Synthese und Optimierung von Konstruktionsbäumen aus unstrukturierten räumlichen Daten

Markus Friedrich

Dissertation
an der Fakultät für Mathematik, Informatik und Statistik
der Ludwig-Maximilians-Universität
München

vorgelegt von
Markus Friedrich

München, den 22. Dezember 2021

Synthese und Optimierung von Konstruktionsbäumen aus unstrukturierten räumlichen Daten

Markus Friedrich

Dissertation
an der Fakultät für Mathematik, Informatik und Statistik
der Ludwig-Maximilians-Universität
München

vorgelegt von
Markus Friedrich

1. Berichterstatter/in:	Prof. Dr. rer. nat. Claudia Linnhoff-Popien
2. Berichterstatter/in:	Prof. Dr. rer. nat. habil. Dr. h. c. Alexander Schill
Tag der Einreichung:	22. Dezember 2021
Tag der Disputation:	13. April 2022

Eidesstattliche Versicherung

(siehe Promotionsordnung vom 12.07.11, § 8, Abs. 2 Pkt. 5)

Hiermit erkläre ich an Eides statt, dass die Dissertation von mir selbstständig, ohne unerlaubte Beihilfe angefertigt ist.

München, 20. April 2022 Markus Friedrich

Danksagung

Die vorliegende Dissertation entstand im Rahmen meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Lehrstuhl für Mobile und Verteilte Systeme der Ludwig-Maximilians-Universität München. Während meiner Zeit dort wurde mir die wertvolle Unterstützung von Personen zu Teil, denen ich im Folgenden meinen Dank aussprechen möchte.
Nicht nur für die Gelegenheit, Mitglied ihres Lehrstuhlteams zu werden, sondern auch für die stets respekt- und vertrauensvolle Zusammenarbeit möchte ich der Lehrstuhlinhaberin Frau Prof. Dr. Claudia Linnhoff-Popien danken. Ich konnte sowohl von ihrer Erfahrung bzgl. formaler und inhaltlicher Aspekte der Dissertation als auch von den exzellenten Rahmenbedingungen am Lehrstuhl profitieren, die dort aufgrund ihres nimmermüden Einsatzes anzutreffen sind.
Des Weiteren gilt mein herzlicher Dank Herrn Prof. Dr. rer. nat. habil. Dr. h. c. Alexander Schill, der sich freundlicherweise bereit erklärte, die Rolle des Zweitberichterstatters zu übernehmen. Ebenfalls danken möchte ich Herrn Prof. Dr. Andreas Butz für die Übernahme des Vorsitzes der Prüfungskommission sowie Herrn Prof. Dr. Christian Böhm für seine Bereitschaft, als Ersatzprüfer zur Verfügung zu stehen.
In meinen Jahren am Lehrstuhl hatte ich die Gelegenheit, mit unglaublich fähigen, inspirierenden und integren Kollegen zusammenzuarbeiten. Ihnen gebührt besonderer Dank, haben sie doch großen Anteil am Gelingen meiner Forschungsvorhaben und an meiner persönlichen Weiterentwicklung. Im Besonderen möchte ich Dr. André Ebert danken, der mich von meinen Anfängen am Lehrstuhl an beratend und motivierend begleitete und sich zudem bereiterklärte, inhaltliche Fragen zur Dissertation ausgiebig zu diskutieren. Für ihr wertvolles Feedback zur vorliegenden Arbeit geht mein Dank ebenfalls an Dr. Sebastian Feld und Sebastian Zielinski. Bei Pierre-Alain Fayolle von der Universität Aizu (Aizuwakamatsu, Japan) möchte ich mich herzlich für eine außergewöhnliche Zusammenarbeit bedanken, die meine Forschung am Lehrstuhl prägte und bereicherte. Merci beaucoup, Pierre-Alain!
Danken möchte ich auch meiner Schwester Anna Maria sowie meinen Eltern Rita und Thomas, die mir auch in schwierigen Phasen zur Seite standen und stets an mich und meine Ziele glaubten. Zuletzt gilt mein Dank Matilde für ihre liebevolle Zuwendung und Engelsgeduld.

Dankeschön.

Zusammenfassung

Sensorsysteme für die dreidimensionale Abtastung von Objektoberflächen sind in vielen Bereichen des täglichen Lebens omnipräsent. Moderne Smartphones und Spielekonsolen im Heimanwenderbereich sowie professionelle Systeme, z.B. eingesetzt zur Qualitätssicherung in Fertigungsprozessen, beinhalten Hard- und Softwarekomponenten zur Ermittlung räumlicher Daten. Aus entsprechenden Quellen stammende Datensätze bestehen meist aus einzelnen dreidimensionalen Punkten, die in unstrukturierter Form und potentiell messfehlerbehaftet vorliegen.
Oft ist die Erzeugung dieser Punktwolke nur der erste Schritt innerhalb eines komplexen Verarbeitungsprozesses, an dessen Ende eine Repräsentation der räumlichen Daten steht, die für den entsprechenden Anwendungsfall als optimal angesehen wird. Ein solcher Anwendungsfall ist z.B. die automatische Erzeugung von Architekturplänen oder das *Reverse Engineering* (RE), also die Analyse des Aufbaus und der Funktionsweise eines Produkts. In beiden Fällen ist eine Repräsentation von Vorteil, die unnötige Details abstrahiert und dabei weiterführende Information über den Aufbau des Objekts und dessen elementare Bausteine beinhaltet. Eine solche Darstellung ist die sog. *Constructive Solid Geometry* (CSG)-Repräsentation, die Modelle als Baumstruktur bestehend aus Booleschen Mengenoperatoren in den inneren Knoten und geometrischen Primitiven in den äußeren Knoten beschreibt. Dabei ist die manuelle Erzeugung dieses sog. *Konstruktionsbaums* (KB) aus einer Punktwolke zeitaufwendig und für komplexe Datensätze kaum zu bewerkstelligen.
Aus diesem Grund werden in dieser Arbeit Methoden vorgestellt, die das Problem der automatischen Synthese von KBs aus fehlerbehafteten Punktwolken robust und effizient lösen. Die vorgestellten Verfahren werden dabei in eine eigens entwickelte Prozess-Pipeline eingebettet und miteinander verknüpft. Den Anfang macht die Einführung eines Systems, das geometrische Primitive, wie Kugeln, Zylinder und allgemeine konvexe Polytope, mittels *Maschinellem Lernen* (ML) und *Evolutionären Algorithmen* (EA) in der Eingabepunktwolke detektiert und in diese einpasst. Dieser folgt die Vorstellung einer Methode, die das eigentliche KB-Syntheseproblem für bekannte Primitive löst und dazu auf graphbasierte Partitionierungs- und Vereinfachungsstrategien zur Steigerung von Laufzeiteffizienz und Robustheit zurückgreift. Da die Repräsentation als KB nicht eindeutig ist, lassen sich zusätzliche Metriken, wie z.B. die Baumgröße, bestimmen und existierende KBs entsprechend optimieren. Dieses Problem steht abschließend im Fokus dieser Arbeit, zu dessen vorgestellter Lösung ein Spektrum unterschiedlicher Lösungsstrategien evaluiert und diskutiert wird.

Abstract

Sensor systems for three-dimensional object surface scanning are omnipresent in many areas of daily life. Modern smartphones and game consoles for home users and professional systems, e.g., used for quality assurance in manufacturing processes, contain hard- and software components for measuring spatial data. Data sets originating from such sources usually consist of individual three-dimensional points which are unstructured and potentially subject to measurement errors.
Often, the generation of this so-called point cloud is only the first step within a complex processing procedure which results in a spatial data representation that is considered optimal for a specific use case. Such a use case is, for example, the automatic generation of architectural plans or *Reverse Engineering* (RE), i.e., the analysis of a product's structure and functionality without any prior knowledge. In both cases, it is advantageous to obtain a representation that abstracts unnecessary details while providing more information about an object's structure and elementary building blocks. Such a representation is called *Constructive Solid Geometry* (CSG), which describes models as a tree structure consisting of Boolean set operators in the inner nodes and geometric primitives in the outer nodes. However, the manual generation of these so-called *Construction Trees* (CTs) based on a measured point cloud is time-consuming and hardly feasible for complex data sets.
For this reason, this work presents methods that can robustly and efficiently solve the problem of automatically synthesizing CTs from error-prone point clouds. The presented methods are thereby embedded and interconnected in a newly developed process pipeline. At first, a system is introduced that detects and fits geometric primitives such as spheres, cylinders and general convex polytopes in the input point cloud using *Machine Learning* (ML) and *Evolutionary Algorithms* (EA). This is followed by an introduction of a method that solves the automatic CT synthesis problem for known primitives using graph-based partitioning and simplification strategies to increase runtime efficiency and robustness. Since the representation as CTs is not unique additional metrics such as tree size can be determined, and existing CTs can be optimized accordingly. This problem is the last this work addresses for which a spectrum of different solution strategies is evaluated and discussed.

Inhaltsverzeichnis

1. Einleitung

Die weitläufige Verfügbarkeit von Sensorsystemen zur Digitalisierung von Oberflächenstrukturen ermöglicht ein breites Spektrum nützlicher Anwendungsfälle in einer Vielzahl wichtiger Wirtschaftssektoren [73, 110, 111]. So sind beispielsweise manuell oder automatisiert angefertigte 3D-Scans von Bauteilen in der Automobil- und anderen Fertigungsindustrien elementarer Bestandteil des Qualitätssicherungsprozesses, um bereits kleinste Überschreitungen festgelegter Toleranzintervalle zu detektieren [38, 102, 201]. Derartige 3D-Scans können zudem als Grundlage für ein virtuelles Modell eines physikalischen Produkts dienen. Dieser sog. *Digital Twin* ermöglicht umfassende Effizienzverbesserungen innerhalb eines digitalisierten Produktentwicklungsprozesses und ist damit zentraler Bestandteil der vierten industriellen Revolution [69, 165].
Im Unterhaltungssektor kann entsprechender Modellierungsaufwand bei der Erstellung digitaler 3D-Inhalte reduziert werden, indem vorher abgetastete Echtweltmodelle als Vorlage fungieren [78]. In beiden Anwendungsbereichen steht die Minimierung von Aufwänden, die entweder während des Produktionsprozesses (Erstellung von 3D-Inhalten) oder im Anschluss (Qualitätssicherung) entstehen, im Vordergrund.
Existierende Geräte zur Oberflächenabtastung erzeugen eine Menge dreidimensionaler Messpunkte. Diese werden ohne räumliche Struktur und in Form einer unsortierten Liste von 3D-Punktkoordinaten, einer sog. Punktwolke, an die weiteren Verarbeitungsmodule übertragen. Dabei ist es aber beispielsweise im rechnerunterstützten Konstruieren oder bei der automatischen Erzeugung eines *Digital Twins* durch *Reverse Engineering* (RE) nötig, auch weiterführende Objektinformationen zu nutzen. Dazu zählen Nachbarschaftsbeziehungen zwischen Punkten und wie diese die Oberfläche des Objekts aufspannen. Weiterhin ist von Interesse, aus welchen Grundbausteinen sich das Objekt zusammensetzt. Dabei handelt es sich oft um geometrische Primitive, wie z.B Kugeln, Ebenen oder Zylinder, die das Objekt entweder exakt oder auch nur näherungsweise beschreiben. Hat man eine Menge von geometrischen Primitiven, stellt sich darüber hinaus die Frage, wie diese miteinander kombiniert wurden, um das Objekt zu beschreiben. So könnten z.B. mehrere kleinere Kugeln aus einem Zylinder ausgeschnitten oder sechs Ebenen zu einem Quader kombiniert worden sein. Derartige Objektinformationen lassen sich in Form einer Baumstruktur darstellen, die das Objekt in eine Hierarchie aus Booleschen Mengenoperatoren (innere Knoten) und geometrischen Primitiven (Blätter) zerlegt und damit direkt für weitere manuell durchgeführte Modellierungs-

