{K}^n$ möglich, die auf der Hypersphäre liegen ($||\mathbf{x}|| = 1$) und ist durch

$$\begin{aligned} \mathcal{K}^{-1} : \mathbb{K}^n &\longrightarrow \mathbb{E}^n \\ \mathbf{x} &\longmapsto \frac{1}{1-\mathbf{x}\cdot\mathsf{e}_+}\sum_{i=1}^{n}(\mathbf{x}\cdot\mathsf{e}_i)\mathsf{e}_i \end{aligned} \tag{3.21}$$

gegeben.

Die Einbettung $\mathcal{P}$ des konformen Raumes $\mathbb{K}^n$ in den projektiven Raum $\mathbb{PK}^n$ geschieht durch die homogene Erweiterung:

$$\begin{aligned} \mathcal{P} : \mathbb{K}^n &\longrightarrow \mathbb{PK}^n \equiv \mathbb{R}^{n+1,1}\backslash 0 \\ \mathbf{x} &\longmapsto \mathbf{x} + \mathsf{e}_-. \end{aligned} \tag{3.22}$$

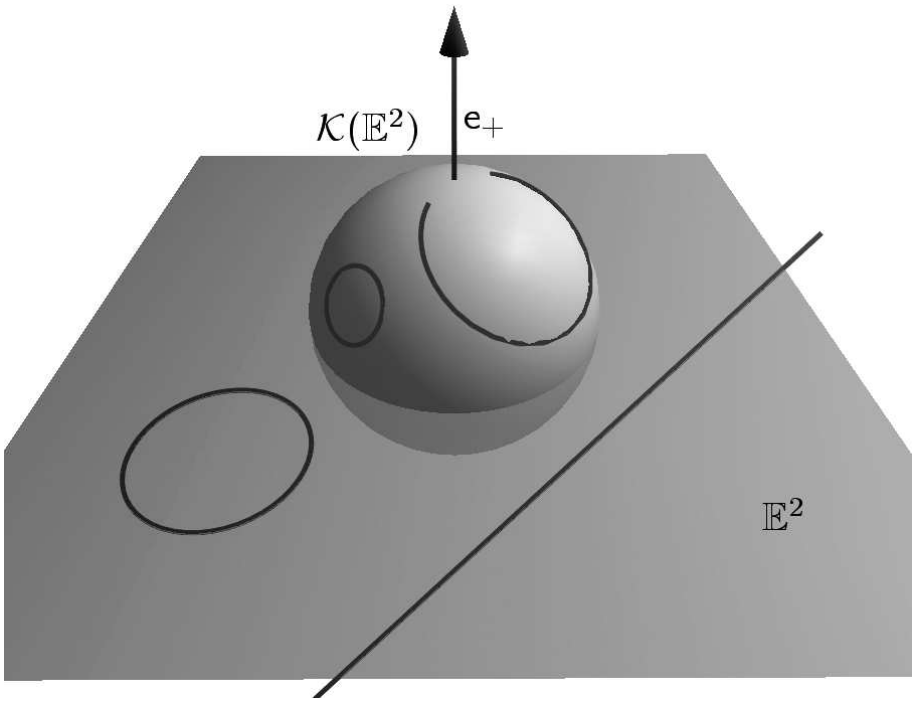


Abbildung 3.2: Die stereographische Projektion einer Geraden und eines Kreises aus $\mathbb{E}^2$ auf die Einheitskugel in $\mathbb{K}^2$.

Die orthonormale Basis von $\mathbb{PK}^n$ enthält $n+1$ Basisvektoren $\{\mathsf{e}_1, \cdots, \mathsf{e}_n, \mathsf{e}_{n+1} \equiv \mathsf{e}_+\}$ mit positiver Signatur ($\mathsf{e}_i \cdot \mathsf{e}_i = 1, \quad 1 < i < n+1$) und einen Basisvektor e_- mit negativer Signatur ($\mathsf{e}_- \cdot \mathsf{e}_- = -1$). Solche Räume werden auch Minkowski Räume genannt.

Die Einbettung in einen konformen und anschließend in einen projektiven Raum impliziert, dass der Euklidische Raum $\mathbb{E}^n$ nicht auf den gesamten Raum $\mathbb{PK}^n$, sondern auf eine spezielle Mannigfaltigkeit in $\mathbb{PK}^n$, den so genannten Nullkegel, abgebildet wird. Der Name Nullkegel kommt daher, dass alle Punkte auf dem Kegel zu Null quadrieren:

$$\mathcal{P}(\mathcal{K}(\mathbb{E}^n))^2 = 0.$$

In der Abbildung 3.3 ist die Einbettung für den eindimensionalen Fall dargestellt. Der Punkt $\mathbf{x} \in \mathbb{E}^1$ wird auf den Einheitskreis auf dem Nullkegel in $\mathbb{PK}^1$ abgebildet. Die homogene Darstellung bewirkt, dass die gesamte Parabel auf dem Nullkegel denselben Euklidischen Punkt repräsentiert.

Die inverse Transformation $\mathcal{P}^{-1}$ ist für alle Elemente $\mathbf{X} \in \mathbb{PK}^n$ mit $\mathbf{X} \cdot \mathsf{e}_- \neq 0$

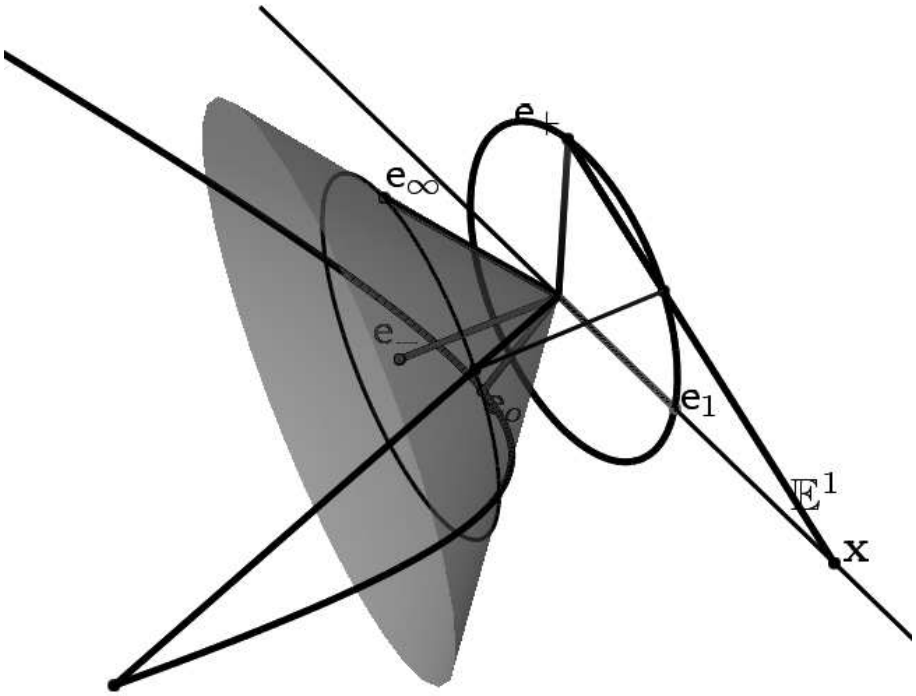


Abbildung 3.3: Einbettung vom Euklidischen Raum $\mathbb{E}^1$ in den projektiven konformen Raum $\mathbb{PK}^1$.

auf folgende Weise definiert:

$$\begin{aligned} \mathcal{P}^{-1} : \mathbb{PK}^n &\longrightarrow \mathbb{K}^n \\ \mathbf{X} &\longmapsto \frac{1}{\mathbf{X} \cdot \mathsf{e}_-} \sum_{i=1}^{n+1} (\mathbf{X} \cdot \mathsf{e}_i)\mathsf{e}_i. \end{aligned} \tag{3.23}$$

Die gesamte Einbettung eines Euklidischen Vektors $\mathbf{x} \in \mathbb{E}^n$ in den projektiven konformen Raum $\mathbb{PK}^n$ ist durch die Abbildungsvorschrift

$$\mathcal{P}(\mathcal{K}(\mathbf{x})) = \frac{2}{\mathbf{x}^2 + 1}\mathbf{x} + \frac{\mathbf{x}^2 - 1}{\mathbf{x}^2 + 1}\mathsf{e}_+ + \mathsf{e}_- \tag{3.24}$$

gegeben.

Da in projektiven Räumen eine Skalierung keine Auswirkung auf die Repräsentation des zugrunde liegenden Euklidischen Raumes hat, kann es für die Erhöhung der Berechnungseffizienz hilfreich sein, eine geeignete Skalierung zu wählen. Im

Fall von (3.24) ist eine Skalierung mit $\frac{1}{2}(\mathbf{x}^2+1)$ nützlich, um den Nenner zu eliminieren:

$$\begin{aligned}\tfrac{1}{2}(\mathbf{x}^2+1)\mathcal{P}(\mathcal{K}(\mathbf{x})) &= \mathbf{x}+\tfrac{1}{2}(\mathbf{x}^2-1)\mathsf{e}_+ + \tfrac{1}{2}(\mathbf{x}^2+1)e_- \\ &= \mathbf{x}+\tfrac{1}{2}\mathbf{x}^2(\mathsf{e}_- + \mathsf{e}_+) + \tfrac{1}{2}(\mathsf{e}_- - \mathsf{e}_+) \\ &= \mathbf{x}+\tfrac{1}{2}\mathbf{x}^2\mathsf{e}_\infty + \mathsf{e}_o,\end{aligned} \tag{3.25}$$

wobei e_∞ und e_o als

$$\mathsf{e}_\infty := \mathsf{e}_- + \mathsf{e}_+ \qquad \mathsf{e}_o := \tfrac{1}{2}(\mathsf{e}_- - \mathsf{e}_+) \tag{3.26}$$

definiert sind und folgende Eigenschaften aufweisen:

$$\mathsf{e}_\infty \cdot \mathsf{e}_\infty = \mathsf{e}_o \cdot \mathsf{e}_o = 0 \qquad \mathsf{e}_\infty \cdot \mathsf{e}_o = -1. \tag{3.27}$$

Anhand von Gleichung (3.25) wird der Einbettungsoperator $\mathcal{C}$ definiert:

$$\begin{aligned}\mathcal{C} : \mathbb{E}^n &\longrightarrow \mathbb{PK}^n \\ \mathbf{x} &\longmapsto \mathbf{x}+\tfrac{1}{2}\mathbf{x}^2\mathsf{e}_\infty + \mathsf{e}_o.\end{aligned} \tag{3.28}$$

Eine konforme geometrische Algebra über einen n-dimensionalen Euklidischen Raum $\mathbb{E}^n$ wird gebildet, indem man die Algebra $\mathcal{G}_{n+1,1}$ über den konformen projektiven Vektorraum $\mathbb{PK}^n$ mit der Basis $\{\mathsf{e}_1, \cdots, \mathsf{e}_n, \mathsf{e}_+, \mathsf{e}_-\}$ aufspannt. Die induzierte Algebra $\mathcal{G}_{n+1,1}$ hat 2^{n+2} Basiselemente.

3.3.1 Darstellung von Entitäten in $\mathbb{PK}^n$

Jede gewählte geometrische Algebra besitzt fundamentale geometrische Entitäten, die als Grundlage für Modellierung aller weiteren Entitäten benutzt werden. In Algebren über Euklidischen Räumen sind dies Punkte. Geraden und Ebenen werden als Menge von Punkten konstruiert, die gewissen Zwängen unterliegen. In Algebren über konforme Räume wird diese Grundentität durch Hypersphären repräsentiert.

Gleichung (3.28) beschreibt die Einbettung von $\mathbb{E}^n$ auf den Nullkegel in $\mathbb{PK}^n$:

$$\mathbf{X} = \mathbf{x}+\tfrac{1}{2}\mathbf{x}^2\mathsf{e}_\infty + \mathsf{e}_o. \tag{3.29}$$

Es stellt sich die Frage, welche geometrische Bedeutung in dem zugrunde liegenden Euklidischen Raum Punkte des konformen Raumes[5] haben, die außerhalb des Nullkegels liegen.

[5] Im Sinne von Koordinatentupeln.

Betrachtet man einen Punkt $\mathbf{S}$ außerhalb des Nullkegels

$$\mathbf{S} = \mathbf{C} - \tfrac{1}{2}r^2\mathsf{e}_\infty = \mathbf{c} + \tfrac{1}{2}(\mathbf{c}^2 - r^2)\mathsf{e}_\infty + \mathsf{e}_o, \quad r^2 \in \mathbb{R} \tag{3.30}$$

wobei $\mathbf{C}$ ein Punkt auf den Nullkegel ist, so gilt:

$$\mathbf{X} \cdot \mathbf{S} = \mathbf{X} \cdot \mathbf{C} - \tfrac{1}{2}\, r^2\, \mathbf{X} \cdot \mathsf{e}_\infty = -\tfrac{1}{2}(\mathbf{x} - \mathbf{c})^2 + \tfrac{1}{2}\, r^2. \tag{3.31}$$

Daraus ergibt sich

$$\mathbf{X} \cdot \mathbf{S} \Longleftrightarrow (\mathbf{x} - \mathbf{c})^2 = r^2 \Longleftrightarrow |\mathbf{x} - \mathbf{c}| = |r|.$$

Dies bedeutet, dass das innere Produkt zwischen $\mathbf{X}$ und $\mathbf{S}$ genau dann Null ist, wenn der Abstand zwischen den Punkten $\mathbf{x}$ und $\mathbf{c}$ in dem zugrunde liegenden Euklidische Raum r beträgt. Mit anderen Worten beschreibt der Nullraum von $\mathbf{S}$ bezüglich des inneren Produktes eine Hypersphäre mit dem Zentrum in $\mathbf{c}$ und dem Radius r. Da der Skalarfaktor r^2 auch negative Werte annehmen kann, ist es möglich, auch imaginäre Hypersphären darzustellen.

Das Ergebnis der Gleichung (3.31) kann benutzt werden, um festzustellen, ob ein Punkt innerhalb, außerhalb oder auf der Hypersphäre liegt:

$$\mathbf{X} \cdot \mathbf{S} = \begin{cases} > 0: & \mathbf{x} \text{ innerhalb der Hypersphäre} \\ = 0: & \mathbf{x} \text{ auf der Hypersphäre} \\ < 0: & \mathbf{x} \text{ außerhalb der Hypersphäre} \end{cases} \tag{3.32}$$

Aus der Definition einer Hypersphäre folgt, dass ein Punkt $\mathbf{X}$ auf dem Nullkegel – also $\mathbb{PK}^n$-Repräsentation eines Euklidischen Punktes) – als eine Hypersphäre mit dem Radius Null interpretiert werden kann. Andererseits stellt eine Hypersphäre mit unendlichem Radius eine Euklidische Hyperebene dar. Da $\mathbb{PK}^n$ ein homogener Raum ist, und geometrische Entitäten bis auf einen Skalarfaktor definiert sind, kann eine Hypersphäre mit unendlichem Radius mit einem endlichen Ausdruck repräsentiert werden, dessen e_o-Komponente gleich Null ist.

Eine Hyperebene $\mathbf{P}$ kann auch als Differenz von zwei Vektoren $\mathbf{A}$ und $\mathbf{B}$ auf dem Nullkegel interpretiert werden:

$$\mathbf{P} = \mathbf{A} - \mathbf{B} = (\mathbf{a} - \mathbf{b}) + \tfrac{1}{2}(\mathbf{a}^2 - \mathbf{b}^2)\mathsf{e}_\infty. \tag{3.33}$$

Das innere Produkt zwischen $\mathbf{P}$ und $\mathbf{X}$ ist durch

$$\mathbf{X} \cdot \mathbf{P} = -\tfrac{1}{2}(\mathbf{a} - \mathbf{x})^2 + \tfrac{1}{2}(\mathbf{b} - \mathbf{x})^2 \tag{3.34}$$

gegeben. Es folgt

$$\mathbf{X} \cdot \mathbf{P} = 0 \Longleftrightarrow (\mathbf{a} - \mathbf{x})^2 = (\mathbf{b} - \mathbf{x})^2 \Longleftrightarrow |\mathbf{a} - \mathbf{x}| = |\mathbf{b} - \mathbf{x}|.$$

Mit anderen Worten, der Nullraum von $\mathbf{P}$ bezüglich des inneren Produktes beschreibt eine Ebene im Euklidischen Raum, die äquidistant zu $\mathbf{a}$ und $\mathbf{b}$ ist und als Normale $\mathbf{a}-\mathbf{b}$ hat. Analog zu (3.32) lässt sich das Ergebnis von (3.34) auswerten, um die relative Position von $\mathbf{x}$ im Bezug auf die Lage der Hyperebene im Euklidischen Raum festzustellen: Das Produkt wird für Punkte, die oberhalb, auf oder unterhalb der Hyperebene liegen, positiv, Null bzw. negativ.

3.3.2 Rotationen und Translationen in $\mathbb{PK}^n$

In einer konformen geometrischen Algebra kann jede konforme Abbildung $\mathcal{O}$ als ein Versorprodukt dargestellt werden:

$$\mathbf{Y} = \mathcal{O}(\mathbf{X}) = O\mathbf{X}O^{-1}. \tag{3.35}$$

Unter einer konformen Abbildung wird dabei aus geometrischer Sicht eine winkelerhaltende Transformation verstanden. Die Gruppe der konformen Transformationen deckt unter anderem die Menge solcher einfachen geometrischen Transformationen wie Reflektionen, Inversionen, Translationen und Rotationen ab. Im Kontext dieser Arbeit ist die Gruppe der Euklidischen Transformationen $SE(n)$ besonders relevant. Da diese Gruppe als Kombination von Rotationen und Translationen modelliert wird, ist die einfache Repräsentation solcher Transformationen als lineare multiplikative Gruppe in konformer Algebra von besonderer Bedeutung.

Genauso wie in Euklidischen Räumen werden Rotationen in konformen Räumen durch Rotoren dargestellt:

$$R = \exp\left(-\frac{\theta}{2}\mathbf{U}\right), \tag{3.36}$$

wobei θ den Rotationswinkel und $\mathbf{U}$ die Rotationsachse angeben. Die eigentliche Rotation wird durch die Anwendung des Versorproduktes durchgeführt:

$$\mathbf{Y} = R\mathbf{X}\widetilde{R}.$$

Die Translation um einen Euklidischen Vektor $\mathbf{t}$, der so genannte Translator, ist gegeben durch

$$T = 1 - \tfrac{1}{2}\mathbf{t}\mathsf{e}_\infty. \tag{3.37}$$

Ein Translator ist nichts anderes als eine Rotation mit dem Zentrum im Unendlichen und kann auch in exponentieller Form dargestellt werden:

$$T = \exp\left(-\tfrac{1}{2}\mathbf{t}\mathsf{e}_\infty\right). \tag{3.38}$$

Da ein Translator ein spezieller Rotor ist, wird die Gruppe der Translationen multiplikativ und auf eine lineare Weise beschrieben. Wie jeder Rotor, kann ein Translator auch auf beliebige Elemente der Algebra wie geometrische Entitäten aber auch auf andere Rotoren angewendet werden.

Die Gruppe von Euklidischen Bewegungen wird in einer konformen geometrischen Algebra durch allgemeine Rotationen modelliert, die Rotationen um eine beliebige Achse und nicht nur um den Ursprung erlauben. Solche allgemeine Rotationen werden durch Elemente der Algebra der Form

$$M = TR\widetilde{T} = \cos\left(\frac{\theta}{2}\right) + \sin\left(\frac{\theta}{2}\right)\mathbf{U} + \sin\left(\frac{\theta}{2}\right)(\mathbf{U}\cdot\mathbf{t})\,\mathsf{e}_\infty \tag{3.39}$$

repräsentiert. Die allgemeinen Rotationen können die Gruppe der Euklidischen Transformationen SE fast vollständig darstellen. Einzig pure Translationen können bedingt durch die Art der Einbettung nicht repräsentiert werden, da eine Nullrotation durch einen Skalar (1) gegeben wird, und die Gleichung (3.39) zu einer Form ausartet, die keine Interpretation als eine Translation zulässt:

$$M = T1\widetilde{T} = T\widetilde{T} = 1.$$

3.4 Zusammenfassung

Die in diesem Kapitel gegebene Einführung bietet nur einen groben Überblick über das Potenzial von Clifford-Algebren. Grosser Wert wurde auf die Erklärung von Eigenschaften gelegt, die im Rahmen dieser Arbeit praktische Anwendung finden. Speziell sind das die Möglichkeiten, reine Rotationen in Euklidischen Räumen effizient zu berechnen bzw. Hypersphären und allgemeine Rotationen in konformen Räumen auf eine lineare Weise beschreiben zu können.

Kapitel 4

EXISTIERENDE NEURONALE MODELLE

Dieses Kapitel bietet einen Überblick über die in der Praxis häufig eingesetzten neuronalen Modelle sowie eine Einführung in das Training von neuronalen Netzen. Dabei werden insbesondere diejenigen Konzepte betont, die als Ausgangspunkt zur Entwicklung neuer Architekturen in Rahmen dieser Arbeit verwendet werden. Der Überblick umfasst das Perzeptron-Modell, SLP und MLP, RBF-Netze sowie die DCS-Architektur.

4.1 Abstraktes Modell

Die kleinsten Berechnungseinheiten eines neuronalen Netzes stellen Neuronen dar. Sie bilden mehrdimensionale Eingabedaten auf eine typischerweise eindimensionale Ausgabe ab.

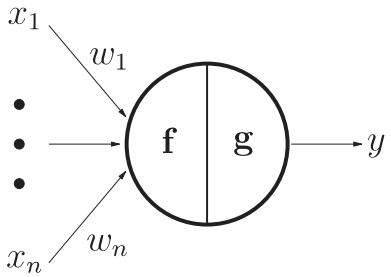


Abbildung 4.1: Generisches Neuron.

Die Berechnung wird üblicherweise in zwei Schritten durchgeführt. Zuerst werden die Eingabedaten $\mathbf{x}$ mit internen Parametern des Neurons, den so genannten Gewichten $\mathbf{w}$, mit Hilfe einer Propagierungsfunktion $\mathbf{f}$ assoziiert. Im zweiten

Propagierung	Aktivierung	Neuronentyp
Skalarprodukt	Sigmoid	Perzeptron
Skalarprodukt	Identität	Linearer Assoziator
Geometrisches Produkt	Identität	Clifford-Neuron
L_2-Norm	Gauß	RBF-Neuron

Tabelle 4.1: Neuronentyp in Abhängigkeit von gewählter Propagierungs- und Aktivierungsfunktion.

Schritt wird das Ergebnis mit der so genannten Aktivierungsfunktion g ausgewertet und ausgegeben. Die Wahl von Propagierungs- und Aktivierungsfunktionen ist ausschlaggebend für den Entwurf einer Netzarchitektur. In Tabelle 4.1 sind entsprechende Funktionen für einige bekannte Netztypen präsentiert.

Allgemein kann man ein neuronales Netz als einen gerichteten Graphen mit mehreren verbundenen Knoten – Neuronen – auffassen. Die einzelnen Knoten besitzen mehrere unabhängige Eingänge und einen Ausgang, das heißt die Information fließt in eine bestimmte Richtung. Die Kanten des Netzes sind gerichtete Informationskanäle, welche die Information von einem Neuron zum anderen transportieren. Dieser Sachverhalt wird folgendermaßen mathematisch formalisiert.

Ein neuronales Netz ist ein Tupel $\mathcal{N} = (N, E, W, \mathcal{F}, \mathcal{G}, I, O)$ mit

- N – der Menge von Neuronen,
- $E \subseteq N \times N$ – der Vernetzungsstruktur,
- W – der Menge von Gewichten,
- $\mathcal{F}$ – der Menge von Propagierungsfunktionen,
- $\mathcal{G}$ – der Menge von Aktivierungsfunktionen,
- $I \subseteq N$ und $O \subseteq N$ – den Ein- und Ausgabeneuronen.

Vorwärts gerichtete Netze werden durch weitere erfüllte Eigenschaften definiert:

- (N, E) ist ein azyklischer Graph,
- Neuronen aus I haben keine Vorgänger,
- Neuronen aus O haben keine Nachfolger.

Im Rahmen dieser Arbeit werden nur vorwärts gerichtete Netze betrachtet.

4.2 Lernen

Neuronale Netze beschreiben auf eine parametrische Weise das Modell einer zu approximierenden Funktion. Die Parameter des Modells sind als Gewichte der Neuronen gegeben. Das Lernen besteht darin, mit Hilfe eines Lernalgorithmus die zu einer gestellten Aufgabe passenden Gewichte zu finden. Die Lernalgorithmen können nach verschiedenen Kriterien klassifiziert werden. Im Kontext dieser Arbeit ist die Unterscheidung zwischen überwachtem und unüberwachtem Lernen besonders wichtig.

4.2.1 Überwachtes Lernen

Überwachtes Lernen bezeichnet eine Methode, bei der zu jeder Eingabe eine zugehörige Soll-Ausgabe gegeben ist und zur Adaption des Modells genutzt wird. Typischerweise wird überwachtes Lernen durch die Angabe einer Fehlerfunktion und eines Optimierungsverfahrens zur Minimierung der Fehlerfunktion spezifiziert. Die übliche Fehlerfunktion ist die so genannte Minkowski-P-Fehlerfunktion

$$\sum_{i=1}^{N} |V(\mathbf{x}_i) - \mathbf{y}_i|^P \,, \tag{4.1}$$

wobei $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1\cdots N}$ die Paare von Eingabe und Soll-Ausgabe, und $V(\mathbf{x}_i)$ die Ist-Ausgaben des Netzes sind. Die wichtigsten Spezialfälle der Minkowski-P-Fehlerfunktion sind die City-Block-Metrik ($P = 1$) und die quadratische Fehlerfunktion ($P = 2$).

Im Rahmen dieser Arbeit wird die quadratische Fehlerfunktion eingesetzt, da diese für Regressions- und Klassifizierungsaufgaben besonders geeignet ist: Es lässt sich zeigen, dass der quadratische Fehler einerseits genau dann minimal wird, wenn die Ist-Ausgabe dem Erwartungswert der Soll-Ausgabe entspricht, andererseits die Fehlerminimierung eine Bayes-optimale Klassifikation erlaubt [12].

Die Adaption von Parametern kann mit verschiedenen Optimierungsverfahren erfolgen. Für viele Algorithmen ist es erforderlich, dass die Fehlerfunktion differenzierbar ist, insbesondere wenn zur Optimierung gradientenbasierte Verfahren wie zum Beispiel der Gradientenabstieg eingesetzt werden.

4.2.2 Unüberwachtes Lernen

Unüberwachtes oder selbstorganisierendes Lernen wird dann eingesetzt, wenn die zu einer Eingabe gehörende Ausgabe unbekannt ist. Kennzeichen von selbstorganisierenden Verfahren ist, dass häufig lokale Informationen verarbeitet werden, um die bezüglich der Topologie des Netzes optimale Datenrepäsentation herauszufinden. Eine sinnvolle Informationsextraktion ist nur dann möglich bzw. notwendig, wenn in den Daten Redundanz enthalten ist: Ohne Redundanz sind die Daten selbst ihre beste Repräsentation.

Typische Einsatzgebiete von unüberwachtem Lernen sind solche Aufgaben, wie zum Beispiel das Lernen der Ähnlichkeit in den Daten, die Dimensionsreduktion oder auch die Suche nach der Topologie des Eingaberaumes.

Es existiert eine Vielfalt von Algorithmen für selbstorganisierendes Lernen. Gut bekannt sind unter anderem die Vektorquantisierung oder auch die Kohonenregeln [54, 73].

4.3 Perzeptron

Das Perzeptron [68] ist eine der am meisten verbreiteten neuronalen Architekturen.

4.3.1 Einfaches Perzeptron

Das Perzeptron kann aus dem generischen Neuron abgeleitet werden, indem man als Propagierung das Skalarprodukt einsetzt:

$$\begin{aligned} \mathbf{f} : \mathbb{R}^{n+1} &\longrightarrow \mathbb{R} \\ (\mathbf{x}, 1) &\longmapsto (\mathbf{x}, 1) \cdot \mathbf{w} = \sum_{i=1}^{n} x_i w_i + 1 \cdot w_{n+1}, \end{aligned} \tag{4.2}$$

wobei $\mathbf{x} = (x_1, \cdots, x_n)$ die n-dimensionale Eingabe und $\mathbf{w} = (w_1, \cdots, w_{n+1})$ der Gewichtsvektor sind, und als Aktivierung eine "treppenähnliche" Funktion spezifiziert ist. Typischerweise wählt man als Aktivierungsfunktion eine differenzierbare Funktion, oft die Sigmoidfunktion:

$$\begin{aligned} \mathbf{g} : \mathbb{R} &\longrightarrow \mathbb{R} \\ \mathbf{x} &\longmapsto (1 + e^{-\lambda \mathbf{x}})^{-1}, \quad \lambda \in \mathbb{R}_+. \end{aligned} \tag{4.3}$$

Die homogene Erweiterung des Eingaberaumes führt dazu, dass das Perzeptron flexibler wird und die ”treppenähnliche” Aktivierungsfunktion implizit nicht nur am Ursprung, sondern an jeder Position platziert werden kann. Die zusätzliche Gewichtskomponente w_{n+1} wird Bias genannt.

Die Lernaufgabe für ein Perzeptron besteht darin, die Eingabedaten auf lineare Trennbarkeit zu untersuchen. Für ein Zwei-Klassen-Problem lässt sich das folgendermaßen formalisieren: Für eine Trainingsmenge

$$\{(\mathbf{x}_i, \mathbf{y}_i) \in \mathbb{R}^n \times \{0, 1\}\}_{i=1,\cdots,N}$$

finde einen Gewichtsvektor $\mathbf{w}$, so dass

$$(\mathbf{x}_i, 1) \cdot \mathbf{w} \geq 0, \text{ falls } \mathbf{y}_i = 1 \qquad (\mathbf{x}_i, 1) \cdot \mathbf{w} < 0, \text{ falls } \mathbf{y}_i = 0.$$

Diese Aufgabe lässt auch eine Interpretation im geometrischen Sinne zu: Ein Perzeptron teilt den Eingaberaum auf lineare Weise in zwei Hälften mittels einer Hyperebene auf. Der Gewichtsvektor beschreibt dabei die Normale der Hyperebene. Das Skalarprodukt wird auf folgende Weise interpretiert:

$$(\mathbf{x}_i, 1) \cdot \mathbf{w} = \begin{cases} > 0: & \mathbf{x} \text{ in einem Halbraum} \\ = 0: & \mathbf{x} \text{ auf der Hyperebene} \\ < 0: & \mathbf{x} \text{ in anderem Halbraum} \end{cases} \tag{4.4}$$

Anders ausgedrückt: Homogen eingebettete Daten werden auf den Gewichtsvektor projiziert. Diese Projektion entspricht einer Reduktion der Dimension mit gleichzeitiger Koordinatentransformation, so dass die Daten auf eine Zahlengerade abgebildet werden. Die so transformierten Daten werden dann anhand des Vorzeichens auf der Zahlengerade getrennt. Diese Transformation ist globaler Natur, wirkt auf den gesamten Eingaberaum und ist in Literatur als Fisher-Diskriminante [32] bekannt.

4.3.2 Mehrschichtiges Perzeptron-Netz

Falls die Daten nicht linear trennbar sind, wird es mit einem einzelnen Perzeptron nicht möglich sein, die Klassifizierungsaufgabe erfolgreich zu lösen. In diesem Fall ist es notwendig, eine komplexere Funktion zur Trennung einzusetzen. Die entsprechende Modellierung kann durch Einsatz von mehreren Perzeptronen erfolgen, die parallel in einer Schicht geschaltet und gegebenenfalls in mehreren Schichten angeordnet sind. Alle Schichten bis auf die letzte heißen versteckte Schichten. Die letzte Schicht wird Ausgabeschicht genannt.

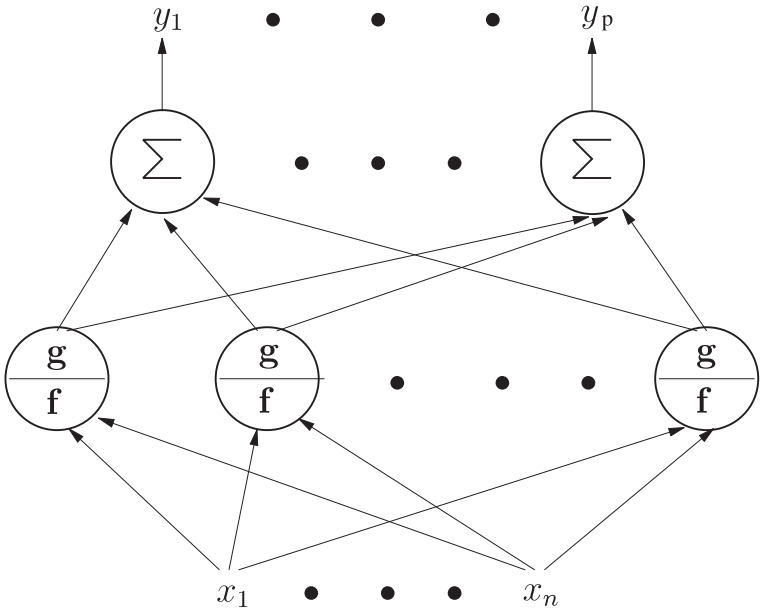


Abbildung 4.2: Multi-Layer-Perceptron.

Abbildung 4.2 veranschaulicht diese Art von Architektur. Solche Netze heißen Single-Layer-Perceptron (SLP), falls nur eine Schicht vorhanden ist, bzw. Multi-Layer-Perceptron (MLP), falls mehrere Schichten vorhanden sind. Die Berechnung wird schichtweise synchron durchgeführt.

Aus geometrischer Sicht wird in jeder Schicht eine Trennfläche berechnet, die aus mehreren Hyperebenen bzw. einzelnen Perzeptrone) auf lineare Weise zusammengesetzt ist. Es ist sogar möglich, jede stetige Hyperfläche beliebig exakt zu approximieren, da ein zweischichtiges MLP ein universeller Approximator ist [24, 50].

Im Vergleich zum einzelnen Perzeptron werden Daten nicht mehr auf eine eindimensionale Zahlengerade (einen Gewichtsvektor), sondern auf mehrere Vektoren projiziert. Die dabei entstehende Transformation kann, abhängig von der Eingabedimension und der Anzahl von Perzeptronen in der Schicht, sowohl einer Reduktion der Dimension als auch einer Einbettung in einen höherdimensionalen Raum entsprechen. Diese Transformation ist global und wirkt auf dem gesamten Raum. Im Lernprozess wird versucht, die Gewichtsvektoren so zu adaptieren, dass die transformierten Daten in der letzten Schicht linear trennbar sind.

4.3.3 Lernen in MLP

Das Lernen geschieht in MLP typischerweise mittels der Minimierung der quadratischen Fehlerfunktion. Das größte Problem dabei ist der Entwurf eines Suchalgorithmus für das globale Minimum. Die bekannteste Methode stellt ein gradientenbasiertes Verfahren, der so genannte Backpropagation-Algorithmus [91], dar. Das Grundprinzip des Backpropagation-Algorithmus liegt in der wiederholten Anwendung von Kettenregeln für partielle Ableitungen bezüglich der Gewichten von einzelnen Perzeptronen, um die Abstiegsrichtung für den Gradienten der Fehlerfunktion zu berechnen. Der Gradient der Fehlerfunktion muss also für alle Punkte des Gewichteraumes existieren, d.h. die partiellen Ableitungen müssen überall definiert sein. Die Verwendung von differenzierbaren Sigmoidfunktionen als Aktivierung genügt diesen Voraussetzungen. Ein Nachteil der sigmoidalen Aktivierung besteht darin, dass die Fehlfunktion nicht konvex ist, d.h. es entstehen lokale Minima, die bei einer Treppenfunktion nicht existent waren. Solche lokale Minima stellen für die im Rahmen dieser Arbeit eingesetzten Algorithmen eine große Herausforderung dar.

Der formelle Ablauf des Backpropagation-Verfahren für MLP mit der sigmoidalen Aktivierung wird im Algorithmus 4.1 skizziert.

Algorithmus 4.1 Backpropagation für MLP mit sigmoidaler Aktivierung

Initialisiere Gewichte $\mathbf{w}$ mit zufälligen Werten
while Wert der Fehlerfunktion groß **do**
$\Delta\mathbf{w} := 0$
for all Datenpaare $(\mathbf{x}_p, \mathbf{y}_p)$ **do**

$$
\begin{aligned}
o_i &:= \begin{cases} x_{pi} & i \text{ ist ein Eingabeneuron} \\ \mathbf{g}(\sum_{j \to i} w_{ij} o_j) & \text{sonst} \end{cases} \\
\delta_j &:= \begin{cases} (o_j - y_{pj}) o_j (1 - o_j) & j \text{ ist ein Ausgabeneuron} \\ \sum_{k \to j} w_{jk} \delta_k o_j (1 - o_j) & \text{sonst} \end{cases} \\
\Delta w_{ij} &:= \Delta w_{ij} - o_i \delta_j
\end{aligned}
$$

end for
$\mathbf{w} := \mathbf{w} + \eta \Delta \mathbf{w}$, $\eta \in \mathbb{R}_{>0}$ Lernrate
end while

Eine ausführliche Beschreibung des Ablaufes des Algorithmus kann in [25, 88] gefunden werden. Der Aufwand des Backpropagation-Algorithmus hängt von der

Anzahl der nötigen Schleifendurchläufe ab, die sich je nach gewählter Lernrate η und Situation ändern kann. Der Aufwand für eine Schleife hängt von der Anzahl der Datenpaare und der Größe des MLPs auf eine lineare Weise ab. Das Verfahren ist mit einigen Problemen behaftet, da die Form der Fehlerfunktion und die gewählte Lernrate dazu führen können, dass z.B. ein lokales Minimum gefunden wird, das globale Minimum übersprungen wird oder der Algorithmus oszilliert. Es existiert eine Reihe von Modifikationen, die das Verhalten des Algorithmus verbessern können. Einige generelle Methoden zur Verbesserung der Optimierungsabläufe können in [40, 101] gefunden werden. In dieser Arbeit wird eine modifizierte Version [65] des Algorithmus [85] benutzt.

4.4 RBF-Netze

Im Vergleich zu den MLPs können Netzwerke radialer Basisfunktionen eine größere Anzahl von Neuronen benötigen, bieten aber oft Vorteile in Bezug auf die Geschwindigkeit des Lernens [18, 69]. RBF-Netze sind vorwärts gerichtete zweischichtige Netze mit einer versteckten Schicht, an deren Ausgängen eine gewichtete Linearkombination von Projektionen auf Basisfunktionen berechnet wird.

4.4.1 Architektur

Ein RBF-Neuron wird von dem generischen Neuron abgeleitet, indem man die Propagierung als

$$\begin{aligned} \mathbf{f} : \mathbb{R}^n &\longrightarrow \mathbb{R} \\ \mathbf{x} &\longmapsto ||\mathbf{x} - \mathbf{w}||_2 \end{aligned} \tag{4.5}$$

spezifiziert und als Aktivierung eine monoton fallende, stetig differenzierbare Funktion

$$\mathbf{g} : \mathbb{R}_{\geq 0} \longrightarrow [0, 1] \qquad \text{mit } \mathbf{g}(0) = 1 \text{ und } \mathbf{g}(\infty) = 0 \tag{4.6}$$

wählt.

Die Ausgabe des Netzes wird gemäß der Evaluierungsfunktion

$$\begin{aligned} \mathbf{V} : \mathbb{R}^n &\longrightarrow \mathbb{R}^m \\ \mathbf{x} &\longmapsto \sum_{i=1}^{K} \mathbf{o}_i \mathbf{g}_i(\mathbf{f}(\mathbf{x})), \quad \mathbf{o}_i \in \mathbb{R}^m \end{aligned} \tag{4.7}$$

berechnet. Dabei gibt K die Anzahl von RBF-Neuronen an. Die Gewichte $\mathbf{w}_i$ von einzelnen RBF-Neuronen werden auch als Zentren der Basisfunktionen bezeichnet und können aus geometrischer Sicht als Positionen von Neuronen im Eingaberaum interpretiert werden. Die Vektoren $\mathbf{o}_i$ stellen die mit jedem Knoten des Netzwerkes assoziierten Ausgabevektoren dar. Es gibt eine Vielzahl von Funktionen, die den Bedingungen (4.6) für die Aktivierung genügen [45, 74, 111].

Eine oft verwendete Aktivierung (Basisfunktion) stellt die Gaussfunktion dar:

$$\mathbf{g}(\mathbf{y}) = \exp\left(-\frac{\mathbf{y}^2}{2\sigma^2}\right). \tag{4.8}$$

Für so gewählte Basisfunktion nimmt die Evaluierungsfunktion (4.7) folgende Gestalt an:

$$\mathbf{V}(\mathbf{x}) = \sum_{i=1}^{K} \mathbf{o}_i \exp\left(-\frac{||\mathbf{x} - \mathbf{w}_i||_2^2}{2\sigma_i^2}\right). \tag{4.9}$$

Der σ_i-Parameter beschreibt dabei die lokale Ausdehnung vom Einzugsbereich des i-ten RBF-Neurons und wird als Radius der Basisfunktion bezeichnet. Im Kontext dieser Arbeit werden RBF-Netze mit ausschließlich Gauss'schen Aktivierungsfunktion betrachtet.

RBF-Netze mit Gaussfunktion als Aktivierung sind universelle Approximatoren [42, 44]. Diese Eigenschaft lässt sich auf eine grosse Klasse von Basisfunktionen verallgemeinern [77]. Es lässt sich sogar für viele RBF-Netze zeigen, dass diese im Gegensatz zu MLPs die Bestapproximation-Eigenschaft erfüllen [38]. Das heißt, dass aus einer möglichen Menge von Approximierungsfunktionen RBF-Netze die Funktionen darstellen können, die den minimalen Abstand zur gesuchten Funktion bezüglich dieser Menge haben.

4.4.2 Lernen in RBF-Netzen

Die Gestalt von (4.6) und (4.7) lässt darauf schließen, dass RBF-Netze lokal sind. Es existiert eine Reihe von Algorithmen, welche diese Tatsache ausnutzen, um schnelle Trainingsverfahren zu realisieren.

Das Training findet in zwei Phasen statt. Dabei werden in einer ersten Trainingsphase die Zentren $\mathbf{w}_i$ und die Radien σ_i mittels eines unüberwachten Trainingsverfahrens gelernt. Typischerweise setzt man für diese Phase den K-Means-Algorithmus [27] ein. Dabei wird der Eingaberaum in eine feste Anzahl K von disjunkten

Teilmengen $\mathcal{L}_j$ basierend auf der Minimierung der Intra-Cluster-Varianz IVar partitioniert:

$$\text{IVar} = \sum_{j=1}^{K} \sum_{\mathbf{x}_i \in \mathcal{L}_j} ||\mathbf{x}_i - \mathbf{w}_j||^2, \quad \mathbf{w}_j = \frac{1}{|\mathcal{L}_j|} \sum_{\mathbf{x}_i \in \mathcal{L}_j} \mathbf{x}_i.$$

Algorithmus 4.2 K-Means

```
Initialisiere Gewichte w mit zufälligen Werten
while w sich ändern do
  for all Eingaben x_i do
    Bestimme j := arg min_{l∈{1,···,K}} ||x_i − w_l|| := best matching unit (bmu)
    Füge x_i zu L_j hinzu
  end for
  for k := 1 to k = K do
    w_k := 1/|L_k| Σ_{x_i∈L_k} x_i
  end for
end while
```

Eine Abwandlung dieses Algorithmus stellt der ISODATA-Algorithmus dar [106], der im Gegensatz zu K-Means auch die Anzahl von Partitionen variieren kann.

In einer zweiten Phase wird die lineare Optimierung der Ausgabeschicht durchgeführt. Ein mögliches Verfahren stellt lineare Regression dar. Mathematisch gesehen, lässt sich diese Aufgabe in Matrixform folgendermaßen darstellen:

$$\begin{pmatrix} \mathbf{g}_1(\mathbf{f}(\mathbf{x}_1)) & \cdots & \mathbf{g}_K(\mathbf{f}(\mathbf{x}_1)) \\ \vdots & & \vdots \\ \mathbf{g}_1(\mathbf{f}(\mathbf{x}_N)) & \cdots & \mathbf{g}_K(\mathbf{f}(\mathbf{x}_N)) \end{pmatrix} \begin{pmatrix} \mathbf{o}_1 \\ \vdots \\ \mathbf{o}_K \end{pmatrix} = \begin{pmatrix} \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_N \end{pmatrix},$$

wobei Vektoren $\mathbf{o}_j$ für bekannte Werte $\mathbf{g}_j(\mathbf{f}(\mathbf{x}_i))$ und $\mathbf{y}_i$ gesucht werden.

Algorithmus 4.3 fasst die beiden Schritte des Lernverfahrens zusammen:

Algorithmus 4.3 Lernen in RBF-Netzen

```
Suche mit K-Means-Algorithmus die Zentren w_i für K Neurone
for all K Neuronen do
  Berechne Radien σ_i
end for
Bestimme Vektoren o_i der Ausgabeschicht
```

Eine ausführliche Beschreibung des RBF-Lernalgorithmus kann z.B. in [10] gefunden werden.

4.5 Dynamische Zellstrukturen

Die so genannten dynamische Zellstrukturen oder DCS-Netze [12] stellen eine Verallgemeinerung von RBF-Netzen dar. Dabei wird die Architektur eines RBF-Netzes um eine Struktur erweitert, welche die Topologie des Eingaberaumes beschreiben kann. Diese Struktur ist durch die Euklidischen k-Nachbarschaften $\mathcal{N}_k$ und die topologischen k-Nachbarschaften $\mathcal{N}^k$ [67] der Neuronen im Eingaberaum gegeben:

$$\mathcal{N}_k(\mathbf{x}) := \{\mathbf{w}_i | \, 0 \leq i \leq K : \text{ord}(\mathbf{x}, i) \leq k\} \tag{4.10}$$

und

$$\mathcal{N}^k(\mathbf{x}) := \{\mathbf{w}_i | \, \exists \text{Pfad}(\mathbf{w}_i, \mathcal{N}_1(\mathbf{x})) : |\text{Pfad}(\mathbf{w}_i, \mathcal{N}_1(\mathbf{x}))| \leq k\}, \tag{4.11}$$

wobei $\text{ord}(\mathbf{x}, i)$ die Euklidische Nachbarschaftsrangfolge vom Zentrum i zur Eingabe $\mathbf{x}$ angibt: Ist $\{\mathbf{w}_{i_1}, \cdots, \mathbf{w}_{i_K}\}$ die bezüglich des Euklidischen Abstandes zu $\mathbf{x}$ aufsteigend sortierte Folge von Zentrumsvektoren, so ist

$$\text{ord}(\mathbf{x}, i_j) := j - 1.$$

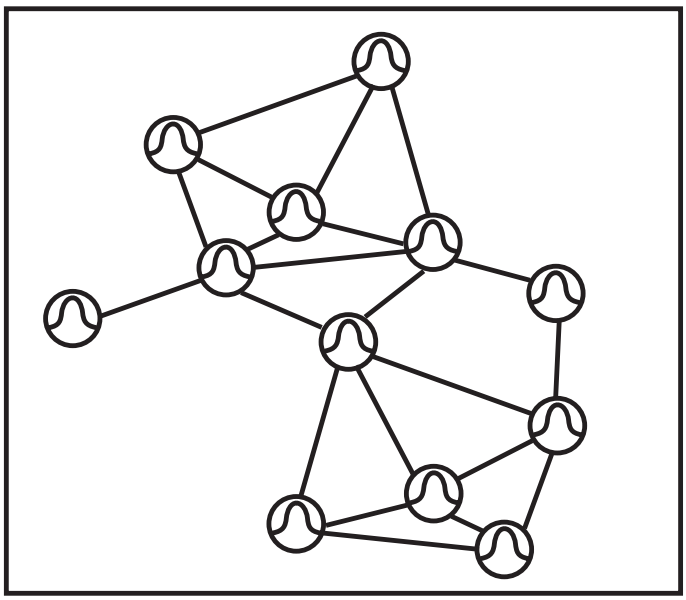


Abbildung 4.3: Die Nachbarschaftsstruktur eines DCS-Netzes gibt die Topologie des abgetasteten Eingaberaumes wieder.

Die ersten beiden Elemente in dieser Folge heißen Best Matching Unit (BMU) und Second Matching Unit (SMU). Der Aufbau einer topologischen Nachbarschaft

mit Hebbschem Wettbewerbslernen induziert eine Abtastung des Eingaberaumes auf eine optimal topologieerhaltende Weise [107, 108]. Die daraus entstehende Netzarchitektur wird optimal topologieerhaltende Karte (OTPM) genannt [12].

Die Ausgabe des DCS-Netzes wird gemäß der Evaluierungsfunktion

$$\begin{aligned} \mathbf{V} : \mathbb{R}^n &\longrightarrow \mathbb{R}^m \\ \mathbf{x} &\longmapsto \sum_{i \in \mathcal{N}^k(\mathbf{x})} \mathbf{o}_i \mathbf{g}_i(\mathbf{f}(\mathbf{x})), \quad \mathbf{o}_i \in \mathbb{R}^m \end{aligned} \tag{4.12}$$

berechnet, wobei k den Grad der topologischen Nachbarschaft angibt. Es werden also nicht alle K Neuronen aktiviert, sondern nur die im Bezug auf die topologische Nachbarschaft lokal wichtigen.

Dehnt man die Nachbarschaft auf das gesamte Netz aus, so entspricht das Netz einem RBF-Netz (vergleiche (4.7) und (4.12)). Schränkt man die Nachbarschaft auf die BMU ein, so entsteht eine Struktur, die als Neuronengas bzw. erweitertes Neuronengas [66] bekannt ist.

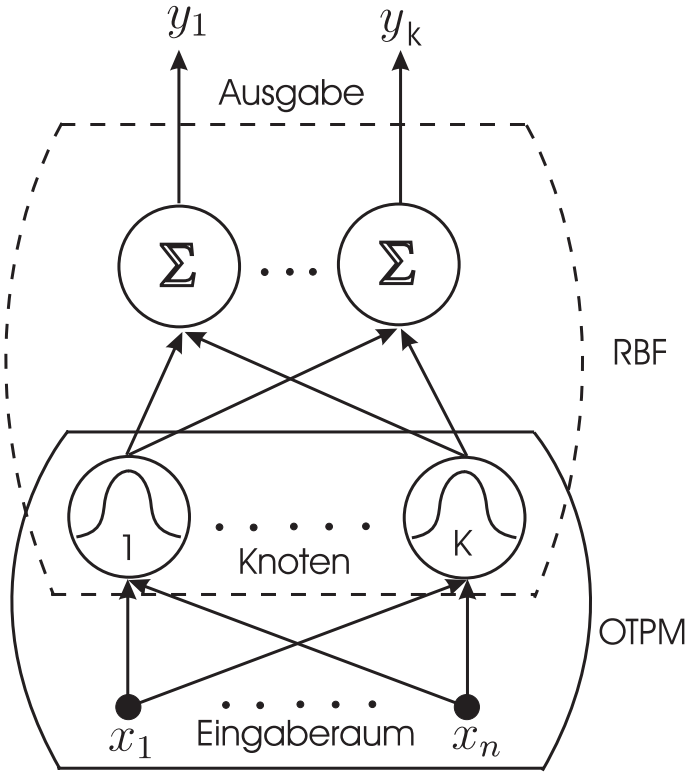


Abbildung 4.4: Dynamische Zellstrukturen kombinieren RBF-Netze mit OTPMs.

Eine typische Aktivierung eines DCS-Netzes ist durch die modifizierte Gaussfunktion der Form

$$\mathbf{g}_i(\mathbf{x}) := \begin{cases} \exp\left(-\frac{||\mathbf{x}-\mathbf{w}_i||^2}{\sigma_i^2}(-\ln(p))\right) & \mathbf{w}_i \in \mathcal{N}^1(\mathbf{x}) \\ 0 & \text{sonst} \end{cases} \tag{4.13}$$

gegeben. Dabei ist $p \in \mathbb{R}_+$ ein Skalarfaktor, der angibt, wie groß die Aktivierung des Neurons i im Zentrum der benachbarten Neuronen ist, d.h. wie stark sich die Aktivierungsfunktionen von benachbarten Neuronen überlappen.

In DCS-Netzen sind die Aktivierungsfunktionen der einzelnen Neuronen verschieden und werden an die lokale Struktur des Netzes angepasst. Um die Aktivierungsfunktionen optimiert zu gestalten, werden die Breiten der Gaussfunktionen in Abhängigkeit von der mittleren quadratischen Entfernung zu den benachbarten Neuronen berechnet:

$$\sigma_i^2 = \frac{\sum\limits_{j \in \mathcal{N}^1(\mathbf{w}_i)} ||\mathbf{w}_i - \mathbf{w}_j||^2}{|\mathcal{N}^1(\mathbf{w}_i)|}. \tag{4.14}$$

Ein DCS-Netz ist ein dynamisches System, welches während der Lernphase seine Struktur ändert. Dabei verändert sich sowohl die Anzahl von Neuronen als auch die Nachbarschaftsstruktur. Diese Eigenschaft ermöglicht dem Netz, sich der Topologie des Eingaberaumes anzupassen.

Das Lernen wird in mehreren Schritten durchgeführt. Im ersten Schritt wird das Netz mit einem Neuron initialisiert, dessen Zentrum und Ausgabevektor zufällig auf einen der Trainingsdaten gesetzt werden kann. In jedem weiteren Schritt wird dem Netz ein neues Neuron in den Einzugsbereich eines bestehenden Neurons hinzugefügt, wo der durchschnittliche lokale Trainingsfehler am größten ist. Das Zentrum des neuen Neurons wird auf das Datenpaar mit dem größten Fehler gesetzt.

Algorithmus 4.4 Einfügen eines neuen Neurons in ein DCS-Netz

for all Neuronen **do**
  Berechne den durchschnittlichen lokalen Trainingsfehler
  $F_j := \frac{1}{|\mathcal{L}_j|} \sum\limits_{(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{L}_j} |\mathbf{V}(\mathbf{x}_i) - \mathbf{y}_i|$
end for
$k := \arg\max F_j$
$l := \arg \max\limits_{(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{L}_k} |\mathbf{V}(\mathbf{x}_i) - \mathbf{y}_i|$
Füge neues Neuron mit $\mathbf{w} := \mathbf{x}_l$ und $\mathbf{o} := \mathbf{y}_l$ ein

In einem zweiten Schritt werden die Neuronen mit dem K-Means Algorithmus 4.2 neu platziert.

Danach wird die Nachbarschaftsstruktur gelernt. Dazu wird für jedes Datum $\mathbf{x}_i$ die BMU und die SMU berechnet und die entsprechenden Neuronen miteinander verbunden.

Algorithmus 4.5 Bestimmung der Nachbarschaftsstruktur im DCS-Netz

```
Lösche alte Nachbarschaftsstruktur
for all x_i do
    Berechne BMU und SMU
    Verbinde BMU und SMU
end for
```

Anschließend wird analog zum RBF-Netz die lineare Ausgabeschicht bestimmt.

Das gesamte Lernverfahren ist im Algorithmus 4.6 zusammengefasst.

Algorithmus 4.6 Lernen im DCS-Netz

```
Initialisiere ein Neuron mit zufälligen Werten
while Wert der Fehlerfunktion groß do
    Füge neues Neuron ein
    Verteile die Zentren w_i der Neuronen mit dem K-Means-Algorithmus neu
    Bestimme Nachbarschaftsstruktur
    for all K Neuronen do
        Berechne Radien σ_i
    end for
    Finde Ausgabevektoren o_i
end while
```

In [1, 12] sind weitere Modifikationen und Erweiterungen des Lernens mit DCS-Netzen ausführlich beschrieben.

4.6 Zusammenfassung

Die hier vorgestellten Netze und Lernverfahren geben nur einen kleinen Einblick in die Vielfalt der möglichen neuronalen Architekturen. Mit MLPs und RBF-Netzen sind dabei Modelle angegeben, die sowohl zwecks Approximation als auch für die Klassifizierungsaufgaben in der Praxis häufig Einsatz finden. Die Wahl einer geeigneten Netzarchitektur soll anhand vorhandenen a priori Wissens stattfinden. Ausgehend von der Gleichung (2.9) werden durch die Einschränkungen für die Art der Approximierungsfunktion $\mathbf{V}$, der Wichtungsfunktion $\mathbf{L}$ und die Anzahl K der lokalen Bereiche $\mathcal{L}$ verschiedene neuronale Strukturen entworfen.

	Perzeptron	MLP	RBF	DCS
V	$\frac{1}{1+\exp(-\lambda \mathbf{x}\cdot\mathbf{w})}$	$\sum \frac{1}{1+\exp(-\lambda \sum(\cdots))}$	$\mathbf{o} = (o_1, \cdots o_m)$	$\mathbf{o} = (o_1, \cdots o_m)$
L	1	1	$\exp\left(-\frac{\|\|\mathbf{x}-\mathbf{w}\|\|^2}{2\sigma^2}\right)$	$\exp\left(\frac{\|\|\mathbf{x}-\mathbf{w}_i\|\|^2}{\sigma_i^2}\ln(p)\right)$
K	1	1	> 1	> 1
Glob.	+	+	-	-
Dyn.	-	-	+	+
Top.	-	-	-	+

Abkürzung	Bedeutung
V	Approximierungsfunktion
L	Wichtungsfunktion
K	Anzahl von lokalen Bereichen
Glob.	Globale Wirkung
Dyn.	Dynamische Veränderung der Anzahl von Neuronen
Top.	Anpassung an die Topologie des Eingaberaumes

Tabelle 4.2: Vergleich der Eigenschaften von Perzeptron, MLP, RBF und DCS.

Tabelle 4.2 fasst die im Rahmen dieser Arbeit relevanten Eigenschaften der vorgestellten Netze zusammen.

Kapitel 5

DAS HYPERSPHÄREN-NEURON

Das in Kapitel 4 vorgestellte Perzeptron-Modell kann geometrisch interpretiert werden – die Gewichte eines Perzeptrons beschreiben eine lineare Trennfläche im Eingaberaum. Falls die Daten im Eingaberaum nicht linear trennbar sind, so sind mehrere Perzeptrone erforderlich (gegebenenfalls in mehreren Schichten angeordnet – MLP), um eine Klassifizierung zu ermöglichen. Dabei wird eine nicht lineare Trennfläche stückweise linear approximiert. Allerdings kann dabei die Lern- und Berechnungskomplexität sehr hoch ausfallen. Sind die zu lernenden Daten im Eingaberaum auf eine nichtlineare Weise verteilt, so kann es für die Erhöhung der Berechnungseffizienz hilfreich sein, eine neuronale Architektur einzusetzen, die in der Lage ist, diese Nichtlinearität zu berücksichtigen.

Durch eine spezielle Einbettung von Daten in einen konformen Raum wird es möglich, eine einfache lineare Repräsentation von Hypersphären zu erzielen und das Konzept eines klassischen Perzeptrons um hypersphärische Trennflächen zu erweitern.

5.1 Einbettung

In Kapitel 3 wurde die Einbettung eines Euklidischen Raumes in einen konformen Raum beschrieben. Insbesondere wurde in (3.30) gezeigt, dass Euklidische Hypersphären als einfache Entitäten in diesem Raum repräsentiert werden können. Weitere Entitäten wie Punkte und Flächen sind bei dieser Darstellung nur Spezialfälle einer Hypersphäre ((3.28) und (3.33)). Die geometrische Position eines Punktes im Bezug zu einer gewählten Hypersphäre wird dabei durch das innere Produkt gegeben, wie in (3.31) und (3.32) beschrieben.

Die Grundidee eines Hypersphären-Neurons ist, das vorhandene Perzeptron durch die Einbettung von Daten in höherdimensionale Euklidische Räume so zu modi-

fizieren, dass das Skalarprodukt der Propagierungsfunktion eine Hypersphäre als Trennfläche berechnet. Durch die Beschränkung der Einbettung auf Euklidische Räume wird die Möglichkeit der Anwendung von klassischen neuronalen Architekturen gesichert.

Die Propagierungsfunktion eines Hypersphären-Neurons kann als ein Skalarprodukt beschrieben werden: bettet man einen Punkt $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$ in $\mathbb{R}^{n+2}$ (*keine* konforme Einbettung) als

$$\vec{X} = (x_1, \ldots, x_n, -1, -\tfrac{1}{2}\,\mathbf{x}^2) \in \mathbb{R}^{n+2}, \quad \mathbf{x}^2 = \mathbf{x} \cdot \mathbf{x} \tag{5.1}$$

ein, so gilt:

$$\begin{aligned} \vec{X} \cdot \vec{S} &= \sum_{i=1}^{n} x_i c_i - \tfrac{1}{2}\big(\mathbf{c}^2 - r^2\big) - \tfrac{1}{2}\,\mathbf{x}^2 \\ &= -\tfrac{1}{2}\big(\mathbf{x}^2 - 2\,\mathbf{x} \cdot \mathbf{c} + \mathbf{c}^2\big) + \tfrac{1}{2} r^2 \\ &= -\tfrac{1}{2}\big(\mathbf{x} - \mathbf{c}\big)^2 + \tfrac{1}{2} r^2 \\ &= X \cdot S, \end{aligned} \tag{5.2}$$

wobei

$$\vec{S} = (c_1, \ldots, c_n, \tfrac{1}{2}(\mathbf{c}^2 - r^2), 1) \in \mathbb{R}^{n+2} \tag{5.3}$$

und

$$S = \mathbf{c} + \tfrac{1}{2}\,(\mathbf{c}^2 - r^2)\,\mathsf{e}_\infty + \mathsf{e}_o \in \mathbb{PK}^n \tag{5.4}$$

die Repräsentationen der Euklidischen Hypersphäre mit Zentrum in $\mathbf{c} = (c_1, \ldots, c_n) \in \mathbb{R}^n$ und Radius $r \in \mathbb{R}$ im Euklidischen Raum $\mathbb{R}^{n+2}$ bzw. im konformen Raum $\mathbb{PK}^n$ sind, und $X \in \mathbb{PK}^n$ die konforme Einbettung von $\mathbf{x} \in \mathbb{R}^n$ darstellt.

Wenn die vorliegenden Daten der Gleichung (5.1) entsprechend kodiert sind, so können die Komponenten des Vektors $\vec{S} \in \mathbb{R}^{n+2}$ als die Gewichte eines Perzeptrons interpretiert werden, die eine Trennhypersphäre in $\mathbb{R}^n$ beschreiben. In diesem Fall kann das Hypersphären-Neuron als ein klassisches Perzeptron mit zwei "Bias" Komponenten interpretiert werden. Eine lineare Trennung von Daten in der $\mathbb{R}^{n+2}$-Einbettung entspricht einer Trennung mit einer Hypersphäre in dem zugrunde liegenden Raum.

5.2 VC-Dimension und Trennbarkeit

Da eine Hypersphäre mit unendlichen Radius zu einer Hyperebene ausartet, die allerdings durch einen endlichen Vektor in einem konformen Raum repräsentiert werden kann, ist ein klassisches Neuron ein Spezialfall des Hypersphären-Neurons. Somit ist die VC-Dimension [2] einer Hypersphäre mindestens genau so hoch wie die von einer Hyperebene. Da eine eindimensionale Hypersphäre durch zwei Punkte repräsentiert wird, ist ihre VC-Dimension in diesem Fall drei im Gegensatz zu einer eindimensionalen Hyperebene (ein Punkt), wo die VC-Dimension zwei ist. Für die höherdimensionalen Räume sind die VC-Dimensionen von Hyperebene und Hypersphäre gleich groß (ein zweidimensionales XOR-Problem ist mit einer Hypersphäre nicht trennbar).

Allerdings ist es möglich, mit nur einem Hypersphären-Neuron Daten zu trennen, die bezüglich eines Punktes isotrop verteilt sind. Als Beispiel werden $\{\mathbf{x}_i\} \subseteq \mathbb{R}^n$ und $\{\mathbf{y}_i\} \subseteq \mathbb{R}^n$ als Daten zweier verschiedener Klassen betrachtet. Falls ein Punkt $\mathbf{c} \in \mathbb{R}^n$ existiert, so dass $\max_i |\mathbf{x}_i - \mathbf{c}| < \min_i |\mathbf{y}_i - \mathbf{c}|$ oder $\max_i |\mathbf{y}_i - \mathbf{c}| < \min_i |\mathbf{x}_i - \mathbf{c}|$ gilt, so ist die Klassifizierungsaufgabe ein intrinsisch eindimensionales Problem, und die beide Klassen können von nur einem Hypersphären-Neuron unabhängig von der Einbettungsdimension getrennt werden. Die Abbildung 5.1 veranschaulicht dieses Konzept.

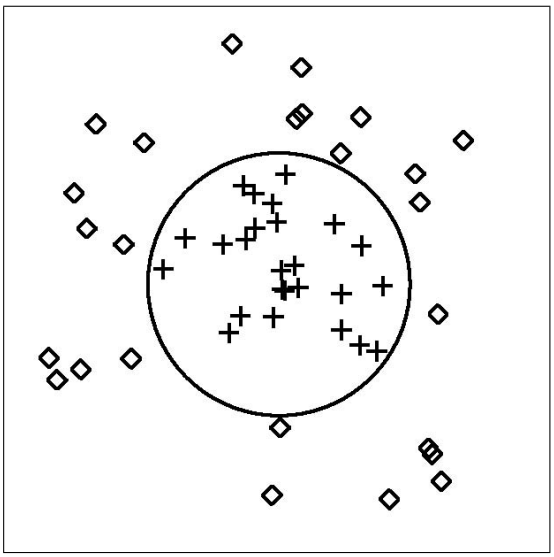


Abbildung 5.1: Eine Trennung zweier Klassen mit einem Hypersphären-Neuron ist möglich, wenn beide bezüglich der Orientierung intrinsisch eindimensional sind.

Es ist möglich, ein kompaktes n-dimensionales Gebiet schon mit nur einem Hypersphären-

Neuron abzudecken. Dagegen benötigt man mindestens $n + 1$ Perzeptrone für denselben Zweck. Im Allgemeinen sind weniger Hypersphären-Neuronen als Perzeptrone notwendig, um kompakte Klassen zu trennen. Ein mehrschichtiges Netz mit Hypersphären-Neuronen versucht, die Daten in solche orientierungsinvariante Cluster aufzuteilen.

Die Abbildung 5.2 gibt ein Beispiel an. Es wird dabei eine zweidimensionale Klassifizierungsaufgabe mit einem zweischichtigen Netz gelöst. Die versteckte Schicht besteht aus drei Hypersphären-Neuronen (Kreise, in der Abbildung teilweise nur ausschnittsweise zu sehen). Die Ausgabeschicht besteht aus einem dreidimensionalen Hypersphären-Neuron (Sphäre). Um den dargestellten kompakten Bereich (durch Kreuze repräsentiert) einzuhüllen, werden mindestens drei Kreise oder vier Geraden benötigt. Die Daten werden von der versteckten Schicht eines auf den Hypersphären-Neuronen basierenden Netzes in einen dreidimensionalen Raum abgebildet, wo eine im konformen Raum lineare Trennung mit einer Kugel stattfindet.

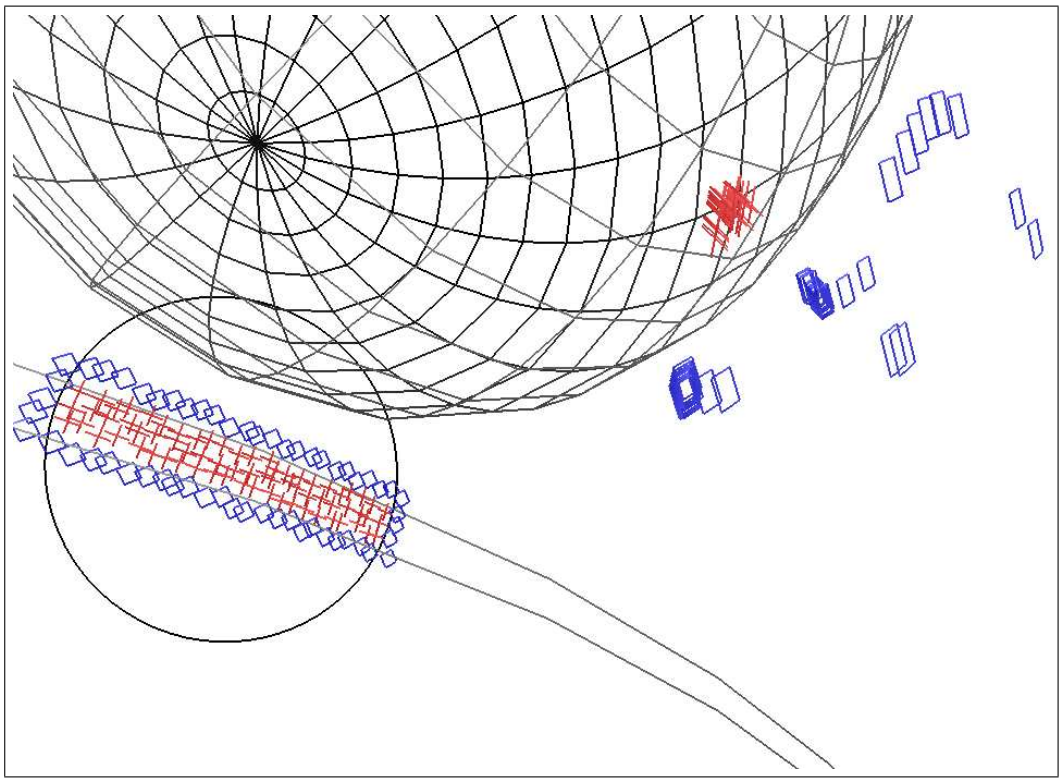


Abbildung 5.2: Klassifizierung mit Hypersphären-Neuronen.

Eine weitere Eigenschaft dieser Architektur folgt unmittelbar aus der Tatsache, dass ein Hypersphären-Neuron eine Verallgemeinerung eines Perzeptrons ist. Da

mehrschichtige Perzeptrone universelle Approximatoren sind [24, 50], sind auch vorwärts gerichtete Netze mit Hypersphären-Neuronen universelle Approximatoren.

5.3 Wahl der Parameter der Aktivierungsfunktion

Eine Hypersphäre trennt den eingebetteten Raum in zwei Klassen: $\mathcal{I}$ und $\mathcal{O}$ für die Punkte, die innerhalb und außerhalb dieser Hypersphäre liegen. Aus Gleichung (3.31) folgt, dass die Größe des Skalarproduktes zwischen einer normalisierten Hypersphäre und einem Punkt aus $\mathcal{I}$ durch den Radius der Hypersphäre begrenzt ist. Dagegen sind die Werte für die außerhalb liegenden Punkte nicht begrenzt. Da das Ergebnis des Skalarproduktes die Eingabe für die Aktivierungsfunktion ist, beeinflusst die Wahl der Aktivierungsfunktion die Größe des Radius von der Hypersphäre. Allerdings werden die Gewichte eines Hypersphären-Neurons als unabhängige Werte betrachtet – die Hypersphäre ist bis auf einen Skalierungsfaktor definiert. Da dieser Skalierungsfaktor nicht beschränkt ist, kann das Ergebnis des Produktes für die Klasse $\mathcal{I}$ betragsmäßig beliebig große Werte annehmen.

Dieses Verhalten wird im Folgenden beispielhaft verdeutlicht. Sei X die konforme Einbettung des Punktes $\mathbf{x}$, S eine Hypersphäre mit dem Radius r und Zentrum $\mathbf{c}$ und $\mathbf{g}(\lambda, z) = (1 + e^{-\lambda z})^{-1}$ die sigmoidale Aktivierung. Der Punkt $\mathbf{x}$ gehört zur Klasse $\mathcal{I}$, wenn die Werte für $\mathbf{c}$ und r so gewählt sind, dass $\mathbf{g}(\lambda, \vec{X} \cdot \vec{S}) > 1 - \varepsilon$ für den gegebenen Schwellwert $\varepsilon \in \mathbb{R}_{\geq 0}$ ist. Soll $\mathbf{x}$ außerhalb der Hypersphäre liegen, so gilt $\mathbf{g}(\lambda, \vec{X} \cdot \vec{S}) < \varepsilon$. Die Umformung der beiden Ungleichungen ergibt

$$r^2 > \frac{2}{\lambda} \ln \frac{1-\varepsilon}{\varepsilon} + (\mathbf{c} - \mathbf{x})^2 \quad \text{falls} \quad \mathbf{x} \in \mathcal{I}, \tag{5.5}$$

$$r^2 < \frac{2}{\lambda} \ln \frac{\varepsilon}{1-\varepsilon} + (\mathbf{c} - \mathbf{x})^2 \quad \text{falls} \quad \mathbf{x} \in \mathcal{O}. \tag{5.6}$$

Für ein festes $\varepsilon > 0$ und $\mathbf{c}$ hängt der Radius der Hypersphäre hängt von der Wahl des Parameters λ ab. Je kleiner λ ist, desto größer ist die Hypersphäre. Im Allgemeinen heißt das, dass man "flachere" Aktivierungsfunktion wählen soll, um den Radius einer Hypersphäre zu vergrößern.

5.4 Konfidenzen

Es ist bei einer konformen Einbettung möglich, ein Maß für die Zuverlässigkeit der Eingabedaten zu definieren. Diese Konfidenz stellt eine Art von a priori Wis-

sen dar. Für einen gegebenen Punkt $\mathbf{x}$ mit der Konfidenz r_{konf} sieht die Einbettung in den konformen Raum folgendermaßen aus:

$$X_{\text{konf}} = \mathbf{x} + \tfrac{1}{2}\left(\mathbf{x}^2 + r_{\text{konf}}^2\right)\mathsf{e}_\infty + \mathsf{e}_o. \tag{5.7}$$

Ein Vergleich zwischen den Gleichungen (3.30) und (5.7) verdeutlicht, dass aus geometrischer Sicht diese Einbettung einer Hypersphäre mit einem imaginären Radius entspricht. Die entsprechende Einbettung in $\mathbb{R}^{n+2}$ lautet

$$\vec{X}_{\text{konf}} = \left(x_1, \ldots, x_n, -1, -\tfrac{1}{2}(\mathbf{x}^2 + r_{\text{konf}}^2)\right) \in \mathbb{R}^{n+2}. \tag{5.8}$$

Das Produkt des mit einer Konfidenz versehenen Punktes X_{konf} mit einer Hypersphäre S liefert einerseits

$$S \cdot X_{\text{konf}} = \tfrac{1}{2}\Big(r^2 - \left((\mathbf{c} - \mathbf{x})^2 + r_{\text{konf}}^2\right)\Big). \tag{5.9}$$

Andererseits kann das Produkt zwischen einer Hypersphäre und einem Punkt gemäß Gleichung (3.31) als eine skalierte und mit Vorzeichen versehene Distanz zwischen dem Punkt und der Oberfläche der Hypersphäre auf folgende Weise geometrisch interpretiert werden:

$$S \cdot X = \begin{cases} > 0, & \text{falls } \mathbf{x} \text{ innerhalb der Hypersphäre liegt} \\ = 0, & \text{falls } \mathbf{x} \text{ auf der Hypersphäre liegt} \\ < 0, & \text{falls } \mathbf{x} \text{ außerhalb der Hypersphäre liegt.} \end{cases} \tag{5.10}$$

Falls der Abstand zwischen dem Punkt $\mathbf{x}$ und dem Zentrum der Hypersphäre $\mathbf{c}$ genau dem Radius der Hypersphäre r entspricht, d.h. der Punkt auf der Oberfläche der Hypersphäre liegt, so ist das Ergebnis der Gleichung (5.9) durch die Beziehung $-r_{\text{konf}}^2 < 0$ gegeben. Laut Gleichung (5.10) bedeutet dies, dass der Punkt schon außerhalb der Hypersphäre liegen soll – im Allgemeinen scheint der Punkt $\mathbf{x}$ weiter von dem Zentrum der Hypersphäre $\mathbf{c}$ zu liegen, als es tatsächlich der Fall ist. Dies hat Auswirkungen auf den Verlauf des Lernens. Der Trainingsalgorithmus versucht, den Punkt weiter von der Trennfläche weg zu platzieren.

Dieses Prinzip ist in der Abbildung 5.3 veranschaulicht. Dabei sind die Daten zweier verschiedener Klassen durch kreuzförmige und quadratische Punkte dargestellt. Die schwarzen und grauen Kreise repräsentieren die Hypersphären bzw. die Konfidenzen. Das linke Bild zeigt die Lage der Entscheidungshypersphäre für die Punkte mit gleicher Konfidenz. Nach dem Erhöhen der Konfidenz für den linken unteren inneren Punkt bewegt sich die Entscheidungshypersphäre so, dass der betroffene Punkt näher zum Ursprung der Hypersphäre liegt.

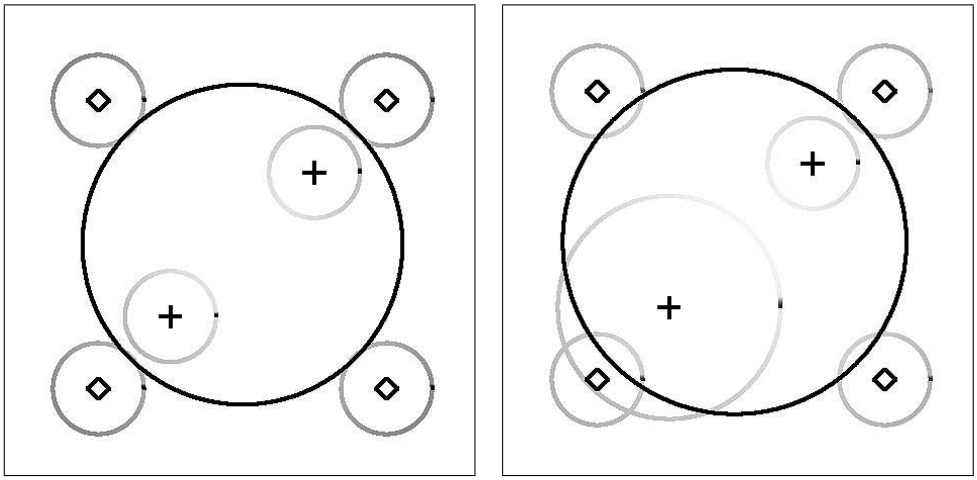


Abbildung 5.3: Die Platzierung der Hypersphäre (schwarzer Kreis) kann durch die Konfidenzen (graue Kreise) beeinflusst werden.

Betrachtet man das Produkt der imaginären Hypersphäre X_{konf} mit der Hypersphäre S gemäß Gleichung (5.9) als Produkt zwischen einem Punkt $\mathbf{y}$ und der Hypersphäre S gemäß Gleichung (3.31) so gilt:

$$\begin{aligned} S \cdot X_{\text{konf}} &= S \cdot Y \\ \tfrac{1}{2}\Big(r^2 - \big((\mathbf{x}-\mathbf{c})^2 + r_{\text{konf}}^2\big)\Big) &= -\tfrac{1}{2}(\mathbf{y}-\mathbf{c})^2 + \tfrac{1}{2}\,r^2 \\ (\mathbf{y}-\mathbf{c})^2 &= (\mathbf{x}-\mathbf{c})^2 + r_{\text{konf}}^2. \end{aligned} \tag{5.11}$$

Die Lage des Punktes $\mathbf{y}$ – und somit der Abstand zwischen dem Punkt und der Hypersphäre S – hängt nicht nur vom Radius r_{konf} der imaginären Hypersphäre X_{konf} ab, sondern auch von der Lage des Zentrums $\mathbf{c}$ der Hypersphäre S. Da $\mathbf{c}$ in der Lernphase verändert wird, ist die Konfidenz, die durch r_{konf} gegeben ist, nur ein qualitatives aber kein quantitatives Maß. Mit anderen Worten, höhere Konfidenz garantiert, dass ein Punkt weiter von der Hypersphärenoberfläche entfernt ist, macht aber keine Aussage über den Abstand zu dieser.