schritte nutzbar macht. Die automatische Synthese dieses sog. *Konstruktionsbaumes* (KB) aus einem Datensatz mit unorganisierter räumlicher Struktur ist rechnerisch aufwendig und bisher noch nicht zufriedenstellend gelöst worden, v.a. hinsichtlich der Prozessdauer und den unterstützten unterschiedlichen Primitiventypen. Ziel dieser Arbeit ist es daher, Lösungen zu entwickeln, die eine automatisierte und robuste Wiederherstellung von KBs ermöglichen.
Ein weiteres zu lösendes Problem betrifft die Beschaffenheit des resultierenden KBs: Ein und dasselbe Modell kann über eine unendliche Anzahl verschiedener KBs repräsentiert werden. Diese unterscheiden sich meist hinsichtlich ihrer Größe, Höhe, Breite und anderer Eigenschaften. Für die manuelle Weiterverarbeitung ist es dabei essentiell, einen KB zu nutzen, der keine redundanten Strukturen aufweist und auch hinsichtlich der enthaltenen geometrischen Primitive ein hohes Maß an Editierbarkeit garantiert. In bisherigen Veröffentlichungen wurde nur die Baumgröße als der zu optimierende Parameter berücksichtigt. Um diese Lücke zu schließen, wird im Rahmen dieser Arbeit ein neuartiges System vorgestellt, das existierende KBs hinsichtlich multipler Parameter automatisch optimiert und es dem Benutzer gleichzeitig ermöglicht, deren Gewichtung feingranular zu justieren.
Bei der Auswahl möglicher Primitive ist es ein weiteres Ziel der Arbeit, auch bisher wenig beforschte Rekonstruktionsprobleme zu lösen. So wird z.B. nicht nur die Einpassung einfacher Ebenen in die Eingabepunktwolke betrachtet, sondern auch die Kombination von Ebenen zu komplexeren Objekten, wie den konvex geformten Polyedern.
Die so gewonnene, höherwertige Information kann verwendet werden, um darauf aufbauende Variationen oder bloße Kopien des Ausgangsobjekts zu entwerfen. Dies kann, verglichen mit der komplett manuell durchgeführten Modellierung, in einem effizienteren Prozess geschehen. Dieser Effizienzgewinn im Produktentwicklungsprozess lässt sich auf vielfältige Weise nutzen. So können z.B. schnellere Iterationszyklen die Produktentwicklung flexibilisieren und damit auch den kostengünstigen Entwurf hochindividualisierter Gebrauchsgegenstände ermöglichen.
Natürlich stellt sich die Frage, ob die Ausdrucksstärke der Repräsentation mit einfachen geometrischen Grundformen ausreicht, um eine variationsreiche Modellierung von später physikalisch hergestellten Objekten zu ermöglichen. Für viele Anwendungsgebiete ist das nachweislich der Fall. So sind mechanische Bauteile für Maschinen in den verschiedensten Fertigungsbranchen zu einem hohen Prozentsatz aus Primitiven, wie Kegel, Kugel, Zylinder und Ebenen modellierbar [149].
Auch in der Architektur und im Design finden sich Stilrichtungen, die eine klare Formensprache basierend auf einer Grammatik einfacher geometrischer Grundelemente präferieren. Als Beispiel kann der weltbekannte Bauhausstil genannt werden, der auf die gleichnamige, von Walter Gropius im Jahr 1919 in Weimar gegründete Kunstschule zurückgeht und die Benutzbarkeit von Architektur und Gebrauchsgegenständen in den Fokus rückt: *Die Form folgt der*

Abbildung 1.1.: Designstücke im Bauhausstil. Ausgestellt in der Pinakothek der Moderne in München im Rahmen der Ausstellung "REFLEX BAUHAUS" zum hundertjährigen Jubiläum der Kunstschule Bauhaus 2019-2021.

Funktion. Beispiele der Bauhausschule sind in Abbildung 1.1 zu sehen. Es zeigt sich also, dass die Beschränkung auf einfache Geometrien als Grundkörper das mögliche Anwendungsspektrum kaum schmälert.
Im weiteren Verlauf dieses Kapitels werden die dieser Arbeit zugrundeliegenden und bereits veröffentlichten Arbeiten beschrieben (siehe Kapitel 1.1) sowie der Aufbau der Arbeit erläutert (siehe Kapitel 1.2).

1.1. Zugrundeliegende Vorarbeiten

Die in dieser Arbeit zusammengefassten Forschungsinhalte wurden bereits zu überwiegendem Teil auf entsprechenden Fachkonferenzen publiziert und vor internationalem Fachpublikum präsentiert. Um eine genaue Zuordnung von bereits publizierten Ergebnissen zu den entsprechenden Inhaltskapiteln zu ermöglichen, werden im Folgenden entsprechende Details über für die Arbeit relevante Publikationen gegeben und auch auf den Eigenanteil des Autors dieser Arbeit eingegangen. Die Aufzählung erfolgt dabei in der Reihenfolge der Referenzierung in den nachfolgenden Inhaltskapiteln, wenn immer dies möglich ist.

- **Evolutionary Generation of Primitive-Based Mesh Abstractions [54].** Kern dieser Arbeit bildet ein System, das dreidimensionale Geometrie in Form von Dreiecksnetzen und Punktwolken in abstrahierte Modelle, bestehend aus geometrischen Primitiven, umwandelt. Der Erzeugungsprozess wird zu diesem Zweck als kombinatorisches Optimierungsproblem formuliert und mit einem *Evolutionären Algorithmus* (EA) gelöst. Die Idee für das System stammt dabei vom Autor, dessen Implementierung und Evaluation wurde im Rahmen der Bachelorarbeit von Felip Guimèra Cuevas durchgeführt. Andreas Sedlmeier und André Ebert standen für äußerst fruchtbare Diskussionen zu inhaltlichen Fragen der Veröffentlichung zur Verfügung. Zu den Eigenanteilen des Autors zählen

neben der grundsätzlichen Idee auch die Ausarbeitung der Veröffentlichung. Inhalte dieser Publikation finden sich im in Kapitel 4.4.1.1 beschriebenen System zur automatischen Generierung von geometrischen Primitiven und in der Vorevaluation von Architekturen für das *Maschinelle Lernen* (ML) auf Punktwolken in Kapitel 4.6.3.1.

- **A Hybrid Approach for Segmenting and Fitting Solid Primitives to 3D Point Clouds [55].** In dieser Arbeit wird ein System vorgestellt, welches einfache Primitive, wie Kugeln, Zylinder und Quader in einer Punktwolke detektiert und deren Parameter schätzt. Dabei zentral sind zwei Aspekte: Zum einen ein *Künstliches Neuronales Netz* (KNN), das für jeden Punkt der Punktwolke einen Primitiventyp prädiziert und damit einen Partitionierungsschritt zur Vereinfachung des Gesamtproblems speist. Zum anderen im Mittelpunkt ist ein EA, der erkannte Ebenen zu Quadern zusammenfügt. Zum Eigenanteil des Autors gehören die Idee, deren Implementierung, die Evaluation des Systems sowie die Ausarbeitung der daraus resultierenden Publikation. Von Steffen Illium wurden diverse Anpassungen an der gewählten Netzarchitektur vorgenommen sowie deren Training durchgeführt. Pierre-Alain Fayolle half bei der Vorrecherche, bei der Fehlerbereinigung sowie bei der Evaluation des Systems und war neben dem Autor hauptsächlich verantwortlich für das Verfassen des Manuskripts. Claudia Linnhoff-Popien stand für inhaltliche und strukturgebende Diskussionen der vorgestellten Konzepte zur Verfügung. Die Publikation ist grundlegend für die Inhalte von Kapitel 4 und dient ebenfalls als Basis für die folgend erwähnte Publikation.

- **Reconstruction of Convex Polytope Compositions from 3D Point-clouds [51].** Aufbauend auf [55] ist der Fokus in dieser Arbeit auf der effizienten Rekonstruktion von arbiträren konvexen Polytopen aus Punktwolken. Im Vergleich zum in [55] vorgestellten System, das nur den Spezialfall des Quaders als konvexes Polytop unterstützt, erhöht sich damit die Problemkomplexität signifikant. Um diese zu reduzieren, werden entsprechende Partitionierungsverfahren untersucht, die das Gesamtproblem in Teilprobleme aufteilen, diese lösen und abschließend zu einer Gesamtlösung zusammenführen. Zum Eigenanteil des Autors zählt dabei die Entwicklung der Idee, die Implementierung sowie die Evaluation der Systemerweiterungen und das Ausarbeiten des Manuskripts. Pierre-Alain Fayolle war in die Vorrecherche sowie in die Fehlerbereinigung des Systems involviert. Zudem half er bei der Ausarbeitung der Veröffentlichung. Konzepte dieser Publikation finden sich in Kapitel 4 wieder, genauer bei der in Kapitel 4.5 beschriebenen Erweiterung des Basissystems.

- **CSG Tree Extraction from 3D Point Clouds and Meshes using a Hybrid Approach [56].** In dieser Arbeit wird eine komplette Prozess-Pipeline vorgestellt, die einen KB aus einer Punktwolke extrahiert und

optimiert. Zur Detektion und Einpassung von Primitiven kommt dabei das Verfahren aus [55] zum Einsatz, welches um die Optimierung von Zylinderdeckflächen und die Erkennung einfacher konvexer Polytope erweitert wurde. Die KB-Synthese erfolgt mittels eines EAs. Optimiert werden resultierende KBs per regelbasierter Redundanzentfernung, wie sie schon in [58] vorgestellt wurde. Zusätzlich enthält die Veröffentlichung eine detaillierte Charakterisierung des KB-Syntheseproblems. Dazu werden verschiedene Problemkategorien klassifiziert und deren Komplexität hergeleitet. Die Prozess-Pipeline wurde vom Autor konzeptioniert, implementiert und evaluiert. Steffen Illium führte das Training des genutzten KNNs durch. Die enthaltenen Herleitungen zur Problemkomplexität wurden von Pierre-Alain Fayolle und dem Autor zu gleichen Teilen erarbeitet. Die Publikation wurde federführend vom Autor verfasst, wobei Pierre-Alain Fayolle und Claudia Linnhoff-Popien konzeptionell beteiligt waren. Elemente veröffentlichter Inhalte sind Teil der Kapitel 3, 4 und 5.

- **Accelerating Evolutionary Construction Tree Extraction via Graph Partitioning [53].** Diese Publikation beschreibt ein graphbasiertes Verfahren zur Beschleunigung der Synthese von KBs aus einer Menge von Primitiven und einer assoziierten Punktwolke. Der sog. Intersektiongraph wird dabei in maximale Cliquen partitioniert, jedes Teilproblem per EA gelöst und die Teilergebnisse mittels eines eigens entworfenen Algorithmus zu einem Ergebnis-KB zusammengefügt. Damit lässt sich die Gesamtlaufzeit des Syntheseprozesses signifikant reduzieren, bei gleichzeitiger Verbesserung der Robustheit desselben. Die Idee zu dieser Partitionierungsstrategie stammt vom Autor. Das Verfahren zur Zusammenführung der Teilergebnisse wurde von Sebastian Feld und dem Autor zu gleichen Teilen konzipiert. Implementierung und Evaluation des Verfahrens führte der Autor durch. Die Publikation wurde hauptsächlich vom Autor verfasst, wobei Pierre-Alain Fayolle sowie Sebastian Feld und Thomy Phan beteiligt waren. Die Inhalte dieser Veröffentlichung finden sich in Kapitel 5.4.3.2 wieder.