Mit Konfidenzen ist es unter anderem möglich, das Verhalten von Entscheidungsoberflächen in den Grenzbereichen zwischen verschiedenen Klassen zu steuern. Durch das Erhöhen von Konfidenzen von Grenzpunkten kann man die Trennung verschärfen.

Dieses Prinzip ist in der Abbildung 5.4 veranschaulicht. Im linken Bild ist ein Fall dargestellt, bei dem einige wenige Punkte einer Klasse innerhalb einer anderen Klasse eingeschlossen sind. Erhöht man die Konfidenzen für die eingeschlossenen Punkte, so wird die grosse Klasse in zwei nicht zusammenhängende Bereiche aufgeteilt (rechtes Bild).

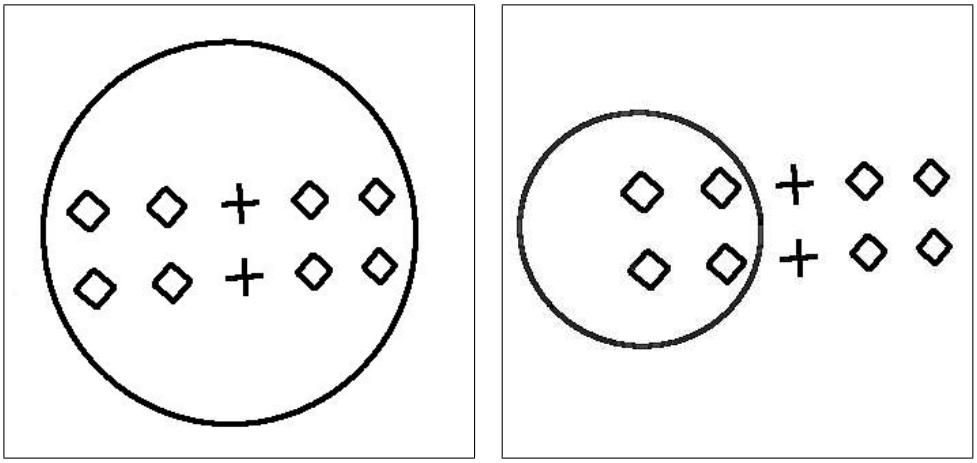


Abbildung 5.4: Der Verlauf der Trennoberfläche kann mit Konfidenzen gesteuert werden.

5.5 Berechnungskomplexität

Durch die Einbettung der Daten gemäß Gleichung (5.1) erhöht sich die Berechnungskomplexität im Vergleich zum klassischen Perzeptron. Die Eingabe $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$ wird als $\vec{X} = (x_1, \ldots, x_n, -1, -\frac{1}{2}\,\mathbf{x}^2) \in \mathbb{R}^{n+2}$ kodiert. Die letzte Komponente von $\vec{X}$ hängt von den ersten n Komponenten ab. Sie kann als Skalarprodukt zwischen zwei n-dimensionalen Vektoren beschrieben werden:

$$\mathbf{x}^2 = \mathbf{x} \cdot \mathbf{x}. \tag{5.12}$$

Es ist also notwendig, zwei Skalarprodukte von n-dimensionalen Vektoren auszuwerten – einmal für die Berechnung von $\mathbf{x}^2$ und ein zweites mal für die Auswertung der Propagierungsfunktion. Dies entspricht der Verdoppelung der Anzahl von Operationen im Vergleich zum klassischen Perzeptron. Besteht das Netz aus mehreren Neuronen, so muss diese Berechnung nur einmal pro Schicht gemacht werden. Bezüglich der Berechnungskomplexität entspricht dies dem Einfügen von je einem klassischen Perzeptron in jede Schicht.

5.6 Lernen

Die Architektur eines Multi-Layer-Hypersphere-Perceptron (MLHP) entspricht vollständig der Struktur eines MLP. Das Lernen geschieht auf dieselbe Weise wie in einem MLP – über einen Backpropagation-Algorithmus.

Sei $\vec{X}^l = (x_1^l, \ldots, x_{N^l+2}^l)$ die Ausgabe eines MLHP in der l-ten Schicht, wobei N^l die Nummer des Neurons in der l-ten Schicht ist (Bias-Komponenten werden nicht gezählt). Die Eingabeschicht sei mit Null nummeriert. Weiterhin seien $\vec{W}_p^l = (w_{p1}^l, \ldots, w_{p(N^l+2)}^l)$ die Gewichte des p-ten Neurons in der l-ten Schicht. Das Ergebnis der Evaluierung der Propagierungsfunktion für das p-te Neuron in der l-ten Schicht sei mit $s_p^l = \vec{W}_p^l \cdot \vec{X}^{l-1}$ bezeichnet. Die Ausgabe des Neurons sei

$$x_p^l = \mathbf{g}(s_p^l), \tag{5.13}$$

wobei g die Aktivierungsfunktion ist.

Es werden zwei Arten von Ableitungen benötigt, um ein MLHP mit dem Backpropagation-Algorithmus zu trainieren: Die Ableitung für ein Neuron bezüglich seiner Gewichte und die Ableitung für ein Neuron bezüglich der Ausgabe der vorhergehenden Schicht. Alle benötigten Terme können aus diesen beiden Arten gebildet werden. Die erste Art sieht folgendermaßen aus:

$$\partial_{w_{pi}^l} x_p^l = \mathbf{g}'(s_p^l)\, x_i^{l-1}, \tag{5.14}$$

wobei g′ die Ableitungen der Aktivierungsfunktionen g sind. Die zweite Art von Ableitungen tritt auf, wenn diese bezüglich der Gewichte in der vorhergehenden Schicht gesucht werden:

$$\partial_{w_{qj}^{l-1}} x_p^l = \left(\partial_{x_q^{l-1}} x_p^l\right) \left(\partial_{w_{qj}^{l-1}} x_q^{l-1}\right). \tag{5.15}$$

Die Auswertung der ersten partiellen Ableitung ergibt

$$\partial_{x_q^{l-1}} x_p^l = \mathbf{g}'(s_p^l) \left(w_{pq}^l - \underbrace{w_{p(N^l+2)}^l\, x_q^{l-1}}_{\text{zusätzlicher Term}}\right). \tag{5.16}$$

Der zusätzliche Term entsteht durch die quadratische Komponente von $\vec{X}^{l-1}$.

Folgende Kürzel können benutzt werden, um alle benötigten Ableitungen zu beschreiben:

$$\begin{aligned} \partial_{w_{pi}^l} x_p^l &= d_{pi}^l \\ \partial_{w_{qj}^{l-1}} x_p^l &= c_{pq}^l\, d_{qj}^{l-1} \end{aligned} \tag{5.17}$$

Damit kann man zum Beispiel folgende Ableitung auf einfache Weise ausdrücken:

$$\partial_{w_{rk}^{l-2}} x_p^l = \sum_{q=1}^{N^{l-1}} \left(c_{pq}^l\, c_{qr}^{l-1}\right) d_{rk}^{l-2}.$$

In vielen Fällen ist es allerdings günstiger, die quadratische Komponente als eine unabhängige Komponente zu betrachten, da dadurch die zu minimierende Fehlerfunktion glatter wird, was sich auf die Stabilität und die Konvergenzgeschwindigkeit positiv auswirkt. Die Netze, die nach diesem Prinzip trainiert werden, heißen Multi-Layer-Hypersphere-Perceptron with Simplified Derivatives (MLHPS).

5.7 Das Hypersphären-Neuron als RBF-Berechnungseinheit

Wählt man als Aktivierungsfunktion die Sigmoidfunktion $\mathbf{g}(z, \lambda) = (1+e^{-\lambda z})^{-1}$, so lässt sich die Ausgabe o eines Hypersphären-Neurons unter der Berücksichtigung der Gleichungen (3.31) und (5.13) folgendermaßen aufschreiben:

$$o = \mathbf{g}(\vec{X} \cdot \vec{S}) = \mathbf{g}(-(\pm\tfrac{1}{2}(\|\mathbf{x}-\mathbf{c}\|_2^2 - r^2))) = \left(1 + e^{\mp\lambda'(\|\mathbf{x}-\mathbf{c}\|_2^2 - r^2)}\right)^{-1} \quad (5.18)$$

Das Ergebnis der obigen Gleichung ist eine isotrope Gauss-ähnliche Funktion, die ein Extremum erreicht, wenn $\|\mathbf{x}-\mathbf{c}\| = 0$ ist und sich asymptotisch bei $\|\mathbf{x}-\mathbf{c}\| \to \infty$ verhält.

Da die Hypersphäre – das heißt Zentrum und Radius – bis auf einen Skalarfaktor definiert wird, ist das Vorzeichen in dem Sigmoidfunktion-Innenterm nicht festgelegt, sondern wird in der Trainingsphase gelernt – die Hypersphäre kann sowohl einschließend als auch ausschließend sein. Das heißt, dass das Extremum nicht ausschließlich ein Maximum, sondern auch ein Minimum sein kann. Bildlich gesprochen bedeutet dies, dass nicht nur eine Gauss-ähnliche "Glocke" sondern auch eine "umgekippte Glocke" definiert werden kann. In einem MLHP können gleichzeitig beide Arten auftreten. Mögliche Formen der Aktivierungsfunktion sind in Abbildung 5.5 veranschaulicht.

Weiterhin werden im Gegensatz zu einem RBF-Netz nicht nur die Zentren, sondern auch die Radien, die der Varianz der Gaussfunktion entsprechen, gelernt. Dies ist eine Eigenschaft, die weittragende Konsequenzen nach sich zieht. Konstruktions- und lerntechnisch bedingt überlappen sich die Einzugbereiche in einem RBF-Netz nur leicht. In einem MLHP kann dagegen eine Hypersphäre mehrere weitere Hypersphären umschließen oder auch nur teilweise schneiden. Dies zusammen mit der Tatsache, dass in einem MLHP mehrere verschiedene Hypersphären gleichzeitig auftreten können (einschließend/ausschließend, reellwertig/imaginär, Punkt/Hyperebene), verhilft dieser Architektur zu einer größeren Flexibilität bei der Modellierung der Trennoberfläche im Vergleich zu einem RBF-Netz bei gleicher Anzahl von Neuronen.

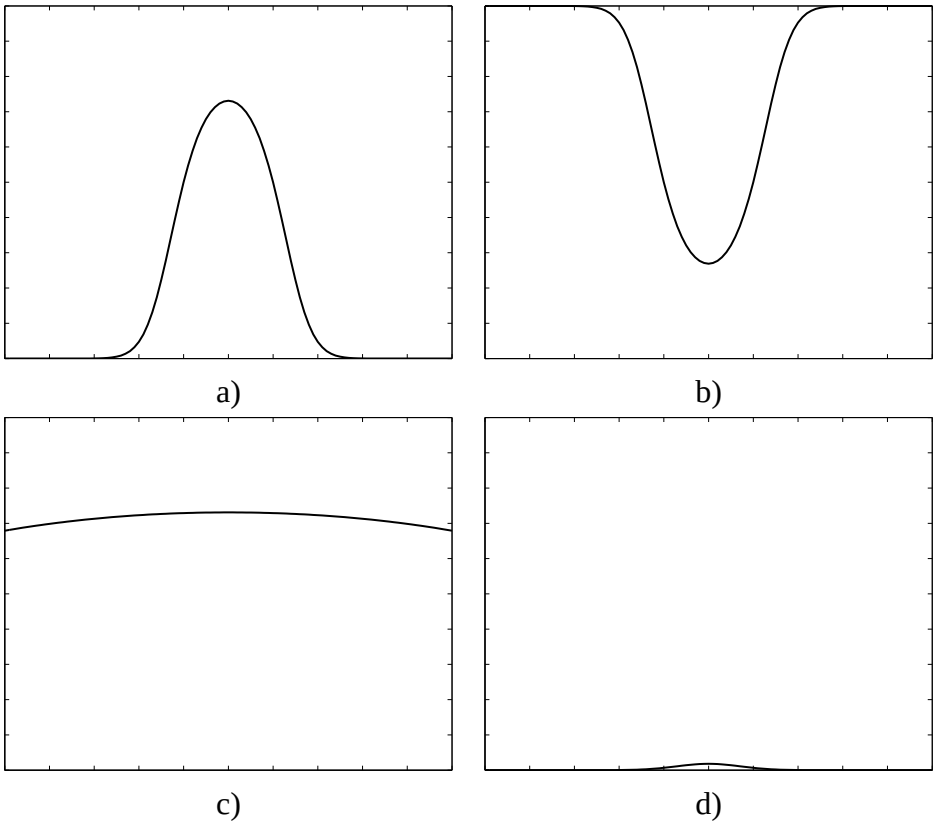


Abbildung 5.5: Aktivierungsfunktion eines Hypersphären-Neurons: Ausgabe über der Aeingabe. a) – Die Aktivierung kann eine Gauss-ähnliche Form annehmen. b) – Bei einer negativen Skalierung von Hypersphären wird die Funktion invertiert. c) – Bei großen Radien wirkt die Aktivierung global. d) – Imaginäre Radien lassen die Funktion verschwinden.

In einem klassischen RBF-Netz wird nur die versteckte Schicht mit lokalen rezeptiven Feldern beschrieben, die Ausgabeschicht entspricht einem global wirkenden SLP. Im MLHPs hingegen können alle Schichten gleich gut sowohl lokal als auch global agieren. Ein MLHP ist also auch eine Verallgemeinerung eines RBF-Netzes mit gleicher Anzahl von Neuronen.

Da ein MLHP an sich ein globales Netz ist, verhält es sich weniger anfällig gegenüber dem ”curse of dimensionality” als ein ausschließlich lokal wirkendes RBF-Netz.

Ein Vorteil der RBF-Netze ist dagegen, dass im Vergleich zur starren Architektur eines MLHP die Architektur eines RBF-Netzes dynamisch ist, d.h. die Anzahl

	MLHP	MLP	RBF
Globale Aktivierung	+	+	-
Lokale Aktivierung	+	-	+
Dynamische Anpassung der Anzahl von Neuronen	-	-	+
Dynamische Anpassung der Aktivierung	+	-	-
Geringer Einfluss von ”curse of dimensionality”	+	+	-

Tabelle 5.1: Vergleich der Eigenschaften von MLHP, MLP und RBF.

von Neuronen eines RBF-Netzes kann sich während der Lernphase ändern.

In der Tabelle 5.1 sind die Eigenschaften von MLHP, MLP und RBF zusammengefasst.

5.8 Experimente

In diesem Abschnitt wird anhand einiger Klassifikationsprobleme die Lernfähigkeit von MLHPs demonstriert und mit der anderer Netztypen verglichen.

5.8.1 Iris-Benchmark

Im ersten Experiment ist die Leistung eines SLHP[1] anhand von Fisher's Iris-Datensatzes [32] untersucht worden. Dieser Datensatz besteht aus 150 vierdimensionalen Vektoren, die in drei Klassen aufgeteilt sind. Jede Klasse enthält 50 Daten, die Merkmale von verschiedenen Typen der Iris-Pflanze beschreiben. Eine der Klassen ist von den beiden anderen linear trennbar. Bei den restlichen Klassen besteht teilweise eine Überlappung [52]. Die Daten werden zufällig in zwei Mengen aufgeteilt: Die Trainingsmenge besteht aus 39 Vektoren, die restlichen 111 Vektoren bilden die Testmenge.

Die Kodierung der Ausgabe ist auf zwei verschiedene Arten beschrieben. In der ersten Konfiguration (C1) sind die Ausgaben als dreidimensionale binäre Vektoren $(1,0,0)$, $(0,1,0)$ und $(0,0,1)$ kodiert. In der zweiten Konfiguration (C2) ist die Kodierung $(1,0)$, $(0,1)$ und $(1,1)$ gewählt. Sowohl SLHP und SLP als auch RBF und DCS wurden auf beiden Konfigurationen getestet.

[1] Single-Layer-Hypersphere-Perceptron

Tabelle 5.2 und Tabelle 5.3 geben die Ergebnisse dieser Tests wieder (Anzahl der falsch klassifizierten Daten).

Netztyp	Anzahl Neuronen	Anzahl Gewichte	Trainingsdaten (C1)	Testdaten (C1)
SLHP	3	18	0	7
SLP	3	15	0	2
RBF3	3	15	2	11
RBF10	10	70	0	2
DCS3	3	15	2	10
DCS10	10	70	0	2

Tabelle 5.2: Iris-Klassifizierung: Die Klassen sind als dreidimensionale binäre Vektoren $(1,0,0)$, $(0,1,0)$ und $(0,0,1)$ kodiert. Die Tabelleneinträge geben die Anzahl der falsch klassifizierten Daten wieder.

Netztyp	Anzahl Neuronen	Anzahl Gewichte	Trainingsdaten (C2)	Testdaten (C2)
SLHP	2	12	0	7
SLP	2	10	9	31
RBF2	2	10	10	20
RBF10	10	60	0	2
DCS2	2	10	9	20
DCS10	10	60	0	2

Tabelle 5.3: Iris-Klassifizierung: Die drei Klassen sind als zweidimensionale Vektoren $(1,0)$, $(0,1)$ und $(1,1)$ kodiert. Die Tabelleneinträge geben die Anzahl der falsch klassifizierten Daten wieder.

Eine gesonderte Rolle nimmt ein Vergleich zwischen dem SLHP und einem High-Order-Neuron (HON), wie in [60] vorgestellt, ein. Bei diesem Verfahren handelt es sich um ein Netz, das irreguläre Bereiche in Eingaberaum mit Hilfe der Hebbschen Lernregeln segmentieren kann. Die Anzahl der Neuronen in solchen Netzen entspricht dabei der Anzahl der zu segmentierenden Klassen. Die Anzahl der Gewichte hängt von der Eingabedimension sowie von der gewählten Ordnung des Neurons ab. Entsprechend dem in [60] vorgestellten Testverfahren sind die Daten in 120 Trainingsmuster und 30 Testmuster aufgeteilt. In Tabelle 5.4 sind die Vergleichsergebnisse zusammengefasst.

Die Ergebnisse der Klassifizierung zeigen, dass SLHPs im Durchschnitt die beste Stabilität und Generalisierungsfähigkeit unter den getesteten Netztypen bei glei-

Netztyp	Anzahl Neuronen	Anzahl Gewichte	Fehlklassifizierung	
			Gemittelt	Kleinste
SLHP	3	18	7.9	0
HON 2-ter Ordnung	3	75	3.08	1

Tabelle 5.4: Iris-Klassifizierung: SLHP mit 3 Neuronen vs. HON 2-ter Ordnung – Anzahl von falsch klassifizierten Daten über die gesamte Menge (Trainings- und Testdaten). Gemittelte Fehlklassifizierung und kleinste Fehlklassifizierung sind angegeben für 250 Wiederholungen.

cher Anzahl von Neuronen in diesem Benchmark aufweist. Die Lerngeschwindigkeit aller getesteten Netztypen lag in der gleichen Größenordnung von wenigen Sekunden. Die Berechnungen wurden an einem Pentium III 700MHz System durchgeführt. Alle im Rahmen dieser Arbeit entwickelte Verfahren wurden als C++-Klassen implementiert.

5.8.2 2-Spiralen-Benchmark

In diesem Experiment wurden die Eigenschaften des MLHP mit dem 2-Spiralen-Benchmark [31] untersucht. Die Abbildung 5.6 zeigt die Daten für das 2-Spiralen-Problem. Die Spiralen bestehen aus 192 Punkten in einer Ebene, und gehören zu zwei ineinander verwobenen Spiralen, die sich drei mal um den Ursprung drehen. Das Ziel in diesem Test ist, die beiden Spirale zu trennen. Diese Aufgabe ist für viele vorwärts gerichtete Netze nur sehr schwer lösbar [31, 55, 112]

Abbildung 5.7 zeigt, dass die Lernfähigkeit eines Netzes mit Hypersphären-Neuronen für die hier getesteten Daten viel höher als die eines MLP ist. Auch unter der Berücksichtigung der Berechnungskomplexität – ein zweischichtiges MLHP mit n Neuronen ist von der Berechnungskomplexität äquivalent zu einem zweischichtigen MLP mit $n + 2$ Neuronen – lernt ein MLHP besser als ein MLP. Zum Beispiel liefert ein MLHP mit 4 Neuronen in der versteckten Schicht (20 Gewichte) bessere Ergebnisse als ein MLP mit 6 Neuronen (21 Gewichte).

Die Generalisierungsfähigkeit beider getesteten Netztypen entspricht den jeweiligen Ergebnissen in der Lernphase.

MLHPs und MLHPSs unterscheiden sich in den Ableitungsregeln für den Backpropagation-Algorithmus. Das MLHPS benutzt vereinfachte Ableitungen unter der Annahme, dass die quadratische Komponente des Eingabevektors unabhän-

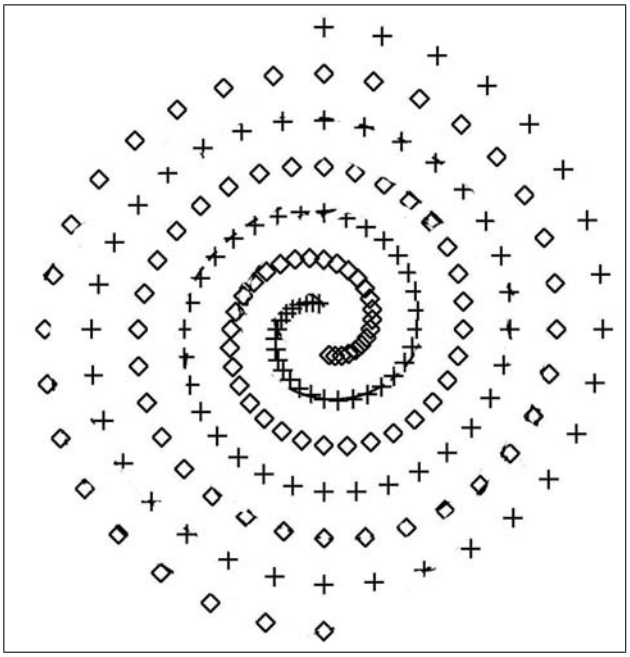
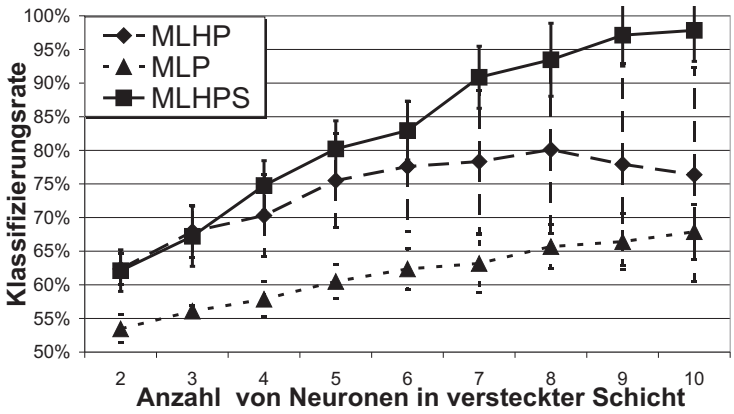


Abbildung 5.6: Die Daten für den 2-Spiralen-Benchmark.

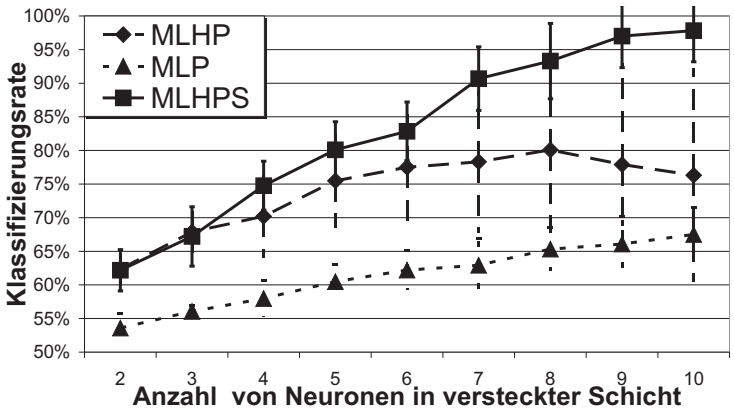
gig ist. Diese vereinfachten Ableitungen entsprechen zwar nicht exakt der besten Richtung auf der Minimierungsoberfläche der Fehlerfunktion, stabilisieren aber die Ergebnisse der Minimierung. Die Standardabweichungen in der Abbildung 5.7 geben eine Auskunft über die Stabilität des jeweiligen Lernalgorithmus. Die Konvergenz des Algorithmus auf einer ”geglätteten” Minimierungsoberfläche wird auch beschleunigt und ist vergleichbar mit der Lerngeschwindigkeit eines MLP (siehe Abbildung 5.8).

Es ist möglich, schon mit 8 Hypersphären-Neuronen in der versteckten Schicht eine fehlerfreie Trennung der Spiralen zu erzielen (Abbildung 5.9).

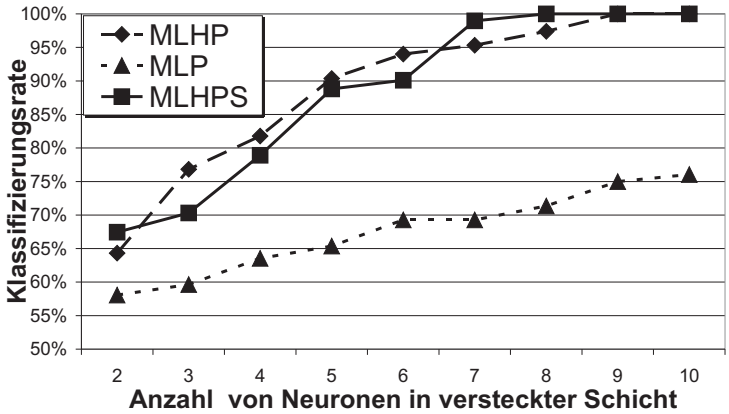
Zum weiteren Vergleich wurden lokal wirkende Netztypen – RBF- und DCS-Netze – hinzugezogen. Die Abbildung 5.10 stellt die Klassifizierungsergebnisse für die getesteten Netztypen visuell dar. In der Tabelle 5.5 sind die wichtigsten Daten dieser Tests zusammengefasst.



a) Trainingsdaten



b) Testdaten



c) Die beste Klassifizierung aus 100 Versuchen

Abbildung 5.7: 2-Spiralen-Benchmark: Vergleich von Klassifizierungsraten (y-Achse) für MLP, MLHP und MLHPS für die verschiedene Anzahl von Neuronen in der versteckten Schicht (x-Achse). Gemittelt über 100 Versuche.

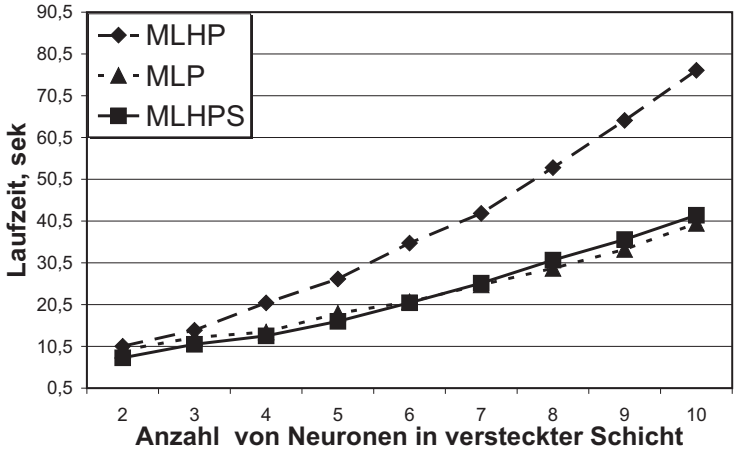


Abbildung 5.8: Vergleich von Laufzeiten für MLHP, MLHPS und MLP.

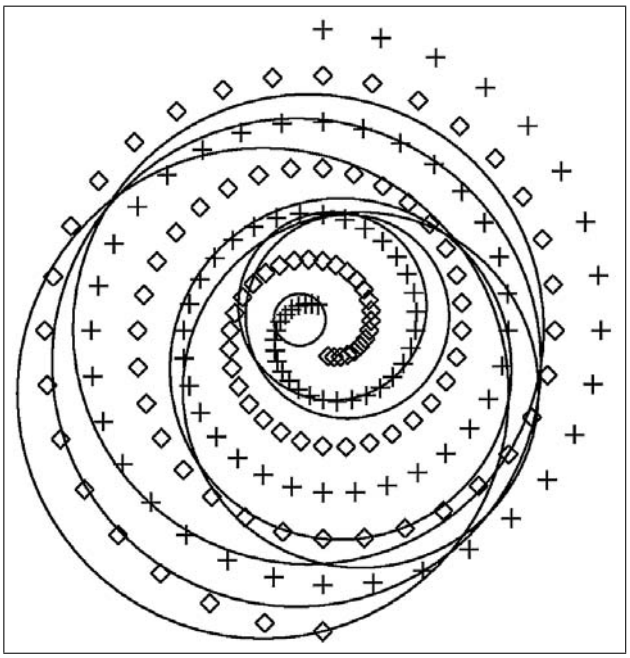


Abbildung 5.9: Visualisierung der Trennfunktionen in der versteckten Schicht (8 Neuronen) eines MLHPS im Fall einer fehlerfreien Trennung.

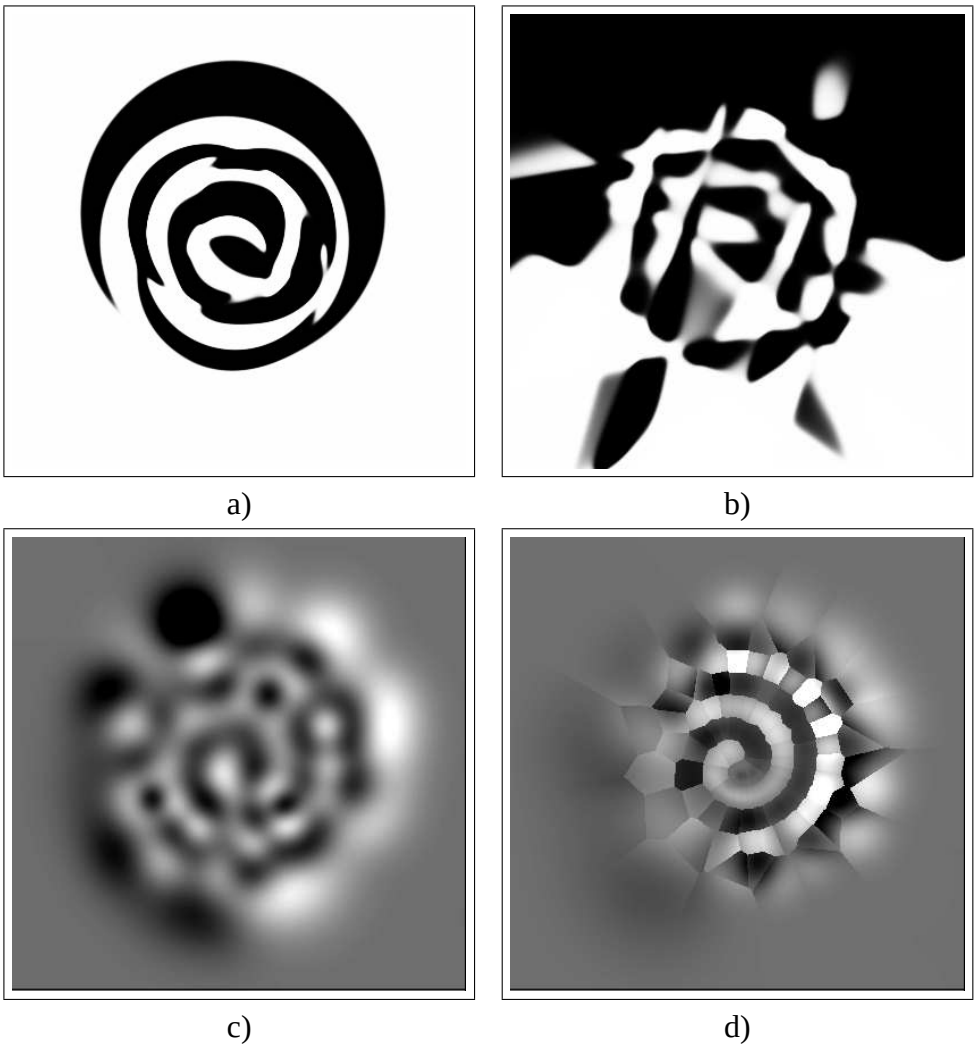


Abbildung 5.10: Visualisierung einer fehlerfreien Klassifizierung für verschiedene Netztypen: a) – MLHP mit 10 Neuronen, b) – MLP mit 60 Neuronen, c) – RBF mit 80 Neuronen, c) – DCS mit 110 Neuronen. Schwarze und weiße Bereiche repräsentieren beide Klassen. Die grauen Bereiche beschreiben Gebiete, wo keine zuverlässige Klassifizierung möglich ist.

	MLHPS	MLP	RBF	DCS
Minimale Anzahl Neuronen für eine fehlerfreie Klassifizierung	9	> 30	76	83
Anzahl Gewichte	34	>90	226	247
Laufzeit	~ 30s	> 300s	~ 15s	~ 20s

Tabelle 5.5: Vergleich der Ergebnisse des 2-Spiralen-Benchmarks für MLHPS, MLP, DCS und RBF.

5.8.3 Glass-Benchmark

Der Glass-Benchmark stammt aus der Proben1-Kollektion [84]. Die Ergebnisse einer chemischen Analyse (Bestandteile von 8 verschiedenen Substanzen) sowie der Brechungskoeffizient werden zur Klassifizierung von verschiedenen Glassorten benutzt. Diese Aufgabe wurde durch den gerichtlichen Bedarf in kriminalistischen Untersuchungen motiviert.

Die Daten bestehen aus 214 neun-dimensionalen reellwertigen Eingabevektoren und sechs-dimensionalen binären Ausgabevektoren. Die Größen der sechs Klassen betragen jeweils 70, 76, 17, 13, 9 und 29 Datensätze. Die Daten sind in je-

Netztyp	Anzahl Neuronen in versteckter Schicht	Anzahl Gewichte	Klassifizierungsergebnisse an Testdaten,%
MLHP	7	131	69
MLP	8	134	66

Tabelle 5.6: Vergleich der Ergebnisse vom Glass-Benchmark für MLHP und MLP.

weils 107 Trainings- und Testpaare aufgeteilt. Für die vorliegenden Daten ist es sehr schwer, eine gute Generalisierung zu erreichen. Die Ergebnisse der Klassifizierung für die Testdaten sind in Tabelle 5.6 zusammengefasst. Die Resultate sind über 30 Testläufe gemittelt. Sowohl ein MLP als auch ein MLHP mit einer vergleichbaren Anzahl von Gewichten liefern Erkennungsraten, die unter 70% liegen, wobei das MLHP etwas besser ist. Es scheint, dass weder Hyperebenen noch Hypersphären geeignet für eine gute Modellierung des Problems sind.

5.8.4 Vowel-Benchmark

Der Vowel-Benchmark stammt aus dem Bereich der Spracherkennung [26]. Die Daten enthalten die Aufnahmen von Vokalen, die von sowohl männlichen als auch weiblichen englischsprachigen Personen stammen. Die Daten enthalten 528 Trainings - und 462 Testmuster, wobei jedes Muster 10 Merkmale enthält, die zu einer von 11 Klassen gehören. Diese Aufgabe stellt sehr hohe Anforderungen an das Netz [26, 33].

In Tabelle 5.7 sind die Ergebnisse der Klassifizierung an der Testmenge für verschiedene Netztypen zusammengetragen. Ein Teil der Tabelle findet sich in [12] und [87].

Netztyp	Anzahl Neuronen in versteckter Schicht	Anzahl Gewichte	Klassifizierungsergebnisse an Testdaten, %
MLHP	11	275	62
MLP	10	231	57
MLP	88	2024	51
RBF88	88	1947	48
RBF528	528	11099	53
DCS	105	2216	65

Tabelle 5.7: Vergleich der Ergebnisse vom Vowel-Benchmark für verschiedene Netztypen.

Die Resultate zeigen, dass Vokalen-Klassifizierung ein schwieriges Problem darstellt. Insbesondere Bei MLP führt das Erhöhen der Anzahl von Gewichten zu einer Verschlechterung der Klassifizierung. Dies ist ein Indiz für Overfitting.

Die besten Ergebnisse liefern MLHP und DCS Netze, wobei MLHP zwar um ein Paar Prozente zurückliegt, aber eine wesentlich kleinere Architektur aufweist.

5.8.5 Visuell basierte Objektklassifizierung

In diesem Experiment wurden die Klassifizierungseigenschaften von MLHPs auf Daten getestet, die aus dem Bereich der Bildverarbeitung stammen. Das Ziel dieses Experimentes war, die Aufnahmen eines Objektes in Aufsicht mit einer von drei Klassen – Schraube, Brücke oder Dreieck – zu assoziieren.

Die Daten sind auf folgende Weise aufgenommen. Für jedes Objekt wurden 360 Aufsichten erstellt, wobei das Objekt zwischen den Aufnahmen jeweils um ein Grad weiter gedreht wurde. Jedes Objekt wurde durch ein Programm [19] automatisch erkannt, das Bild wurde entsprechend abgeschnitten und auf die Größe von 35×35 Pixel skaliert. In der Abbildung 5.11 sind einige Ansichten vorgestellt.

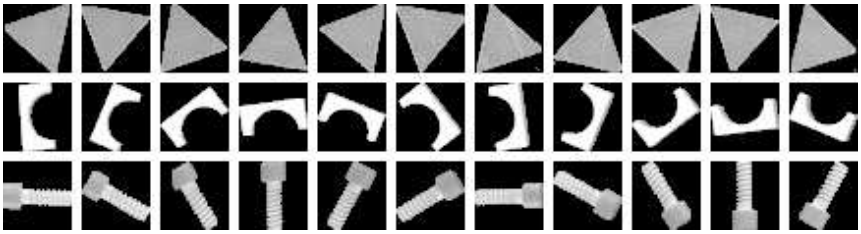


Abbildung 5.11: Jedes Objekt wurde automatisch erkannt und auf die Bildgröße von 35×35 Pixel skaliert.

Für die Weiterverarbeitung wurden die Daten als 1225-dimensionale Vektoren abgespeichert.

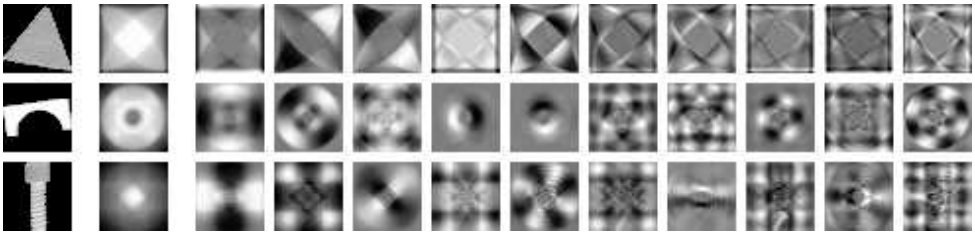
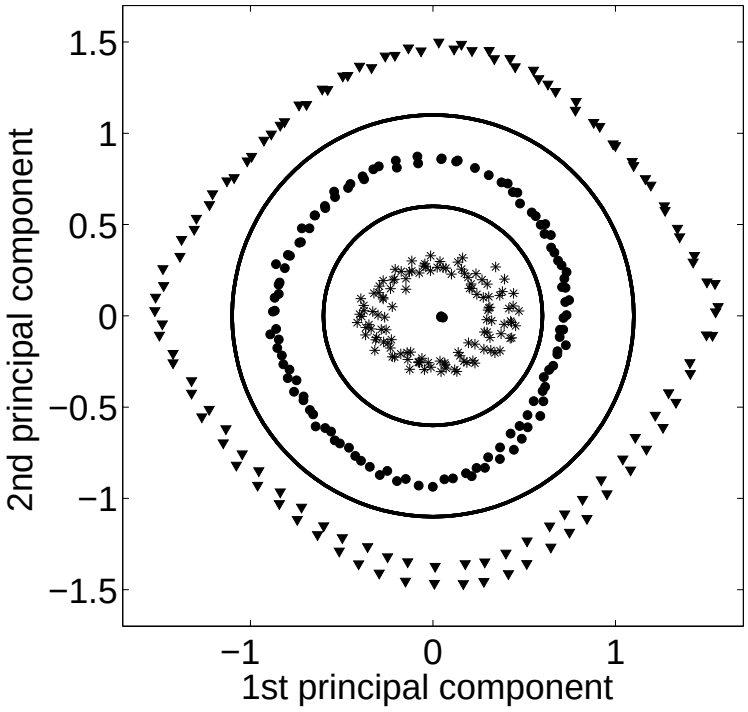


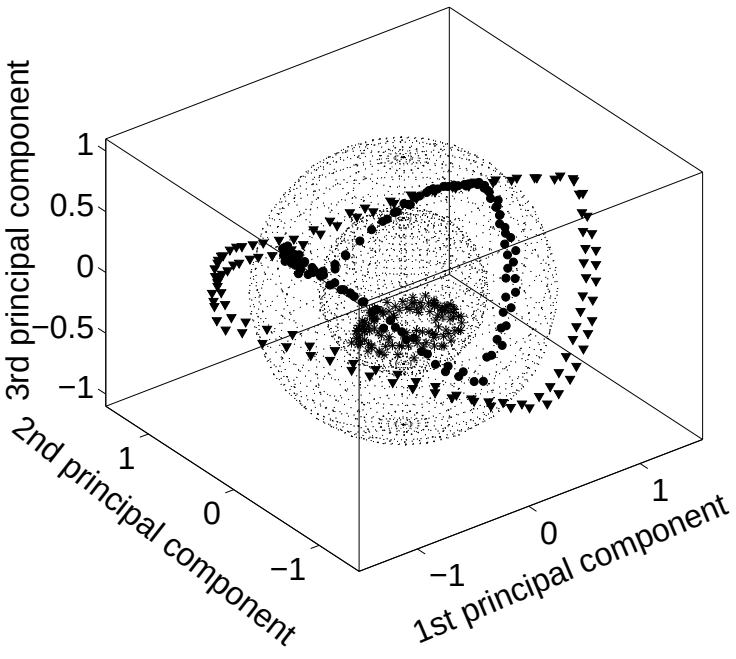
Abbildung 5.12: Mittelwert und die ersten zehn Hauptkomponenten für Dreieck (oben), Brücke (mitte) und Schraube (unten).

Für jede Menge von 360 Datenpaaren wurde die Hauptkomponentenanalyse durchgeführt (Abbildung 5.12). Danach wurden alle Datenvektoren aus allen drei Klassen auf die ersten zwei bzw. drei Hauptkomponenten der Brücke projiziert. Die drei Klassen wurden auf binäre Weise durch die Vektoren $(1, 0)$, $(0, 1)$ und $(1, 1)$ kodiert. Von insgesamt 1080 Daten wurden 360 für das Training und die restlichen 720 für den Test genommen. Es wurden verschiedene Netzwerkarchitekturen getestet. Die Ergebnisse der Klassifizierung mit MLHP sind in der Abbildung 5.13 dargestellt. Der gesamte Vergleich für den 2D- und 3D-Fall ist in den Tabellen 5.8 bzw. 5.9 gegeben.

Die Anzahl der zu einer nahezu fehlerfreien Klassifizierung benötigten Gewichte ist bei einem MLHP wesentlich geringer als bei klassischen MLP bzw. RBF-Netzen.



a)



b)

Abbildung 5.13: Visualisierung der Klassifizierungsergebnisse: Zwei Kreise bzw. Sphären repräsentieren die Trennflächen. a) – Projektion auf die ersten beiden Hauptkomponenten, b) – Projektion auf die ersten drei Hauptkomponenten.

Netztyp	Anzahl Neuronen	Anzahl Gewichte
MLHP	2	20
MLP	8	51
RBF	>20	>61

Tabelle 5.8: Minimale Anzahl von Neuronen in der versteckten Schicht für eine nahezu fehlerfreie Klassifizierung für das zweidimensionale Erkennungsproblem.

Netztyp	Anzahl Neuronen	Anzahl Gewichte
MLHP	2	22
MLP	3	24
RBF	8	33

Tabelle 5.9: Minimale Anzahl von Neuronen in der versteckten Schicht für eine nahezu fehlerfreie Klassifizierung für das dreidimensionale Erkennungsproblem.

5.9 Zusammenfassung

Das Hypersphären-Neuron stellt eine sinnvolle Erweiterung des klassischen Konzeptes der MLP und RBF-Netze dar. Die Effizienz der neuronalen Berechnung lässt sich durch die Einbettung in einen konformen Raum in vielen Fällen steigern. Insbesondere werden die Vorteile sowohl von globalen als auch von lokalen Netzen kombiniert. Speziell für Daten, die kompakte Klassen im Eingaberaum bilden bzw. von anderen Klassen umschlossen sind, lassen sich leistungsfähige Strukturen mit wenigen Neuronen entwickeln. Die Kosten für die Einbettung in einen konformen Raum bleiben unabhängig von der Dimension des ursprünglichen Problems gering.

Kapitel 6

NEURONALE MODELLE MIT LOKALEN REZEPTIVEN FELDERN ZUR SEGMENTIERUNG DES OPTISCHEN FLUSSES

Eine häufig gestellte Aufgabe in Computer Vision besteht darin, ein Bild in mehrere zusammenhängende Bereiche zu unterteilen, die einem angenommenem Modell entsprechen. Das zugrunde liegende Modell wird für jeden disjunkten Bereich meistens durch eine Reihe von Parametern beschrieben. Es wird dabei ein Zusammenhang zwischen den Parametern und den gemessenen Merkmalen im Bild, die oft als Punkte beschrieben werden, hergestellt.

Das Ziel einer Segmentierung ist es, die Position jedes Pixels einem Satz von Parametern unter Berücksichtigung der Nebenbedingungen, die einerseits den Fehler zwischen der Ausgabe des Modells und den tatsächlichen Messungen minimieren und andererseits die Anzahl der Segmente klein halten [58], zuzuordnen. Die Suche nach der Balance zwischen den beiden in Konkurrenz zueinander stehenden Zielsetzungen entspricht exakt dem Bias-Varianz-Dilemma, welches im Kapitel 2 beschrieben ist (Ausdruck (2.8)).

Eine Reihe von Techniken wurde von verschiedenen Autoren entwickelt, um das Segmentierungsproblem zu lösen. Ist die Beziehung zwischen den Modellparametern und den Merkmalen an jedem Punkt invertierbar, so kann die Segmentierung mit einer einfachen Hough-Transformation erfolgen [30]. Bei region-basierten Techniken werden die Parameter anhand mehrerer Teile des Bildes initialisiert und anschließend zusammengefügt oder gemäß einigen spezifischen Kriterien aufgeteilt [51]. Energie-basierte Ansätze reduzieren die Anzahl der Grenzübergänge in einem Bild durch Minimierung eines Funktionales [70]. Statistik-basierte Methoden benutzen statistische Maße, die aus Residuen von jedem Bereich berechnet werden, um eine Segmentierung durchzuführen. Dabei werden sowohl Helligkeit

[56] als auch optischer Fluss [11, 71, 109] als Merkmale für die Modellierung benutzt. Ein Überblick über weitere Techniken ist in [96] gegeben.

In dieser Arbeit wird das Segmentierungsproblem als eine Art von lokaler Approximation – wie in (2.9) allgemein beschrieben – formalisiert. Speziell in diesem Fall sind die Wichtungsfunktionen als disjunkte Lokalisator-Funktionen (2.13) zu wählen. Eine neuronale Architektur mit lokalen rezeptiven Feldern eignet sich dabei besonders gut für eine solche Clustering-Aufgabe. Ausgehend von der Annahme, dass die Bilder durch die Projektion der Bewegung von starren Objekten erzeugt werden, ist der für diesen Ansatz gewählte Merkmalsraum durch den optischen Fluss repräsentiert. Die Modellierung wird dabei durch die geometrischen Aspekte des Problems – Lochkamera, Projektion der starren Bewegungen, Epipolargeometrie – beeinflusst. Der Algorithmus zur Segmentierung eines Bildes bzw. einer Bildfolge in Bereiche mit gleicher induzierender 3D-Bewegung modelliert den Verlauf der Grenzübergänge zwischen einzelnen Objekten nicht explizit – a priori Wissen über die Form oder Größe einzelner Objekte ist nicht notwendig.

6.1 Netzarchitektur zur Segmentierung des optischen Flusses

Im Folgenden wird das Problem formalisiert und eine generische neuronale Netzarchitektur zusammen mit einem passenden Lernalgorithmus vorgestellt.

6.1.1 Problemstellung

Gegeben sei eine Szene bestehend aus N 3D-Punkten, die zu mehreren starren Objekten gehören. Die Objekte bewegen sich über die Zeit mit unterschiedlicher Geschwindigkeit und werden mit einer Kamera als eine Bildfolge der Länge M aufgenommen. Die projizierten Bewegungen erzeugen dabei einen optischen Fluss, der die Verschiebung (Translation) von 2D-Punkten beschreibt. Diese 2D-Punkte sind die Projektionen von demselben 3D-Punkt über mehrere Bilder. Mit anderen Worten, jeder 3D-Punkt $\mathbf{X}_i$ erzeugt in einer Bildfolge einen Korrespondenzvektor

$$\mathbf{x}_i = (\mathbf{x}_{1i}, \cdots, \mathbf{x}_{Mi}) = \begin{pmatrix} x^1_{1i} & \cdots & x^1_{Mi} \\ x^2_{1i} & \cdots & x^2_{Mi} \end{pmatrix}, \quad x^l_{ji} \in \mathbb{R}.$$

Anhand dieses Vektors kann zu der Projektion $\mathbf{x}_{ji}$ des Punktes $\mathbf{X}_i$ im Bild j ein Merkmal $\mathbf{y}_i$ entweder als direkte Korrespondenz im Bild k ($\mathbf{y}_i = \mathbf{x}_{ki}$) oder durch eine Weiterverarbeitung ($\mathbf{y}_i = f(\mathbf{x}_i)$) konstruiert werden.

Die Aufgabe besteht darin, die Projektionen einzelner Objekte zu unterscheiden bzw. zu segmentieren und die Parameter ihrer Bewegungen in der Bildebene abzuschätzen. Die Datenpaare $\{(\mathbf{x}_{ji}, \mathbf{y}_i)\}_{i=1\cdots N}$ bilden dabei den Merkmalsraum, der bezüglich eines angenommenen parametrischen Modells (Basisfunktion) $\mathbf{V}(., \mathbf{p}_k)$ lokal homogen ist.

Die Segmentierung wird durchgeführt, indem jedem Punkt $\mathbf{x}_{ji}$ im Bild j ein Satz von Parametern $\mathbf{p}_k$ zugeordnet wird, wobei sowohl die Differenz zwischen der Ausgabe des Modells $\mathbf{V}(\mathbf{x}_{ji}, \mathbf{p}_k)$ und den Merkmalen $\mathbf{y}_i$ als auch die Anzahl der lokalen Regionen K gleichzeitig minimiert werden:

$$\min_{K,\{\mathbf{p}_k\}} \sum_{i=1}^{N} \sum_{k=1}^{K} ||\mathbf{y}_i - \mathbf{V}(\mathbf{x}_{ji}, \mathbf{p}_k)||^2. \tag{6.1}$$

6.1.2 Generische Implementierung

Die Minimierungsaufgabe (6.1) lässt sich neuronal mit einer DCS-ähnlichen Architektur lösen, die für eine Segmentierung angepasst ist:

- Anstelle von klassischen RBF-Neuronen werden die Knoten des Netzes durch ggf. komplexere Strukturen (Experten) ersetzt, die ausschließlich das angenommene Modell (Basisfunktion $\mathbf{V}$) und dessen Parameter $\mathbf{p}$ berechnen bzw. anpassen können.
- Es existiert im Netz keine Nachbarschaftsstruktur, da die segmentierten Bereiche $\mathcal{L}_k$ disjunkt zueinander sind.
- Als Eingabe werden die Koordinaten eines Bildpunktes mit dem entsprechenden Merkmal erwartet.
- Als Ausgabe werden die Parameter des Modells (2D-Bewegung) sowie die Zuordnungen der Daten zu einzelnen Bereichen geliefert.

Das Netz wird unüberwacht mit einer speziell angepassten Version des DCS-Lernalgorithmus trainiert, die eine Abwandlung der Lernregeln für ein wachsendes Neuronengas [35] darstellt.

Der Algorithmus startet mit einem Satz von Parametern, d.h. mit einem Bereich $\mathcal{L}_1$. Der Ausgabefehler des angenommenen Modells in einzelnen Bereichen wird als Konfidenz benutzt, um die zu segmentierenden Daten einem Bereich zuzuordnen oder dem Netz ggf. einen neuen Bereich hinzuzufügen. Anschließend werden

die Parameter dieses Bereiches neu adaptiert, um das Modell sowohl an die schon vorhandenen als auch an die neu zugeordneten Daten optimal anzupassen. Die ganze Prozedur wird wiederholt bis alle Daten verteilt sind.

Algorithmus 6.1 Generischer Algorithmus zur Segmentierung

Setze Anzahl K von disjunkten Bereichen auf 1: $K := 1$
Initialisiere Basisfunktion $\mathbf{V}$ mit einem Satz von Parametern $\mathbf{p}_1$
while $\max_{i,k} ||\mathbf{y}_i - \mathbf{V}(\mathbf{x}_{ji}, \mathbf{p}_k)||^2$ > Schwellwert **do**
  Lösche alle Zuordnungen, ohne die Parameter $\mathbf{p}$ zu verändern
  for all Daten $(\mathbf{x}_{ji}, \mathbf{y}_i)$ **do**
    $l := \arg\min_k ||\mathbf{y}_i - \mathbf{V}(\mathbf{x}_{ji}, \mathbf{p}_k)||^2$
    if $||\mathbf{y}_i - \mathbf{V}(\mathbf{x}_{ji}, \mathbf{p}_l)||^2$ > Schwellwert **then**
      Füge neuen Bereich hinzu: $K := K + 1$
      $l := K$
    end if
    Ordne $(\mathbf{x}_{ji}, \mathbf{y}_i)$ dem Bereich $\mathcal{L}_l$ zu
    Passe die Modellparameter $\mathbf{p}_l$ unter Berücksichtigung aller Daten, die dem Bereich $\mathcal{L}_l$ zugeordnet sind, neu an
  end for
  Lösche Bereiche, denen keine Daten zugeordnet sind
end while

Eine geeignete Segmentierung kann nur dann erreicht werden, wenn die Bereiche am Anfang mit passenden Parametern initialisiert werden. Je nach angenommenem Modell werden mehrere Daten bzw. Punkte benötigt, um alle Freiheitsgrade des Modells bei der Bestimmung von Parametern auszuschöpfen. Das Verhalten des Systems kann stabilisiert werden, indem Lokalität vorausgesetzt wird – die Bereiche werden erst durch die Gruppierung von benachbarten Pixeln und der Annahme, dass diese Gruppen überwiegend Punkte enthalten, die zu einem Satz von Parametern gehören, initialisiert. Diese Anforderung wird implementiert, indem einerseits Daten in einer Reihenfolge an das Netz gereicht werden, die der Pixelnachbarschaft in der Bildebene entspricht (bzw. Zeilen- oder Spaltenweise). Andererseits wird der ganze Ablauf, wie im oberen Abschnitt beschrieben, mehrmals wiederholt und alle Daten immer wieder neu verteilt, bis ein gefordertes Konfidenzniveau erreicht wird. Dabei werden Bereiche, die nach der Umverteilung keine zugeordneten Daten enthalten, aus dem Netz gelöscht. Ursprünglich falsch initialisierte Bereiche werden dadurch eliminiert.

Der gesamte Ablauf in allgemeiner Form ist in Algorithmus 6.1 skizziert. Dieser Algorithmus beschreibt eine generische neuronale Struktur zur Segmentierung auf einem abstrakten, von der Wahl der Merkmale und folglich von der Modellierung

der Basisfunktion unabhängigem Niveau. Die einzige Voraussetzung besteht darin, dass der verwendete Merkmalsraum metrisch ist oder zumindest ein Distanzmaß besitzt, um den Fehler zwischen Ist- und Sollausgabe berechnen zu können. Die Güte der Segmentierung und die Lerngeschwindigkeit hängen allerdings von der Wahl des verwendeten Modells ab.

Mit dem Schwellwert für die Konfidenz kann dann die Balance zwischen der Anzahl von Segmenten und der Qualität der Parameterschätzung gesteuert werden.

Falls die Parameter einer Transformation als Modell benutzt werden, so ist es unter der Annahme einer konstanten Bewegung über mehrere Bilder einer Bildfolge möglich, eine Prädiktion durchzuführen bzw. die Segmentierung in den folgenden Frames der Bildsequenz zu beschleunigen.

6.2 Modellierung von lokalen Experten

Bewegt sich ein starres Objekt im 3D-Raum, so lässt sich diese Bewegung durch eine allgemeine Rotation als eine lineare Transformation in dem entsprechenden homogenen Raum beschreiben. Das Lochkamera-Modell[1] lässt geometriebedingt im Allgemeinen aber keine Beschreibung der projizierten Bewegung auf lineare Weise zu. Die Zerlegung der 3D-Bewegung in eine zu der Bildebene parallele bzw. senkrechte Komponente [53] erlaubt, die Bewegung in der Bildebene besser zu interpretieren.

Findet die 3D-Bewegung eines Objektes in einer Richtung, die zu der Bildebene parallel verläuft, statt, so ist die Projektion fast linear. Die Projektion der Bewegung in einer Ebene, die senkrecht zu der Bildebene steht und das optische Zentrum enthält, ist dagegen im Rahmen dieser Skala maximal nichtlinear. Je größer der parallele Anteil ist, desto einfacher ist es, die gesamte Projektion linear zu approximieren. Im Weiteren werden die Möglichkeiten sowohl der linearen als auch der nichtlinearen Approximation mittels neuronaler Netze diskutiert.

6.2.1 Approximation mittels allgemeiner Rotationen

Unter der Annahme, dass die Position von Objekten in den benachbarten Bildern einer Bildfolge sich nicht stark (auch in der Tiefe) unterscheidet, lassen sich die projizierten Bewegungen mittels allgemeiner Rotationen approximieren, obwohl

[1] siehe Anhang A.1.

diese Modellierung nicht exakt den geometrischen Zwängen eines Lochkamera-Modells unterliegt. Dieser Sachverhalt ist im Anhang A.2 näher erläutert.

In Kapitel 3 wurde beschrieben, auf welche Weise sich allgemeine Rotationen effizient in einer konformen geometrischen Algebra durch ein Versorprodukt ausdrücken lassen (3.39). Die Daten werden als Elemente einer konformen geometrischen Algebra – Multivektoren – eingebettet. Eine Rotation wird auf Daten angewandt, indem man die Multivektoren einmal von links mit einem Versor und einmal von rechts mit dem Reversen des Versors multipliziert – die Wirkung des Versorproduktes kann also als zweimalige Anwendung des geometrischen Produktes interpretiert werden[2]. Um solch eine Berechnungsvorschrift neuronal zu implementieren, bedarf es erst des Verständnisses, wie sich ein einfaches geometrisches Produkt in einem Neuron kodieren lässt und welche Vorteile es gegenüber dem klassischen Paradigma aufweist.

In den folgenden Abschnitten wird ein Clifford-Neuron vorgestellt und darauf basierend eine Struktur zur Berechnung der allgemeinen Rotationen entworfen.

6.2.1.1 Clifford-Neuron

Ein Clifford-Neuron wird für eine gegebene geometrische Algebra $\mathcal{G}_{p,q,r}$ folgendermaßen definiert [16]:

Die Eingabe wird als ein Multivektor $\mathbf{X}$ aus dieser Algebra kodiert. Die Gewichte $\mathbf{W}$ werden durch einen weiteren Multivektor repräsentiert. Das Neuron berechnet als Ausgabe $\mathbf{Y}$ das geometrische Produkt:

$$\mathbf{Y} = \mathbf{W}\mathbf{X}. \tag{6.2}$$

Die Propagierungsfunktion eines Clifford-Neurons wird also als das durch die gegebene Algebra $\mathcal{G}_{p,q,r}$ induzierte geometrische Produkt definiert. Als Aktivierungsfunktion kann eine der klassischen Aktivierungen gewählt werden. Eine Anzahl solcher Aktivierungen wurde in [14, 17] vorgestellt. Im Rahmen dieser Arbeit wird ausschließlich die Identität-Aktivierungsfunktion bei der Modellierung eingesetzt.

[2] Hier wird das Versorprodukt erst in einer allgemeinen Form, und nicht als Spinorprodukt wie in [13] definert, beschrieben.

Implementierung

Betrachtet man die Multivektoren $\mathbf{X}, \mathbf{W}$ und $\mathbf{Y}$ aus $\mathcal{G}_n$ als 2^n-dimensionale Vektoren $\mathbf{x} = (x_1, x_2, \ldots, x_{2^n}), \mathbf{w} = (w_1, w_2, \ldots, w_{2^n})$ und $\mathbf{y} = (y_1, y_2, \ldots, y_{2^n})$ im Euklidischen Raum $\mathbb{R}^{2^n}$, so lässt sich die Wirkung des geometrischen Produktes als ein mehrfach ausgeführtes Skalarprodukt (3.7) auffassen:

$$y_i =< \text{Permutation}(\mathbf{w}, \text{Signum}(w_j)), \mathbf{x} > . \tag{6.3}$$

Die Regeln zur Permutation vom Gewichtsvektor $\mathbf{w}$ werden durch die Struktur der gewählten Algebra $\mathcal{G}_{p,q,r}$ allgemein bzw. durch den Multiplikationstensor G speziell gegeben, wie in (3.5) beschrieben.

Die Berechnungsregeln (6.3) verdeutlichen, dass ein Clifford-Neuron in der Funktionsweise einem reellwertigen linearen Assoziator ähnlich ist: die Ausgabe wird als die Multiplikation zwischen der Gewichtsmatrix und dem Eingabevektor berechnet. Die Struktur der gewählten Algebra schränkt aber die Freiheitsgrade des Clifford-Modells ein – sowohl die Dimension des Eingaberaumes als auch die Dimension des Ausgaberaumes sind bei einem Clifford-Neuron durch die Dimension n der gewählten Algebra – wie in Abbildung 6.1 veranschaulicht – auf 2^n festgelegt. Im Gegensatz zu einem reellwertigen linearen Assoziator von gleicher Dimension ist die Anzahl der unabhängigen Gewichte auf 2^n eingeschränkt (2^{2n} bei einem reellwertigen linearen Assoziator).

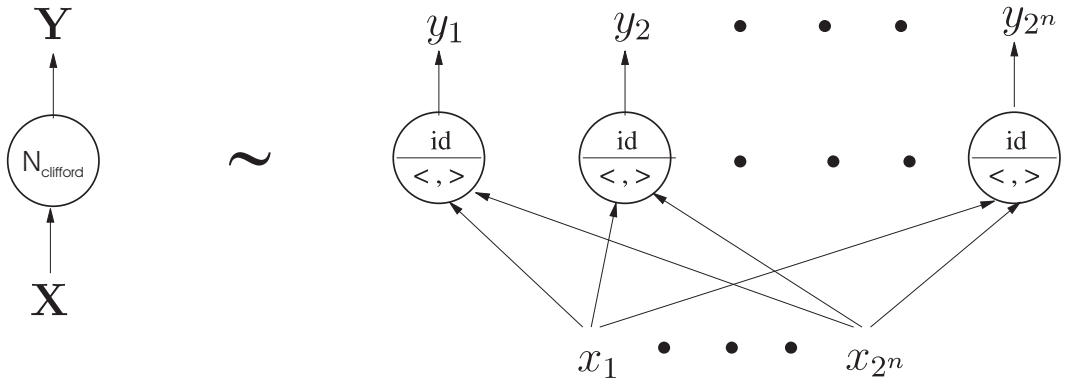


Abbildung 6.1: Ein Clifford-Neuron aus $\mathcal{G}_n$ ist äquivalent zu einem speziellen einschichtigen linearen Assoziator mit 2^n Neuronen.

Diese Einschränkungen zwingen ein Clifford-Neuron nur solche Gewichte zu lernen, welche die intrinsische Struktur der gewählten Algebra wiedergeben. Mit anderen Worten, die Daten werden in die Algebra projiziert.

Lernen

Die Berechnungen mit einem Clifford-Neuron werden im Allgemeinen durchgeführt, indem vorliegende Daten in eine geeignete Algebra einbettet werden. Diese Einbettung ist prinzipbedingt auf Räume beschränkt, deren Dimension eine Zweierpotenz ist. Da Clifford-Algebren spezielle Strukturen besitzen, spielt es auch eine Rolle, in welcher Reihenfolge Daten eingebettet werden, d.h. welchen Teilen eines Multivektors die jeweiligen Elemente von ursprünglichen Daten entsprechen. Die Lernregeln in einem Clifford-Neuron werden von den klassischen Regeln abgeleitet, indem man Nebenbedingungen, die dem geometrischen Produkt der gewählten Algebra entsprechen, aufstellt. Im Rahmen dieser Arbeit wurde der modifizierte Backpropagation-Algorithmus 4.1 zum Lernen benutzt. Dabei wurden die unabhängigen Gewichte dem Multiplikationstensor G (3.5) für die jeweilige Algebra entsprechend angepasst.

Zusammenfassung

Die oben genannten Einschränkungen haben eine direkte Auswirkung auf die grundlegenden Eigenschaften des Neurons. Das gilt insbesondere, wenn die Wahl der Algebra auf dem a priori Wissen über die Struktur der Daten basiert: Die Generalisierungsfähigkeit des Clifford-Neurons wird im Vergleich zu einem klassischen linearen Assoziator erhöht bei gleichzeitiger Reduktion des Rauscheinflusses.

Falls keine geeignete algebraische Repräsentation existiert oder eine ”falsche” Algebra gewählt wird, können die Ergebnisse schlechter ausfallen, als bei einem reellwertigen linearen Assoziator. Dieses Verhalten wird im folgenden Beispiel verdeutlicht.

Rauschen	0%	5%	10%	20%
reellwertiger linearer Assoziator	0.2%	3.6%	11.9%	30.3%
Clifford-Neuron $\mathcal{G}_{3,0,0}$	0.1%	2.4%	6.4%	24.1%
Clifford-Neuron $\mathcal{G}_{1,1,1}$	4.7%	8.2%	19.0%	41.6%

Tabelle 6.1: $\mathcal{G}_{3,0,0}$: Relativer Fehler in Abhängigkeit vom Rauschen für Testdaten.

Im vorgestellten Experiment wurden 100 Datensätze als Paare von normierten

Multivektoren

$$\{(\mathbf{X}_i, \mathbf{Y}_i = \mathbf{W}\mathbf{X}_i)|\mathbf{X}_i, \mathbf{Y}_i, \mathbf{W} \in \mathcal{G}_{3,0,0}, ||\mathbf{X}_i||_2 = 1\}_{i=1\cdots100}$$

zufällig erzeugt. Die auf diese Weise erzeugten acht-dimensionalen Daten wurden zunehmend mit Gauss'schem Rauschen belegt. Dreißig Prozent der Daten wurden zum Training und siebzig Prozent zum Testen benutzt. Drei verschiedene Netzarchitekturen wurden verglichen: ein reellwertiger linearer Assoziator, ein $\mathcal{G}_{3,0,0}$ Clifford-Neuron, sowie ein $\mathcal{G}_{1,1,1}$ Clifford-Neuron. Das Experiment wurde zehn mal für verschiedene $\mathbf{W}$ wiederholt. Die gemittelten Ergebnisse verdeutlichen – wie in Tabelle 6.1 zusammengefasst – dass die Wahl des Modells zur Beschreibung eines Problems erheblichen Einfluss sowohl im positiven als auch im negativen Sinne auf die Güte der Approximation ausüben kann.

6.2.1.2 Neuron zur Berechnung einer allgemeinen Rotation

Um ein Versorprodukt (3.35) neuronal zu implementieren, bedarf es der Anwendung des geometrischen Produktes mit dem Versor selbst, sowie mit seinem Reversen. Die Aufgabe eines solchen neuronalen Modells besteht darin, anhand der vorliegenden Trainingsbeispiele $\{(\mathbf{X}_i, \mathbf{Y}_i)\}_{i=1,\cdots,N}$, die zugrunde liegende allgemeine Rotation (durch Rotor $\mathbf{W}$ gegeben), durch Minimierung die der quadratischen Fehlerfunktion (2.4) zu lernen:

$$\min_{\mathbf{W}\in SE(3)} \sum_{i=1}^{N} ||\mathbf{Y}_i - \mathbf{W}\mathbf{X}_i\widetilde{\mathbf{W}}||_2^2. \tag{6.4}$$

Die beiden Teile des Versorproduktes sind nicht komplett voneinander unabhängig, da es zwischen einem Multivektor und seinem Reversen einen Zusammenhang, wie in (3.16) beschrieben, gibt – das Reverse eines Multivektors wird durch den Vorzeichenwechsel Elementen gebildet. Es ergibt sich daraus, dass das Ergebnis des Versorproduktes in einer quadratischen Weise von den Komponenten des Rotors abhängt.

Ein neuronales Modell, welches die Berechnung des Versorproduktes durchführt, kann folglich als ein Neuron mit einer in Bezug auf Gewichte polynomialen (quadratischen) Propagierungsfunktion oder als ein spezielles zweischichtiges Clifford-Netz, mit schichtenweise voneinander abhängigen Gewichten [15] implementiert werden.

Implementierung

Grundsätzlich kann ein Versor-Neuron alle in der gewählten Algebra durch Multivektoren repräsentierbaren Entitäten auf dieselbe Weise verarbeiten. Da für die Kodierung verschiedener Entitäten unterschiedliche Teile des Multivektors benutzt werden, benötigt man ggf. alle 2^n Komponenten einer Algebra. In dieser Arbeit werden nur solche Entitäten wie Punkte betrachtet. Speziell der 2D-Fall ist im Rahmen dieser Arbeit von Bedeutung. Diese Einschränkung reduziert den benötigten Berechnungsaufwand erheblich. Folgende Gleichungen dienen als Grundlage zur Entwicklung eines Versor-Neurons für den 2D-Fall.

Die vorliegenden Datenpaare $(\mathbf{x} = (x_1, x_2), \mathbf{y} = (y_1, y_2)) \in \mathbb{R}^2 \times \mathbb{R}^2$ werden in die konforme geometrische Algebra $\mathcal{G}_{3,1}$, wie in Kapitel 3 beschrieben, eingebettet:

$$\begin{aligned} \mathbf{X} &= x_1\mathsf{e}_1 + x_2\mathsf{e}_2 + \tfrac{1}{2}\mathbf{x}^2\mathsf{e}_\infty + \mathsf{e}_o, \\ \mathbf{Y} &= y_1\mathsf{e}_1 + y_2\mathsf{e}_2 + \tfrac{1}{2}\mathbf{y}^2\mathsf{e}_\infty + \mathsf{e}_o. \end{aligned} \tag{6.5}$$

Eine allgemeine Rotation mit dem Zentrum in $\mathbf{t} = (t_x, t_y)$ um den Winkel θ wird durch den folgenden Multivektor repräsentiert:

$$\begin{aligned} \mathbf{W} &= \cos\left(\frac{\theta}{2}\right) + \sin\left(\frac{\theta}{2}\right)\mathbf{U} + \sin\left(\frac{\theta}{2}\right)(\mathbf{U}\cdot\mathbf{t})\,\mathsf{e}_\infty \\ &= \cos\left(\frac{\theta}{2}\right) + \sin\left(\frac{\theta}{2}\right)\mathsf{e}_{12} + \sin\left(\frac{\theta}{2}\right)(\mathsf{e}_{12}\cdot(t_x\mathsf{e}_1 + t_y\mathsf{e}_2))\,\mathsf{e}_\infty \\ &= \cos\left(\frac{\theta}{2}\right) + \sin\left(\frac{\theta}{2}\right)\mathsf{e}_{12} + t_y\sin\left(\frac{\theta}{2}\right)(\mathsf{e}_{1+} - \mathsf{e}_{-1}) \\ &+ t_x\sin\left(\frac{\theta}{2}\right)(\mathsf{e}_{-2} - \mathsf{e}_{2+}) \\ &:= w_1 + w_2\mathsf{e}_{12} + w_3(\mathsf{e}_{1+} - \mathsf{e}_{-1}) + w_4(\mathsf{e}_{-2} - \mathsf{e}_{2+}), \end{aligned} \tag{6.6}$$

und das entsprechende Reverse mit

$$\widetilde{\mathbf{W}} = w_1 - w_2\mathsf{e}_{12} - w_3(\mathsf{e}_{1+} - \mathsf{e}_{-1}) - w_4(\mathsf{e}_{-2} - \mathsf{e}_{2+})$$

kodiert.

Die 2D- Koordinaten (y_1, y_2) des transformierten Punktes $\mathbf{y}$ lassen sich durch die Anwendung des Versorproduktes auf die algebraische Repräsentation $\mathbf{X}$ des 2D-Punktes $\mathbf{x}$ berechnen:

$$\begin{aligned} &\mathbf{Y} = \mathbf{W}\mathbf{X}\widetilde{\mathbf{W}} \\ &\quad \Downarrow \\ &y_1 = (w_1^2 - w_2^2)x_1 + 2w_1w_2x_2 + 2w_2w_4 - 2w_1w_3 \\ &y_2 = 2w_1w_2x_1 + (w_1^2 - w_2^2)x_2 + 2w_2w_3 + 2w_1w_4 \end{aligned} \tag{6.7}$$

oder in Matrixform:

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} w_1^2 - w_2^2 & 2w_1w_2 & 2w_2w_4 & -2w_1w_3 \\ 2w_1w_2 & w_1^2 - w_2^2 & 2w_2w_3 & 2w_1w_4 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ 1 \\ 1 \end{pmatrix}. \tag{6.8}$$

Die Darstellung (6.8) verdeutlicht, dass ein Neuron zur Berechnung einer allgemeinen Rotation mit dem Versorprodukt, genauso wie ein Clifford-Neuron, einem speziellen linearen Assoziator (in diesem Fall 4D zu 2D) entspricht. Die Daten werden ähnlich wie bei einem Hypersphären-Neuron nicht nur homogen eingebettet, sondern auch mit einer zweiten 'Bias'-Komponente versehen. Die Nichtlinearität bezüglich der Gewichte w_i kann allerdings die Qualität des Lernens beeinträchtigen.

Lernen

Eine Möglichkeit, das Training zu vereinfachen, besteht darin, die Struktur des Neurons so zu transformieren, dass eine lineare Beziehung bezüglich der Gewichte entsteht. Dabei wird die Versorprodukt-Gleichung zu einer alternativen äquivalenten Darstellung folgendermaßen umgeformt:

$$\begin{aligned} &\mathbf{Y} = \mathbf{W}\mathbf{X}\widetilde{\mathbf{W}} \\ \Longleftrightarrow\ &\mathbf{Y}\mathbf{W} = \mathbf{W}\mathbf{X}\widetilde{\mathbf{W}}\mathbf{W} \\ \Longleftrightarrow\ &\mathbf{Y}\mathbf{W} = \mathbf{W}\mathbf{X} \\ \Longleftrightarrow\ &\mathbf{W}\mathbf{X} - \mathbf{Y}\mathbf{W} = 0. \end{aligned} \tag{6.9}$$

In Abbildung 6.2 sind die oben ausgeführten Repräsentationsmöglichkeiten veranschaulicht.

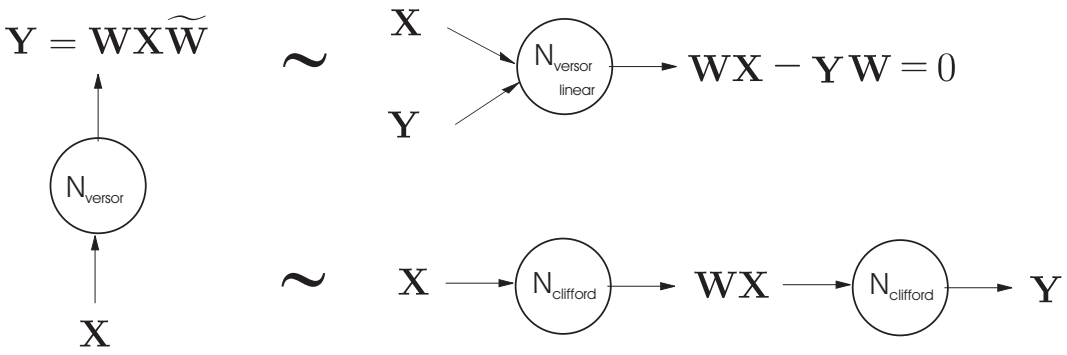


Abbildung 6.2: Die Berechnung der Gewichte eines Neurons mit Versorprodukt-Propagierung kann durch Umformung auf verschiedene Arten erfolgen.

Im Rahmen dieser Arbeit wird zum Lernen die linearisierte Form eingesetzt. Der Ausdruck (6.9) lässt sich folgendermaßen umformen:

$$\mathbf{WX} - \mathbf{YW} = \begin{pmatrix} \mathsf{e}_1 \\ \mathsf{e}_2 \\ \mathsf{e}_\infty \\ \mathsf{e}_{-12} + \mathsf{e}_{12+} \end{pmatrix}^T A \begin{pmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{pmatrix} = 0 \tag{6.10}$$

mit

$$A = \begin{pmatrix} (x_1 - y_1) & (x_2 + y_2) & -2 & 0 \\ (x_2 - y_2) & -(x_1 + y_1) & 0 & 2 \\ \frac{1}{2}(\mathbf{x}^2 - \mathbf{y}^2) & 0 & -(x_1 + y_1) & (x_2 + y_2) \\ 0 & \frac{1}{2}(\mathbf{x}^2 - \mathbf{y}^2) & (x_2 - y_2) & -(x_1 - y_1) \end{pmatrix}. \tag{6.11}$$

Daraus ergibt sich ein lineares Gleichungssystem:

$$A \begin{pmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{pmatrix} = 0. \tag{6.12}$$

Jedes Datenpaar $(\mathbf{x}_i, \mathbf{y}_i)$ einer Trainingsmenge $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1\cdots n}$ liefert ein Gleichungssystem. Bei dieser Darstellung lässt sich die Minimierungsaufgabe (6.4)

folgendermaßen umformulieren:

$$\min_{(w_1,w_2,w_3,w_4)} \left\| \begin{pmatrix} A_1 \\ \vdots \\ A_n \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{pmatrix} \right\|_2^2, \tag{6.13}$$

wobei die A_i wie in (6.11) definierte Matrizen für Datenpaare $(\mathbf{x}_i, \mathbf{y}_i)$ sind. Die Minimierung an sich kann sowohl mit einem auf SVD basierenden Verfahren wie in [82] beschrieben, als auch über einen Gradientenabstieg wie bei Backpropagation erfolgen. In dieser Arbeit wird der Backpropagation-Algorithmus 4.1 benutzt. Die zu minimierende Fehlerfunktion lässt sich direkt aus (6.13) auslesen. Da die Fehlerfunktion quadratisch bezüglich der gesuchten Gewichte und somit konvex ist, existiert nur eine globale Lösung. Dies wirkt sich positiv auf die Stabilität und die Konvergenzgeschwindigkeit des Algorithmus aus. Im Allgemeinen werden die gefundenen Gewichte nicht normiert – je nach dem verwendeten Minimierungsalgorithmus kann eine anschließende Normierung $\mathbf{W}\widetilde{\mathbf{W}} = 1$ notwendig sein.

Zusammenfassung

Genauso wie im Fall eines Clifford-Neurons schränkt die Einbettung des Problems der Schätzung einer allgemeinen Rottion in eine Clifford-Algebra die Freiheitsgrade des Modells ein. Dies erhöht die Generalisierungsfähigkeit und Robustheit des Versor-Neurons im Vergleich zu einem reellwertigen linearen Assoziator.

Rauschen	0%	3%	5%	10%
reellwertiger linearer Assoziator	0.1%	2.3%	4.4%	9.2%
Versor-Neuron	0.1%	1.5%	2.8%	5.9%

Tabelle 6.2: Allgemeine Rotation: Relativer Fehler in Abhängigkeit des Rauschens auf den Testdaten.