- **Optimizing Evolutionary CSG Tree Extraction [52]** Aufbauend auf dem in [53] eingeführten Intersektionsgraphen beschreibt diese Arbeit ein weiteres Verfahren zur Partitionierung des KB-Syntheseproblems. Im Vergleich zu [53] lässt sich jedoch die Zusammenführung der Teillösungen einfacher und effizienter hinsichtlich der Größe des Ergebnis-KBs durchführen. Des weiteren wurden Verfahren vorgestellt, die eine schnellere Konvergenz des verwendeten EAs durch gezielt eingesetzte Optimierungsschritte ermöglichen. Die Idee für die Partitionierungsstrategie stammt vom Autor und Pierre-Alain Fayolle zu gleichen Teilen. Die zusätzlichen Optimierungsstrategien wurden vom Autor konzipiert. Zum weiteren Eigenanteil des Autors zählt die Implementierung sowie die Evaluation des gesamten Verfahrens. Das Manuskript wurde federführend

vom Autor verfasst, wobei Pierre-Alain Fayolle sowie Thomas Gabor bei Strukturierung und Fehlerbereinigung halfen. Claudia Linnhoff-Popien stand für inhaltliche und organisatorische Diskussionen rund um die Veröffentlichung zur Verfügung. Das vorgestellte Partitionierungsverfahren ist Teil von Kapitel 5.4.3.2.

- **Optimizing Geometry Compression using Quantum Annealing [48].** Diese Arbeit enthält die Idee zur Nutzung von *Quadratic Unconstrained Binary Optimization* (QUBO)-Formulierungen im Kontext von Problemstellungen im Bereich der KB-Optimierung und -Synthese. Dabei wird für ein hybrides Modell plädiert, das Probleme in mehrere Teilprobleme aufteilt, die dann entweder klassisch oder per *Quanten-Annealing* (QA) gelöst werden. Die Idee für diesen Ansatz wurde vom Autor zusammen mit Sebastian Feld in gleichen Teilen entwickelt Gleiches gilt für die Ausarbeitung der Veröffentlichung. Dabei stammen die domänenspezifischen Teile vom Autor, wohingegen Sebastian Feld die entsprechenden QA-spezifischen Abschnitte beisteuerte. Claudia Linnhoff-Popien stand während der Konzeptphase als Diskussionspartnerin zur Verfügung und half bei der Strukturierung der Inhalte zur Darstellung in der Publikation. Die grundsätzliche Idee zur hybriden Formulierung eines Problems aus der KB-Optimierung für die potentielle Ausnutzung von QA-Hardware, wie sie in Kapitel 6 umgesetzt wird, ist dieser Publikation entnommen.

- **A Flexible Pipeline for the Optimization of Construction Trees [58].** Kern dieser Arbeit ist eine Prozess-Pipeline, die es ermöglicht, KBs hinsichtlich verschiedener Metriken zu optimieren. Neben der Vorstellung und detaillierten Diskussion von flexibel kombinierbaren Prozessstufen wird auch eine zur Ausdrucksgröße komplementäre Optimierungsmetrik eingeführt, die durch die rekursive Messung der räumlichen Überdeckung von Teilbäumen eine zusätzliche Dimension zur Quantifizierung der manuellen Editierbarkeit von KBs hinzufügt. Das zur Optimierung der Metriken formulierte Optimierungsproblem wird mithilfe eines ganzen Spektrums an passenden Strategien gelöst, welches neben der Nutzung von EAs auch klassische Verfahren zur Minimierung Boolescher Schaltkreise sowie eine Formulierung als QUBO-Problem vorsieht. Dabei stammt die Idee für das Gesamtsystem vom Autor, genauso wie dessen Implementierung und Evaluation. Auch die Veröffentlichung wurde federführend vom Autor angefertigt. Christoph Roch entwickelte die Implementierung der QUBO-Formulierung und schrieb die entsprechenden Kapitel zum Thema. Carsten Hahn und Sebastian Feld halfen bei der Strukturierung der Publikation und standen für zahlreiche Diskussionen zu deren Inhalten zur Verfügung. Pierre-Alain Fayolle half während der Testphase der Implementierung und der Anfertigung des Manuskripts. Die Inhalte von Kapitel 6 entstammen zu großem Teil aus dieser Veröffentlichung.

- **Combining Gesture and Voice Control for Mid-Air Manipulation of CAD Models in VR Environments [57].** Fokus dieser Arbeit ist die Implementierung und Evaluation eines VR-basierten Systems zur manuellen Bearbeitung von 3D-Modellen, die als KBs repräsentiert werden. Als Benutzereingabe dient neben einem Gestenerkennungssystem auch ein Spracherkennungsmodul, das die Verwendung einfacher Sprachkommandos ermöglicht. Innerhalb der Umgebung kann der Benutzer mittels Gesten und Sprachkommandos Primitive eines Baumes selektieren und mittels einfacher Transformationen (Translation, Rotation, Skalierung) modifizieren. Neben der Darstellung des Gesamtmodells wird dabei auch der KB visualisiert, was v.a. der intuitiven Selektion von Primitiven dient. Die Publikation umfasst auch eine qualitative Evaluation des prototypisch implementierten Interaktionskonzepts in Form einer kleinen Nutzerstudie ($n = 5$). Die Idee für das System stammt vom Autor. Implementiert wurde es im Rahmen eines Einzelpraktikums von Fabian Frey. Die Veröffentlichung wurde dabei maßgeblich vom Autor konzipiert und geschrieben, wobei Stefan Langer beratend zur Seite stand. Im Ausblick von Kapitel 6 wird eine Kombination des in dieser Veröffentlichung beschriebenen Interaktionskonzepts mit der Prozess-Pipeline zur automatischen Optimierung von KBs vorgeschlagen.

Zur besseren Übersicht werden im Folgenden die in den Inhaltskapiteln thematisierten Kerninhalte den entsprechenden Publikationen zugeordnet:
Die Kerninhalte von Kapitel 4, das die Extraktion geometrischer Primitive aus Punktwolken zum Inhalt hat, wurden bereits in [54], [55], [56] und [51] veröffentlicht. Die Vorevaluation geeigneter Netzarchitekturen für die Primitivendetektion in Punktwolken (siehe Kapitel 4.6.3.1) findet sich in [54]; ebenso die Idee zur automatischen Generierung von Trainingsdatensätzen bestehend aus Punktwolken (siehe Kapitel 4.4.1.1). Aus [55] stammt dabei die Partitionierung der Eingabepunktwolke durch ein Clusteringverfahren basierend auf Punktkoordinaten, -normalen und prädiziertem Primitiventyp zur Verbesserung der Robustheit der Primitivendetektion und -einpassung. Aus derselben Publikation stammt die Zusammenführung von Ebenen zu Quadern mittels eines EAs. Aus [56] stammt die Idee zur Ermittlung von Zylinderdeckflächen. [51] hingegen ist die Erweiterung des Systems zur Unterstützung von arbiträren konvexen Polytopen mittels einer Partitionierungsstrategie basierend auf einer konvexen Punktwolkensegmentierung entnommen. Neu ist die Evaluation in Kapitel 4.6, die auf den Datensätzen aus [55] und [51] sowie auf einem neuen zusätzlichen Datensatz (Modell M12) durchgeführt wurde. Eine Ausnahme bildet hier die Evaluation des trainierten KNNs zur Primitivendetektion, die aus [56] stammt. Die Abbildungen 4.5 und 4.28 a)-e) finden sich in [55], die Abbildungen 4.13, 4.15a, 4.15c und 4.19 in [51] wieder. Aus [56] stammen die Abbildungen 4.19a, 4.19b, 4.2a, 4.2b, 4.6c und 4.6d.
Das in Kapitel 5 vorgestellte Verfahren zur effizienten Synthese von KBs aus gegebenen Primitiven und einer Punktwolke wurde noch nicht vorveröffentlicht.

Jedoch basiert es auf den in [53] und [52] publizierten Partitionierungsstrategien sowie in Teilen auf der KB-Synthesemethode aus [56]. Genannte Verfahren wurden für diese Arbeit jedoch stark erweitert und verbessert. Auch die in diesem Kapitel enthaltenen Ausführungen zur Problemkomplexität wurden bereits in [56] veröffentlicht. Abbildung 5.2 stammt aus [56]. Die Abbildungen 5.6, 5.5 und 5.4 sind [53] entnommen, wurden jedoch leicht modifiziert. Des weiteren wurde Abbildung 5.7 aus [52] übernommen.
Die zentralen Konzepte von Kapitel 6, welches sich mit der multikriteriellen Optimierung von KBs auseinandersetzt, wurden bereits in [48], [58] und [57] publiziert. Dabei diente v.a. [58] als Vorlage für dieses Kapitel. Aus diesem stammt das Konzept, wobei die Evaluation der vorgestellten Prozess-Pipeline für diese Arbeit noch einmal vollständig neu durchgeführt wurde. Grundlegend für die dort aufgegriffene Idee zur Nutzung von QUBO-Formulierungen zur Lösung von Problemen aus der KB-Domäne ist [48]. Das in Kapitel 6.6 diskutierte Interaktionskonzept zur manuellen Bearbeitung von KBs ist [57] entnommen. Die Abbildungen 6.2, 6.3, 6.4, 6.5 sowie Tabelle 6.3 stammen aus [58], wobei Abbildung 6.3 marginal modifiziert wurde. Aus [57] wurden die beiden Abbildungen 6.15 und 6.16 übernommen.
Zusätzlich zu dieser Auflistung findet sich in jedem Inhaltskapitel (Kapitel 4, 5 und 6) eine ausführliche Zuordnung von bereits veröffentlichten Inhalten zu Kapiteln, Abbildungen und Ergebnissen.

1.2. Aufbau dieser Arbeit

Diese Arbeit ist wie folgt aufgebaut: In Kapitel 2 werden für die Arbeit relevante Grundlagen systematisch erläutert und wichtige Prinzipien, Formalismen und Methoden eingeführt. Im darauffolgenden Kapitel (Kapitel 3) wird die in dieser Arbeit behandelte Problemstellung detailliert erläutert und in einen Gesamtkontext eingebettet, dem nicht nur die Struktur der weiteren Kapitel folgt, sondern auch einen Blick über die inhaltlich gezogenen Grenzen der Arbeit hinaus ermöglicht.
Den beiden Kapiteln angeschlossen sind drei große Inhaltskapitel, die Kernaspekte der vorgestellten Forschung im Detail behandeln. Den Anfang macht dabei Kapitel 4 mit der Vorstellung eines Systems zur Extraktion geometrischer Primitive aus Punktwolken. Diesem folgt Kapitel 5, das eine neuartige Methode zur Ermittlung eines KBs aus einer Punktwolke mit gegebenen Primitiven vorstellt und dabei auch die theoretischen Aspekte des Problems beleuchtet. Das letzte Inhaltskapitel, Kapitel 6, widmet sich dem bisher wenig beforschten Aspekt der Optimierung bestehender KBs hinsichtlich Ausdruckslänge und anderer relevanter Attribute und stellt zu diesem Zweck ein flexibles System mit austauschbaren Teilkomponenten vor.
Den Abschluss der Arbeit bildet Kapitel 7 mit einer Zusammenfassung der präsentierten Inhalte und einem Ausblick auf zukünftige Forschungsrichtungen, die auf den hier erarbeiteten Erkenntnissen aufbauen.

2. Definitionen und Grundlagen

Dieses Kapitel beschreibt Grundlagen, die essentiell für das Verständnis der darauf aufbauenden Kerninhalte dieser Arbeit sind. Den Anfang macht die Einführung von affinen Punkträumen und Transformationen (siehe Kapitel 2.1) gefolgt von einer mathematischen Beschreibung von Echtweltobjekten als sog. Festkörper (siehe Kapitel 2.2).
Mit der formalen Definition des Festkörpers werden in Kapitel 2.3 dessen wichtigsten Repräsentationsformen eingeführt. Diese nutzen unterschiedliche Strukturen und Strategien zur Speicherung der geometrischen Information des Festkörpers und haben dabei spezifische Vor- und Nachteile. Dies ist insofern relevant, da sich das Problem der Synthese von Konstruktionsbäumen aus unstrukturierten räumlichen Daten als eine Konversion von der Repräsentation als Menge von Messpunkten in eine KB-basierte Repräsentation verstehen lässt.
Darauf aufbauend werden in den folgenden Kapiteln grundlegende Aspekte aus den Bereichen der Optimierung (siehe Kapitel 2.4) und des MLs (siehe Kapitel 2.5) vorgestellt, die als Basis für die vorgestellten Systeme dienen. Abschließend findet sich eine Zusammenfassung aller erläuterten Grundlagen in Kapitel 2.6.

2.1. Affine Punkträume und Transformationen

Für die Darstellung von dreidimensionalen Körpern und ihrer Bewegung im Raum ist die Unterscheidung von Raumpunkten und -vektoren sinnvoll. Zur formalen Beschreibung dient ein affiner Punktraum $\mathbb{A} = (S, V, A)$, der sich allgemein über einen Punktraum S, einen Vektorraum V und einer Abbildung $A\colon S \times V \to S$ (Addition) definiert (siehe z.B. [186]). In dieser Arbeit wird der Euklidische Punktraum $\mathbb{E}^3$ verwendet. Als Vektorraum dient hierbei der dreidimensionale Euklidische Vektorraum (ein reeller Vektorraum mit Skalarprodukt $\circ$) mit orthonormaler Basis $B = \{\mathbf{b_1}, \mathbf{b_2}, \mathbf{b_3}\}$ (Vektoren werden zur Unterscheidung von Punkten fett hervorgehoben). Als Punktraum wird der dreidimensionale reelle Punktraum $\mathbb{R}^3$ verwendet. Die Abbildung A wird durch

$$A(s, \mathbf{v}) \mapsto (s^1 + \mathbf{v}^1, s^2 + \mathbf{v}^2, s^3 + \mathbf{v}^3) \tag{2.1}$$

festgelegt, wobei $s \in S$, $\mathbf{v} \in V$ und s^i bzw. $\mathbf{v}^i$ die i-te Komponente des Punkts $s \in S$ bzw. des Vektors $\mathbf{v} \in V$ ist. Zusammen mit einem frei wählbaren Ursprung $o \in S$ ergibt die Basis B ein affines System (o, B), welches benutzt

werden kann, um jeden Punkt $s \in S$ eindeutig über einen 3-Tupel $x \in \mathbb{R}^3$ zu beschreiben (als sog. affine Koordinaten):

$$s = o + \sum_{i=1}^{3} x^i \mathbf{b_i}. \tag{2.2}$$

Für jedes Punktpaar ($s \in S, u \in S$) existiert genau ein einzigartiger Vektor $\mathbf{v_{u-s}}$, für den $u = s + \mathbf{v_{u-s}}$ gilt. Daraus ergibt sich, dass der Punktraum $\mathbb{E}^3$ mit der Distanzfunktion $d(s,u) = |\mathbf{v_{u-s}}|$ metrisch ist, wobei $|\cdot|$ die Norm ist, die durch das Skalarprodukt des verwendeten Euklidischen Vektorraums induziert wird. Weiterhin gilt $s + \mathbf{0} = s$ für $s \in S$ und den Nullvektor $\mathbf{0} \in V$ sowie $s + (\mathbf{v} + \mathbf{w}) = (s + \mathbf{v}) + \mathbf{w}$ für alle $s \in S$ und $\mathbf{v}, \mathbf{w} \in V$.
Transformationen $F : U \to W$ zwischen zwei affinen Räumen U und W (z.B. Rotation, Translation und Skalierung) werden durch die Abbildung

$$F(s) \mapsto M_{3\times3} s + \mathbf{t}, \tag{2.3}$$

dargestellt, wobei s ein Punkt aus dem Punktraum von U, $M_{3\times3}$ eine reelle 3×3 Matrix und $\mathbf{t}$ ein Translationsvektor aus dem Vektorraum von U ist. Die so beschreibbaren, sog. affinen Transformationen erhalten Distanzverhältnisse, Parallelen und Kollinearitäten zwischen Punkten und umfassen Translation, Reflektion, Skalierung, Rotation und die Scherung. Um jede affine Transformation (z.B. auch die Translation) mit nur einer Matrixmultiplikation formulieren zu können, werden dreidimensionale affine Punkte in einen vierdimensionalen Raum eingebettet und mithilfe sog. homogener Koordinaten beschrieben. Die Einbettung erfolgt durch

$$H \colon S \to \mathbb{R}^4, \quad H(s) \mapsto (w \cdot s^1, w \cdot s^2, w \cdot s^3, w), \tag{2.4}$$

wobei $w \in \mathbb{R} \setminus 0$ und $w = 1$, also $(s^1, s^2, s^3, 1)$, die sog. homogene Standardrepräsentation des Punkts s ist. Nun kann jede affine Transformation durch

$$T(s) \mapsto M_{4\times4}\, h, \tag{2.5}$$

dargestellt werden, wobei h der homogenisierte Punkt ist. Um wieder den dreidimensionalen affinen Punkt zu erhalten, wird im Anschluss die sog. De-Homogenisierung durchgeführt:

$$H^{-1} \colon \mathbb{R}^4 \to S, \quad H^{-1}(h) \mapsto (h^1/h^4, h^2/h^4, h^3/h^4). \tag{2.6}$$

Der vierdimensionale Einbettungsraum ist dabei ein projektiver Raum, der neben den affinen auch projektive Punkttransformationen zulässt, wie sie z.B. bei photogrammetrischen Methoden (siehe Kapitel 3.2.1.2) benötigt werden. Betrachtet man nur Rotation und Translation, spricht man von Euklidischen Transformationen. Dabei haben Körper im Raum, neben ihrer Ausdehnung

eine Position (Translation) sowie eine Orientierung (Rotation). Beides zusammengefasst wird als sog. Pose bezeichnet, die durch die Transformationsmatrix

$$P_{4\times4} = \left[\begin{array}{c|c} R & \mathbf{t} \\ \hline 0_{1\times3} & 1 \end{array}\right] \tag{2.7}$$

beschrieben wird, wobei $\mathbf{t}$ ein Translationsvektor und R eine 3×3 Rotationsmatrix ist.

2.2. Festkörper

Unter dreidimensionalen geometrischen Objekten, wie sie in dieser Arbeit relevant sind, werden feste Körper von homogener Materialbeschaffenheit verstanden, die in einem Produktionsprozess, z.B. einem 3D-Druck, auch tatsächlich physikalisch herstellbar sind. Daraus ergeben sich sogenannte Starr- oder Festkörper mit folgenden Eigenschaften [147]:

1. Der Körper ändert seine Form nicht unter translatorischer oder rotatorischer Bewegung (Rigidität).
2. Kein Element des Körpers hat eine Dimensionalität $\neq 3$ (Homogene Dimensionalität).
3. Die Abmessungen des Körpers sind endlich (Finitheit).
4. Das Wegnehmen oder Hinzufügen von Material resultiert wieder in einem Festkörper (Abgeschlossenheit bezüglich Boolescher Mengenoperationen). Rotationen oder Verschiebungen des Körpers resultieren wieder in einem Festkörper (Abgeschlossenheit bezüglich Euklidischer Transformationen).
5. Der Körper lässt sich mit einer endlichen Menge aus Teilstücken beschreiben (finite Beschreibbarkeit).
6. Der Rand des Körpers lässt sich eindeutig beschreiben (Randdeterminismus).

Diese Kriterien lassen sich im sogenannten kontinuierlichen Punktmengenmodell [170] wie folgt formalisieren: Festkörper werden als Teilmengen des Euklidischen Punktraums $\mathbb{E}^3$ definiert. Jeder Festkörper wird über eine Punktmenge $S \in \mathbb{E}^3$ beschrieben, die folgende Eigenschaften erfüllt:

- S ist semi-analytisch, d.h., S kann mithilfe einer endlichen Anzahl Boolescher Kombinationen von Mengen der Form $\{x \mid F_i(x) \leq 0, x \in \mathbb{E}^3\}$ dargestellt werden, wobei $F_i : \mathbb{E}^3 \to \mathbb{R}$ analytische Funktionen sind.
- S ist beschränkt, d.h., es existiert eine Kugel mit endlichem Radius $r > 0$, die S komplett enthält.

- S ist regulär und enthält damit keine Elemente heterogener Dimensionalität, jedoch ihren kompletten Rand δS. Eine Teilmenge $S \subset \mathbb{E}^3$ ist regulär, wenn sie gleich dem Inneren (i) ihres Abschlusses (k) ist, also wenn gilt $S = \text{ki}(S)$ (Formalismus aus [147]). Für eine intuitive Darstellung des Inneren und des Abschlusses einer Menge, siehe Abbildung 2.1. Das Entfernen und Hinzufügen von Elementen wird mithilfe von regularisierten Mengenoperatoren realisiert:
  - Vereinigung: $S_1 \cup^* S_2 := \text{ki}(S_1 \cup S_2)$
  - Schnitt: $S_1 \cap^* S_2 := \text{ki}(S_1 \cap S_2)$
  - Differenz: $S_1 -^* S_2 := \text{ki}(S_1 - S_2)$

  Mit der Verwendung der regularisierten Mengenoperatoren ist S abgeschlossen bezüglich Boolescher Mengenoperationen.

- S ist Teilmenge des affinen Punktraums $\mathbb{E}^3$ und damit abgeschlossen bezüglich Euklidischer Transformationen (z.B. Translation und Rotation, siehe Kapitel 2.1).

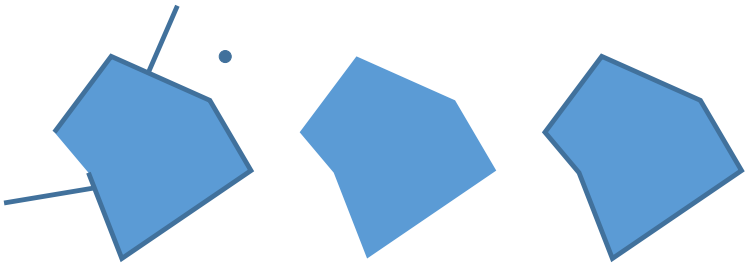


Abbildung 2.1.: Links: Nicht-reguläre Menge S mit heterogener Dimensionalität (Inneres $i(S)$ in Hellblau, Rand δS in Dunkelblau). Mitte: das Innere von S, $\text{i}(S)$. Rechts: Der Abschluss des Inneren von S, $\text{ki}(S)$. $\text{ki}(S)$ ist eine reguläre Menge mit Elementen der Dimension 2.

Teilmengen von $\mathbb{E}^3$ mit den genannten Eigenschaften werden in der Literatur gemeinhin als R-Mengen bezeichnet und bilden zusammen mit den genannten regulären Mengenoperatoren $\cup^*$, $\cap^*$ und $-^*$ einen Ring [172].
Generell besteht die Möglichkeit, auch das regularisierte Komplement auf Festkörpern zu definieren:

$$\setminus^* S := \text{ki}(\setminus S), \tag{2.8}$$

wobei der Zusammenhang $\setminus^* S = W -^* S$ besteht (W ist die Universal- oder Grundmenge, die alles enthält [172], hier: $W := \mathbb{E}^3$). Bei der Anwendung des Komplements kann nicht mehr garantiert werden, dass das Resultat ein beschränkter Festkörper ist und somit sind R-Mengen nicht abgeschlossen bezüglich des Komplements [147]. Lässt man bei der Definition von Festkörpern

die Beschränktheit als Kriterium fallen, erlaubt also potentiell unendlich ausgedehnte Festkörper, erhält man erneut Abgeschlossenheit bezüglich des regularisierten Komplements. Mit den Operationen $\cup^*$, $\cap^*$ und $\setminus^*$ ergibt sich in diesem Fall eine Boolesche Algebra (Beweis in [148]). Im ersten Inhaltskapitel (siehe Kapitel 4) wird von beschränkten Festkörpern ausgegangen, wohingegen im zweiten und dritten Inhaltskapitel (siehe Kapitel 5 und Kapitel 6) das Komplement benutzt wird und damit theoretisch Festkörper unendlichen Volumens möglich sind. Praktisch hat dies jedoch keinen Einfluss auf die dort vorgestellten Verfahren.
In manchen Fällen wird in dieser Arbeit auf eine verkürzte Schreibweise der regularisierten Mengenoperatoren zurückgegriffen: $A + B := A \cup^* B$, $A \cdot B := A \cap^* B$ und $\overline{A} := \setminus^* A$.
Die hier gewählte Definition eines Festkörpers ist vereinfacht gegenüber der ursprünglich in [146] getroffenen. Diese definiert Festkörper als Äquivalenzklassen von Teilmengen des $\mathbb{E}^3$. Dabei sind zwei Teilmengen äquivalent, wenn sie sich mithilfe einer Euklidischen Transformation (siehe Kapitel 2.1) exakt aufeinander abbilden lassen. Jedes Element einer Äquivalenzklasse, in dieser Definition auch Instanz eines Festkörpers genannt [146], wird über eine R-Menge beschrieben. Diese Definition wird hier nur der Korrektheit halber aufgeführt und hat für den weiteren Verlauf der Arbeit untergeordnete Bedeutung.