Die stabilisierende Wirkung der Einbettung ist im folgenden Beispiel veranschaulicht. Eine allgemeine Rotation (durch den Rotor $\mathbf{W}$ repräsentiert) mit der Translationskomponente 1 wurde zufällig gewählt. Anschließend wurden 100 normierte Punkte erzeugt und mit $\mathbf{W}$ transformiert:

$$\{(\mathbf{X}_i, \mathbf{Y}_i = \mathbf{W}\mathbf{X}_i\widetilde{\mathbf{W}}) | \mathbf{X}_i, \mathbf{Y}_i, \mathbf{W} \in \mathcal{G}_{3,1}, ||\mathbf{X}_i||_2 = 1, \mathbf{W}\widetilde{\mathbf{W}} = 1\}_{i=1\cdots100}.$$

Rauschen	0%	3%	5%	10%
reellwertiger linearer Assoziator, Winkel [°]	0.2	1.4	2.4	5.6
Versor-Neuron, Winkel [°]	0.3	1.1	1.8	3.3
reellwertiger linearer Assoziator, Translation	0.01	0.17	0.26	0.30
Versor-Neuron, Translation	0.01	0.08	0.09	0.13

Tabelle 6.3: Allgemeine Rotation: Fehler in Winkel- und Translationskomponente.

Die auf diese Weise erzeugten Daten wurden zunehmend mit Gauss'schem Rauschen belegt. Dreißig Prozent der Daten wurden zum Training und siebzig Prozent zum Testen benutzt. Weiterhin wurden die Parameter der allgemeinen Rotation (Winkel- und Translationskomponente) aus den Gewichten extrahiert und mit den ursprünglichen Parametern verglichen. Das Experiment wurde zehn mal mit verschiedenen zufällig gewählten Parametern wiederholt. Die gemittelten Ergebnisse sind in Tabelle 6.2 und Tabelle 6.3 zusammengefasst.

Die Darstellung von allgemeinen Rotationen in konformen geometrischen Algebren bringt auch eine Einschränkung auf die Art der möglichen repräsentierbaren Transformationen mit sich – wie in Kapitel 3 beschrieben, lassen sich reine Translationen nicht durch allgemeine Rotationen beschreiben. Diese Einschränkung stellt aber kein praktisches Problem dar, da die vorliegenden Daten meistens verrauscht sind und auch aus numerischer Sicht die Implementierung der Minimierungsalgorithmen leichte Ungenauigkeiten aufweist. Dadurch entsteht immer eine, wenn auch kleine, Rotationskomponente. Weiterhin wird das gutartige Verhalten durch die Tatsache begünstigt, dass der zu minimierende Term bis auf einen Skalarfaktor definiert ist und sich immer durch einen endlichen Ausdruck repräsentieren lässt.

6.2.1.3 Entwurf von Experten

Von einer Segmentierung des optischen Flusses mittels allgemeiner Rotationen ausgehend, lassen sich verschiedene Experten-Modelle aufstellen.

Korrespondenzen als Merkmale

Wird die Korrespondenz in Bild k zu einem Punkt $\mathbf{x}_{ji}$ im Bild j als Merkmal

gewählt

$$\mathbf{y}_i = \mathbf{x}_{ki},$$

so lässt sich das Neuron zur Berechnung einer allgemeinen Rotation, wie in (6.8) beschrieben, direkt als lokaler Experte in die allgemeine Netzarchitektur einsetzen. Da der Merkmalsraum durch die Punkte in der Bildebene repräsentiert wird, existiert auch eine Euklidische Metrik, die zur Berechnung der Konfidenzen im Netz erforderlich ist. In der Lernphase werden die Daten als Punktpaare $(\mathbf{x}_{ji}, \mathbf{x}_{ki})$ an das Netz übergeben. Nach der erfolgten Segmentierung lassen sich die Anzahl der Segmente (Anzahl Neuronen), die zugeordneten Punkte und die Parameter der allgemeinen Rotationen (Gewichte der Neuronen) direkt aus dem Netz auslesen.

Parameter von allgemeinen Rotationen als Merkmale

Unter der Annahme, dass die projizierte Bewegung über mehrere Bilder einer Bildfolge konstant bleibt, lassen sich die Merkmale als Parameter der konstanten Bewegung einzelner Punkte über mehrere Bilder modellieren.

Zur Schätzung der Parameter für einzelne Punkte wird das Neuron zur Berechnung der allgemeinen Rotation benutzt. Zum Training werden die Daten paarweise $(\mathbf{x}_{ji}, \mathbf{x}_{(j+1)i})$ aus dem Korrespondenzvektor $\mathbf{x}_i = (\mathbf{x}_{1i}, \cdots, \mathbf{x}_{Mi})$ entnommen und an das Neuron übergeben. Nach der Schätzung werden die Parameter der allgemeinen Rotationen als Merkmale für die Segmentierung verwendet:

$$\mathbf{y}_i = \mathbf{w}_i = (w_{i2}, w_{i3}, w_{i4}).$$

Das Netz gruppiert dann die Pixel mit einer ähnlichen Transformation.

Da unabhängig von der Wahl einer Parametrisierung keine Metrik für allgemeine Rotationen existiert [113], ist der Entwurf eines geeigneten Distanzmaßes erforderlich, um den Algorithmus 6.1 anwenden zu können. Im Rahmen dieser Arbeit wird das im Anhang C.1 vorgestellte Distanzmaß – wie in Gleichung (C.2) angegeben – eingesetzt.

6.2.2 Approximation mit Kreisen

Die Annahme, dass sich die projizierten Bewegungen mittels allgemeiner Rotationen approximieren lassen, bildet ein Fundament für den weiteren Modellentwurf. Die Trajektorien einzelner Bildpunkte unter der Wirkung einer allgemeinen

Rotation sind kreisförmig (beispielhaft in Abbildung 6.3 visualisiert). Unter Berücksichtigung von Korrespondenzen einzelner Punkte in mehreren Bildern einer Bildfolge lassen sich die entsprechenden Trajektorien berechnen und zu Segmentierungszwecken benutzen.

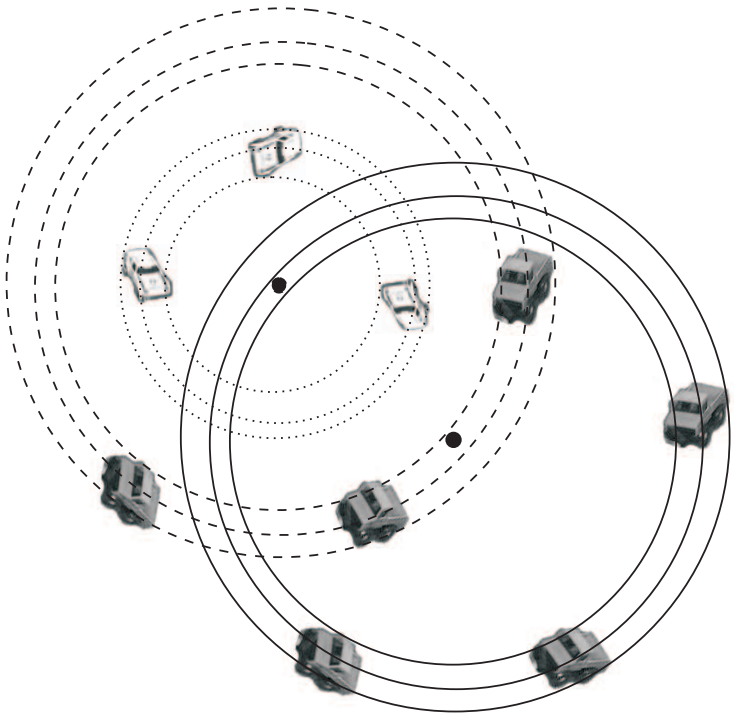


Abbildung 6.3: Trajektorien der Bewegungen von starren Objekten in der Bildebene können auf kurzen Distanzen durch Kreise beschrieben werden.

6.2.2.1 Implementierung

Ein Kreis lässt sich unabhängig von der Dimension des Raumes durch drei Punkte definieren. Ein Kreis aus $\mathbb{E}^3$ kann im konformen Raum $\mathbb{PK}^3$ durch den Nullraum eines Trivektors repräsentiert werden [81]. Liegen die Punkte $\mathbf{x}, \mathbf{y}$ und $\mathbf{z} \in \mathbb{E}^3$ auf einem Kreis, so gilt für jeden weiteren Punkt $\mathbf{a} \in \mathbb{E}^3$

$$(\mathcal{P}(\mathcal{K}(\mathbf{x})) \wedge \mathcal{P}(\mathcal{K}(\mathbf{y})) \wedge \mathcal{P}(\mathcal{K}(\mathbf{z}))) \wedge \mathcal{P}(\mathcal{K}(\mathbf{a})) = \mathbf{C} \wedge \mathcal{P}(\mathcal{K}(\mathbf{a})) = 0, \quad (6.14)$$

falls $\mathbf{a}$ auf dem Kreis liegt. Das Zentrum des Kreises wird durch den Vektor

$$\mathbf{O} = (\mathbf{C} \cdot \mathsf{e}_\infty) \cdot (\mathbf{C} \wedge \mathsf{e}_\infty) \in \mathbb{PK}^3 \quad (6.15)$$

repräsentiert.

Aus (6.14) lässt sich ein lineares Gleichungssystem konstruieren und lösen, falls mehr als drei Punkte zur Bestimmung des Kreises vorliegen [81].

Die Euklidischen Koordinaten $\mathbf{o} = (o_1, o_2)$ des Zentrums für den zweidimensionalen Fall werden geschätzt, indem die den Kreis erzeugenden 2D-Punkte erst um eine Nullkomponente (dritte Koordinate) erweitert werden. Sodann wird der Kreis bzw. das Zentrum gemäß (6.14) und (6.15) berechnet und anschließend das 2D-Zentrum folgendermaßen bestimmt:

$$o_1 = \mathbf{O} \cdot \mathsf{e}_1 \quad \text{und} \quad o_2 = \mathbf{O} \cdot \mathsf{e}_2.$$

6.2.2.2 Entwurf von Experten

Die Zentren der kreisförmigen Trajektorien einzelner Punkte über mehrere Bilder einer Bildfolge können als Merkmale zur Segmentierung gewählt werden:

$$\mathbf{y}_i = \mathbf{o}_i.$$

Da der Merkmalsraum Euklidisch ist, existiert darin auch eine Metrik, die zur Berechnung des Fehlers verwendet werden kann.

Zur Bestimmung von Kreisen werden die Daten aus dem Korrespondenzvektor $\mathbf{x}_i = (\mathbf{x}_{1i}, \cdots, \mathbf{x}_{Mi})$ entnommen und in (6.14) eingesetzt. Da es mindestens dreier Punkte bedarf, um einen Kreis zu schätzen, sollte die Bildfolge eine Mindestlänge von drei haben. Anschließend werden die Zentren vom Netz gruppiert.

Diese Segmentierung ist schwächer als eine Schätzung mit allgemeinen Rotationen, da verschiedene Transformationen ähnliche Trajektorien erzeugen können. So werden z.B. zwei Objekte, die mit verschiedenen Geschwindigkeiten um einen Punkt rotieren, von dem vorgestellten Experten nicht voneinander differenziert.

6.2.3 Approximation mittels Fundamentalmatrizen

Die Epipolargeometrie eines Stereosystems stellt eine Reihe von Zwangsbedingungen zwischen den projizierten Punkten eines starren 3D-Objektes auf. Für ein einzelnes starres Objekt lässt sich dieses Szenario durch eine äquivalente Problemstellung in einem monokularen System beschreiben – die Bewegung eines rigiden Objektes wird mit einer Kamera aufgenommen. Dabei kann diese Bewegung auch als Transformation zwischen zwei Kameras in einem Stereosystem aufgefasst werden (in Abbildung 6.4 beispielhaft veranschaulicht). Die epipolaren

Zwangsbedingungen sind eindeutig für jedes sich bewegende Objekt bestimmt und können zur Segmentierung benutzt werden.

Die Epipolargeometrie wurde von mehreren Autoren sowohl zur Rekonstruktion der Gestalt und Bewegung [3, 103] als auch zur Segmentierung [72, 97, 104, 110] eingesetzt. Im Rahmen dieser Arbeit werden die epipolaren Zwangsbedingungen benutzt, um einen Experten zur Segmentierung des optischen Flusses neuronal zu modellieren.

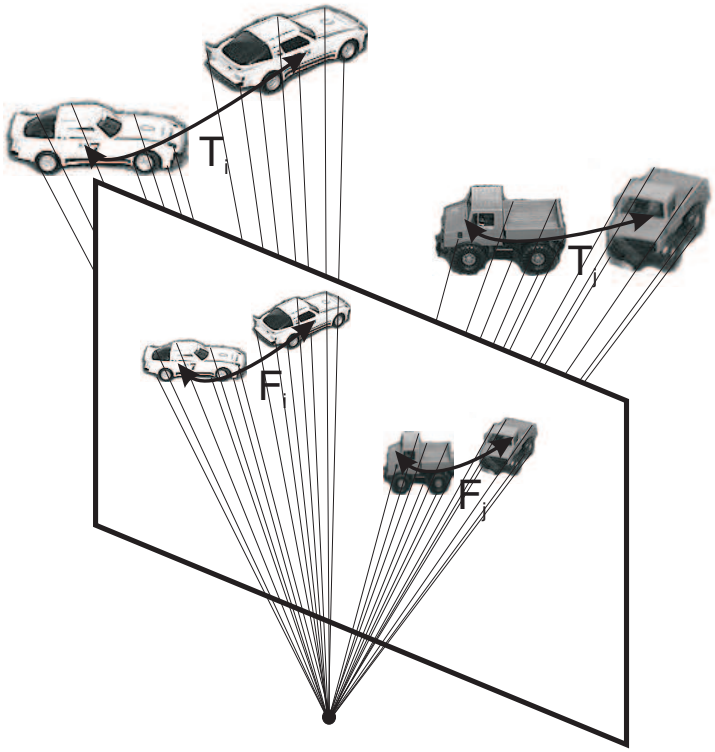


Abbildung 6.4: Ein exaktes mathematisches Modell der Projektion von 3D-Bewegungen kann durch die Epipolargeometrie mit Hilfe von Fundamentalmatrizen beschrieben werden.

6.2.3.1 Implementierung

Im Anhang A.3 wird die Epipolargeometrie eines Stereosystems diskutiert. Die Beziehung zwischen zwei korrespondierenden Punkten $\mathbf{x}_{ji} = (x^1_{ji}, x^2_{ji}) \in \mathbb{R}^2$ und $\mathbf{x}_{ki} = (x^1_{ki}, x^2_{ki}) \in \mathbb{R}^2$ in den Bildern j und k lässt sich mit der Fundamentalmatrix

[61] beschreiben[3]:

$$(\mathbf{x}_{ki},1)F(\mathbf{x}_{ji},1)^T = \begin{pmatrix} x^1_{ki} & x^2_{ki} & 1 \end{pmatrix} \begin{pmatrix} f_1 & f_2 & f_3 \\ f_4 & f_5 & f_6 \\ f_7 & f_8 & 1 \end{pmatrix} \begin{pmatrix} x^1_{ji} \\ x^2_{ji} \\ 1 \end{pmatrix} = 0. \quad (6.16)$$

Durch die Umformung

$$\begin{aligned} & x^1_{ji}x^1_{ki}f_1 + x^2_{ji}x^1_{ki}f_2 + x^1_{ki}f_3 \\ +& x^1_{ji}x^2_{ki}f_4 + x^2_{ji}x^2_{ki}f_5 + x^2_{ki}f_6 \\ & + x^1_{ji}f_7 + x^2_{ji}f_8 + 1 = 0 \end{aligned}$$

erhält man eine Darstellung der Gleichung (6.16) als Skalarprodukt von zwei Vektoren:

$$< \widetilde{\mathbf{x}}_i, \mathbf{w} >= -1 \quad (6.17)$$

mit

$$\widetilde{\mathbf{x}}_i = (x^1_{ji}x^1_{ki}, x^2_{ji}x^1_{ki}, x^1_{ki}, x^1_{ji}x^2_{ki}, x^2_{ji}x^2_{ki}, x^2_{ki}, x^1_{ji}, x^2_{ji}) \quad (6.18)$$

und

$$\mathbf{w} = (f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8).$$

Die Gleichung (6.17) lässt sich als Propagierungsfunktion eines High-Order-Neurons mit einer Struktur, die ähnlich der neuronalen Architektur [37] ist und eine Identität-Aktivierung besitzt, implementieren. Das Training kann mit dem üblichen Backpropagation-Algorithmus 4.1 erfolgen. Als Eingabe wird ein Vektor der Form (6.18) erwartet. Die Ausgabe ist ein Skalar, dessen Wert die Konfidenz der eingebetten Daten angibt – der Wert spiegelt die Qualität der epipolaren Zwangsbedingungen wieder (je näher an -1 desto besser).

Aus (6.18) folgt, dass mindestens acht Trainingsdaten benötigt werden, um alle Freiheitsgrade des Neurons beim Lernen auszuschöpfen. Dies stellt eine grosse Anforderung an das Neuron im Vergleich zu einem Neuron für die Berechnung der allgemeinen Rotation (zwei Datenpaare werden benötigt) dar, da der kombinatorische Aufwand zur Suche nach mindestens acht Punkten, die zu einem Bereich gehören, sehr hoch ausfallen kann. Es ist möglich, den Ablauf zu beschleunigen, indem zuerst eine Vorsegmentierung mit allgemeiner Rotation durchgeführt wird.

[3] siehe Anhang A.4.

6.2.3.2 Entwurf von Experten

Ähnlich wie bei den allgemeinen Rotationen lassen sich zwei Experten-Modelle aufstellen.

High-Order-Neuron-Einbettung als Merkmalsraum

Das Neuron zur Schätzung der Fundamentalmatrix kann direkt in Algorithmus 6.1 eingesetzt werden. Die Eingabedaten werden durch die Einbettung der Vektoren $(\mathbf{x}_{ji}, \mathbf{x}_{ki})$ des optischen Flusses gemäß (6.17) erzeugt. Die Ausgabe (Merkmal) wird auf Minus eins gesetzt:

$$\mathbf{y}_i = -1 \in \mathbb{R}.$$

Da die Ausgabe rellwertig ist (Euklidischer Raum), existiert auch eine passende Metrik zur Berechnung des Fehlers der Ausgabe im gesamten Netz.

Komponente der Fundamentalmatrix als Merkmale

Nimmt man an, dass die Bewegung über mehrere Bilder einer Bildfolge konstant ist, so folgt, dass auch die epipolaren Zwangsbedingungen für einzelne Punkte über mehrere Bilder konstant bleiben und folglich mit einem Neuron zur Schätzung der Fundamentalmatrix berechnet werden können. Zum Training werden die Daten paarweise $(\mathbf{x}_{ji}, \mathbf{x}_{(j+1)i})$ aus dem Korrespondenzvektor $\mathbf{x}_i = (\mathbf{x}_{1i}, \cdots, \mathbf{x}_{Mi})$ entnommen, gemäß (6.17) eingebettet und an das Neuron übergeben. Da mindestens acht Datenpaare zum Training notwendig sind, ist eine Bildfolge der Mindestlänge neun für die Schätzung erforderlich. Nach der erfolgten Schätzung werden die Komponenten der Fundamentalmatrix als Merkmale für die Segmentierung verwendet:

$$\mathbf{y}_i = \mathbf{w} = (f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8).$$

Anschließend werden die Pixel mit ähnlichen epipolaren Zwangsbedingungen gruppiert. Ein Distanzmaß für die Fundamentalmatrizen ist im Anhang C.2 angegeben.

6.3 Experimente

Viele Algorithmen zur Segmentierung des optischen Flusses verfügen weder über eine frei zugängliche Implementierung, noch gibt es für diese einheitliche Benchmarks. Oft bewirken kleine Änderungen der Steuerungsparameter eines Algorithmus erhebliche Qualitätsänderungen bei den Segmentierungsergebnissen. Aus diesem Grund sind im Rahmen dieser Arbeit keine direkten Vergleiche mit anderen Algorithmen aufgeführt. Vielmehr werden die bekannten Benchmarks zur Schätzung des optischen Flusses auch für die Evaluierung der Segmentierungsqualität benutzt.

Einige der getesteten Sequenzen liegen nur als reine Bildfolgen vor – es sind keine Informationen über den optischen Fluss verfügbar. Um diese Bildfolgen in eine Form zu bringen, die eine Weiterverarbeitung mit dem Algorithmus 6.1 ermöglicht, wurde eine Schätzung des optischen Flusses mit dem Kanade-Lucas-Tomasi-Algorithmus (KLT) durchgeführt. Der KLT-Algorithmus wurde aus einer Reihe von bekannten Verfahren [7, 75] aus mehreren Gründen ausgewählt:

- Es existiert ein Maß für die Zuverlässigkeit der extrahierten Merkmale. Durch eine Beschränkung auf Korrespondenzen mit hoher Konfidenz kann der Signal-Rausch-Abstand und somit die Qualität der Segmentierung erhöht werden.
- Der Algorithmus ist auf modernen Rechnern echtzeitfähig.
- Eine freie C-Implementierung ist verfügbar [9] und konnte ohne großen Aufwand in die zur Segmentierung entwickelte Softwarearchitektur als C++-Klasse integriert werden.

Weitere Details zur Implementierung des Algorithmus sind im Anhang B aufgeführt.

6.3.1 Würfel-und-Sphäre-Sequenz

Die Würfel-und-Sphäre-Sequenz ist eine künstlich erzeugte Bildsequenz und stammt aus einer Datenbank [22], die Benchmarks zum Testen von Algorithmen zur Schätzung des optischen Flusses enthält. Die Informationen über den echten optischen Fluss sind für diese Bildfolge bekannt.

Die Folge besteht aus 10 Bildern mit der Auflösung 400×300 Pixel. Es finden zwei Bewegungen über die gesamte Sequenz statt: Die Kugel rotiert entlang der

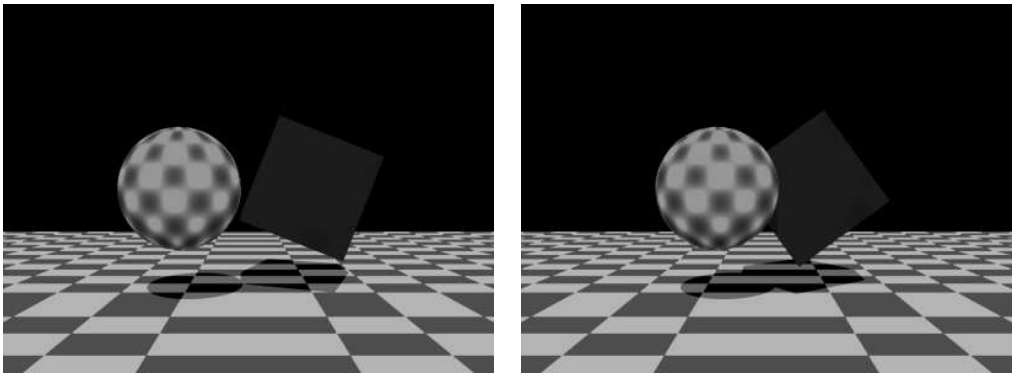


Abbildung 6.5: Zwei Bilder aus der Würfel-und-Sphäre-Sequenz.

vertikalen Achse parallel zur Bildebene nach rechts. Der Würfel rotiert entlang der zur Bildebene senkrechten Achse im Uhrzeigersinn und bewegt sich gleichzeitig nach links. In Abbildung 6.5 wird ein Ausschnitt aus der Bildfolge präsentiert.

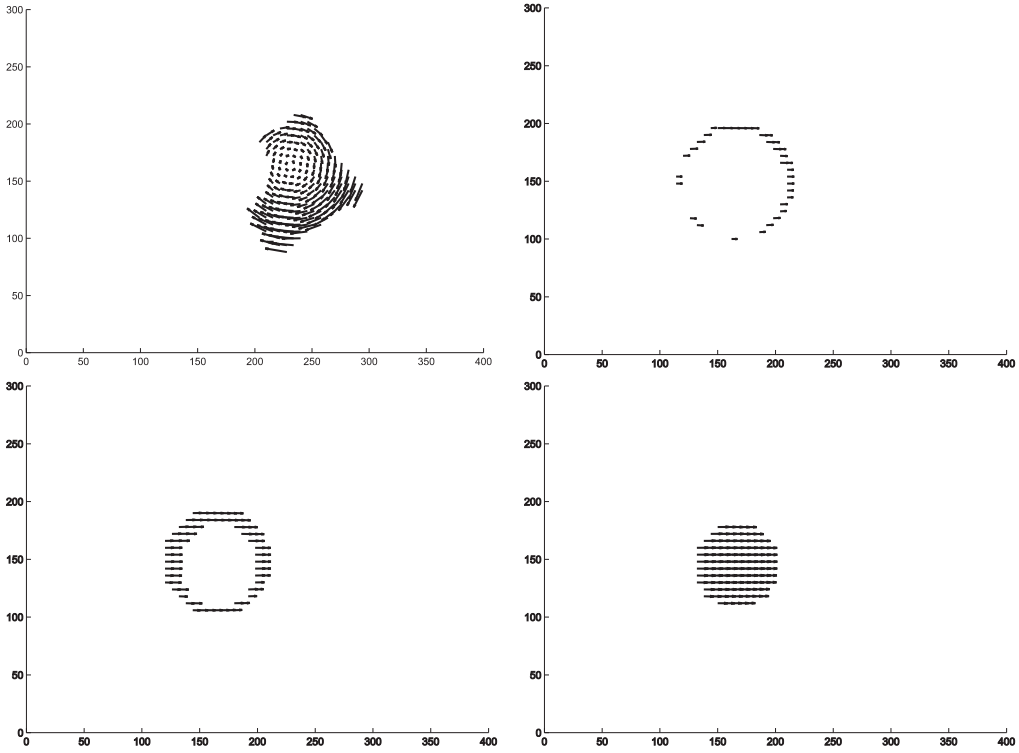


Abbildung 6.6: Würfel-und-Sphäre-Sequenz: Segmentierung mit der allgemeinen Rotationen.

Diese Bildfolge repräsentiert die Extremfälle der Projektion einer starren Bewe-

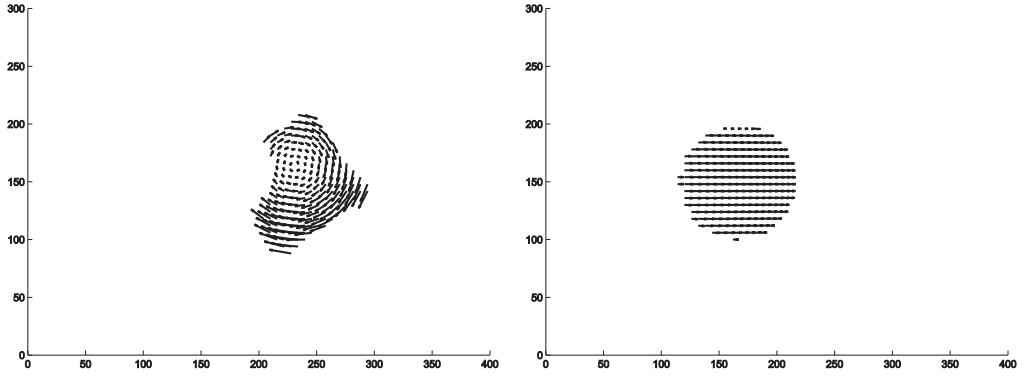


Abbildung 6.7: Würfel-und-Sphäre-Sequenz: Segmentierung mit Fundamentalmatrizen.

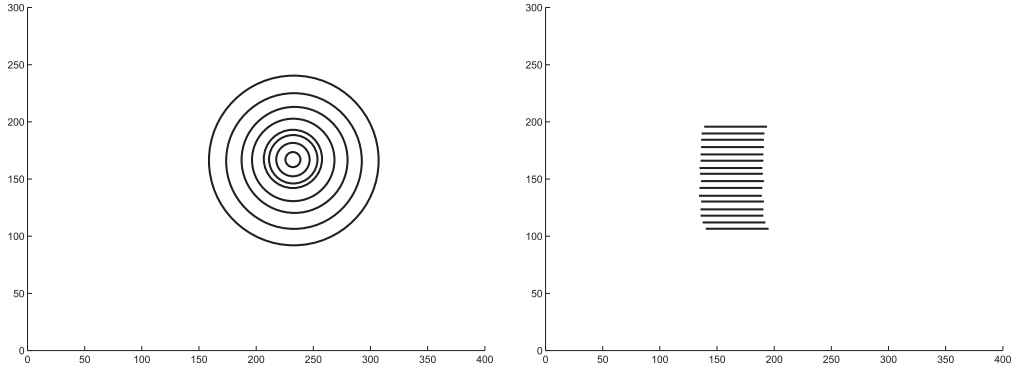


Abbildung 6.8: Würfel-und-Sphäre-Sequenz: Segmentierung mit Kreisen.

gung – die Bewegung in der optischen Ebene (Kugel) lässt sich schlecht linear approximieren, die Projektion der zur Bildebene parallelen Bewegung (Würfel) passt dagegen vollständig in das Konzept der Segmentierung mit allgemeinen Rotationen.

In den Abbildungen 6.6 und 6.7 sind die Ergebnisse einer Segmentierung mit allgemeinen Rotationen bzw. Fundamentalmatrizen visualisiert. Wie erwartet, fallen die Ergebnisse der Segmentierung für den Würfel gleich gut aus, dagegen wird die Kugel mit dem Experten besser erkannt, der die Zwänge der Kamerageometrie erfüllt (Fundamentalmatrix). Aus der Abbildung 6.6 lässt sich erkennen, dass die Kugel in Bereiche mit gleichen Translationskomponenten zerlegt wurde – der Experte hat die Daten maximal an das angenommene Modell (allgemeine Rotation) angepasst.

Ein Sonderfall stellt eine Segmentierung mit Kreisen dar. Da dieses Modell auf der gleichen Hypothese wie das Modell mit allgemeinen Rotationen aufbaut, fallen

die Ergebnisse bei dem Würfel entsprechend gut aus. Da Kreise mit unendlichen Radien sich als Geraden [81] ausdrücken lassen und durch einen endlichen Term in (6.14) beschrieben werden, ist es möglich, diese nach dem Merkmal "Richtung" zu gruppieren. Speziell die Kugel wird dadurch nur mit einem Bereich beschrieben. Die Ergebnisse für diesen Fall sind in der Abbildung 6.8 veranschaulicht.

Rauschen	0	0.5	1.0	1.5	2.0	2.5	3.0	3.5
AR	100%	98%	93%	90%	89%	84%	80%	75%
FM	100%	96%	90%	87%	87%	79%	71%	68%

Tabelle 6.4: Prozentualer Anteil von korrekt dem Würfel zugeordneten Punkten unter dem Rauscheinfluss (als Standardabweichung in Pixel angegeben) für die Segmentierung mit allgemeinen Rotationen (AR) bzw. mit Fundamentalmatrizen (FM).

Rauschen	0	0.5	1.0	1.5	2.0	2.5	3.0	3.5
AR	63%	62%	62%	60%	58%	55%	57%	52%
FM	100%	91%	84%	78%	74%	70%	69%	65%

Tabelle 6.5: Prozentualer Anteil von korrekt der Sphäre zugeordneten Punkten unter dem Rauscheinfluss (als Standardabweichung in Pixel angegeben) für die Segmentierung mit allgemeinen Rotationen (AR) bzw. mit Fundamentalmatrizen (FM).

Um die Robustheit des Systems zu testen, wurden die Daten mit zunehmendem Gauss'schem Rauschen belegt und an das Netz übergeben. Dabei wurde die Qualität der Segmentierung einzeln für beide Objekte überprüft. Die Ergebnisse sind in den Tabellen 6.4 und 6.5 zusammengefasst. Es lässt sich eine Tendenz bezüglich der Robustheit zugunsten der Segmentierung mit allgemeinen Rotationen erkennen. Die Qualität der Segmentierung entspricht zwar modellbedingt nicht immer exakt der Grundwahrheit, trotzdem bleiben die (teilweise falschen) Ergebnisse relativ stabil unter dem Einfluss von Rauschen.

In einem weiteren Test wurde die Fähigkeit des Systems überprüft, die Bewegungen von Punkten im Bild auch unter dem Einfluss von Rauschen im Voraus zu schätzen. Dabei wurden die ersten beiden Bilder der Folge zur Segmentierung bzw. Schätzung der Bewegungsparameter benutzt. Darauf basierend wurde die Prädiktion für die restlichen acht Frames (Bilder) berechnet. Da eine Prädiktion nur für die Fälle möglich ist, bei denen eine explizite Transformation zwischen dem Anfangs- und dem Endpunkt angegeben ist, lässt sich für diese Aufgabe nur

$\frac{\Delta\text{-Frame}}{\text{Rauschen}}$	1	2	3	4	5	6	7	8
0	0.1	0.1	0.2	0.2	0.2	0.3	0.3	0.3
1	1.2	1.2	1.2	1.3	1.3	1.5	1.5	1.5
2	2.2	2.3	2.5	2.6	2.6	2.6	2.8	3.0
3	3.1	3.3	3.4	3.6	3.8	4.0	4.2	4.3

Tabelle 6.6: Absoluter Fehler (in Pixel) für die Prädiktion der Bewegung des Würfels über mehrere Frames in Abhängigkeit vom Rauschen (als Standardabweichung in Pixel angegeben).

das auf den allgemeinen Rotationen basierende Netz einsetzen. Ausgehend von den vorherigen Ergebnissen wurde für dieses Szenario die Voraussage sinnvollerweise nur für den Würfel berechnet. In Tabelle 6.6 ist der absolute Fehler (in Pixel) in Abhängigkeit vom Δ-Frame und vom Rauschniveau angegeben. Nur die Punkte, die über die gesamte Folge sichtbar sind, wurden berücksichtigt. Die Ergebnisse zeigen, dass die Qualität der Prädiktion stark von dem Niveau des Grundrauschens in einer fast linearen Weise abhängt.

6.3.2 Automobil-Sequenzen

Folgende Ergebnisse stammen aus eine Reihe von Experimenten, die in einer kontrollierten Umgebung (konstante Beleuchtung, die Bewegungen der zu beobachtenden Objekte sind steuerbar) entstanden sind, und stellen einen Zwischenschritt zu einer frei gestalteten Szene dar.

Für die Experimente wurden zwei Spielzeugautos eingesetzt. Verschiedene Szenarien wurden getestet. 10 Bildfolgen der Länge 30 bist 70 Frames wurden im Robotiklabor des Lehrstuhls mit der Auflösung 384×288 Pixel aufgenommen. Anschließend wurde der optische Fluss mit dem KLT-Algorithmus geschätzt. Die Grenze für die Anzahl der zu extrahierenden Korrespondenzen wurde basierend auf dem Wert, den die Entwickler des Algorithmus für die im Experiment benutzte Bildgröße vorschlagen, auf 250 gesetzt. Es wurden nur die Korrespondenzen berücksichtigt, die über die gesamte Länge der jeweiligen Bildfolge auffindbar waren. Nur eine Segmentierung mit allgemeinen Rotationen lieferte während der gesamten Experimente brauchbare Resultate, da einzelne Objekte teilweise aus weniger als acht Punkten bestanden und somit eine sinnvolle Segmentierung mit Fundamentalmatrizen unmöglich machten. Die Schätzung des optischen Flusses und die anschließende Segmentierung nahm für jedes Bildpaar einer Bildfolge auf einem Pentium IV 2.4GHz System insgesamt zwischen 100ms und 800ms in

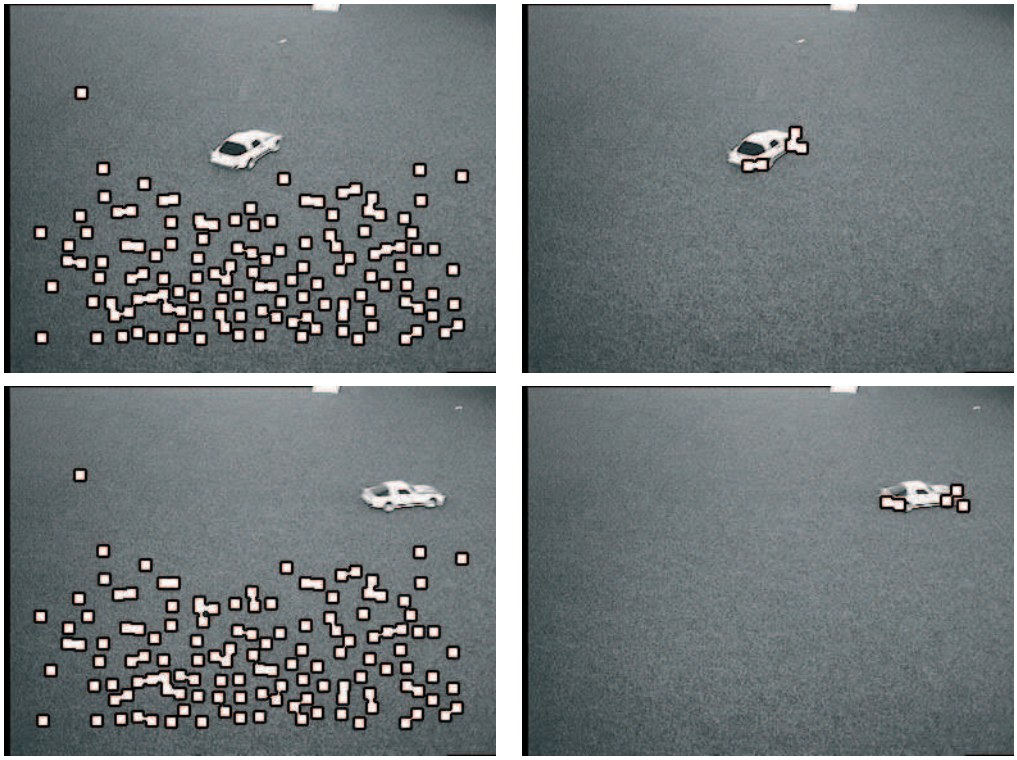


Abbildung 6.9: Die Ergebnisse einer Segmentierung einer kreisförmigen (3D) Bewegung eines Spielzeugautos aus einer Bildfolge der Länge 20. Links – Hintergrund. Rechts – Bewegung. Oben – Anfangsphase der Bewegung. Unten – Endphase der Bewegung.

Anspruch, wobei eine Initialisierung des Netzes mit den Ergebnissen der vorhergegangenen Segmentierung die Laufzeit um ca. 30% reduzierte.