2.3. Repräsentationsschemas

Auf Basis der Definition von Festkörpern durch R-Mengen lassen sich nun verschiedene Repräsentationsschemas für deren Darstellung entwickeln. Dabei wird in einem Repräsentationsschema festgelegt, welche "Sprache" verwendet wird, um Festkörper zu beschreiben. Formal lässt sich ein Repräsentationsschema wie folgt definieren [147]: Ein sog. Modellierungsraum M enthält alle zu beschreibenden Festkörper. Dem wird die Menge der syntaktisch korrekten Repräsentationen R eines Repräsentationsschemas gegenübergestellt, die meist durch eine formale Grammatik erzeugt wird. Ein Repräsentationsschema ist nun eine Abbildung $F\colon M \to R$, die einem Festkörper aus M eine Repräsentation aus R zuordnet. Jede Repräsentation im Bild von F wird als valide bezeichnet, da sie syntaktisch korrekt ist und ihr ein Element aus M zugeordnet ist. Eine valide Repräsentation ist eindeutig, wenn ihr genau ein Festkörper aus M zugeordnet wird; sie ist einzigartig, wenn ihr zugeordneter Festkörper nur durch sie (also genau durch eine Repräsentation) dargestellt werden kann. Die Eigenschaften einer einzelnen Repräsentation lassen sich auf Repräsentationsschemas erweitern: Ein Repräsentationsschema ist eindeutig, wenn alle enthaltenen validen Repräsentationen eindeutig sind; es ist einzigartig, wenn alle enthaltenen validen Repräsentationen einzigartig sind. Bei einem eindeutigen und einzigartigem Repräsentationsschema ist die dazugehörige Abbildung F bijektiv. Abbildung 2.2 veranschaulicht den Zusammenhang von Modellierungsraum, Abbildung F und der Menge der syntaktisch validen

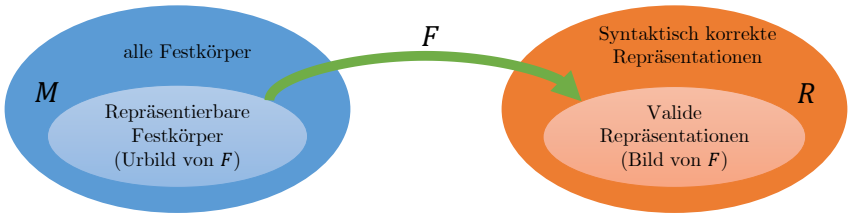


Abbildung 2.2.: Ein Repräsentationsschema wird durch eine Abbildung F definiert, welche eine Teilmenge des Modellierungsraums M auf eine Teilmenge der Menge der syntaktisch korrekten Repräsentationen R abbildet (Abbildung entlehnt aus [147]).

Repräsentationen. Darauf aufbauend lässt sich ein Repräsentationsschema anhand folgender Kriterien charakterisieren [147]:

- Ausdrucksstärke: Die Mächtigkeit des Urbilds von F im Verhältnis zur Mächtigkeit von M. Bestenfalls können alle Festkörper aus M repräsentiert werden.

- Validität: Die Anzahl valider Repräsentationen (Mächtigkeit des Bilds von F) in Relation zur Anzahl möglicher, syntaktisch korrekter Repräsentationen (Mächtigkeit von R). Bestenfalls ist jede Repräsentation aus R auch gleichzeitig valide.

- Eindeutigkeit & Einzigartigkeit: Ist das Repräsentationsschema eindeutig und einzigartig, so vereinfacht sich der Test auf Gleichheit zweier darstellbarer Festkörper auf einen (syntaktischen) Test auf Gleichheit seiner Repräsentationen.

- Prägnanz: Die Speichereffizienz des Repräsentationsschemas hinsichtlich redundanten Datenstrukturen. Je nach Anwendungsfall kann eine effiziente Datenübertragung im Vordergrund stehen oder jedoch eine redundante Speicherung für schnelleren Datenzugriff erwünscht sein.

- Komfort bei der Bearbeitung und Erstellung: Repräsentationsschemas, die für das manuelle Editieren von Festkörpern eingesetzt werden sollen, müssen hinsichtlich ihrer Gebrauchstauglichkeit optimiert worden sein.

- Applikationsspezifische Eignung: Die konkrete Applikation bestimmt die Auswahl zu verwendender Algorithmen, welche den Einsatz bestimmter Repräsentationsschemas voraussetzen.

Im Folgenden soll nun auf einige, für diese Arbeit relevante Repräsentationsschemas hinsichtlich der diskutierten Kriterien eingegangen werden. Dabei wird eine Einteilung vorgenommen, wie sie in [170] und [168] vorgeschlagen wurde.

2.3.1. Implizite Repräsentationsschemas

Implizite Repräsentationsschemas beschreiben die Punktmenge S_m eines Festkörpers $m \in M$ implizit über eine Gleichung der Form [170]:

$$S_m = \{x \mid a_m(x) = 1, \forall x \in \mathbb{E}^3\}, \tag{2.9}$$

wobei $a_m \colon \mathbb{E}^3 \to \{1, 0\}$ ein Prädikat darstellt, das für einen beliebigen Punkt $x \in \mathbb{E}^3$ angibt, ob er Teil der Punktmenge des Festkörpers m ist (1) oder nicht (0). Da die darzustellende Punktmenge S_m semi-analytisch ist und damit aus beliebig komplexen Booleschen Mengenkombinationen besteht, kann das Prädikat a_m als eine logische Verknüpfung von einfacheren Prädikaten a_{m1}, a_{m2}

$$a_m(x) = a_{m1}(x) \odot a_{m2}(x) \tag{2.10}$$

formuliert werden, wobei $\odot \in \{\wedge, \vee, \neg\}$. Analog lässt sich die mengentheoretische Variante über $x \in (S_{m1} \oplus S_{m2}) \implies (x \in S_{m1} \odot (x \in S_{m2})$ als

$$S_m = S_{m1} \oplus S_{m2} \tag{2.11}$$

schreiben, wobei $\oplus \in \{\cap, \cup, -\}$ die mit der logischen Operation $\odot$ korrespondierende Mengenoperation ist. Da Festkörper nicht nur semi-analytische, sondern auch reguläre Teilmengen des $\mathbb{E}^3$ sind, müssen die Booleschen Mengenoperatoren $\{\cap, \cup, -\}$ durch ihre regularisierten Varianten $\{\cap^*, \cup^*, -^*\}$ ersetzt werden. Die Zerlegung der Prädikate lässt sich rekursiv fortführen, was in einer Baumstruktur resultiert, in der innere Knoten logische Verknüpfungen und Blätter arbiträre Prädikate darstellen. Analoges gilt für die Mengenschreibweise. Meist beschreiben Blattprädikate einfache geometrische Grundformen wie Kugeln, Quader oder Zylinder, die als beschränkte und damit endliche Halbräume des $\mathbb{E}^3$ definiert werden. Diese werden als geometrische Primitive oder vereinfacht als Primitive bezeichnet. Die Darstellung eines Festkörpers mittels einer Baumstruktur nennt sich *Konstruktionsbaum* (KB), das dazugehörige Repräsentationsschema *Constructive Solid Geometry* (CSG). Anstelle des in der Literatur häufig anzutreffenden Begriffs des CSG-Baums wird hier aus Gründen der Konsistenz der Begriff KB oder KB-Ausdruck verwendet.

2.3.1.1. Constructive Solid Geometry (CSG)

Ein KB Φ kann als formale Grammatik beschrieben werden, die alle syntaktisch korrekten Repräsentationen erzeugt. Abbildung 2.3 zeigt diese Grammatik in *Backus-Naur-Form* (BNF)(ausschließlich binäre Operatoren). In der Praxis wird für die Beschreibung der Prädikate in den Blattknoten oft auf sog. *vorzeichenbehaftete Distanzfunktionen* (VDFs) zurückgegriffen [133]:

$$F(x) = \begin{cases} -d_{\min}(x, \delta S_m) \text{ falls } x \in S_m \\ d_{\min}(x, \delta S_m) \text{ sonst,} \end{cases} \tag{2.12}$$

⟨*KB*⟩ ::= ⟨*Praedikat-Blatt*⟩
| ⟨*KB*⟩ ⟨*Operator-Knoten*⟩ ⟨*KB*⟩
| ⟨*KB*⟩ ⟨*Pose-Knoten*⟩ ⟨*KB*⟩

Abbildung 2.3.: Formale Beschreibung der KB-Syntax mittels BNF. Pose-Knoten: Lage und Orientierung des darunterliegenden Teilbaums im Raum mittels einer 4×4-Matrix (siehe Gleichung 2.7), Prädikat-Blatt: Für gewöhnlich geometrische Primitive, Operator-Knoten: Regularisierte Mengenoperatoren (siehe Kapitel 2.3). Abbildung angepasst aus [147].

wobei $d_{\min}(x, \delta S_m) = \min(d(x, y))\ \forall y \in \delta S_m$, also die kürzeste Distanz von Punkt $x \in \mathbb{E}^3$ zur Oberfläche δS_m beschreibt. Der Gradient von F erfüllt die Eikonal-Gleichung $|\nabla F| = 1$. Daraus ergibt sich der zu x nächstliegendste Punkt auf der Oberfläche δS_m, x_c, aus $x_c = x - F(x) \cdot \nabla F(x)$, falls x_c einzigartig ist [133]. Für geometrische Primitive existieren exakte oder approximative VDFs, wie zum Beispiel für die Kugel:

$$F_{Kugel}(x) = d(x, c) - r, \tag{2.13}$$

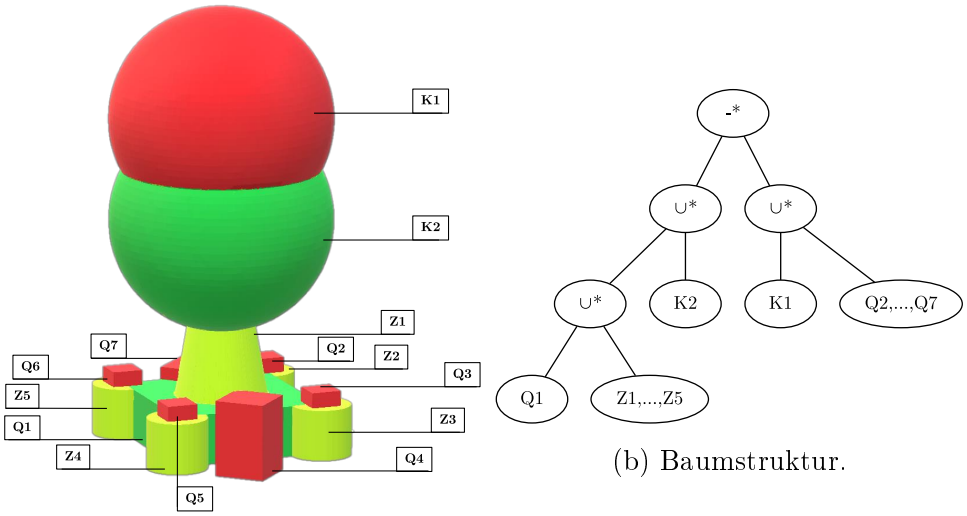
wobei r der Radius und c das Zentrum der Kugel ist. Ein Festkörper m wird durch einen KB Φ und einer Menge von Primitiven H exakt beschrieben, wenn $S_m = |\Phi(H)|$ gilt, d.h. die durch den KB Φ angewandt auf H induzierte Punktmenge der Punktmenge S_m entspricht. Weiterhin können reguläre Boolesche Mengenoperatoren auf unterschiedliche Arten approximiert werden. Eine Möglichkeit besteht in der Nutzung von Min- und Max-Funktionen [150]:

$$|\Phi| \cap^* |\Psi| := \max(F_\Phi, F_\Psi) \tag{2.14}$$
$$|\Phi| \cup^* |\Psi| := \min(F_\Phi, F_\Psi) \tag{2.15}$$
$$\setminus^* |\Phi| := -F_\Phi \tag{2.16}$$
$$|\Phi| -^* |\Psi| := \max(F_\Phi, -F_\Psi), \tag{2.17}$$

wobei F_Φ und F_Ψ die VDFs der KBs Φ und Ψ sind und $|\cdot|$ die Punktmengen darstellen, die durch die jeweiligen KBs beschrieben werden. Darüber hinaus lassen sich zu diesem Zweck auch R-Funktionen [171] verwenden:

$$|\Phi| \cap^* |\Psi| := \frac{1}{2}(F_\Phi + F_\Psi + \sqrt{(F_\Phi - F_\Psi)^2 + \alpha}) \tag{2.18}$$
$$|\Phi| \cup^* |\Psi| := \frac{1}{2}(F_\Phi + F_\Psi - \sqrt{(F_\Phi - F_\Psi)^2 + \alpha}) \tag{2.19}$$
$$\setminus^* |\Phi| := -F_\Phi \tag{2.20}$$
$$|\Phi| -^* |\Psi| := \frac{1}{2}(F_\Phi - F_\Psi - \sqrt{(F_\Phi - F_\Psi)^2 + \alpha}), \tag{2.21}$$

wobei $\alpha \in \mathbb{R}$. Für $\alpha = 0$ sind die R-Funktionen äquivalent zu den Min- und Max-Funktionen. Setzt man hingegen $\alpha \neq 0$, kann man differenzierbare Approximationen der Min- und Max-Funktionen erzeugen, was sie für die Verwendung in Zielfunktionen von Optimierungsproblemen prädestiniert, die mit gradientenbasierten Verfahren optimiert werden sollen (siehe Kapitel 2.4.3.1). Generell ist zu beachten, dass die so approximierten Mengenoperatoren im Allgemeinen nicht regulär sind.
Das CSG-Repräsentationsschema (siehe Abbildung 2.4) ist eindeutig, jedoch nicht einzigartig, da ein einzelner Festkörper durch eine unendliche Anzahl von KBs dargestellt werden kann. Seine Ausdrucksstärke ist determiniert durch die Wahl der Primitive und Mengenoperatoren. Durch die Eigenschaften der R-Mengen und die Verwendung von regularisierten Mengenoperatoren sind syntaktisch korrekte KBs immer valide. Dies gilt nicht für KBs, die das regularisierte Komplement nutzen, da dieses zu unbeschränkten Festkörpern führen kann. Die Speichereffizienz ist abhängig von der Repräsentation der Primitive. Sie ist hoch, wenn ausschließlich geometrische Grundformen zum Einsatz kommen, was jedoch die Ausdrucksstärke einschränkt. Die manuelle Modellierung ist intuitiv, da auch Laien eine Vorstellung von der Funktionsweise Boolescher Mengenoperatoren haben. KBs lassen sich auf einfache Weise rekursiv durchlaufen und bieten einen hohen Grad an Abstraktion. Für Algorithmen, die auf einen direkten Punktmengenzugriff (z.B. zur Visualisierung) oder auf die effiziente Ermittlung von Punktnachbarschaften angewiesen sind, ist das CSG-Repräsentationsschema eher ungeeignet. [147, 170]



(a) Primitive.

(b) Baumstruktur.

Abbildung 2.4.: Beispiel-KB. Rote Primitive in a) sind im resultierenden Festkörper nicht sichtbar. In b) wurde auf die Darstellung von Pose-Knoten aus Platzgründen verzichtet.

2.3.1.2. Andere implizite Repräsentationsschemas

Ein ebenfalls breit erforschtes implizites Repräsentationsschema ist die sog. Sweep-Repräsentation [87]. Dabei wird eine Teilpunktmenge X des $\mathbb{E}^3$ (z.B. ein Festkörper) über die zeitabhängige, kontinuierliche Bewegungsfunktion $M\colon \mathbb{R} \times \mathbb{E}^3 \to \mathbb{E}^3$ transformiert. Der neue Festkörper ergibt sich dann aus

$$\text{sweep}_M(X) = \bigcup_{t \in [t_0, t_1]} M(t, X), \tag{2.22}$$

also aus der Vereinigung aller Teilmengen erzeugt durch M für ein Zeitintervall $[t_0, t_1]$. Ist M eine Euklidische Transformation, wird im Folgenden von Euklidischen Sweeps gesprochen. Abbildung 2.5 zeigt ein einfaches Beispiel eines Euklidischen Sweeps.

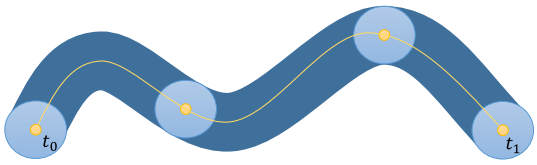


Abbildung 2.5.: Euklidischer Sweep: Die zeitabhängige Transformationsfunktion M beschreibt eine Translation entlang eines Kurvenverlaufs (orange). Die Eingabeteilmenge X ist hier ein Kreis (hellblau). Der erzeugte Körper wird in Dunkelblau dargestellt.

2.3.2. Aufzählende Repräsentationsschemas

Repräsentationsschemas dieser Kategorie definieren Regeln, um Punkte zu erzeugen bzw. aufzuzählen, die zur Punktmenge S_m eines Festkörpers $m \in M$ gehören.

2.3.2.1. Parametrisierungen

Parametrische Repräsentationsschemas definieren Abbildungen der Form $F\colon [0,1]^3 \to \mathbb{E}^3$, um einen Festkörper zu beschreiben. Um Punkte aufzuzählen, wird der Parameterraum entlang aller drei Achsen in äquidistanten Schritten durchlaufen und F für jede Parameterkombination ausgewertet.

2.3.2.2. Zellgruppierungen

Bei Repräsentationsschemas dieser Kategorie wird ein Festkörper in Zellen gleichen geometrischen Typs und gleicher Dimensionalität aufgeteilt. Als Zellgeometrie können z.B. Quader gleicher Größe dienen, die in einem Gitternetz angeordnet sind (sog. Voxel-Repräsentation, siehe Abbildung 2.6).
Ein weiteres Beispiel ist die spatiale Partitionierung mithilfe eines Octrees. Dabei wird ein den Festkörper vollständig umschließender Quader rekursiv in acht

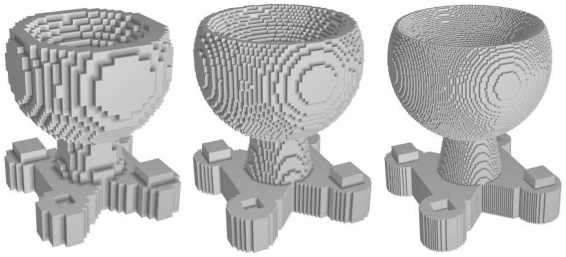


Abbildung 2.6.: Voxel-Repräsentation eines Modells in verschiedenen Auflösungen (von links nach rechts: $26 \times 32 \times 26$, $52 \times 64 \times 52$ und $103 \times 128 \times 103$ Zellen). Gut zu sehen sind die quaderförmigen Modellteile, die schon mit geringer Auflösung ohne Approximationsfehler repräsentiert werden können, wohingegen runde Formen immer nur angenähert werden können.

Zellen unterteilt, bis eine Zelle entweder vollständig innerhalb oder vollständig außerhalb des Festkörpers liegt. Somit wird der Festkörper durch eine Baumstruktur mit Quadern unterschiedlicher Größe als Blattknoten repräsentiert (siehe Abbildung 2.7). Repräsentationsschemas basierend auf Zellgruppierun-

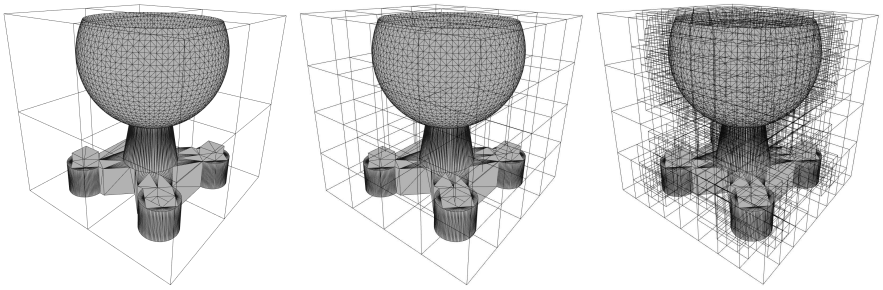


Abbildung 2.7.: Octree-Repräsentation eines Modells in verschiedenen Rekursionstiefen (von links nach rechts: 1, 2 und 4). Zur Anschauung ist zusätzlich noch jeweils das Modell als Dreiecksnetz dargestellt.

gen sind einzigartig, jedoch nicht eindeutig, da durch die Granularität der Zellaufteilung mehrere Festkörper auf dieselbe Repräsentation abgebildet werden können. Zellgruppierungen haben hohe Ausdrucksstärke, da generell allen möglichen Festkörpern aus M eine Repräsentation zugeordnet werden kann. Jede Repräsentation ist zudem auch valide, wobei beachtet werden muss, dass nicht jeder Festkörper exakt repräsentiert werden kann. Die Speichereffizienz ist hingegen vergleichsweise gering. So ist der Speicherverbrauch der Voxel-Repräsentation von kubischer Komplexität hinsichtlich der Höhe, Breite und Tiefe des initialen Quaders. Die gewählte Zellauflösung ist dabei abhängig von der gewünschten Detailtreue der Repräsentation. Das manuelle Modellieren ist eher mühsam, da für einzelne Zellen festgelegt werden muss, ob sie Teil des Festkörpers sind oder nicht. Für bestimmte Applikationen besteht besonde-

re Eignung. So lassen sich Nachbarschaften und Punktzuordnungen (ist der Punkt innerhalb oder außerhalb des Festkörpers?) effizient ermitteln.

2.3.2.3. Punktwolke

Eine endliche Menge von Punkten im Raum wird als Punktwolke bezeichnet. Punktwolken sind meist das Resultat von Messungen, die z.B. mit einem Laserscanner durchgeführt wurden. Andere Quellen umfassen zweidimensionale Bilder oder Videos, aus denen mit photogrammetrischen Verfahren dreidimensionale Information rekonstruiert wurde. Je nach Quelle können Punktwolken prinzipbedingte Fehler enthalten, wie z.B Messrauschen oder eine fehlende Oberflächenabdeckung. Aus diesem Grund wird vor der Weiterverarbeitung meist ein Vorverarbeitungsschritt (engl. pre-processing) durchgeführt, indem versucht wird, etwaige Fehler algorithmisch zu minimieren oder bestenfalls zu beseitigen. Punktwolken repräsentieren meist nur die Oberfläche eines physisch vorhandenen Festkörpers. Je nach Quelle kann für jeden Punkt neben seinen Koordinaten auch der Normalenvektor der an diesem Punkt gemessenen Oberfläche vorliegen. Ist das nicht der Fall, kann diese entsprechend geschätzt werden. Die Repräsentation eines Festkörpers durch eine Punktwolke ist weder eindeutig noch einzigartig, jedoch kann jeder Festkörper mithilfe einer syntaktisch korrekten Punktwolke approximativ repräsentiert werden. Dabei können Punktwolken existieren, die keinen Festkörper repräsentieren. Der Speicherverbrauch ist direkt von der Größe der gemessenen Oberfläche sowie der Punktdichte abhängig. Da Punktwolken selten manuell erstellt werden, ist der mangelnde Erstellungskomfort nicht relevant. Schwer wiegt das Fehlen topologischer Struktur (z.B. Nachbarschaften oder Randzugehörigkeit), welche, wenn nötig, geschätzt werden muss. Somit lassen sich Nachbarschaften und Punktzuordnungen nicht eindeutig oder nicht effizient ermitteln. Abbildung 2.8 zeigt Beispielpunktwolken mit verschiedenen Punktdichten.