In den Abbildungen 6.9 und 6.10 sind zwei Fälle dargestellt, die von besonderem Interesse sind. Die erste Abbildung demonstriert die Fähigkeit des Netzes, eine Bewegung aus einem verrauschten optischen Fluss zu segmentieren, die sich stark von einer reinen Verschiebung unterscheidet – eine Gruppierung mit einem häufig eingesetzten Verfahren, das nach ähnlichen Vektoren aus dem optischen Fluss sucht (gleiche Translation), würde im besten Fall das Auto als mehrere Objekte erkennen. In der zweiten Abbildung sind die Ergebnisse einer Segmentierung von zwei Spielzeugautos, die sich auf die gleiche Weise geradlinig bewegen, dargestellt. Da die zugrunde liegende allgemeine Rotation für beide Objekte gleich ist, erkennt das Netz nur eine Bewegung. Diese Eigenschaft kann sowohl im negativen, als auch im positiven Sinne gewertet werden. Einerseits ist die (Un)Fähigkeit des Netzes, Objekte mit einer gleichen Bewegung nicht differenzieren zu können,

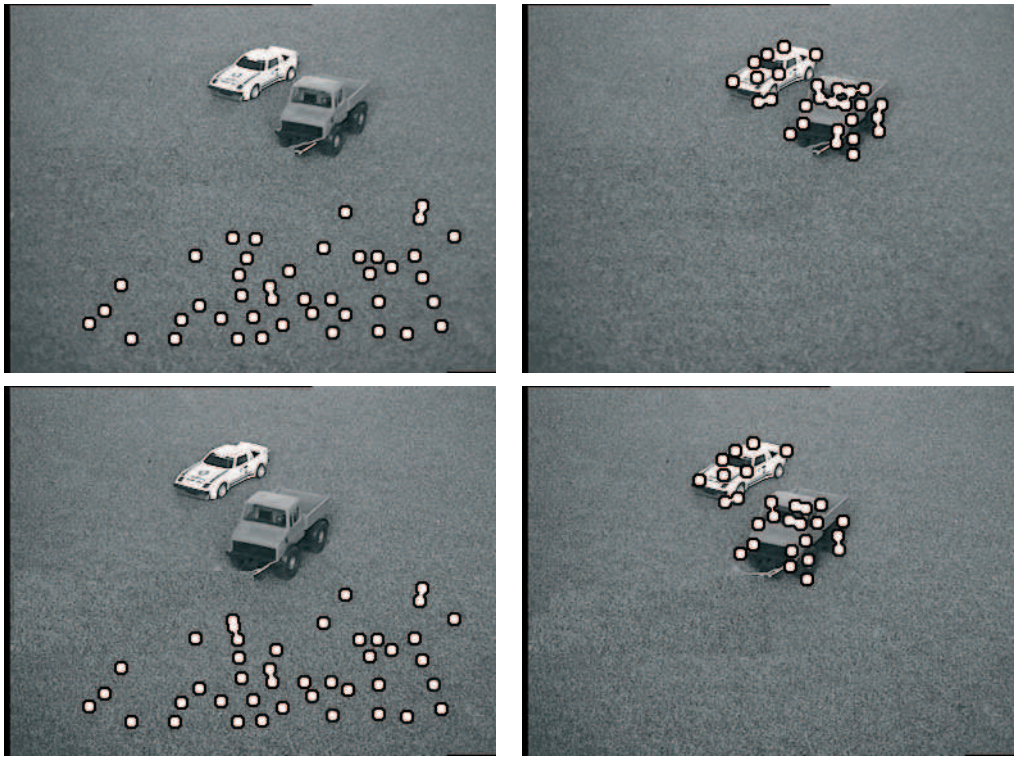


Abbildung 6.10: Die Ergebnisse einer Segmentierung der Bewegung von zwei Spielzeugautos aus einer Bildfolge der Länge 30. Links – Hintergrund. Rechts – Bewegung. Oben – Anfangsphase der Bewegung. Unten – Endphase der Bewegung.

nicht wünschenswert, andererseits kann diese Eigenschaft von Vorteil sein, wenn durch Verdeckungen einige Teile eines Objektes nicht sichtbar sind.

6.3.3 Hamburger-Taxi-Sequenz

In diesem Experiment wurde das Netz mit einer bekannten Hamburger-Taxi-Bildfolge [6] trainiert, die oft als Benchmark zur Schätzung des optischen Flusses eingesetzt wird. Die Sequenz besteht aus 20 Bildern mit der Auflösung 256×190 Pixel und stellt ein schwieriges Problem für viele Algorithmen dar.

Im Rahmen des Experimentes wurde der optische Fluss über die gesamte Bildfolge mit dem KLT-Verfahren berechnet. Dabei wurde die Anzahl der maximal zu findenden Korrespondenzen auf 150 gesetzt. Anschließend wurde eine Segmentierung mit allgemeinen Rotationen durchgeführt. Um den Signal-Rausch-Abstand

Abbildung 6.11: Taxi-Sequenz: Der gesamte optische Fluss im dritten und im dreizehnten Bild.

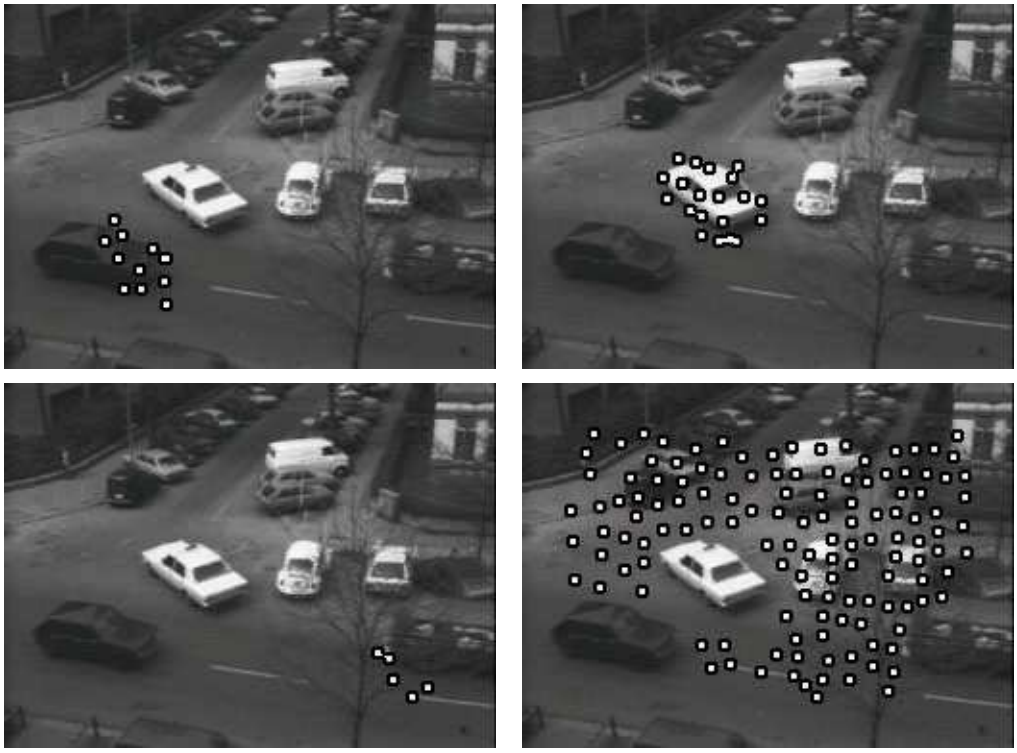


Abbildung 6.12: Taxi-Sequenz: Die Ergebnisse einer Segmentierung.

zu vergrößern, wurden die Trainingsdaten nicht aus benachbarten Bildern der Folge, sondern aus den Bildern mit Δ-Frame 10 erzeugt, da die Qualität des geschätzten optischen Flusses nicht sehr gut ausfiel: So wanderten teilweise die extrahierten Punkte des statischen Hintergrundes erheblich über die gesamte Sequenz, wie in Abbildung 6.11 beispielhaft dargestellt ist. Ein Auto wurde während der Bewegung in der Bildfolge ab dem 13-ten Frame teilweise von einem Baum verdeckt, was die Qualität der Schätzung zusätzlich reduzierte. Der Berechnungsaufwand für jedes Bildpaar betrug auf einem Pentium IV 2.4GHz System zwischen 150ms und 400ms. Von 10 getesteten Bildpaaren lieferten nur die ersten 3 eine zufriedenstellende Segmentierung. In der Abbildung 6.12 sind die Ergebnisse für das Bildpaar $(3, 13)$ präsentiert.

Diese Ergebnisse zeigen deutlich, dass ein Algorithmus, der rein auf Punktbewegungen basiert, und keine weiteren Informationen wie Helligkeit, Farbe oder Gestalt usw. berücksichtigt, insbesondere bei stark verrauschten Bildern nur bedingt zur Segmentierung geeignet ist.

6.3.4 Kieler-Kreuzung-Sequenz

Um einerseits den Signal-Rausch-Abstand zu vergrößern, und andererseits die Prädiktionsfähigkeit des Netzes besser testen zu können, wurde eine Bildfolge der Länge 60 aufgenommen. Die Auflösung betrug 768×576 Pixel. Ausschnitte der Bildfolge sind in der Abbildung 6.13 dargestellt. Anschließend wurde der optische Fluss mit KLT-Algorithmus geschätzt. Der Schwellwert für die maximale Anzahl von Korrespondenzen wurde auf 1000 gesetzt. Es wurden nur diejenigen Korrespondenzen genommen, die über die gesamte Bildfolge auffindbar waren.

Eine Segmentierung über die gesamte Sequenz wurde durchgeführt. Zur Erhöhung des Signal-Rausch-Abstandes sind die Bilder mit Δ-Frame 5 zum Training benutzt worden. Die Laufzeit des Verfahrens betrug für das erste Bildpaar ca. 1200ms auf einem Pentium IV 2,4GHz System (allgemeine Rotationen). Für die nachfolgenden Bildpaare wurde diese Zeit durch eine Vorinitialisierung auf ca. 700ms reduziert. Ein Netz mit Fundamentalmatrizen-Experten lieferte (visuell) nicht zufriedenstellende Ergebnisse.

Ein Beispiel der Segmentierung mit allgemeinen Rotationen ist in der Abbildung 6.14 gegeben. Es lässt sich erkennen, dass Objekte (auch Schatten), die eine ähnliche Bewegung durchführen als ein Segment – wie schon im Experiment 6.10 – erkannt werden (Abbildung 6.14 oben rechts). Die Bewegungen auf ähnlichen Trajektorien mit verschiedenen Geschwindigkeiten werden aber vom Netz differenziert (Abbildung 6.14 unten).

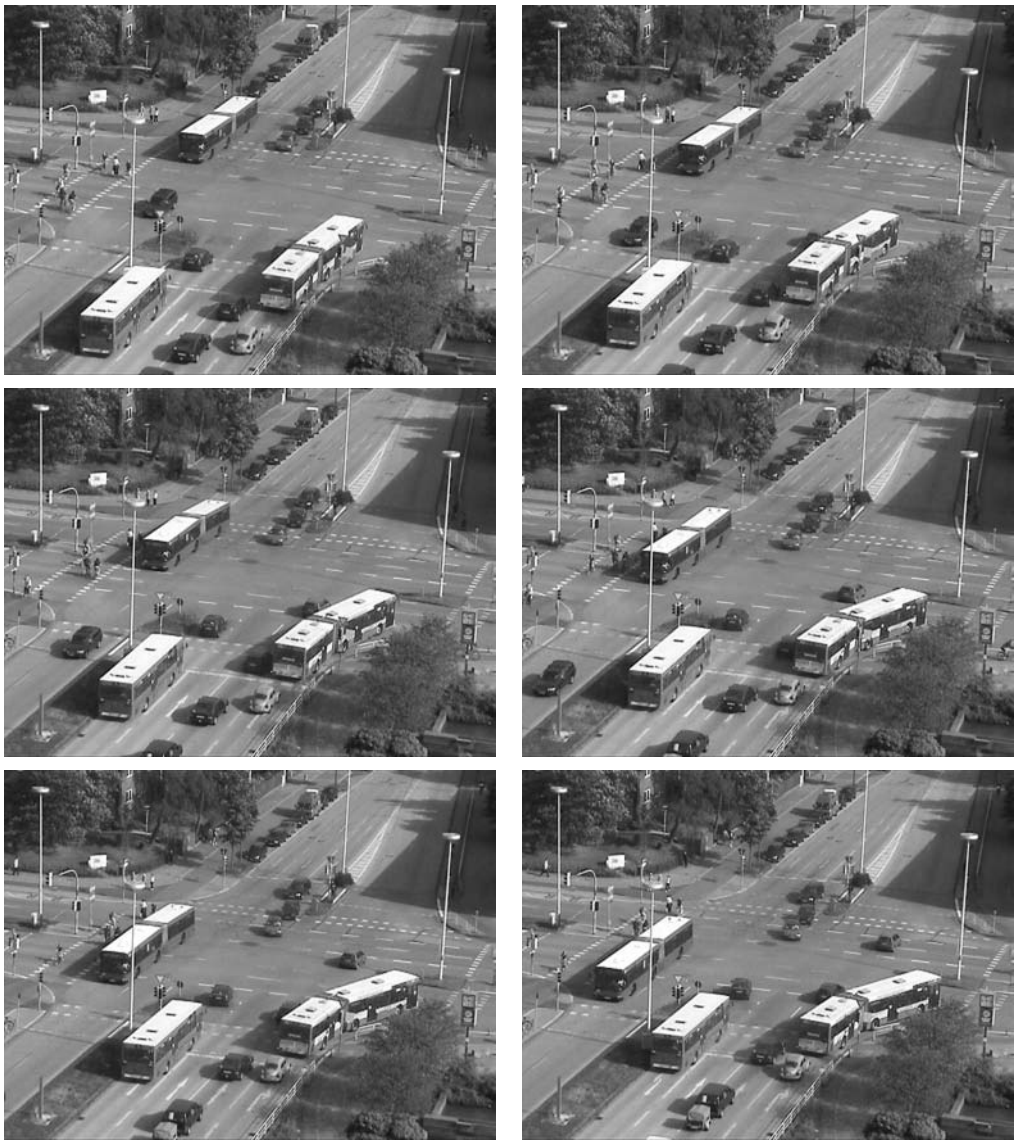


Abbildung 6.13: Kieler-Kreuzung-Sequenz: Ein Ausschnitt aus der Bildfolge mit Δ-Frame 10.

Abbildung 6.14: Kieler-Kreuzung-Sequenz: Die Ergebnisse einer Segmentierung mit allgemeinen Rotationen.

Die nach der Gleichung (6.14) berechneten kreisförmigen Trajektorien sind in der Abbildung 6.15 visualisiert. Es wurden Korrespondenzen über 10 Bilder der Sequenz zur Schätzung der Kreise eingesetzt. Die Ergebnisse zeigen deutlich, dass in echten verrauschten Bildern die Trajektorien nicht mit der zur Segmentierung erforderlichen Präzision berechnet werden können, und somit ein nur auf kreisförmigen Trajektorien basierender Ansatz nicht praxistauglich ist.

Die Tauglichkeit des Netzes zur Segmentierung nach Parametern der allgemeinen Rotation wurde in einem weiteren Experiment untersucht. Dabei wurden Korrespondenzen eines Abschnittes der Länge 10 aus der Bildfolge zur Schätzung der Bewegungsparameter (allgemeine Rotation) einzelner Punkte benutzt. Die Parameter für die gefundenen Transformationen sind als Gruppen in der Abbildung 6.16 visuell dargestellt. Diese entsprechen den Ergebnissen einer Segmentierung, wie sie im Experiment 6.14 durchgeführt wurde. Es lässt sich erkennen, dass die Bewegungsparameter keine kompakten Regionen für einzelne Objekte bilden[4], und somit als Segmentierungskriterium für echte verrauschte Bilder un-

[4] Dies kann teilweise auch an der Qualität des geschätzten optischen Flusses liegen, was aller-

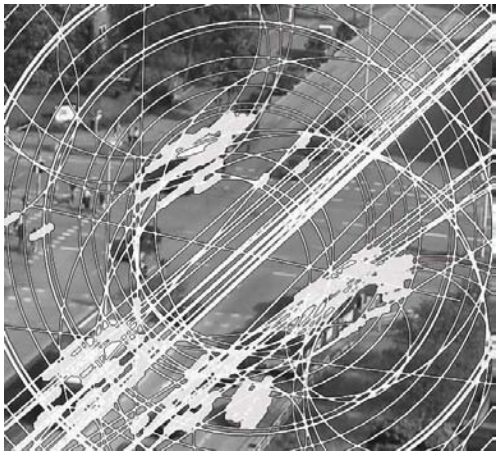


Abbildung 6.15: Kieler-Kreuzung-Sequenz: Die kreisförmigen Trajektorien einzelner Punkte.

geeignet sind.

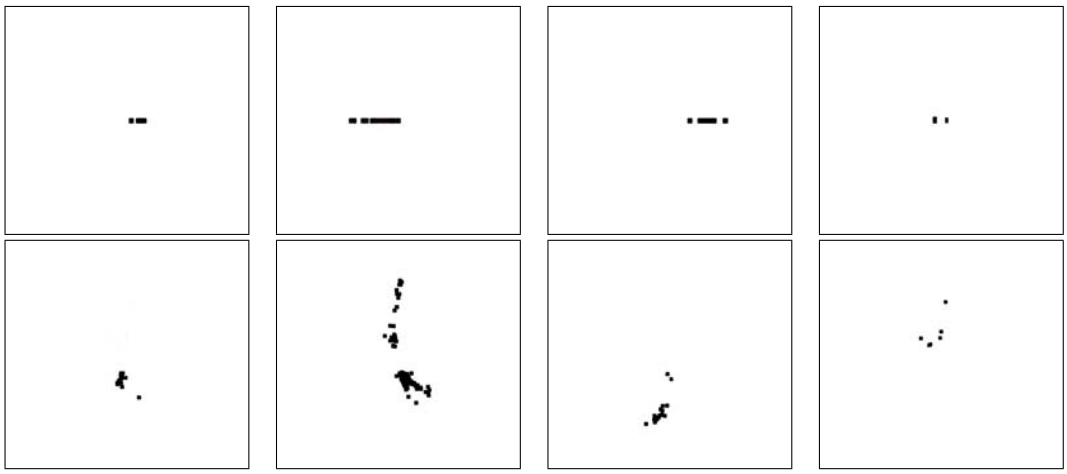


Abbildung 6.16: Kieler-Kreuzung-Sequenz: Oben – Rotationswinkel einzelner Segmente als 1D-Werte. Unten – Translationskomponenten (x, y) einzelner Segmente als 2D-Werte.

Die Fähigkeit des Netzes, eine gute Prädiktion durchzuführen, wurde in einem weiteren Experiment untersucht. Verglichen wurden die Voraussagen mittels allgemeiner Rotationen mit denen einer Prädiktion, die auf einer Translation basierte, welche direkt aus einzelnen Elementen des optischen Flusses über mehrere Bilder berechnent wurde. Abbildung 6.17 gibt die Ergebnisse des Vergleichs wie-

dings das Urteil über die Praxistauglichkeit dieses Ansatzes nur bestätigt.

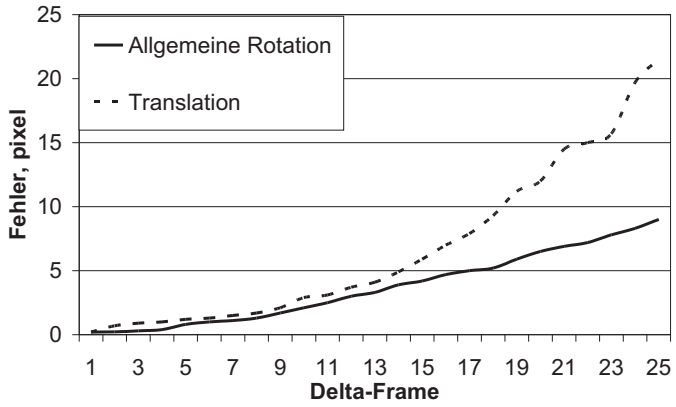


Abbildung 6.17: Kieler-Kreuzung-Sequenz: Prädiktion mit allgemeinen Rotationen im Vergleich zur Prädiktion mit reinen Translationen.

der. Da zur Schätzung der allgemeinen Rotationen (und auch reinen Translationen) die Bildpaare mit einem Δ-Frame 10 benutzt wurden, veranschaulicht diese Grafik sowohl die Inter- als auch die Extrapolationseigenschaften des Netzes. Eine Prädiktion aufgrund der Hypothese der allgemeinen Rotationen weist dabei ein viel gutartigeres Verhalten auf, als eine auf Translationsannahme basierende Technik.

6.4 Zusammenfassung

Die Segmentierung eines Bildes stellt hohe Anforderungen an das verwendete Verfahren. Es steht eine Anzahl von Merkmalen zur Verfügung, die sich für diese Aufgabe eignen. Im Rahmen dieser Arbeit wurden von der Eigenschaft der geometrischen Algebren ausgehend, allgemeine Rotationen effizient berechnen zu können, die auf der Geometrie einer Bewegung basierenden Merkmale – der optische Fluss – für diese Zwecke eingesetzt. Eine generische Architektur zur Segmentierung wurde basierend auf den klassischen neuronalen Konzepten entworfen. Spezielle Experten sind entwickelt worden, die verschiedenen Aspekte der Kamerageometrie modellieren können.

Die durchgeführten Experimente belegen, dass die Komplexität eines Experten eine direkte Auswirkung auf die Robustheit des Systems hat. Konkret bedeutet das: Zwar werden die geometrischen Zwänge bei einem Fundamentalmatrix-Experten komplett erfüllt, aber der zur Segmentierung benötigte Aufwand sowie die Stabi-

Experte	Geom.	Präd.	Eingabe (Dim.)	Ausgabe (Dim.)	Init.	Robustheit	LZ
AR1	o	+	$\mathbf{x}_{ji}(2)$	$\mathbf{x}_{ki}(2)$	2PP	o	+
AR2	o	+	$\mathbf{x}_{ji}(2)$	$R_i(3)$	3F	-	-
K	-	o	$\mathbf{x}_{ji}(2)$	$\mathbf{o}_i(2)$	3F	-	o
FM1	+	-	$\widetilde{\mathbf{x}}_i(8)$	-1(1)	8PP	-	-
FM2	+	-	$\mathbf{x}_{ji}(2)$	$F_i(8)$	9F	-	-

Abkürzung	Bedeutung	Abkürzung	Bedeutung
AR1	allgemeine Rotation (PP)	PP	Punktpaar
AR2	allgemeine Rotation (parametrisch)	F	Frame
K	Kreise	Geom.	erfüllt Geometrie
FM1	Fundamentalmatrix (PP)	Präd.	Prädiktionsfähigkeit
FM2	Fundamentalmatrix (parametrisch)	Init.	benötigte Initialisierung
Dim.	Dimension	LZ	Laufzeit

Tabelle 6.7: Experten zur Segmentierung: Eine Zusammenfassung.

lität gegenüber dem Rauscheinfluss stellen ein Problem dar. Unter allen Experten besitzt der auf den allgemeinen Rotationen basierende die größte Flexibilität. Der Einsatz der geometrischen Algebra stabilisiert dieses neuronale Modell zusätzlich.

Die Fähigkeit, eine sinnvolle Inter- bzw. Extrapolation nach einer Anpassung des Modells anhand der Trainingsdaten durchzuführen, bestimmt die Qualität der Generalisierung eines neuronalen Systems. Speziell im Fall des optischen Flusses werden die Merkmale implizit als Bewegungen von Objekten über mehrere Bilder einer Bildfolge kodiert. Die Generalisierungsfähigkeit eines darauf basierenden Netzes bestünde darin, eine ähnliche Segmentierung auch bezüglich der Bewegungsparameter über mehrere Frames zu behalten, ohne das Netz völlig neu trainieren zu müssen, bzw. eine möglichst präzise Prädiktion der Bewegungen einzelner Objekte zu berechnen. Diese Eigenschaften wurden im experimentellen Teil anhand einer Reihe von Tests (Laufzeit, Prädiktion) untersucht. Auch hier lieferte der auf den allgemeinen Rotationen basierende Experte gute Ergebnisse.

In der Tabelle 6.7 sind alle für die Arbeit relevanten Eigenschaften zusammengefasst.

Kapitel 7

NEURONALE STEUERUNG EINES ROBOTERARMES

Der Entwurf eines Roboter-Systems mit der Fähigkeit, einen Bewegungsablauf zu planen, der einerseits die gestellte Zielstellung erfüllt und andererseits die sich ggf. ändernden Umgebungsbedingungen berücksichtigt, stellt einen wichtigen Aspekt der autonomen Robotik dar. Je nach Aufgabe ist eine Modellierung als ein autonomes Fahrzeug, als ein Roboterarm oder als ein komplexes Robotersystem erforderlich. Die Umgebungsbedingungen können dabei als Einschränkungen sowohl bezüglich der möglichen räumlichen Positionierung des Systems als auch bezüglich der Art der zulässigen Bewegungen in das Modell eingebracht werden.

Die Planung eines Bewegungsablaufs kann allgemein in Termen des Konfigurationsraumes ($\mathcal{C}$-space bzw. $\mathcal{C}$-Raum) [57] – eines Raumes, der die Parameter enthält, welche die Bewegung eines Roboter-Systems steuern – beschrieben werden. Die Verwendung des $\mathcal{C}$-Raumes allein reicht oft nicht aus, um eine robuste Planung durchzuführen – vor allem bei komplexen dynamischen Umgebungsbedingungen. Eine Erweiterung des Konfigurationsraumes um die Komponenten, die Informationen wie z.B. visuelle Rückkopplung, zusätzliche Steuerungsparameter usw. [93, 114] enthalten, kann die Stabilität des Systems erheblich erhöhen. Zulässige Bewegungen finden dabei auf einer Mannigfaltigkeit in diesem Raum statt, die durch die auf den Umgebungsbedingungen basierenden Einschränkungen induziert wird, der so genannten ”perceptual control manifold” (PCM) [57].

Im Rahmen dieser Arbeit wird die Steuerung eines Roboterarmes mittels neuronaler Netze untersucht. Speziell wird ein DCS-Netz zum Lernen der PCM eines (Menschen- oder Roboter-)Armes eingesetzt. Das Ziel dabei besteht darin, eine Abbildung zwischen den $\mathcal{C}$-Räumen von zwei verschiedenen Armen mit typischerweise unterschiedlicher Kinematik zu finden. Als Eingabe dienen Posen (Konfigurationen) des ersten Armes. Die Ausgabe des Netzes liefert die korre-

spondierende Pose des zweiten Armes und wird zur Planung bzw. Durchführung einer Bewegung eingesetzt. Insbesondere ermöglicht dieser Ansatz, menschenähnliche Gesten auf einen Roboterarm zu übertragen oder auch den Bewegungsraum eines Armes auf spezielle Trajektorien zu beschränken. Die Repräsentation der Gelenkstellungen durch Rotoren einer geometrischen Algebra erlaubt dabei dem Netz, die gestellte Aufgabe effizient zu lösen.

Im Folgenden wird die oben beschriebene Aufgabe formalisiert, sowie ein zur Steuerung geeignetes DCS-Netz vorgestellt.

7.1 Problemstellung

Betrachtet man einen (Roboter- bzw. Menschen-)Arm mit n Gelenken und Gliedmaßen einer bekannten festen Länge – beispielhaft in Abbildung 7.1 dargestellt – so ist eine 3D-Pose eines solchen Systems durch die Gelenkstellungen eindeutig repräsentiert. Die Stellung einzelner Gelenke kann je nach den physikalischen Eigenschaften des Systems und der Art der gestellten Aufgabe auf verschiedene Weisen parametrisiert werden. Allgemein werden Parameter, die eine Pose beschreiben, durch einen Punkt im Konfigurationsraum $\mathcal{C}$ des Armes repräsentiert. Die Menge aller zulässigen Posen bildet eine Mannigfaltigkeit in diesem Raum. Eine optimale Bewegung des Armes wird durch den Anfangs- und den Endpunkt, sowie den kürzesten Weg auf der $\mathcal{C}$-Mannigfaltigkeit beschrieben.

Die Steuerung des Armes kann anhand verschiedener Kriterien sowohl explizit über die direkte Veränderung der Gelenkstellungen als auch implizit erfolgen. Bei einem impliziten Ansatz werden die aus externen Messungen gewonnenen Merkmale – wie z.B. im Fall einer visuellen Rückkopplung [93] usw. – zur Ansteuerung des Systems benutzt. Dabei werden Daten aus dem Merkmalsraum $\mathcal{S}$ auf den Konfigurationsraum des Armes allgemein bzw. auf die $\mathcal{C}$-Mannigfaltigkeit speziell abgebildet:

$$\mathbf{C} : \mathcal{S} \longrightarrow \mathcal{C}.$$

Diese Abbildung ist in der Literatur [57] in einer impliziten Form als PCM – das Produkt $\mathcal{S} \times \mathcal{C}$ des Merkmalsraumes und Konfigurationsraumes – bekannt (vergleiche Kapitel 2, Kern-Funktion). Die Steuerung des Armes kann als eine Planung der Trajektorie auf der PCM definiert werden. Die Beschränkung der möglichen Bewegungen auf diese Mannigfaltigkeit stellt dabei sicher, dass die Umgebungsbedingungen während des gesamten Steuerungsvorgangs erfüllt bleiben.

Im Rahmen dieser Arbeit wird die Zielstellung verfolgt, einen Zusammenhang zwischen der Kinematik zweier verschiedener Arme herzustellen: Die Posenver-

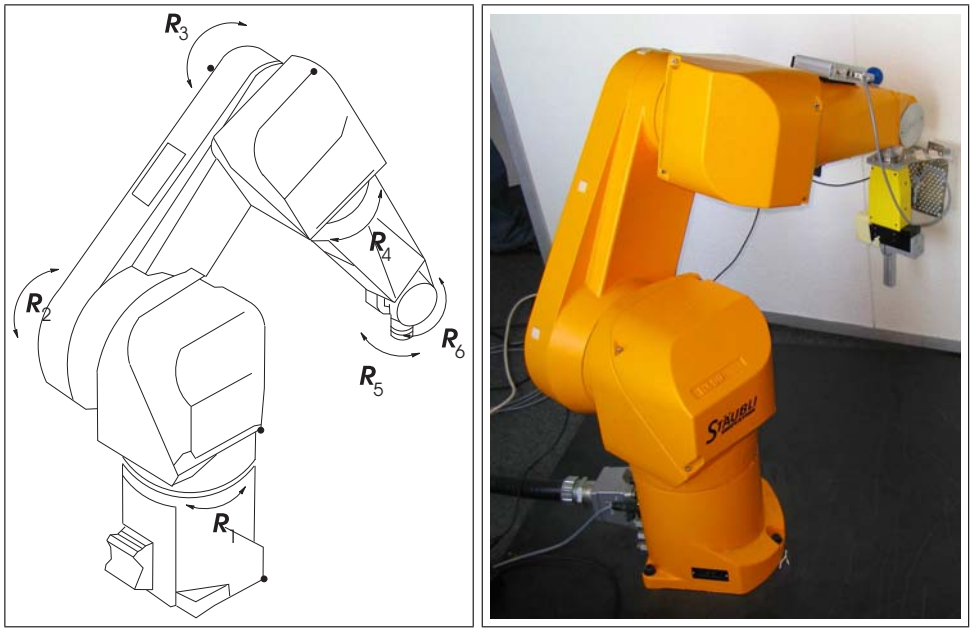


Abbildung 7.1: Stäubli RX90: Ein Roboterarm mit 6 Gelenken.

änderungen eines Armes sollen dabei zur Steuerung von Bewegungen eines anderen Armes unter Berücksichtigung der jeweiligen mechanischen und aufgabespezifischen Einschränkungen eingesetzt werden. Konkret wird dies realisiert, indem der Konfigurationsraum $\mathcal{C}'$ des steuernden Armes als Merkmalsraum in der PCM des gesteuerten Armes verwendet wird:

$$\mathcal{S} \equiv \mathcal{C}'.$$

Die Steuerungsaufgabe ist in diesem Fall als Suche nach einer Abbildung zwischen den Konfigurationsräumen bzw. den entsprechenden $\mathcal{C}$-Mannigfaltigkeiten der beiden Arme

$$\mathbf{C} : \mathcal{C}' \longrightarrow \mathcal{C}$$

formalisiert.

7.2 Implementierung

Mit dem kompletten Wissen über die Kinematik des Armes und die zu jeder Gelenkstellung korrespondierenden Parameter im Merkmalsraum ist eine analytische Modellierung der PCM möglich. In praktischen Anwendungen lässt sich die PCM

analytisch allerdings nur sehr schwer beschreiben, da die vorliegenden Informationen für ein solches Modell fast immer in einer nicht ausreichenden Menge vorhanden sind. Ein neuronaler Ansatz zum Lernen der Topologie dieser Mannigfaltigkeit stellt in diesem Fall eine sinnvolle Alternative dar [114]. Eine DCS-Architektur ist aufgrund ihrer optimal topologieerhaltenden Eigenschaft (siehe Kapitel 4) für diese Aufgabe besonders gut geeignet.

Während der Trainingsphase lernt das Netz anhand der Beispieldaten, die im vorgestellten Szenario in Form von korrespondierenden Posen vorliegen. Dabei werden die Neuronen im Konfigurationsraum – genauer gesagt auf der $\mathcal{C}$-Mannigfaltigkeit – des steuernden Armes (Eingaberaum) verteilt und eine Nachbarschaftsstruktur wird gebildet. Die an jedes Neuron gekoppelten Vektoren liegen dabei im Konfigurationsraum des gesteuerten Armes (Ausgaberaum).

Da zum Lernen mit einem DCS-Netz ein Distanzmaß sowohl für den Eingabe- als auch für den Ausgaberaum benötigt wird – wie in Algorithmus 4.6 skizziert – ist eine geeignete Darstellung der Konfiguration eines Armes von grosser Bedeutung. Typischerweise wird der $\mathcal{C}$-Raum eines Armes durch die Menge aller möglichen Gelenkstellungen – die so genannten Joints – beschrieben. Ein Joint gibt dabei nichts anderes als die Rotation des jeweiligen Gelenkes bezüglich eines Referenz-Koordinatensystems an. Eine Darstellung von Rotationen mittels der Euklidischen geometrischen Algebra als Rotoren wurde im Rahmen dieser Arbeit zur Modellierung von $\mathcal{C}$-Räumen eingesetzt:

$$\begin{aligned} \mathcal{C} &:= \underbrace{\mathcal{G}_m^+ \times \cdots \times \mathcal{G}_m^+}_{n}, \quad m \in \{2,3\} \\ c &:= (R_1, \cdots, R_n) \in \mathcal{C} | \quad \forall\, 0 < i < n : R_i \widetilde{R_i} = 1, \end{aligned} \tag{7.1}$$

wobei n die Anzahl und m die Art (2D oder 3D) der Gelenke des Armes angeben. Die Wahl dieser Parametrisierung erfolgte aufgrund von Vorteilen, die im folgenden Abschnitt diskutiert sind.

7.2.1 Distanzmaß in $\mathcal{C}$-Räumen

Wird ein mehrgelenkiger Arm als eine geordnete Menge von Rotationen modelliert, so können die Berechnung von Abständen zwischen Posen bzw. die Schätzung einer gewichteten mittleren Pose auf der Basis der entsprechenden Berechnungsvorschriften für einzelne Gelenke (Rotationen) erfolgen.

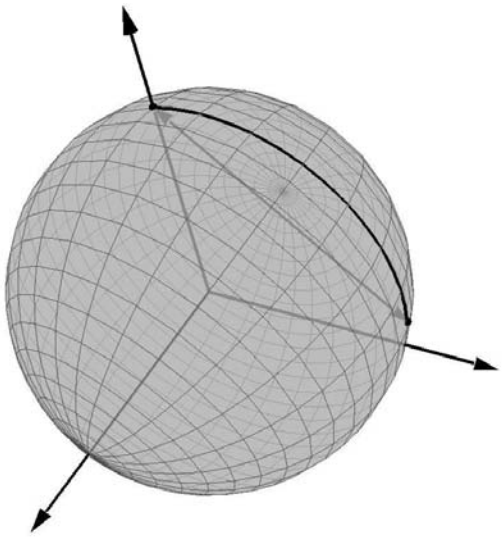


Abbildung 7.2: Der Abstand zwischen zwei Rotationen kann als die geodätische Distanz auf der Einheitshypersphäre berechnet werden.

7.2.1.1 Distanzmaß für Rotationen

Das Problem der Berechnung der Distanz zwischen zwei Rotationen oder einer gewichteten mittleren Rotation liegt darin, dass je nach gewählter Repräsentation diese im Allgemeinen nicht zu einem Vektorraum gehören, sondern auf einer nichtlinearen Mannigfaltigkeit in diesem Raum liegen. Diese Nichtlinearität führt dazu, dass weder das Euklidische Distanzmaß noch die Berechnung der mittleren Rotation als gewichtete Summe eine geometrische Interpretation zulassen, da das Ergebnis nicht zwangsläufig auf der Mannigfaltigkeit liegt.

Eine aus der geometrischen Sicht sinnvolle Darstellung wird durch die Repräsentation von Joints durch Rotoren einer Euklidischen geometrischen Algebra gemäß (3.16) bzw. (3.17) (isomorph zu Quaternionen [83]) gegeben. In dieser Darstellung können Rotationen als Punkte auf der 3D-Einheitshypersphäre in einem vierdimensionalen Vektorraum interpretiert werden. Der Abstand zwischen zwei Rotationen R_1 und R_2 kann in diesem Fall als die geodätische Distanz auf der Einheitshypersphäre berechnet werden [78] (beispielhaft in Abbildung 7.2 illustriert). Da die Hypersphäre normiert ist, entspricht diese Distanz dem Rotationswinkel θ bzw. dem Bogenmaß, der einen Punkt in den anderen Punkt überführt[1]:

$$d(R_1, R_2) := \arg((\widetilde{R_1} R_2)) = \theta(\widetilde{R_1} R_2),$$

[1] $\arg(R)$ gibt den Rotationswinkel θ des Rotors R in der Euler-Darstellung zurück.

wobei $\widetilde{R_1}R_2$ die Rotation angibt, die R_1 in R_2 transformiert. Unter der Berücksichtigung der Rechenregeln der geometrischen Algebra sowie Gleichungen (3.16) und (3.17) lässt sich diese Distanz folgendermaßen ausdrücken:

$$\begin{aligned} d(R_1, R_2) &= 2\arccos < \widetilde{R_1}R_2 >_0 = 2\arccos(\widetilde{R_1} \cdot R_2) \\ &= 2\arccos < \widetilde{R_2}R_1 >_0 = 2\arccos(\widetilde{R_2} \cdot R_1), \end{aligned} \tag{7.2}$$

wobei $< ., . >_i$ ein Operator ist, der den Unterraum des i-ten Grades (in diesem Fall 0-ten – also Skalaranteil) aus dem Multivektor extrahiert.

Die Herleitung dieser Metrik ist einerseits geometrisch motiviert. Andererseits lässt sich diese Metrik im Vergleich zu anderen Parametrisierungen wie z.B. Matrixdarstellung oder Darstellung über mehrere Rotationswinkel (Euler-Winkel) effizienter berechnen [39, 78]. Weitere Bemerkungen zum Entwurf von Distanzmaßen sind im Anhang C aufgeführt.

7.2.1.2 Gewichtete mittlere Rotation

Die Schätzung der gewichteten mittleren Rotation $\bar{R}$ über gegebener Menge Rotationen A

$$\mathbb{A} = \{R_i \in \mathbb{S}\}, \quad \mathbb{S} := \{R \in \mathcal{G}_m^+ | R\widetilde{R} = 1\}, m \in \{2, 3\}$$

lässt sich als folgende Minimierungsaufgabe definieren:

$$\bar{R} = \arg\min_{R \in \mathbb{S}} \sum_{R_i \in \mathbb{A}} \lambda_i \left[d(R, R_i)\right]^2 = \arg\min_{R \in \mathbb{S}} \sum_{R_i \in \mathbb{A}} \lambda_i \left[\theta(\widetilde{R}R_i)\right]^2, \quad \lambda_i \in \mathbb{R}. \tag{7.3}$$

In [39] wurde ein Ansatz zur Linearisierung dieser Minimierungsaufgabe für Quaternionen mittels Taylor-Entwicklung über dem Cosinus vorgestellt. Das Prinzip lässt sich direkt auf die Multivektor-Darstellung übertragen:

$$\begin{aligned} \cos\frac{\theta}{2} &= 1 - \frac{1}{8}\theta^2 + o(\theta^4) \\ &\Updownarrow \\ \left[\theta(\widetilde{R}R_i)\right]^2 &= 8\left(1 - \cos\frac{\theta}{2}\right) + o(\theta^4) \\ &\approx 8(1 - \widetilde{R} \cdot R_i). \end{aligned} \tag{7.4}$$

Es folgt

$$\begin{aligned}\bar{R} &= \arg\min_{R\in\mathbb{S}} \sum_{R_i\in\mathbb{A}} \lambda_i \left[\theta(\widetilde{R}R_i)\right]^2 \approx \arg\min_{R\in\mathbb{S}} \sum_{R_i\in\mathbb{A}} \lambda_i 8(1-\widetilde{R}\cdot R_i) \\ &= \arg\max_{R\in\mathbb{S}} \sum_{R_i\in\mathbb{A}} \lambda_i \widetilde{R}\cdot R_i = \frac{\sum\limits_{R_i\in\mathbb{A}} \lambda_i R_i}{\| \sum\limits_{R_i\in\mathbb{A}} \lambda_i R_i\|} \end{aligned} \tag{7.5}$$

Die Gleichung (7.5) sagt aus, dass die gewichtete mittlere Transformation bei einer Repräsentation von Rotationen mit geometrischer Algebra durch ein einfaches Aufsummieren mit anschließender Renormierung gut approximiert wird.

7.2.1.3 Verallgemeinerung für einen mehrgelenkigen Arm

Die oben vorgestellte Metrik dient zur Berechnung der Abstände für einzelne Gelenke. Wird der Arm als eine geordnete Menge von Gelenken (7.1) modelliert, so lässt sich eine passende Metrik auf eine natürliche Weise als eine Erweiterung der Gleichung (7.2) konstruieren. Dabei werden zuerst die Abstände für die einzelnen Gelenke gemessen und in einem reellwertigen n-dimensionalen Vektor abgespeichert. Die Distanz zwischen zwei Posen für den gesamten Arm wird im nächsten Schritt durch die Bildung der Euklidischen Norm über diesem Vektor berechnet:

$$\begin{aligned}\forall\, c_i = (R_{i1},\cdots,R_{in}),\quad c_j = (R_{j1},\cdots,R_{jn}) \in \mathcal{C}: \\ D(c_i,c_j) := \left(\sum_{k=1}^{n} \left[d(R_{ik},R_{jk})\right]^2\right)^{\frac{1}{2}}.\end{aligned} \tag{7.6}$$

Die Schätzung der gewichteten mittleren Pose des Armes kann in diesem Modell für einzelne Gelenke gemäß (7.5) unabhängig erfolgen.

7.2.2 Planung eines Bewegungsablaufes

Das Lernen der Topologie von PCM mit DCS-Netzen erlaubt, einzelne Posen eines Armes auf einen anderen Arm zu übertragen. Posen des steuernden Armes, die sich außerhalb der gelernten $\mathcal{C}$-Mannigfaltigkeit befinden, werden dabei erst auf diese projiziert[2]. Eine weitergehende Aufgabe besteht jedoch darin, einen Pfad

[2] Dies ist eine Eigenschaft von DCS-Netzen.

von einem beliebigen Startpunkt zum gegebenen Zielpunkt auf der gelernten Mannigfaltigkeit zu finden, mit anderen Worten: Eine zulässige Bewegung von einer Anfangspose zu der Endpose durchzuführen.

Das DCS-Netz bildet eine Nachbarschaftsstruktur im Eingaberaum, die sich mit einem Graph repräsentieren lässt. Die Aufgabe kann also als eine Suche nach dem kürzesten Pfad – der kürzesten verbundenen Kette von Knoten des Graphs – formuliert werden. Es existiert eine Reihe von Algorithmen zur Lösung dieses Problems. Im Rahmen dieser Arbeit ist der Floyd-Warshall-All-Pairs-Shortest-Path-Algorithmus [34] verwendet worden. Der Algorithmus setzt zur Lösung eine Methode aus dem Bereich der dynamischen Programmierung ein. Die Laufzeitkomplexität liegt für einen Graph mit l Knoten bei $O(l^3)$. Von der Adjazenzmatrix W des Graphs ausgehend, werden in l Iterationen die Distanzen $d(i,j)$ zwischen allen Konotenpaare (i,j) gemäß rekursiver Vorschrift

$$d^k(i,j) := \begin{cases} 1 & \text{falls } k = 0 \\ \min\left\{d^{k-1}(i,j), d^{k-1}(i,k) + d^{k-1}(k,j)\right\} & \text{sonst} \end{cases}$$

berechnet. Die entsprechenden Pfade $s(i,j)$ werden dabei auf folgende Weise ebenfalls rekursiv bestimmt:

$$s^0(i,j) := \begin{cases} \emptyset & \text{falls } i = j \text{ oder } w_{ij} \neq 1 \\ \{i\} & \text{sonst} \end{cases}$$

und

$$s^k(i,j) := \begin{cases} s^{k-1}(i,j) & \text{falls } d^{k-1}(i,j) < d^{k-1}(i,k) + d^{k-1}(k,j) \\ s^{k-1}(i,k) \cup s^{k-1}(k,j) & \text{sonst} \end{cases}.$$

Eine detaillierte Beschreibung des Floyd-Warshall-Algorithmus kann in [23] gefunden werden.

Falls es für die gestellte Aufgabe erforderlich ist, die Bewegungen zwischen beliebigen Punkten auf der $\mathcal{C}$-Mannigfaltigkeit durchführen zu können, soll die Existenz der Pfade zwischen allen Knoten des Graphen gewährleistet werden. Mit anderen Worten soll die Nachbarschaftsstruktur des DCS-Netzes zusammenhängend sein. Im Allgemeinen garantiert der übliche Lernalgorithmus die Erfüllung dieser Eigenschaft nicht. Insbesondere beim Wachsen des Netzes erhöht sich die Wahrscheinlichkeit der Nichterfüllung. Eine Erweiterung des Abbruchkriteriums für das Training um die Bedingung, die das Lernen unterbricht, falls die Nachbarschaft des Netzes nicht zusammenhängt, sichert dabei das erwünschte Verhalten des Netzes.

7.2.3 Algorithmus zur Steuerung

Nachdem das Netz trainiert und die kürzesten Pfade für alle möglichen Knotenpaare bestimmt wurden, kann die eigentliche Steuerung erfolgen.

Algorithmus 7.1 Steuerung eines Roboterarmes

Trainiere DCS-Netz
for all Neuronenpaare (i, j) **do**
  Finde kürzesten Pfad $s(i, j)$
end for
while Steuerung erwünscht **do**
  Lese Anfangspose c'_A des steuernden Armes
  Finde c'_{BA} :=BMU(c'_A)
  Lese Endpose c'_E des steuernden Armes
  Finde c'_{BE} :=BMU(c'_E)
  Finde kürzesten Pfad $c'_A \cup s(c'_{BA}, c'_{BE}) \cup c'_{BE} \cup c'_B$
  Berechne mit DCS-Netz die Steuerungssequenz $c_A \cup s(c_{BA}, c_{BE}) \cup c_{BE} \cup c_B$
end while

Dabei werden die Anfangs- und Endpose des steuernden Armes an das Netz übergeben. Es werden BMUs für diese bestimmt, der kürzeste Pfad im Eingaberaum aus der gespeicherten Tabelle ausgelesen und um diese Posen erweitert. Für jedes Element des Pfades wird die Ausgabe berechnet und zur Generierung der Bewegung des gesteuerten Armes eingesetzt. Der gesamte Ablauf ist in Algorithmus 7.1 zusammengefasst.

Im nächsten Abschnitt sind Experimente vorgestellt, welche die Fähigkeiten dieses Ansatzes zur neuronalen Steuerung demonstrieren.

7.3 Experimente

Eine Testumgebung für das Training des DCS-Netzes und die Simulation der Steuerung wurde im Rahmen dieser Arbeit entwickelt. Das System besteht aus Modellen für zwei Arme mit verschiedener Kinematik. Der steuernde Arm ist einem menschlichen Arm nachempfunden: Das Modell besitzt zwei Gelenke, die jeweils eine 3D-Rotation durchführen können. Der gesteuerte Arm entspricht in der Funktionsweise dem zur Verfügung stehenden Roboterarm 'Stäubli RX90': Der Arm besitzt sechs Gelenke, die je um eine feste Achse rotieren können (siehe Abbildung 7.1).

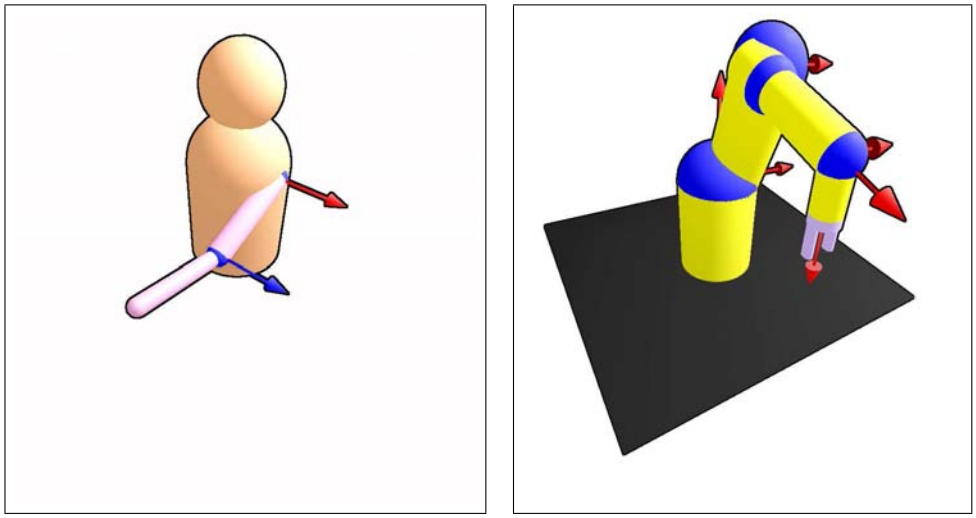


Abbildung 7.3: Testsystem: Links - steuernder Arm; Rechts - gesteuerter Arm.

Die Modellierung erfolgte in der Skriptsprache des Programms CLUCalc [79], die eine Beschreibung von Objekten als Entitäten bzw. Multivektoren einer geometrischen Algebra ermöglicht. Das Programm eignet sich gut zur Visualisierung und Manipulation von solchen Entitäten auf eine interaktive Weise. Abbildung 7.3 zeigt die beiden Modelle (Pfeile repräsentieren die entsprechenden 3D- bzw. 2D-Rotationsachsen).

Die 3D-Gelenke des menschenähnlichen Armes wurden durch die Rotoren $\{R'_i\}_{i=1,2}$ der dreidimensionalen geometrischen Algebra $\mathcal{G}_3$

$$R'_i = r'_{i0} + r'_{i1}\mathsf{e}_{23} + r'_{i2}\mathsf{e}_{31} + r'_{i3}\mathsf{e}_{12} \in \mathcal{G}_3^+, \quad R'_i\widetilde{R'_i} = 1$$

repräsentiert. Die Parametrisierung einer Pose erfolgte dementsprechend mit einem 8-dimensionalen Vektor:

$$c' := (r'_{10}, r'_{11}, r'_{12}, r'_{13}, r'_{20}, r'_{21}, r'_{22}, r'_{23}) \in \mathbb{R}^8 \equiv \mathcal{C}'.$$

Analog wurden 2-D Gelenke des Roboterarmes als Rotoren $\{R_i\}_{i=1\cdots6}$ der geometrischen Algebra $\mathcal{G}_2$

$$R_i = r_{i0} + r_{i1}\mathsf{e}_{12} \in \mathcal{G}_2^+, \quad R_i\widetilde{R_i} = 1$$

und die entsprechenden Posen als Vektoren

$$c := (r_{10}, r_{11}, \cdots, r_{60}, r_{61}) \in \mathbb{R}^{12} \equiv \mathcal{C} \tag{7.7}$$

dargestellt.

Der Algorithmus 7.1 wurde als eine C++ Klasse der auf dem Lehrstuhl entwickelten Bibliothek PACLib [19] implementiert. Zur Entwicklung der Klasse für das um die geodätische Metrik erweiterte DCS-Netz wurde die frei zugängliche Bibliothek zum neuronalen Lernen Torch3 [21] eingesetzt. Es wurde eine weitere C++ Klasse zur Steuerung des Roboterarmes Stäubli RX90 anhand der Simulationsdaten implementiert.

7.3.1 Lernen von Gesten

Das Ziel dieses Experimentes bestand darin, die Gestik eines menschenähnlichen Armes auf den Roboterarm zu übertragen. Diese Aufgabe stellt hohe Anforderungen an das Modell (siehe z.B. [92]), da sowohl die Position im Raum als auch die Dynamik der Bewegung berücksichtigt werden müssen. Schränkt man die Gestik auf die Ähnlichkeit von Posen ein, so lässt sich die Aufgabe mit dem oben vorgestellten Modell lösen.

Für das Experiment wurde eine Bewegungssequenz der Länge 100 erzeugt (beispielhaft in Abbildung 7.4 dargestellt). Die gesamte Geste wurde in drei Phasen durchgeführt. In den ersten beiden Phasen wurde nur je ein Gelenk bewegt. In der dritten Phase rotierten alle zur Verfügung stehende Gelenke gleichzeitig.

Die beiden 3D-Gelenke des menschenähnlichen Armes wurden für die Generierung der Geste unter Berücksichtigung aller physikalisch möglichen Freiheitsgrade eingesetzt. Um die (subjektiv) ähnlichen Posen des Roboterarmes zu erzeugen, wurden nur die ersten drei 2D-Gelenke bewegt. Entsprechend klein im Vergleich zu (7.7) fiel der $\mathcal{C}$-Raum für den Arm aus, um die geforderte Bewegung zu ermöglichen:

$$c := (r_{10}, r_{11}, r_{20}, r_{21}, r_{30}, r_{31}) \in \mathbb{R}^6 \equiv \mathcal{C}.$$

Die Korrespondenzen zwischen den Posen der beiden Arme wurden manuell gesetzt. Ein DCS-Netz wurde gemäß Algorithmus 7.1 mit 30 aus der Bewegungssequenz zufällig gewählten Datenpaaren unter Berücksichtigung der oben vorgestellten Berechnungsvorschriften für den Abstand und die gewichtete mittlere Rotation trainiert. Alternativ wurde ein Netz mit den klassischen Euklidischen Distanzmaßen mit denselben Daten trainiert. Der Schwellwert für die maximale Anzahl von Neuronen lag für beide Netze bei 15, um die Ergebnisse besser vergleichen zu können. Anschließend wurden die restlichen 70 Daten zum Testen eingesetzt.

Um das Verhalten der Netze gegenüber Rauschen zu testen, wurden die Gelenke

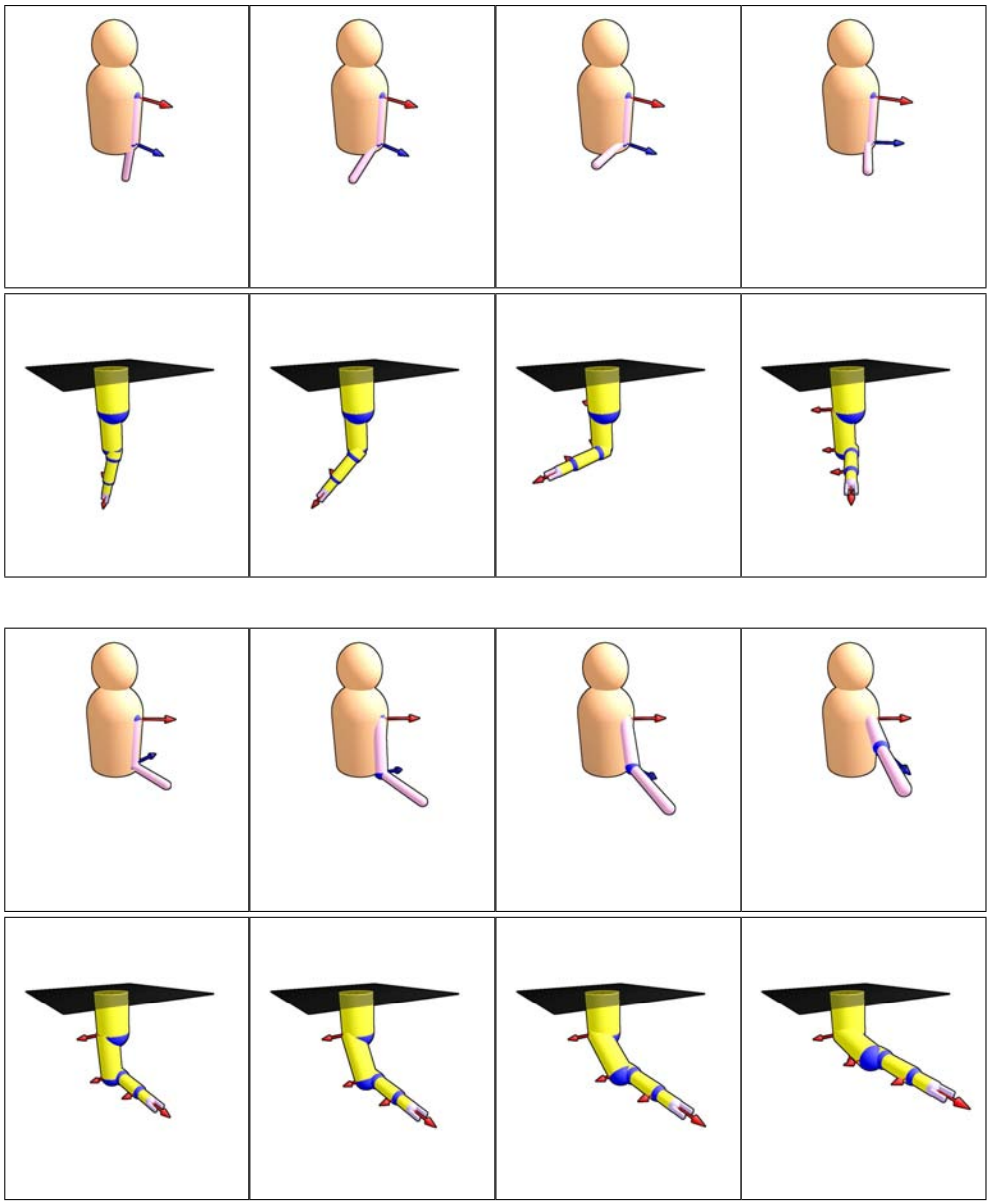


Abbildung 7.4: Übertragung der Gestik eines Armes mit zwei 3D-Gelenken auf einen Arm mit drei 2D-Gelenken.

Rauschen,°	0	1	3	5
DCS (geodätisch)	5.24	6.08	8.72	13.13
DCS (Euklidisch)	6.91	7.90	10.36	14.61

Tabelle 7.1: Fehler in Grad bei der Posenschätzung des gesteuerten Armes bei verrauschten Posen des steuernden Armes.

des steuernden Armes für jede Pose zufällig um 1, 3 und 5 Grad zusätzlich rotiert. Die berechneten Netzausgaben wurden mit den nicht verrauschten Posen des gesteuerten Armes verglichen (gesamter Winkelfehler über drei Gelenke in Grad). Der Versuch wurde für alle Stufen des Rauschens 50 mal wiederholt. Tabelle 7.1 fasst die Ergebnisse zusammen. Erwartungsgemäß weist das Netz, welches die Nichtlinearität der Mannigfaltigkeit von Rotationen berücksichtigt, besseres Verhalten gegenüber dem Rauschen auf.

Weiterhin wurden aus den 70 Testdaten 100 Paare zufällig erzeugt und zur Planung von Bewegungsabläufen benutzt. Die durchschnittliche Pfadlänge lag für beide Netze bei 7 Knoten, was in etwa der Hälfte der Anzahl der verwendeten Neuronen entsprach. Dieses Ergebnis zeigt auf, dass beide Netze eine ähnliche Nachbarschaftsstruktur aufweisen und sich nur in der Position von Neuronen im Eingaberaum unterschieden.

Die Trainingszeit mit anschließender Suche nach Pfaden lag auf einem Pentium IV 2.4Ghz System für beide Netze unter 1 Sekunde.

7.3.2 Vermeiden von statischen Hindernissen

Die Umgebungsbedingungen für beide Arme des System müssen nicht zwangsläufig identisch sein. Da eine Pose eines Armes mit den Gliedmaßen fester Länge implizit auch die 3D-Position angibt, können räumliche Einschränkungen insbesondere bei einem fest installierten Arm als nicht zulässige Konfigurationen im $\mathcal{C}$-Raum repräsentiert werden. Es kann eine Pose für einen Arm auf der $\mathcal{C}$-Mannigfaltigkeit liegen, während die korrespondierende bzw. subjektiv als ähnlich empfundene Pose des anderen Armes nicht zulässig ist. Falls in diesem Szenario nur die Erreichbarkeit von einigen ganz speziellen Zuständen (Posen bzw. 3D-Positionen) gefordert wird, so ist die Ähnlichkeit von Bewegungen nicht auf dem ganzen Raum notwendig.

Abbildung 7.5 illustriert beispielhaft ein solches Szenario. Die Zielstellung war, mit ausgestrecktem Arm die Endposition zu erreichen, die auf der gleichen Höhe

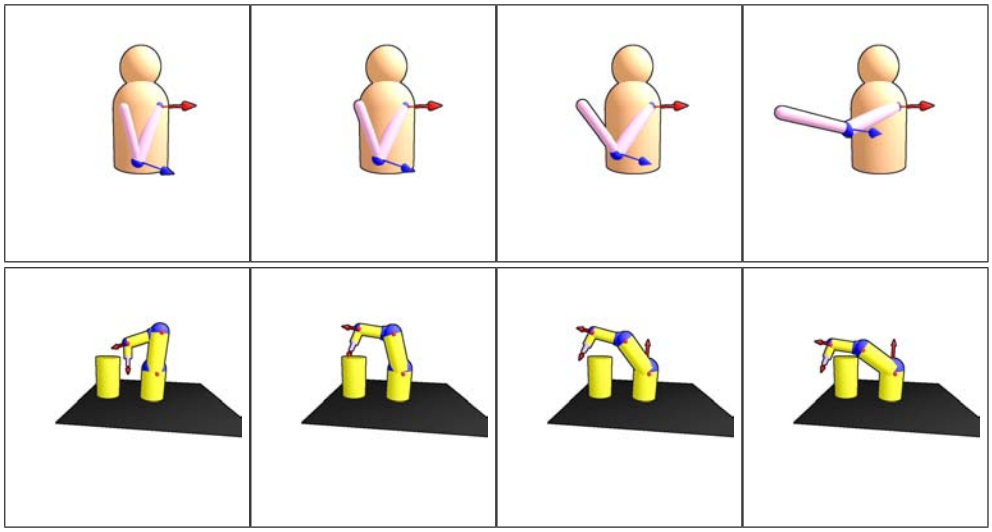


Abbildung 7.5: Zwei Arme: Zulässige Posen des steuernden bzw. gesteuerten Armes induzieren Trajektorien unterschiedlicher Form.

Fehler,°	1	3	5
DCS (geodätisch)	41	27	18
DCS (Euklidisch)	49	32	20

Tabelle 7.2: Anzahl von Neuronen in Abhängigkeit vom Schwellwert des maximal zulässigen Fehlers.

mit der Anfangsposition liegt. Der Endpunkt des steuernden Armes konnte sich auf einer geradlinigen Trajektorie bewegen. Bei dem gesteuerten Arm ließ sich der Endpunkt nicht auf der geradlinigen Trajektorie erreichen, da ein Hindernis im Weg lag. Eine zulässige Trajektorie konnte durch die Ansteuerung des zweiten und dritten Gelenkes generiert werden. Entsprechend wurde der $\mathcal{C}$-Raum des Armes definiert:

$$c := (r_{20}, r_{21}, r_{30}, r_{31}) \in \mathbb{R}^4 \equiv \mathcal{C}.$$

Eine Sequenz aus zulässigen Bewegungen der Länge 80 wurde erzeugt und zum Lernen benutzt. Das Training wurde sowohl mit der geodätischen als auch mit der Euklidischen Metrik durchgeführt und nach dem Senken des Fehlers der Netzausgabe unter 1,3 bzw. 5 Grad abgebrochen.

Die Ergebnisse sind in der Tabelle 7.1 zusammen gefasst. Es lässt sich erkennen, dass insbesondere, wenn eine hohe Genauigkeit bei der Posenschätzung gefordert ist, die geodätische Metrik gegenüber der Euklidischen im Vorteil ist.

Auch in diesem Experiment lag die Trainigszeit unter 1 Sekunde.

7.4 Zusammenfassung

Die Abbildung von Konfigurationsräumen verschiedener Arme aufeinander im Allgemeinen bzw. das Steuern eines Roboterarmes mit Gesten eines anderen im Speziellen ist vielseitig einsetzbar. Der im Rahmen dieser Arbeit vorgestellte Ansatz kann diese Aufgabe für verschiedene Szenarios erfolgreich lösen. Der neuronaler Zugang erlaubt vor allem ein robustes System auch dann entwickeln zu können, wenn die vorliegende Informationen nicht ausreichend für einen analytischen Entwurf sind.

Die Kodierung von Posen als eine geordnete Menge von Rotationen ermöglicht einerseits eine kompakte Repräsentation, die von dem Typ und der Anordnung der Gelenke aber nicht von der Größe einzelner Gliedmaßen abhängt. Andererseits beschreibt sie gleichzeitig auf eine implizite Weise Positionen des Armes mit festen Gliedmaßen. Da das Netz nur mit Posen trainiert wird, kann die eigentliche Steuerung auch mit einem Arm erfolgen, der nicht zum Training eingesetzt wurde. Dagegen ist das Modell an den gesteuerten Arm gebunden, da die 3D-Positionierung des Armes fest mit dessen Kinematik zusammenhängt.

Die Parametrisierung von Gelenkstellungen mit Rotoren einer geometrischen Algebra gestaltet die Berechnung der geodätischen Metrik, welche die Struktur von Gruppe der Rotationen $SO(n)$ berücksichtigt, auf eine effiziente Weise. Die Metrik selbst erlaubt, die Topologie des Konfigurationsraumes eines Armes im Vergleich zu einem Euklidischen Distanzmaß mit geringerer Komplexität zu lernen.

Kapitel 8

DISKUSSION

Die Zielstellung dieser Arbeit war, Clifford-Algebren auf ihre Eignung zur Repräsentation von a priori Wissen und die Integration von Informationen in neuronale Modelle zwecks Verbesserung der Funktionalität und Effizienz zu analysieren.[1] Diese Analyse bedurfte der Antworten auf folgende essentielle Fragen:

- Auf welche Art und in welcher Form kann ein Modellierungsvorgang durch a priori Wissen beeinflusst werden? (WIE)
- Welche Vorteile bringt die Repräsentation mit einer Clifford-Algebra gegenüber den standardmäßig verwendeten Vektorraumkonzepten? (WAS)
- An welchen strukturellen Stellen einer neuronalen Architektur lassen sich Clifford-kodierte Informationen integrieren? (WO)

Die erste Frage wurde in den Kapiteln 2 und 4 diskutiert. Ein mathematisches Gerüst zur allgemeinen Beschreibung der Modellierung für verschiedene Aufgaben wurde vorgestellt und mit einer funktionalen und geometrischen Interpretation versehen. Verschiedene Ansätze wurden untersucht, die das a priori Wissen sowohl in einer dynamischen Form (Wahl der Klasse der Approximierungsfunktion) als auch in Form von Zuständen (Trainingsdaten) zu einer an die gestellte Aufgabe angepassten Modellierung einsetzen. Insbesondere ließen sich die klassischen lokalen und globalen neuronalen Architekturen durch eine geeignete Parametrisierung aus einem generischen Modell ableiten. Den Grundgedanken dieser Arbeit folgend, waren dabei drei Einflussmöglichkeiten besonders wichtig: Eine geeignete Datenrepräsentation (Einbettung) in der Vorverarbeitungsphase, die Wahl der Approximierungsfunktion innerhalb des Modells sowie die Verwendung

[1] Diese Arbeit deckt hauptsächlich praktische Aspekte der Modellierung mit Clifford-Algebren ab. Eine komplementäre Arbeit, deren Schwerpunkt auf der theoretischen Seite liegt, ist [13].

eines auf das vorgegebene Problem abgestimmten Distanzmaßes zu Berechnung der Fehlerfunktion.

Kapitel 3 beschäftigte sich mit der zweiten Frage. Es wurde eine geometrische Algebra über Euklidische Räume eingeführt. Dabei wurde auf die Fähigkeit, Elemente der Algebra als geometrische Entitäten – Punkte und Geraden – zu interpretieren, näher eingegangen. Auch die Möglichkeit, Rotationen als spezielle Multivektoren effizient repräsentieren zu können, stellte einen wichtigen Punkt dar. Darauf aufbauend wurden die Eigenschaften von konformen Algebren als eine durch die stereographische Projektion induzierte Erweiterung erläutert: starre Bewegungen lassen sich als Punkte auf der Einheitskugel entsprechender Dimension im konformen Raum ausdrücken. Weiterhin erlauben konforme Algebren, solche geometrische Entitäten des zugrundeliegenden Euklidischen Raumes wie Hypersphären auf eine lineare Weise zu beschreiben.

a priori Wissen	strukturelle Stelle	Realisierung
Hypersphäre, $\mathcal{G}_{n+1,1}$	Eingaberaum (Einbettung)	MLHP
starre Bewegung, $\mathcal{G}_{3,1}$	Approximierungsfunktion	Segmentierung von OF
Rotation, $\mathcal{G}_2^+$, $\mathcal{G}_3^+$	Distanzmaß	Steuerung eines RA

Tabelle 8.1: Einfluss des a priori Wissens auf die Realisierung des generischen Modells. Abkürzungen: OF - optischer Fluss; RA - Roboterarm.

Die vorangegangenen Kapitel bildeten eine Basis zur Beantwortung der dritten Fragestellung. In Tabelle 8.1 sind die im Rahmen dieser Arbeit entwickelten Modelle (mit dem zur Parametrisierung benötigten a priori Wissen) zusammenfassend veranschaulicht: In Kapitel 5 wurde der Entwurf eines neuronalen Netzes (MLHP) beschrieben, das einerseits eine Erweiterung des klassischen MLP darstellt, andererseits die Funktionalität der RBF-Netze aufweist. Die Grundidee der Erweiterung lag in der Einbettung der Daten in einen konformen geometrischen Raum. Die Struktur eines MLP wurde dabei beibehalten bzw. nur geringfügig verändert. Die mit dem MLHP durchgeführte lineare Aufteilung des Raumes entsprach dabei einer nichtlinearen Zerlegung des zugrunde liegenden Euklidischen Raumes mit Hypersphären. Die Vorteile der neuen Architektur lagen dabei sowohl in der Fähigkeit, globale und lokale Wirkung gleichzeitig zu erzeugen als auch in der teilweise darauf zurückzuführenden hohen Effizienz. Die Ergebnisse wurden in [4, 5, 80] veröffentlicht. Die Darstellung von allgemeinen Rotationen in einer konformen geometrischen Algebra wurde in Kapitel 6 für die Entwicklung von neuronalen Experten zur Segmentierung des optischen Flusses verwendet. Das generische Modell wurde dann als ein lokal wirkendes Netz mit Rotoren als Ap-

proximierungsfunktionen parametrisiert. Das Netz zeigte im Vergleich mit anderen Modellentwürfen sowohl eine bessere Generalisierungsfähigkeit und Robustheit als auch eine höhere Effizienz. Die Suche nach einem Distanzmaß zwischen den Gelenkstellungen eines Roboterarmes bzw. zwischen zwei beliebigen Rotationen bildete den Schwerpunkt des nächsten Kapitels. Der Einsatz von Rotoren einer Clifford-Algebra verhalf dabei zu einem geometrisch motivierten Entwurf einer Metrik. Darauf basierend wurde ein neuronales Netz zur Steuerung eines Roboterarmes durch die Gestik eines zweiten Armes entwickelt.

Die Ergebnisse dieser Dissertation verdeutlichen, dass es ein universell einsetzbares Lernparadigma nicht gibt: Spätestens bei der Anwendung zeigt sich die Überlegenheit eines problembezogen entworfenen neuronalen Modells gegenüber einem generischen Ansatz. Solche Modellierung erfordert allerdings eine individuelle, umfassende Analyse für jede gestellte Aufgabe, die mit entsprechend hohem Aufwand verbunden ist. Um so schwieriger gestaltet sich der Entwurf eines effizienten Systems. Eine Aufteilung des Problems falls möglich in mehrere unabhängige Teilaufgaben kann den Modellierungsvorgang vereinfachen. Clifford-Algebren können dabei ein passendes (aber nicht einziges) Werkzeug zur Modellierung vieler geometrisch motivierter Probleme bieten.

Anhang A

KAMERAGEOMETRIE

A.1 Kameramodell

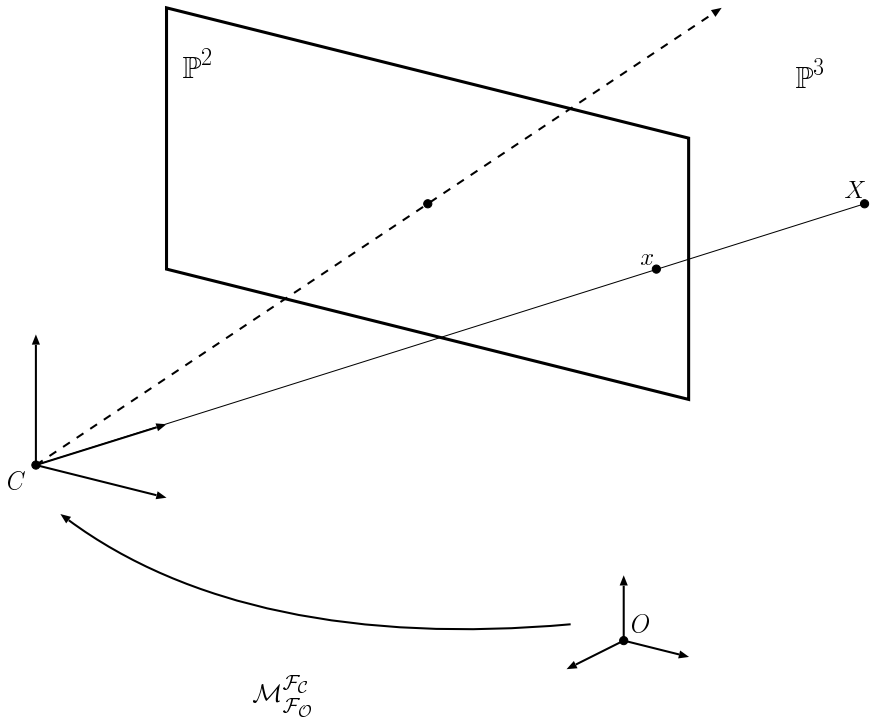


Abbildung A.1: Lochkameramodell.

Typischerweise wird die Projektion eines 3D-Objektes in die Bildebene eines Kamerasystems als eine lineare Abbildung angenommen. Dabei werden in der Abhängigkeit von der Distanz zwischen dem betrachteten Objekt und der Kamera drei Projizierungsmodelle – projektiv, affin und orthographisch– unterschieden.

Unabhängig von der Wahl des Modells lässt sich die Projektion mit Hilfe einer Projektionsmatrix P beschrieben:

$$\mathbf{x} = P\mathbf{X},$$

wobei $\mathbf{x}$ die 2D-Abbildung eines 3D-Punktes $\mathbf{X}$ ist.

Für Objekte, die nahe an der Kamera liegen, wird häufig das Lochkamera-Modell (projektiv) angenommen. In Abbildung A.1 wird die Projektion eines Punktes $\mathbf{X}$ aus einem dreidimensionalen projektiven Raum $\mathbb{P}^3$ (isomorph zu $\mathbb{R}^4$) in die Bildebene für diesen Fall dargestellt. Der abgebildete Punkt $\mathbf{x}$ liegt im zweidimensionalen projektiven Raum $\mathbb{P}^2$(isomorph zu $\mathbb{R}^3$). Das optische Zentrum der Kamera wird mit $\mathbf{C}$ bezeichnet. Der Koordinatenursprung von $\mathbb{P}^3$ liegt in $\mathbf{O}$. Für dieses Modell wird die Projektionsmatrix als

$$P = \begin{bmatrix} t_{11} & t_{12} & t_{13} & t_{14} \\ t_{21} & t_{22} & t_{23} & t_{14} \\ t_{31} & t_{32} & t_{33} & t_{34} \end{bmatrix} \underbrace{\begin{bmatrix} \sigma f & -\cot\alpha & u_0 \\ 0 & f & v_0 \\ 0 & 0 & 1 \end{bmatrix}}_{K_p} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{P_0} M_{F_0}^{F_C} \quad \text{(A.1)}$$

definiert, wobei K_p die Transformation in der Bildebene, die so genannten intrinsischen Parameter, beschreibt:

- Die Brennweite f – der Abstand zwischen dem optischen Zentrum und der Bildebene,
- Das Seitenverhältnis σ – Skalierungsfaktor zwischen den Koordinatenrichtungen in der Bildebene,
- Der Winkel zwischen den Koordinatenrichtungen α,
- Der Schnittpunkt (u_0, v_0) der optischen Achse mit der Bildebene.

Die Matrix $M_{F_0}^{F_C}$ beschreibt die Transformation zwischen den Weltkoordinaten $\mathcal{F}_0$ und Kamerakoordinaten $\mathcal{F}_C$:

$$M_{F_0}^{F_C} = \begin{bmatrix} R & \mathbf{t} \\ 0^T & 1 \end{bmatrix}$$

mit der 3×3 Rotationsmatrix R, dem 3×1 Translationsvektor $\mathbf{t}$ und dem 3×1 Nullvektor 0.

Für weit von der Kamera entfernte Objekte sind die optischen Strahlen fast parallel. In diesem Fall spricht man von einer affinen Kamera. Diese kann als eine Lochkamera mit dem optischen Zentrum im Unendlichen interpretiert werden:

$$P = \underbrace{\begin{bmatrix} \sigma f & 0 & u_0 \\ 0 & f & v_0 \\ 0 & 0 & 1 \end{bmatrix}}_{K_{aff}} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{P_{par}} M_{F_0}^{F_C}, \tag{A.2}$$

wobei P_{par} die parallele Projektion angibt.

Spielen die Skalierungsfaktoren keine Rolle, so wird aus der affinen die orthografische Kamera:

$$P = \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{K_{ort}} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{P_{par}} M_{F_0}^{F_C}. \tag{A.3}$$

A.2 Projektion einer starren Bewegung

Wird ein 3D-Punkt $\mathbf{X}$ mit einer starren Bewegung

$$T_{4\times4} = \begin{pmatrix} R_{3\times3} & \mathbf{t}_{3\times1} \\ 0_{1\times3} & 1 \end{pmatrix}, \quad R \text{ Rotationsmatrix, } \mathbf{t} \text{ Translationsvektor}$$

transformiert, so ergibt die Projektion einen neuen Punkt in der Bildebene:

$$\mathbf{y} = PT\mathbf{X}. \tag{A.4}$$

Falls zwischen der alten Projektion $\mathbf{x}$ und der neuen Projektion $\mathbf{y}$ eine lineare Abhängigkeit existiert,

$$\mathbf{y} = A\mathbf{x},$$

wobei die 3×3 Matrix A die lineare Transformation beschreibt, so lässt sich ein Zusammenhang zwischen dieser Transformation, der starren 3D-Bewegung und der Projektionsmatrix herstellen:

$$AP = PT. \tag{A.5}$$

Beschreibt man die Projektionsmatrix als die Kombination einer 3×3 Matrix K mit einem Spaltenvektor e:

$$P_{3\times4} = \begin{pmatrix} K_{3\times3} & e_{3\times1} \end{pmatrix},$$

so kann Gleichung (A.5) folgendermaßen umgeformt werden:

$$\begin{aligned} AP = A\begin{pmatrix} K & e \end{pmatrix} &= \begin{pmatrix} AK & Ae \end{pmatrix} = PT \\ &= \begin{pmatrix} K & e \end{pmatrix} = \begin{pmatrix} R_{3\times3} & t_{3\times1} \\ 0_{1\times3} & 1 \end{pmatrix} = \begin{pmatrix} KR & Kt+e \end{pmatrix}. \end{aligned}$$

Daraus ergibt sich:

$$AK = KR \quad \text{bzw.} \quad Ae = Kt + e.$$

Da K im Allgemeinen eine invertierbare Matrix ist, können obige Ausdrücke folgendermaßen umgeschrieben werden:

$$A = KRK^{-1}$$

und

$$KRK^{-1}e = Kt + e. \tag{A.6}$$

Aus dieser Darstellung lässt sich ein Zusammenhang zwischen den Kameraparametern und der starren Bewegung des Objektes erkennen. Das bedeutet, dass die Projektion A einer starren Bewegung T nur dann als eine lineare 2D-Transformation darstellbar ist, wenn die 3D-Bewegung der Gleichung (A.6) genügt. Dies ist nur für planare Objekte, die sich parallel zur Bildebene bewegen, der Fall.