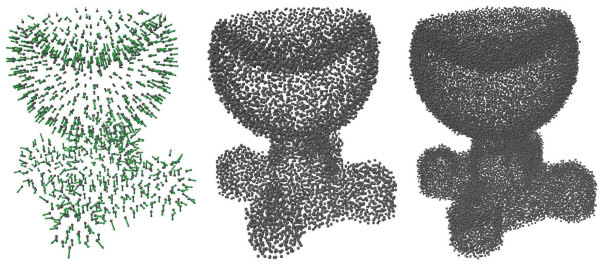


Abbildung 2.8.: Punktwolken eines Modells in verschiedenen Punktdichten (links nach rechts: 1000, 8000 und 32000 Punkte). In der ersten Punktwolke von links sind die Oberflächennormalen zusätzlich als grüne Pfeile dargestellt.

2.3.2.4. Zellkomplexe

Man kann zeigen, dass jeder Festkörper von einer endlichen Menge von nichtüberlappenden Zellen unterschiedlicher Dimensionalität und geometrischer Form exakt repräsentiert werden kann (kombinatorisches Modell) [170]. Dabei ist das kombinatorische Modell äquivalent zum bereits beschriebenen kontinuierlichen Punktmengenmodell. Im kombinatorischen Modell ist ein Festkörper ein dreidimensionales topologisches Polytop, was intuitiv-geometrisch betrachtet ein Polytop mit arbiträr gekrümmten Flächen beschreibt. Repräsentationsschemas, die diesem Modell folgen, sind sogenannte Zellkomplexe [170] oder Zelldekompositionen [147], wobei die Zellgeometrie gemeinhin als geometrischer Träger bezeichnet wird. Im Vergleich zu einfachen Zellgruppen sind die möglichen Nachbarschaftsbeziehungen einzelner Zellen in einem Zellkomplex nicht konstant, sondern je nach Zelle unterschiedlich. Zellkomplexe sind eindeutig aber nicht einzigartig (unendliche viele Dekompositionen eines Festkörpers existieren). Nicht jede syntaktisch korrekte Dekomposition ist auch valide, wobei Validität nicht effizient überprüfbar ist [147]. Die manuelle Modellierung mit Zellkomplexen ist bei gekrümmten Festkörpern nicht intuitiv. Ihr Hauptvorteil gegenüber Zellgruppierungen liegt in der expliziten Darstellung topologischer Eigenschaften, wie z.B. der Rand oder die Verbundenheit des Festkörpers [170].
Ein approximatives Repräsentationsschema aus der Gruppe der Zellkomplexe stellt die Dekomposition eines Festkörpers in dreidimensionale konvexe Polytope dar (im Folgenden einfach konvexe Polytope genannt). Dabei sind konvexe Polytope Spezialfälle dreidimensionaler konvexer Polyeder (im Folgenden einfach konvexe Polyeder genannt) mit der Zusatzeigenschaft, dass sie beschränkt sind, d.h. endliche Teilmengen des Punktraums $\mathbb{E}^3$ umschließen. Für ein genaueres Verständnis konvexer Polytope, wird zuerst der Begriff des konvexen Polyeders eingeführt. Ein konvexes Polyeder P lässt sich über eine Menge von m Ungleichungen

$$P(A,b) := \{x \in \mathbb{E}^3 \mid Ax \leq b\} \tag{2.23}$$

beschreiben, wobei $A \in \mathbb{R}^{m\times 3}$ und $b \in \mathbb{R}^m$. Anschaulich sind diese Ungleichungen Ebenen im dreidimensionalen Raum. Diese Darstellung eines konvexen Polyeders P ist die sog. H-Darstellung. Dem *Minkowski-Weyl*-Theorem folgend [196], existiert für jeden konvexen Polyeder eine Darstellung, die äquivalent zu seiner H-Darstellung ist:

$$P(V,R) := \mathrm{kon}(V) \oplus \mathrm{keg}(R), \tag{2.24}$$

wobei $\mathrm{kon}(V)$ die konvexe Hülle

$$\mathrm{kon}(V) := \{x \in \mathbb{E}^3 \mid x = \sum_{i=1}^{s} a^i v_i, \sum_{i=1}^{s} a^i = 1, a \in \mathbb{R}^s, a \geq \mathbf{0}\} \tag{2.25}$$

beschrieben durch s Extremalpunkte $V = \{v_1, ..., v_s\}$ ist und keg(R) den konvexen Kegel (auch: nichtnegative Hülle)

$$\text{keg}(R) := \{x \in \mathbb{E}^3 \mid x = \sum_{i=1}^{t} \lambda^i \mathbf{r_i}, \lambda \in \mathbb{R}^t, \lambda \geq \mathbf{0}\} \tag{2.26}$$

beschrieben durch t Vektoren $R = \{\mathbf{r_1}, ..., \mathbf{r_t}\}$ darstellt. Weiterhin ist $\oplus$ als die Minkowski-Summe zweier Punktmengen X und Y

$$X \oplus Y := \{x + y \mid x \in X \wedge y \in Y\} \tag{2.27}$$

definiert. Diese alternative Darstellung wird V-Darstellung genannt. Beide Darstellungsformen lassen sich ineinander umwandeln, was effizient und robust mit der sog. Doppelbeschreibungsmethode [59] funktioniert.
Wie bereits erwähnt sind konvexe Polytope Spezialfälle konvexer Polyeder und umfassen dabei genau diejenigen konvexen Polyeder, deren eingeschlossenes Volumen endlich ist und damit durch ihre konvexe Hülle beschrieben werden kann, also $P(V, R) = \text{kon}(V)$ und somit keg$(R) = \emptyset$ gilt.
Möchte man einen konvexen Polyeder als VDF beschreiben, bietet sich

$$F_{\text{k-Polyeder}}(x) = \min(\{\mathbf{e_n} \circ (e_s - x) \mid (e_s, \mathbf{e_n}) \in H\}) \tag{2.28}$$

als Approximation an, wobei das Tupel $(e_s, \mathbf{e_n}) \in H$ eine Ebene mit Ursprung e_s und Normale $\mathbf{e_n}$ darstellt. Auf diese Art lassen sich auch konvexe Polytope darstellen, jedoch kann nicht garantiert werden, dass jede Ebenenkombination ein valides konvexes Polytop darstellt.
Die Dekomposition eines Festkörpers in konvexe Polytope ist ein genaues Repräsentationsschema, falls nur Festkörper in Form geometrischer Polytope dargestellt werden sollen. Anwendung findet sich dafür z.B. in der effizienten Simulation von Festkörperbewegungen, wie sie z.B. in [36] beschrieben wird.

2.3.3. Begrenzungsflächenmodelle

Jeder Festkörper m hat einen Rand δS_m (seine Oberfläche) der ihn eindeutig beschreibt [170]. Diese Eigenschaft erlaubt eine Repräsentation des Festkörpers durch seinen Rand, welcher wiederum von einer Menge von homogenzweidimensionalen Zellen (sog. Begrenzungsflächen) beschrieben werden kann. Für die Beschreibung eines Festkörpers durch ein Begrenzungsflächenmodell werden topologische Informationen (Wie sind geometrische Zellen miteinander verbunden?) und geometrische Information (Wie ist eine Zelle geformt?) voneinander getrennt betrachtet. Die folgende Beschreibung von topologischen und geometrischen Entitäten ist aus [63] entnommen, findet sich aber leicht abgewandelt auch in [147].
Topologisch wird die Festkörperoberfläche aus einer Menge von Seitenflächen zusammengesetzt, die die Hülle des Festkörpers lückenlos beschreiben. Lücken-

los bedeutet in diesem Kontext, dass die zusammengesetzten Seitenflächen an den Rändern vollständig zusammenpassen. Man spricht dabei auch von einer wasserdichten Beschreibung der Festkörperhülle. Die Hülle enthält somit Information über die Nachbarschaften der Seitenflächen. Seitenflächen sind aus einer Menge von Schleifen zusammengesetzt, wobei jede Schleife eine geordnete Menge von Kanten enthält. Damit legt eine Schleife nicht nur die Kanten einer Seitenfläche fest, sondern durch die gegebene Reihenfolge auch ihre Orientierung, also welche Seite der Fläche nach außen und welche in das Innere des Festkörpers zeigt. Jede Kante besteht weiterhin aus zwei Eckpunkten (Vertices).
Aus geometrischer Perspektive werden Seitenflächen durch beliebig gekrümmte Oberflächen beschrieben, Kanten durch Kurven und Vertices durch dreidimensionale Punkte. Geometrische Entitäten sind die sog. geometrischen Träger eines Festkörpers beschrieben durch das Begrenzungsflächenmodell. Für die Beschreibung von Oberflächen bieten sich zwei verschiedene Konzepte an, analog zu den Festkörperrepräsentationen (siehe Kapitel 2.3.2):

- Aufzählend: Zur Oberfläche gehörende Punkte werden explizit aufgezählt.
- Implizit: Implizite Beschreibung der Oberfläche.

Eine Übersicht der geometrischen und topologischen Entitäten findet sich in Abbildung 2.9.

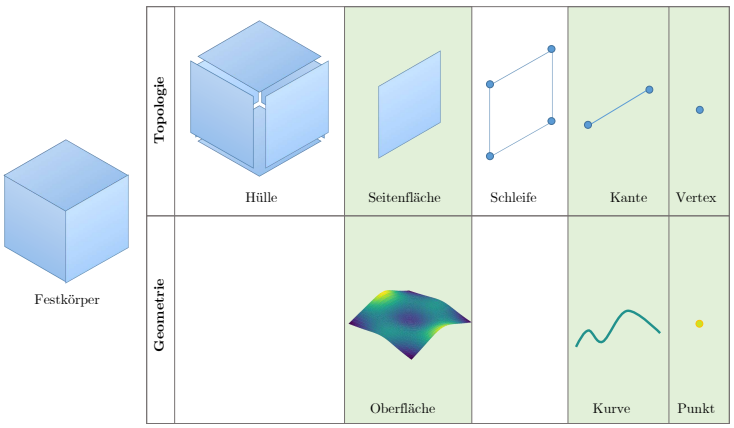


Abbildung 2.9.: Topologische und assoziierte geometrische Entitäten im Begrenzungsflächenmodell. Abbildung angepasst aus [63].

Das Sicherstellen der Validität von Begrenzungsflächenmodellen ist nicht immer trivial zu ermitteln und muss für Topologie und Geometrie getrennt betrachtet werden [147]. Die Ausdrucksstärke eines Begrenzungsflächenmodells hängt von der gewählten Beschreibung der Oberfläche ab. Gleiches gilt für seine Prägnanz, seine Eindeutigkeit sowie für den Komfort bei Bearbeitung und Erstellung [147].

Begrenzungsflächenmodelle sind im Allgemeinen nicht einzigartig, jedoch für viele Anwendungen der Modellierung geeignet (z.B. bevorzugt bei der Entwicklung von Computerspielen) [147]. Im Folgenden werden für die Arbeit relevante geometrische Repräsentationsschemas zur Beschreibung von Oberflächen eingeführt.