A.3 Stereogeometrie

Abbildung A.2 veranschaulicht die Projektion eines 3D-Punktes $\mathbf{X}$ in ein Stereosystem mit zwei Kameras A und B (optische Zentren $\mathbf{A}_0$ und $\mathbf{B}_0$). Die auf der Bildebene liegende 3D-Punkte sind dabei als $\mathbf{A}'$ bzw. $\mathbf{B}'$ dargestellt.

Die so genannten Epipole (in der Abbildung A.2 als 3D-Punkte $\mathbf{E}_{AB}$ und $\mathbf{E}_{BA}$ dargestellt) sind jeweils die Abbildungen des optischen Zentrums einer Kamera in die andere Kamera.

Die Punkte $\mathbf{A}_0$,$\mathbf{B}_0$, $\mathbf{A}'$ und $\mathbf{B}'$ sind koplanar – sie liegen in einer Ebene, die Epipolarebene genannt wird. Falls die Punkte in einer projektiven geometrischen

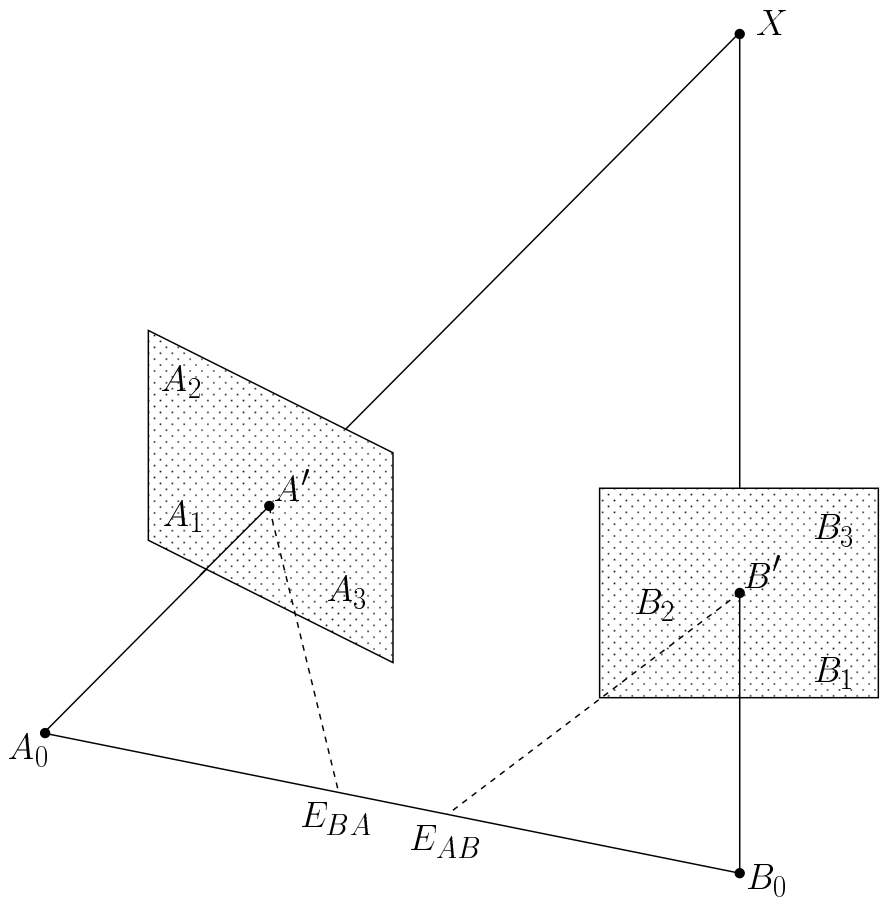


Abbildung A.2: Stereogeometrie.

Algebra repräsentiert sind, kann die Koplanaritätsbedingung folgendermaßen formuliert werden [3]:

$$\mathbf{A}_0 \wedge \mathbf{B}_0 \wedge \mathbf{A}' \wedge \mathbf{B}' = 0. \tag{A.7}$$

Die Punkte $\mathbf{A}'$ und $\mathbf{B}'$ lassen sich in Termen einer frei gewählten Kamerabasis $\{\mathbf{A}_i\}_{i=1\cdots3}$ bzw. $\{\mathbf{B}_j\}_{j=1\cdots3}$ ausdrücken:

$$\mathbf{A}' = \sum_{i=1}^{3} x_i \mathbf{A}_i \quad \text{und} \quad \mathbf{B}' = \sum_{j=1}^{3} y_j \mathbf{B}_j.$$

Unter der Berücksichtigung der neuen Basis wird die Koplanaritätsbedingung

(A.7) wie folgt umformuliert:

$$\sum_{i,j=1}^{3} x_i y_j \{\mathbf{A}_0 \wedge \mathbf{B}_0 \wedge \mathbf{A}_i \wedge \mathbf{B}_j\} = 0$$

bzw.

$$\sum_{i,j=1}^{3} f_{ij} x_i y_j = 0 \tag{A.8}$$

mit

$$f_{ij} := \{\mathbf{A}_0 \wedge \mathbf{B}_0 \wedge \mathbf{A}_i \wedge \mathbf{B}_j\} \mathsf{I}^{-1}.$$

Die Gleichung (A.8) gibt die bekannte Gleichung für die Fundamentalmatrix F wieder [64]:

$$\mathbf{y} F \mathbf{x}^T = (y_1\, y_2\, y_3) \begin{pmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = 0.$$

Die Matrix F lässt sich als eine lineare Funktion, die zwei Vektoren auf einen Skalar abbildet, interpretieren:

$$F(\mathbf{A}, \mathbf{B}) = \{\mathbf{A}_0 \wedge \mathbf{B}_0 \wedge \mathbf{A} \wedge \mathbf{B}\} \mathsf{I}^{-1}.$$

Auf diese Weise können die Koeffizienten f_{ij} der Fundamentalmatrix als eine Funktion von $F(\mathbf{A}, \mathbf{B})$ definiert werden:

$$f_{ij} = F(\mathbf{A}_i, \mathbf{B}_j), \tag{A.9}$$

wobei die $\mathbf{A}_i$ und $\mathbf{B}_j$ Elemente von jeweiligen 3D-Basis der Bildebene sind.

A.4 Der Acht-Punkte-Algorithmus

In [43] wurde ein Algorithmus[1] zur Schätzung einer Fundamentalmatrix mit acht Punktkorrespondenzen beschrieben. Dem Algorithmus liegt die Gleichung (A.8) zugrunde,

[1] als Weiterentwicklung des in [61] beschriebenen Algorithmus zur Berechnung der essentiellen Matrix (”essential matrix”).

$$\begin{aligned} x_1y_1f_{11} + x_2y_1f_{12} + x_3y_1f_{13} &+ \\ x_1y_2f_{21} + x_2y_2f_{22} + x_3y_2f_{23} &+ \\ x_1y_3f_{31} + x_2y_3f_{32} + x_3y_3f_{33} &= 0, \end{aligned} \tag{A.10}$$

die bezüglich der Koeffizienten der Fundamentalmatrix linear ist.

Da jede Fundamentalmatrix bis auf einen Skalar definiert ist, werden mindestens acht Punktkorrespondenzen benötigt, um ein genügend grosses lineares Gleichungssystem zur Schätzung der Matrix aufzustellen.

Die auf die oben beschriebene Weise berechnete Matrix erfüllt zunächst die epipolaren Zwangsbedingungen nicht, da die zur Schätzung eingesetzten Punktkorrespondenzen typischerweise verrauscht sind. Durch die Anpassung an die zu der bezüglich der Frobeniusnorm nächsten Rang-Zwei-Matrix kann das Ergebnis verbessert werden – der kleinste Singulärwert (berechnet mit Singulärwertzerlegung/SVD [85]) der geschätzten Fundamentalmatrix wird dafür auf Null gesetzt.

Anhang B

BERECHNUNG DES OPTISCHEN FLUSSES MIT KANADE-LUCAS-TOMASI-ALGORITHMUS

Der Algorithmus basiert auf einer Arbeit von Lucas und Kanade [63] und wurde von Tomasi und Kanade in [102] weiterentwickelt. Die komplette Beschreibung wurde allerdings erst von Shi und Tomasi [95] publiziert.

Eines der wichtigsten Probleme der Bildverarbeitung besteht darin, die Bewegung der einzelnen Pixel oder Muster in einer Bildfolge – den optischen Fluss – zu messen. Das Ziel ist, ein approximiertes 2D-Bewegungsfeld – eine Projektion der Bewegungen von 3D-Punkten in die Bildebene – zu berechnen [49]. Dabei wird ein Bild als eine zeitabhängige Funktion angenommen:

$$I(x, y, t + \tau) = I(x - \xi(x, y, t, \tau), y - \eta(x, y, t, \tau)). \tag{B.1}$$

Mit anderen Worten, ein Bild I, das zum Zeitpunkt $t + \tau$ aufgenommen wurde, kann durch eine Verschiebung $\delta = (\xi, \eta)$ von einzelnen Punkten $\mathbf{x} = (x, y)$ des Bildes, das zum Zeitpunkt t aufgenommen wurde, berechnet werden.

Der Verschiebungsvektor δ ist eine Funktion, die von Bildkoordinaten $\mathbf{x}$ abhängt. Diese kann auch in einer kleinen Umgebung – Fenster – von einem Aufpunkt stark variieren. Eine affine Repräsentation der Verschiebung stellt in Vergleich zu einer reinen Translation eine bessere Modellierung dar:

$$\delta = D\mathbf{x} + \mathbf{d},$$

wobei

$$D = \begin{pmatrix} d_{xx} & d_{xy} \\ d_{yx} & d_{yy} \end{pmatrix} \quad \text{und} \quad \mathbf{d} = \begin{pmatrix} d_x \\ d_y \end{pmatrix}$$

die Deformationsmatrix und $\mathbf{d}$ den Translationsvektor darstellen. Die Bildkoordinaten $\mathbf{x}$ sind dabei relativ zu dem Aufpunkt des Fensters gemessen.

Die Bewegung eines Punktes $\mathbf{x}$ im Bild I zu dem Punkt $\mathbf{y}$ im Bild J wird durch

$$\mathbf{y} = \left(\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + D \right) \mathbf{x} + \mathbf{d} = A\mathbf{x} + \mathbf{d}$$

oder

$$J(A\mathbf{x} + \mathbf{d}) = I(\mathbf{x}) \tag{B.2}$$

dargestellt. Die Berechnung des optischen Flusses für zwei gegebene Bilder I und J bedeutet in diesem Kontext, die Parameter der Deformation für jeden Bildpunkt in einer gewählten Umgebung (Fenstergröße) zu finden.

Da die Daten verrauscht sind und das affine Modell nicht exakt der Kamerageometrie entspricht, wird eine auf Gleichung (B.2) basierende alternative Minimierungsaufgabe gestellt:

$$\min_{A,\mathbf{d}} \epsilon = \int \int_W (J(A\mathbf{x} + \mathbf{d}) - I(\mathbf{x}))^2 w(\mathbf{x}) d\mathbf{x}, \tag{B.3}$$

wobei W die Fenstergröße und $w(\mathbf{x})$ die Wichtungsfunktion sind. Typischerweise wird die Identitätsfunktion oder eine Gauss-ähnliche Funktion mit Zentrum im Aufpunkt als Wichtungsfunktion gewählt.

Um ϵ zu minimieren, wird (B.3) im Bezug auf die unbekannten Parameter der Transformation abgeleitet und auf Null gesetzt. Das Ergebnis kann durch die Taylor-Approximation linearisiert werden:

$$J(A\mathbf{x} + \mathbf{d}) = J(\mathbf{x}) + \mathbf{g}^T(\mathbf{u}). \tag{B.4}$$

Es ergibt sich folgendes lineares Gleichungssystem der Dimension 6×6 ([94]):

$$T\mathbf{z} = \mathbf{a}, \tag{B.5}$$

wobei

$$\mathbf{z}^T = (d_{xx}\, d_{yx}\, d_{xy}\, d_{yy}\, d_x\, d_y),$$

$$\mathbf{a} = \int\int_W (I(\mathbf{x}) - J(\mathbf{x})) \begin{pmatrix} xg_x \\ xg_y \\ yg_x \\ yg_y \\ g_x \\ g_y \end{pmatrix} w d\mathbf{x}$$

und

$$T = \int\int_W \begin{pmatrix} U & V \\ V^T & Z \end{pmatrix} w d\mathbf{x}$$

mit

$$U = \begin{pmatrix} x^2g_x^2 & x^2g_xg_y & xyg_x^2 & xyg_xg_y \\ x^2g_xg_y & x^2g_y^2 & xyg_xg_y & xyg_y^2 \\ xyg_x^2 & xyg_xg_y & y^2g_x^2 & y^2g_xg_y \\ xyg_xg_y & xyg_y^2 & y^2g_xg_y & y^2g_y^2 \end{pmatrix}, \quad V = \begin{pmatrix} xg_x^2 & xgxg_y \\ xg_xg_y & xg_y^2 \\ yg_x^2 & yg_xg_y \\ yg_xg_y & yg_y^2 \end{pmatrix}$$

und

$$Z = \begin{pmatrix} g_x^2 & g_xg_y \\ g_xg_y & g_y^2 \end{pmatrix}$$

sind. Anschließend lässt sich das linearisierte Problem beispielsweise mit dem Newton-Raphson-Algorithmus [8, 85] iterativ lösen (minimieren).

Anhang C

DISTANZMAßE

Für Aufgaben der Segmentierung des optischen Flusses oder auch zur Steuerung eines Roboterarms ist es in Rahmen dieser Arbeit von größter Bedeutung, die passenden Distanzmaße zu finden, um die Ähnlichkeit zwischen solchen Transformationen $\mathcal{T}$ wie Rotationen ($SO(n)$-Gruppe) bzw. allgemeinen Rotationen ($SE(n)$-Gruppe) messen zu können. Das gesuchte Distanzmaß $d(.,.)$ soll möglichst durch eine geeignete Metrik repräsentierbar sein oder folgende Eigenschaften zumindest lokal erfüllen:

1. $d(\mathcal{T}_i, \mathcal{T}_i) = 0$
2. $d(\mathcal{T}_i, \mathcal{T}_j) = d(\mathcal{T}_j, \mathcal{T}_i)$
3. $0 \leq d(\mathcal{T}_i, \mathcal{T}_k) \leq d(\mathcal{T}_i, \mathcal{T}_j) + d(\mathcal{T}_j, \mathcal{T}_k)$.

Eine weitere wichtige Eigenschaft ist Äquidistanz, die besagt, dass die Abstände zwischen den Transformationen invariant gegenüber ihrer Position im Raum sind:

$$d(\mathcal{T}'\mathcal{T}_i, \mathcal{T}'\mathcal{T}_j) = d(\mathcal{T}_i\mathcal{T}', \mathcal{T}_j\mathcal{T}') = d(\mathcal{T}_i, \mathcal{T}_j).$$

C.1 Distanzmaß für allgemeine Rotationen

Die Gruppe der starren Bewegungen $SE(n)$ kann auf verschiedene Arten parametrisch dargestellt werden - zum Beispiel in einer reellwertigen Matrixform, als ein spezieller Multivektor (Motor) in einer konformen geometrischen Algebra oder auch als Element einer Lie-Gruppe. In [113] wurde bewiesen, dass unabhängig von der Repräsentation keine Riemannsche Metrik für diese Gruppe existiert. Es existieren allerdings mehrere Möglichkeiten, ein Abstandsmaß für solche Transformationen zu definieren.

Die einfachste Möglichkeit besteht darin, den Abstand zwischen zwei Transformationen als Euklidische Metrik im Parameterraum dieser Transformationen aufzufassen:

$$d(\mathcal{T}_i, \mathcal{T}_j) := \| \operatorname{Par}(\mathcal{T}_i) - \operatorname{Par}(\mathcal{T}_j) \|_2, \tag{C.1}$$

wobei $\operatorname{Par}(\mathcal{T}_i)$ den Vektor beschreibt, der die Parameter der Transformation $\mathcal{T}_i$ enthält. Offensichtlich werden in diesem Fall die Metrikeigenschaften erfüllt. Diese Metrik kann man unabhängig von der Parametrisierung verwenden. Die Eigenschaften bleiben sowohl bei der kleinstmöglichen Parametrisierung[1] als auch bei der Matrixrepräsentation (mit der Frobeniusnorm) oder auch bei einer Multivektorrepräsentation erfüllt . Allerdings beschreibt diese Metrik die Abstände im Parameterraum und nicht in dem Raum, in welchem die Transformationen wirken – es existiert keine sinnvolle geometrische Interpretation. Auch die Äquidistanz-Eigenschaft wird nicht erfüllt.

Vom geometrischen Standpunkt stellt der folgendermaßen definierte Abstand eine bessere Alternative dar:

$$d(\mathcal{T}_i, \mathcal{T}_j) := \| \operatorname{Par}(\mathcal{T}_{ij}) \|_2 \tag{C.2}$$

mit $\mathcal{T}_{ij} = \mathcal{T}_j \mathcal{T}_i^{-1}$. Je nach gewählter Parametrisierung werden die ersten beiden Metrikeigenschaften erfüllt [76]. Insbesondere die Versor-Repräsentation erfüllt diese, wenn als Parametrisierung alle besetzen Komponenten des Motor-Multivektors bis auf Skalar genommen werden, da einerseits bei einer Null-Transformation nur die Skalar-Komponente belegt ist und andererseits das Inverse gleich dem Reversen ist und somit

$$\| \operatorname{Par}(\mathcal{T}) \|_2 = \| \operatorname{Par}(\mathcal{T}^{-1}) \|_2$$

gilt. Es folgt daraus, dass auch die Äquidistanzeigenschaft (jedoch nur bei der Rechtsmultiplikation) erfüllt wird:

$$\begin{aligned} d(\mathcal{T}_i, \mathcal{T}_j) &= \| \operatorname{Par}(\mathcal{T}_j \mathcal{T}_i^{-1}) \|_2 = \| \operatorname{Par}(\mathcal{T}_j \mathcal{T}' \mathcal{T}'^{-1} \mathcal{T}_i^{-1}) \|_2 \\ &= \| \operatorname{Par}((\mathcal{T}_j \mathcal{T}')(\mathcal{T}_i \mathcal{T}')^{-1}) \|_2 = d(\mathcal{T}_i \mathcal{T}', \mathcal{T}_j \mathcal{T}') \end{aligned}$$

Für reine Rotation oder reine Translation werden sogar alle geforderten Eigenschaften erfüllt. Die Gleichung (C.2) stellt in diesem Fall eine Riemannsche Metrik dar.

[1] Zum Beispiel ist die kleinstmögliche Parametrisierung einer 2D-Rotation durch den Rotationswinkel (nur ein Parameter) gegeben. Jede weitere Darstellung (Matrix, Quaternion, Multivektor usw.) erfordert mehrere Parameter.

C.2 Distanzmaß für die Fundamentalmatrizen

Die Fundamentalmatrizen sind singuläre 3×3 Matrizen vom Rang 2. Sie genügen der Gleichung (A.8), sind bis auf einen Skalarfaktor definiert und bilden Äquivalenzklassen. Wegen ihrer Singularität ist keine Normierung bezüglich der Determinanten möglich. Typischerweise wird eine Normierung bezüglich des letzten Elementes der Matrix, wenn dieses ungleich Null ist, durchgeführt. In diesem Fall kann ein auf der Gleichung (C.1) basierendes Distanzmaß verwendet werden.

Eine geometrisch anschauliche Interpretation des Abstandes wird durch eine Beschreibung mit Epipolen gegeben. Epipole sind Nullvektoren einer Fundamentalmatrix:

$$F\mathbf{e}_1 = 0 \quad \text{und} \quad \mathbf{e}_2 F = 0$$

und beschreiben die Projektion des optischen Zentrums einer Kamera in die andere Kamera in einem Stereosystem. Ausschlaggebend sind dabei nur die intrinsischen Parameter sowie die relative Position der Kameras zueinander.

Darauf basierend kann ein Distanzmaß folgendermaßen definiert werden:

$$d(F_i, F_j) := \min_{k,l \in \{1,2\}; k \neq l} \left\{ \sqrt{\|\mathbf{e}_k^i - \mathbf{e}_k^j\|_2^2 + \|\mathbf{e}_l^i - \mathbf{e}_l^j\|_2^2 + \|\mathbf{e}_k^i - \mathbf{e}_l^j\|_2^2 + \|\mathbf{e}_l^i - \mathbf{e}_k^j\|_2^2} \right\}, \tag{C.3}$$

wobei $\mathbf{e}_k^i$ den k-ten Epipol der i-ten Fundamentalmatrix bezeichnet.

Die ersten beiden Metrikeigenshaften sind erfüllt, da das Maß auf der Euklidischen Norm aufbaut und die Minimumbildung die Symmetrie sicherstellt.

Eine Schwäche dieser Darstellung besteht jedoch darin, dass die Epipole und die Fundamentalmatrizen die geometrische Konfiguration eines Mehrkamerasystems nicht eindeutig beschreiben: Eine Kamera kann sich entlang der Achse, welche die beiden optischen Zentren eines Stereosystems verbinden, bewegen, ohne die Position der Epipole in der Bildebene bzw. die Komponenten der Fundamentalmatrix zu verändern.

LITERATURVERZEICHNIS

[1] E. Abraham-Mumm. Bestimmung der Gesichtspose mit kuenstlichen neuronalen Netzen. Diplomarbeit, Christian-Albrechts-Universität zu Kiel, Institut für Informatik und Praktische Mathematik, 1998.

[2] Y. S. Abu-Mostafa. The Vapnik-Chervonenkis dimension: Information versus complexity in learning. *Neural Computation*, 1(3):312–317, 1989.

[3] V. Banarer. Rekonstruktion der Gestalt und Bewegung im Rahmen der Geometrischen Algebra. Diplomarbeit, Christian-Albrechts-Universität zu Kiel, Institut für Informatik und Praktische Mathematik, 1999.

[4] V. Banarer, C. Perwass, and G. Sommer. Design of a multilayered feedforward neural network using hypersphere neurons. In N. Petkov and M. A. Westenberg, editors, *Proc. 10th Int. Conf. Computer Analysis of Images and Patterns, CAIP 2003, Groningen, The Netherlands, August 2003*, volume 2756 of *LNCS*, pages 571–578. Springer-Verlag, 2003.

[5] V. Banarer, C. Perwass, and G. Sommer. The hypersphere neuron. In *11th European Symposium on Artificial Neural Networks, ESANN 2003, Bruges*, pages 469–474. d-side publications, Evere, Belgium, 2003.

[6] J. L. Barron. `ftp://csd.uwo.ca/pub/vision/`, 1998.

[7] J. L. Barron, D. J. Fleet, and S. S. Beauchemin. Performance of optical flow techniques. *International Journal Computer Vision IJCV*, 12(1):43–77, 1994.

[8] A. Ben-Israel. A Newton-Raphson method for the solution of systems of equations. *J. Math. Anal. Appl.*, 15:243–252, 1966.

[9] S. Birchfield. `http://www.ces.clemson.edu/~stb/klt`, 2004.

[10] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[11] P. Bouthemy and J. Santillana Rivero. A hierarchical likelihood approach for region segmentation. In *ICCV*, pages 463–467, 1987.

[12] J. Bruske. *Dynamische Zellstrukturen - Theorie und Anwendung eines KNN-Modells*. Dissertation, Bericht Nr. 9809, Christian-Albrechts-Universität zu Kiel, Institut für Informatik und Praktische Mathematik, 1998.

[13] S. Buchholz. *Elements of The Theory of Neural Computation with Clifford Algebras*. PhD thesis, Christian-Albrechts-Universität zu Kiel, Institut für Informatik und Praktische Mathematik, 2004, to be published.

[14] S. Buchholz and G. Sommer. A hyperbolic multilayer perceptron. In S. I. Amari, C. L. Giles, M. Gori, and V. Piuri, editors, *International Joint Conference on Neural Networks, IJCNN 2000, Como, Italy*, volume 2, pages 129–133. IEEE Computer Society Press, 2000.

[15] S. Buchholz and G. Sommer. Learning geometric transformations with Clifford neurons. In G. Sommer and Y. Zeevi, editors, *2nd International Workshop on Algebraic Frames for the Perception-Action Cycle, AFPAC 2000, Kiel*, volume 1888 of *LNCS*, pages 144–153. Springer-Verlag, 2000.

[16] S. Buchholz and G. Sommer. Quaternionic spinor MLP. In *8th European Symposium on Artificial Neural Networks, ESANN 2000, Bruges*, pages 377–382, 2000.

[17] S. Buchholz and G. Sommer. Clifford algebra multilayer perceptrons. In G. Sommer, editor, *Geometric Computing with Clifford Algebra*, pages 315–334. Springer-Verlag, Heidelberg, 2001.

[18] S. Chen, C. F. N. Cowan, and P. M. Grant. Orthogonal least squares learning algorithm for radial basis function networks. *EEE Transactions on Neural Networks*, 2(2):302–309, 1991.

[19] Lehrstuhl Kognitive Systeme Christian-Albrechts-Universität zu Kiel, Institut für Informatik und Praktische Mathematik. Paclib. `http://www.ks.informatik.uni-kiel.de/~paclib/`, 2004.

[20] W. K. Clifford. *Mathematical Papers*. Macmillian, London, 1882.

[21] R. Collobert, S. Bengio, and J. Mariethoz. Torch3. `http://www.torch.ch`, 2004.

[22] University of Otago Dunedin New Zealand Computer Vision Research Group, Department of Computer Science. `http://www.cs.otago.ac.nz/research/vision/Research/OpticalFlow/opticalflow.html`, 2000.

[23] T. Cormen, C. Leiserson, R. Rivest, and C. Stein. Introduction to algorithms. pages 558–562. Cambridge, MA: MIT Pres, 2001.

[24] G. Cybenko. Approximation by superposition of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2:303–314, 1989.

[25] H. B. Demuth and M. H. Beale. *Neural Network Design*. PWS Publishing Company, Boston, MA, 1996.

[26] D. H. Deterding. *Speaker Normalisation for Automatic Speech Recognition*. PhD thesis, University of Cambrige, 1989.

[27] E. Dimitriadou. Convex clustring methods and clustering indexes. `citeseer.nj.nec.com/dimitriadou01convex.html`.

[28] L. Dorst. Honing geometric algebra for its use in the computer sciences. In G. Sommer, editor, *Geometric Computing with Clifford Algebra*, pages 127–152. Springer-Verlag, 2001.

[29] L. Dorst, C. Doran, and J. Lasenby, editors. *Applications of Geometric Algebra in Computer Science and Engineering*. Birkhäuser, 2002.

[30] R. Duda and P. Hart. *Pattern Classification and Scene Analysis*. John Wiley and Sons, 1973.

[31] S. E. Fahlman and C. Lebiere. The cascade-correlation learning architecture. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2, pages 524–532, Denver 1989. M. Kaufmann, San Mateo, 1990.

[32] R. A. Fisher. The use of multiple measurements in axonomic problems. *Annals of Eugenics 7*, pages 179–188, 1936.

[33] G. Flake. Square unit augmented, radially extended, multilayer perceptrons. In Orr and Muller, editors, *Neural Networks: Tricks of the Tade*, pages 145–163. Springer-Verlag, 1998.

[34] R. W. Floyd. Algorithm 97: Shortest path. *Communications of the ACM*, 5(6):345, 1962.

[35] B. Fritzke. A growing neural gas network learns topologies. In G. Tesauro, D. S. Touretzky, and T. K. Leen, editors, *Advances in Neural Information Processing Systems 7*, pages 625–632. MIT Press, Cambridge MA, 1995.

[36] S. Geman, E. Bienenstock, and R. Doursat. Neural networks and bias/variance dilema. *Neural Computation*, 4(1):1–58, 1992.

[37] C. L. Giles, R. D. Griffin, and T. Maxwell. Encoding geometric invariances in higher-order neural networks. In *Neural Information Processing Systems*, pages 301–309, 1988.

[38] F. Girosi and T. Poggio. Networks and the best approximation property. *Biological Cybernetics*, 63(3):169–176, 1990.

[39] Claus Gramkow. On averaging rotations. *Int. J. Comput. Vision*, 42(1-2):7–16, 2001.

[40] A. Griewank. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. Number 19 in Frontiers in Appl. Math. SIAM, Philadelphia, PA, 2000.

[41] A. J. Hahn. *Quadratic Algebras, Clifford Algebras, and Arithmetic Witt Groups*. Springer–Verlag, 1994.

[42] J. Hakala and R. Eckmiller. Partition of unity RBF networks are universal funcion approximators. In *ICANN*, pages 459–462. Amer Elsevier, 1994.

[43] R. Hartley. In defence of the 8-point algorithm. *Pattern Analysis and Machine Intelligence*, 19(6):580–593, 1997.

[44] E. Hartman, J. Keeler, and J. Kowalski. Layered neural networks with Gaussian hidden units as universal approximators. *Neural Computations*, 2(2):210–215, 1990.

[45] M. Hassoun. *Fundamentals of Artificial Neural Networks*. Number 19 in Frontiers in Appl. Math. MIT, 1995.

[46] D. Hestenes, H. Li, and A. Rockwood. New algebraic tools for classical geometry. In G. Sommer, editor, *Geometric Computing with Clifford Algebra*, pages 3–23. Springer-Verlag, 2001.

[47] D. Hestenes and G. Sobczyk. *Clifford Algebra to Geometric Calculus*. Reidel Publ. Comp., 1984.

[48] D. Hestenes and R. Ziegler. Projective geometry with Clifford algebra. *Acta Applicandae Mathematicae*, 23:25–63, 1991.

[49] B. K. P. Horn. *Robot Vision*. MIT Press, Cambridge, 1986.

[50] K. Hornik. Approximation capabilities of multilayer feedforward neural networks. *Neural Networks*, 4:251–257, 1990.

[51] S. L. Horowitz and T. Pavlidis. Picture segmentation by a directed split and merge procedure. In *ICPR*, pages 424–433, 1974.

[52] L. Hoyle. `http://www.ku.edu/cwis/units/IPPBR/java/iris/irisglyph.html`.

[53] J. J. Koenderink and A. J. van Doorn. Affine structure from motion. *Journal optical Society of America*, 8(2):377–385, 1991.

[54] T. Kohonen. *Self-Organizing Maps*. Springer-Verlag, 1995.

[55] K. J. Lang and M. J. Witbrock. Learning to tell two spirals apart. In D. S. Touretzky, G. E. Hinton, and T. Sejnowski, editors, *Connectionist Models Summer School*. M. Kaufmann, 1988.

[56] R. H. Laprade and M. F. Doherty. Split-and-merge segmentation using an f test criterion. In *SPIE*, volume 758.

[57] C.J. Latombe. *Robot Motion Planning*. Kluwer Academic, Boston, MA, 1991.

[58] Y. G. Leclerc. Constructing simple stable descriptions for image partitioning. *International Journal of Computer Vision*, 3:72–102, 1989.

[59] H. Li, D. Hestenes, and A. Rockwood. A universal model for conformal geometries. In G. Sommer, editor, *Geometric Computing with Clifford Algebra*, pages 77–118. Springer-Verlag, 2001.

[60] H. Lipson and H. T. Siegelmann. Clustering irregular shapes using high-order neurons. *Neural Computation*, 12(10):2331–2353, 2000.

[61] H.C. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293:133–135, 1981.

[62] P. Lounesto. *Clifford Algebras and Spinors*. Cambridge University Press, 1997.

[63] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 1981 DARPA Image Understanding Workshop*, pages 121–130, April 1981.

[64] Q. T. Luong and O. D. Faugeras. The fundamental matrix: Theory, algorithms, and stability analysis. *International Journal Computer Vision IJCV*, 17(1):43–75, 1996.

[65] D. MacKay. `http://www.inference.phy.cam.ac.uk/mackay/c/macopt.html`, 2002.

[66] T. Martinez. *Selbstorganisierende Neuronale Netzwerkmodelle zur Bewegungssteuerung*. PhD thesis, Physik-Department, TU Muenchen, 1992.

[67] T. Martinez and J. Schulten. Topology representig networks. *Neural Networks*, 7:505–522, 1994.

[68] M. Minsky and S. Papert. *Perceptrons*. Cambridge: MIT Press, 1969.

[69] J. Moody and C. J. Darken. Fast learning in networks of locally-tuned processing units. *Neural Computation*, 1(2):281–294, 1989.

[70] D. Mumford and J. Shah. Boundary detection by minimizing functionals. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 22–26, 1985.

[71] H.-H. Nagel, G. Socher, H. Kollnig, and M. Otte. Motion boundary detection in image sequences by local stochastic tests. In *European Conference on Computer Vision, ECCV'94*, Stockholm, Sweden, May 2-6, 1994.

[72] E. Nishimura, G. Xu, and S. Tsuji. Motion segmentation and correspondence using epipolar constraint. In *Asian Conference on Computer Vision*, pages 199–204, Osaka, Japan, 1996.

[73] E. Oja and S. Kaski. *Kohonen Maps*. Elsevier, 1999.

[74] M. Orr. Introduction to radial basis function networks. `http://www.anc.ed.ac.uk/~mjo/rbf.html`.

[75] M. Otte and H.-H Nagel. Optical flow estimation: Advances and comparisons. In *European Conference on Computer Vision, ECCV'94*, pages 51–60, Stockholm, Sweden, May 2-6, 1994.

[76] F. C. Park and R.W. Brockett. Kinematic dexterity of robotic mecanism. *International Journal of Robotics Research*, 13(1):1–15, 1994.

[77] J. Park and I. W. Sandberg. Approximation and radial-basis-function networks. *Neural Comput.*, 5(2):305–316, 1993.

[78] X. Pennec. Computing the mean of geometric features - application to the mean rotation. Research Report RR-3371, INRIA, March 1998.

[79] C. Perwass. Clucalc. `http://www.clucalc.info`, 2004.

[80] C. Perwass, V. Banarer, and G. Sommer. Spherical decision surfaces using conformal modelling. In B. Michaelis and G. Krell, editors, *25. Symposium für Mustererkennung, DAGM 2003, Magdeburg*, volume 2781 of *LNCS*, pages 9–16. Springer-Verlag, Berlin, 2003.

[81] C. Perwass and D. Hildebrandt. Aspects of geometric algebra in Euclidean, projective and conformal space. Technical Report Number 0310, Christian-Albrechts-Universität zu Kiel, Institut für Informatik und Praktische Mathematik.

[82] C. Perwass and G. Sommer. Numerical evaluation of versors with Clifford algebra. In L. Dorst, C. Doran, and J. Lasenby, editors, *Applications of Geometric Algebra in Computer Science and Engineering*, pages 341–350. Proc. AGACSE 2001, Cambridge, UK, Birkhäuser Boston, 2002.

[83] I. R. Porteous. *Clifford Algebras and the Classical Groups*. Cambridge University Press, 1995.

[84] L. Prechelt. Proben1: A set of neural network benchmark problems and benchmarking rules. Technical Report 21/94, 1994.

[85] W. H. Press, S. A. Teukolsky, W. T. Vettering, and B. P. Flannery. *Numerical Recipes in C++*. Cambridge University Press, 2002.

[86] M. Riesz. *Clifford Numbers and Spinors*. Kluwer Academic Publishers, 1993.

[87] A. J. Robinson. *Dynamic Error Propagation Networks*. PhD thesis, Cambrige University, 1989.

[88] R. Rojas. *Theorie der Neuronalen Netze*. Springer-Verlag, Heidelberg, 1993.

[89] B. Rosenhahn. *Pose Estimation Revisited*. PhD thesis, Technical Report Number 0308, Christian-Albrechts-Universität zu Kiel, Institut für Informatik und Praktische Mathematik, 2003.

[90] B. Rosenhahn, C. Perwass, and G. Sommer. Pose estimation of 3d free-form contours. Technical Report Number 0207, Christian-Albrechts-Universität zu Kiel, Institut für Informatik und Praktische Mathematik, August 2002.

[91] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In *Parallel Distributed Processing*, pages 318–362. MIT Press, Cambridge MA, 1986.

[92] R. Shadmehr and F. Mussa-Ivaldi. Geometric structure of the adaptive controller of the human arm. Technical Report AIM-1437, 1993.

[93] R. Sharma and H. Sutanto. A framework for robot motion planning with sensor constraints. *IEEE Transactions on Robotics and Automation*, 13(1):61–73, 1997.

[94] J. Shi and C. Tomasi. Good features to track. Technical report, 1993.

[95] J. Shi and C. Tomasi. Good features to track. In *1994 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, pages 593 – 600, 1994.

[96] S. M. Smith. Reviews of optic flow, motion segmentation, edge finding and corner finding. `http://www.fmrib.ox.ac.uk/~steve/review/`, 1997.

[97] S. Soatto and P. Perona. Three dimensional transparent structure segmentation and multiple 3d motion estimation from monocular perspective image sequences. In *IEEE Workshop on Motion of Nonrigid and Articulated Objects*, pages 228–235.

[98] F. Sommen. An algebra of abstract vector variables. *Porugaliae Math.*, 54:287–310, 1997.

[99] F. Sommen. The problem of defining abstract bivectors. *Result. Math.*, 31:148–160, 1997.

[100] G. Sommer, editor. *Geometric Computing with Clifford Algebra*. Springer-Verlag, 2001.

[101] B. Speelpenning. *Compiling Fast Partial Derivatives of Functions Given by Algorithms*. PhD thesis, Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana-Champaign, IL, January 1980.

[102] C. Tomasi and T. Kanade. Shape and motion from image streams: a factorization method - part 3 detection and tracking of point features. Technical Report CMU-CS-91-132, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, April 1991.

[103] C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: a factorization method. *International Journal Computer Vision IJCV*, 9(2):137–54, 1992.

[104] P. H. S. Torr and D. W. Murray. Stochastic motion clustering. In J.-O. Eklundh, editor, *European Conference on Computer Vision, ECCV'94*, pages 328–338. Springer–Verlag, 1994.

[105] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, 1995.

[106] N. B. Venkateswarlu and P. S. V. S. K. Raju. Fast isodata clustering algorithms. *Pattern Recogn.*, 25(3):335–342, 1992.

[107] T. Villmann, R. Der, M. Herrmann, and T. Martinez. Topology preservation in selforganising feature maps: Exact definition and measurement. *ITONN*, 8(3):256–265, 1997.

[108] T. Villmann, R. Der, and T. Martinez. A novel approach to measure the topology preservation of feature maps. *ICANN*, pages 289–301, 1994.

[109] J. Weber. Scene partitioning via statisctic-based region growing. Technical Report UCB/CSD-94-817, Computer Science Division (EECS), University of California, Berkley, 1994.

[110] J. Weber and J. Malik. Rigid-body segmentation and shape-description from dense optical-flow under weak perspective. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(2):139–143, February 1997.

[111] H. Wendland. Piecewise polynomials, positive definite and compactly supported radial basis functions of minimal degree. *Advances in Computational Mathematics*, 4:389–396, 1995.

[112] A. Wieland and S. E. Fahlman. `http://www.ibiblio.org/pub/academic/computer-science/neural-networks/programs/bench/two-spirals`, 1993.

[113] M. Zefran, V. Kumar, and C. Croke. Choice of Riemannian metrics for rigid body kinematics. In *ASME 24th Biennial Mechanisms Conference*, Irvine, CA, 1996.

[114] M. Zeller, R. Sharma, and K. Schulten. Topology representing network for sensor-based robot motion planning. In *World Congress on Neural Networks*, pages 100–103, 1996.

GLOSSAR

BMU	Best Matching Unit
DCS	Dynamic Cell Strukture
KLT	Kanade-Lucas-Tomasi
MLP	Multi-Layer-Perceptron
MLHP	Multi-Layer-Hypersphere-Perceptron
MLHPS	Multi-Layer-Hypersphere-Perceptron with Simplified Derivatives
OTPM	Optimal Topology Preserving Map
PCM	Perceptual Control Manifold
RBF	Radiale Basisfunktion
SLP	Single-Layer-Perceptron
SLHP	Single-Layer-Hypersphere-Perceptron
SMU	Second Matching Unit
SVD	Singular Value Decomposition
SVM	Support Vector Machine

INDEX