2.3.3.1. Polygonnetze

Eine stückweise lineare Approximation der Festkörperoberfläche mithilfe eines geometrischen dreidimensionalen Polyeders wird als Polygonnetz (engl. polygon mesh) oder im speziellen Fall von Dreiecksflächen als Dreiecksnetz bezeichnet (engl. triangle mesh). Dieses aufzählende Repräsentationsschema ist aufgrund der Verfügbarkeit effizienter Visualisierungsalgorithmen im Bereich der Echtzeitgrafikanwendungen (z.B. Computerspiele) weit verbreitet. Für die Darstellung und Verarbeitung von Dreiecksnetzen wird oft auf dedizierte Hardware zurückgegriffen (sog. *Graphics Processing Units* (GPUs)), bei der entsprechende Algorithmen direkt als Schaltkreise implementiert und damit besonders effizient sind. Abbildung 2.10b zeigt ein Beispiel.

2.3.3.2. Algebraische Flächen

Weiterhin lassen sich einzelne Zellen der Oberfläche über die Nullstellenmenge eines Polynoms P definieren:

$$\{x \in \mathbb{E}^3 \mid P(x) = 0\} \tag{2.29}$$

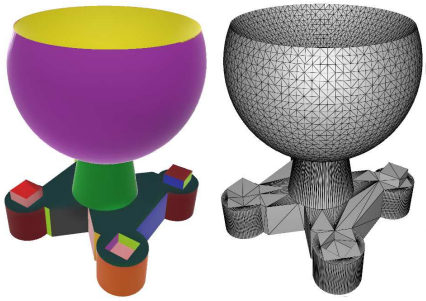
Ist das Polynom zweiten Grades, also

$$\{x \in \mathbb{E}^3 \mid x^T Ax + 2b^T x + c = 0\} \tag{2.30}$$

spricht man von sog. Quadriken oder auch von Oberflächen zweiter Ordnung [199], wobei A eine reelle 3×3 Koeffizientenmatrix, $b \in \mathbb{R}^3$ ein Koeffizientenvektor und $c \in \mathbb{R}$ eine additive skalare Konstante darstellt. Mit Quadriken lassen sich Primitive, wie z.B. Flächen, Kugeln, Zylinder, Kegel und Tori, implizit darstellen. Wichtig ist, dass viele auf diese Weise beschreibbaren Primitive nicht beschränkt sind, also unendliches Volumen füllen. Das trifft z.B. auf Flächen sowie Zylinder und Kegel zu. Abbildung 2.10a zeigt ein Beispiel, in dem Oberflächenstücke mit Quadriken beschrieben wurden.

2.3.3.3. Nicht-uniforme Rationale Basis-Splines (NURBS)

NURBS-Flächen sind parametrische Oberflächenbeschreibungen (jedoch mit einer Abbildung $F\colon [0,1]^2 \to \mathbb{E}^3$, statt wie in Kapitel 2.3.2.1 $F\colon [0,1]^3 \to \mathbb{E}^3$), die v.a. bei der Modellierung von arbiträr geformten Flächen im Entwurf von z.B Flugzeugen, Autos oder Schiffen Anwendung finden [138]. Flächen werden dabei durch mehrere stückweise zusammengesetzte rationale Funktionen



(a) (b)

Abbildung 2.10.: Begrenzungsflächen-Repräsentation eines Modells. a) Implizit mit Quadriken. Einzelne geometrische Träger sind farblich markiert. b) Approximativ aufzählend mit einem Dreiecksnetz. Runde Oberflächen werden stückweise linearer angenähert.

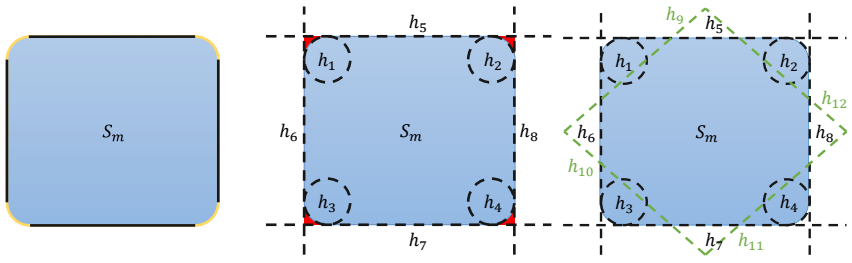
dargestellt. Dabei erfolgt die Flächenbeschreibung über ein Viereckснetz von gewichteten Punkten, dem sog. Kontrollpunktnetz, wobei der lokale Einfluss eines jeden Kontrollpunkts auf die Flächenform durch Skalare im sog. Knotenvektor bestimmt wird. Neben Freiformflächen umfassen NURBS auch den Raum der durch Quadriken (siehe Kapitel 2.3.3.2) darstellbaren Oberflächen. Ein Nachteil der Modellierung von Oberflächen mithilfe von NURBS ist die Beschränkung auf ein Kontrollpunktnetz regulärer Topologie (jeder Knoten hat genau vier Nachbarn). Möchte man komplexere Festkörper darstellen und benötigt in verschiedenen Bereichen der Oberfläche unterschiedliche Detailgrade, werden diese daher oft mittels mehrerer Kontrollnetze beschrieben, dessen zugehörige Oberflächen dann zusammengesetzt werden. An den Rändern der zusammengesetzten Oberflächen können dabei in bestimmten Fällen Diskontinuitäten auftreten, welche eine nicht-wasserdichte Hülle und damit eine fehlerhafte Festkörperbeschreibung zur Folge haben. Abhilfe schaffen andere Beschreibungen, die auch Kontrollnetze nicht-regulärer Topologie und damit Bereiche unterschiedlicher Kontrollpunktauflösung zulassen, wie z.B. T-Splines [166] oder Unterteilungsflächen (z.B. *Catmull-Clark*-Unterteilungsflächen [28]).

2.3.4. Repräsentationskonversionen

Alle hier aufgeführten Repräsentationsschemas haben gemeinsam, dass sie R-Mengen und damit Teilmengen des $\mathbb{E}^3$ repräsentieren und sich nur hinsichtlich der Strukturierung geometrischer und topologischer Information unterscheiden. Dies legt nahe, dass sich verschiedene Repräsentationsschemas ineinander umwandeln lassen [170].
Eine Übersicht der möglichen Konversionen zwischen Repräsentationsschemas bietet Abbildung 2.12, welche sich auf [147] und [170] stützt. Grundsätzlich lassen sich Begrenzungsflächenmodelle, Zellkomplexe, Parametrisierungen und

implizite Repräsentationen in Punktwolken oder Zellgruppierungen umwandeln. Dies geschieht über eine im Allgemeinen verlustbehaftete Abtastung der Festkörperoberfläche oder des Festkörpervolumens in diskreten Abständen (sog. *Sampling*, siehe Kapitel 3.2.1.1). Diese Konversion ist nicht eindeutig, damit nicht umkehrbar und resultiert in einer *Approximation* des Festkörpers. Möchte man Festkörper beschrieben durch parametrische Repräsentationsschemas aus Punktwolken oder Zellgruppierungen gewinnen, muss eine Abbildung $F\colon [0,1]^3 \to \mathbb{E}^3$ gefunden werden, für die Parameter 3-Tupel existieren, die entweder auf exakt die Punkte der Punktwolke bzw. Punkte der Zellen einer Zellgruppierung abgebildet werden (*Interpolation*) oder diese nur annähern (*Approximation*). Die Suche nach einer Abbildung mit diesen Eigenschaften wird auch *Einpassung* genannt. Ist die Ausgangsrepräsentation ein Zellkomplex, müssen einzelne Zellen entsprechend eingepasst werden.
Für die Konversion von Punktwolken und Zellgruppierungen in Begrenzungsflächenmodelle ergibt sich ebenfalls ein Einpassungsproblem. Sollen z.B. Oberflächen mit Quadriken beschrieben werden, müssen zuerst deren Parameter ermittelt werden (z.B. Radius und Zentrum einer Kugel), sodass die Punkte der Punktwolke oder die Zellen der Zellgruppierung einen möglichst geringen Abstand zu den ermittelten Oberflächen haben. Wenn immer weniger Parameter zu ermitteln sind, als es Punkte oder Zellen als Eingabe gibt, handelt es sich um ein Ausgleichsproblem (siehe Kapitel 2.4.2) und bei dessen Lösung damit um eine *Approximation*. Dies ist z.B. bei Quadriken fast immer der Fall. Hingegen lassen sich NURBS-Oberflächen und Dreiecksnetze bilden, die jeden Eingabepunkt bzw. jede Eingabezelle beinhalten und damit eine *Interpolation* der Eingabe darstellen. Mögliche Einpassungsverfahren für die Ermittlung von Quadriken aus Punktwolken sind für diese Arbeit relevant und werden in Kapitel 3.2.4 beschrieben. Auch für die Umwandlung von Punktwolken und Zellgruppierungen in Dreiecksnetze existieren Algorithmen [98, 113].
Die Umwandlung eines KBs in ein Begrenzungsflächenmodell und vice versa (*Begrenzungsflächenauswertung* bzw. *Inverse Begrenzungsflächenauswertung*) ist exakt möglich [147]. Eine Übersicht von Algorithmen zur *Begrenzungsflächenauswertung* findet sich in [139]. Für diese Arbeit relevanter ist das inverse Problem. Zu dessen Lösung müssen im ersten Schritt aus jeder Seitenfläche des Begrenzungsflächenmodells des Festkörpers ein geeignetes Primitiv für den zu erstellenden KB extrahiert werden. Im Anschluss muss dann die Struktur des Baumes ermittelt werden, wie es in [172] für Begrenzungsflächenmodelle bestehend aus Quadriken erläutert wird. Grundsätzlich ergibt sich das Problem, dass die von den Flächenstücken extrahierten Primitive nicht immer ausreichen, um den Festkörper vollständig zu beschreiben. Abhilfe schaffen dabei zusätzliche Primitive, die zur Menge der Primitive des KBs hinzugefügt werden und separierende Primitive genannt werden [173]. Abbildung 2.11 zeigt ein entsprechendes Beispiel. Für Quadriken als Oberflächenbeschreibungen kann gezeigt werden, dass Ebenen als separierende Primitive ausreichen, um jeden als Begrenzungsflächenmodell darstellbaren Festkörper in eine CSG-Repräsentation

exakt zu konvertieren [173].



(a) (b) (c)

Abbildung 2.11.: a) Festkörper (hellblau) mit Begrenzungsflächen (orange, schwarz). b) Mit den Begrenzungsflächen als Primitive $\{h_1, ..., h_8\}$ (schwarz) ist der Festkörper nicht darstellbar (die abgerundeten Kanten fehlen (rot)). c) Durch die Einführung zusätzlicher separierender Primitive $\{h_9, .., h_{12}\}$ (grün) lässt sich der Festkörper als KB repräsentieren. So ist z.B. die linke obere runde Kante über eine Kombination der Primitive $\{h_1, h_5, h_6, h_9\}$ repräsentierbar.

In diesem Zusammenhang lässt sich auch eine einfache und verlustfreie Möglichkeit zur Umwandlung von Zellkomplexen zu KBs beschreiben. Dabei wird jede Zelle des Komplexes als ein Primitiv eines KB-Ausdrucks angesehen und per regularisierter Vereinigungsoperation $\cup^*$ mit allen anderen Primitiven verknüpft (*Zellvereinigung*).
Der Vollständigkeit halber soll noch die Umwandlung von Zellkomplexen in Begrenzungsflächenmodelle erwähnt werden (*Seitenflächenselektion*). Diese ist exakt möglich und lässt sich durch die Selektion aller Zellseitenflächen, die nur einer Zelle zugeordnet sind, bewerkstelligen [1]. Ebenfalls einfach möglich ist die exakte Umwandlung von Euklidischen Sweeps in Begrenzungsflächenmodelle (*Umsetzung*), weil Kanten im zu bewegenden Festkörper durch die Transformationsoperation direkt zu Flächen im resultierenden Begrenzungsflächenmodell werden (und Eckpunkte zu Kanten) [147]. Die inverse Konversion hingegen ist nicht immer möglich, da nicht jedes Begrenzungsflächenmodell durch einen Euklidischen Sweep darstellbar ist.

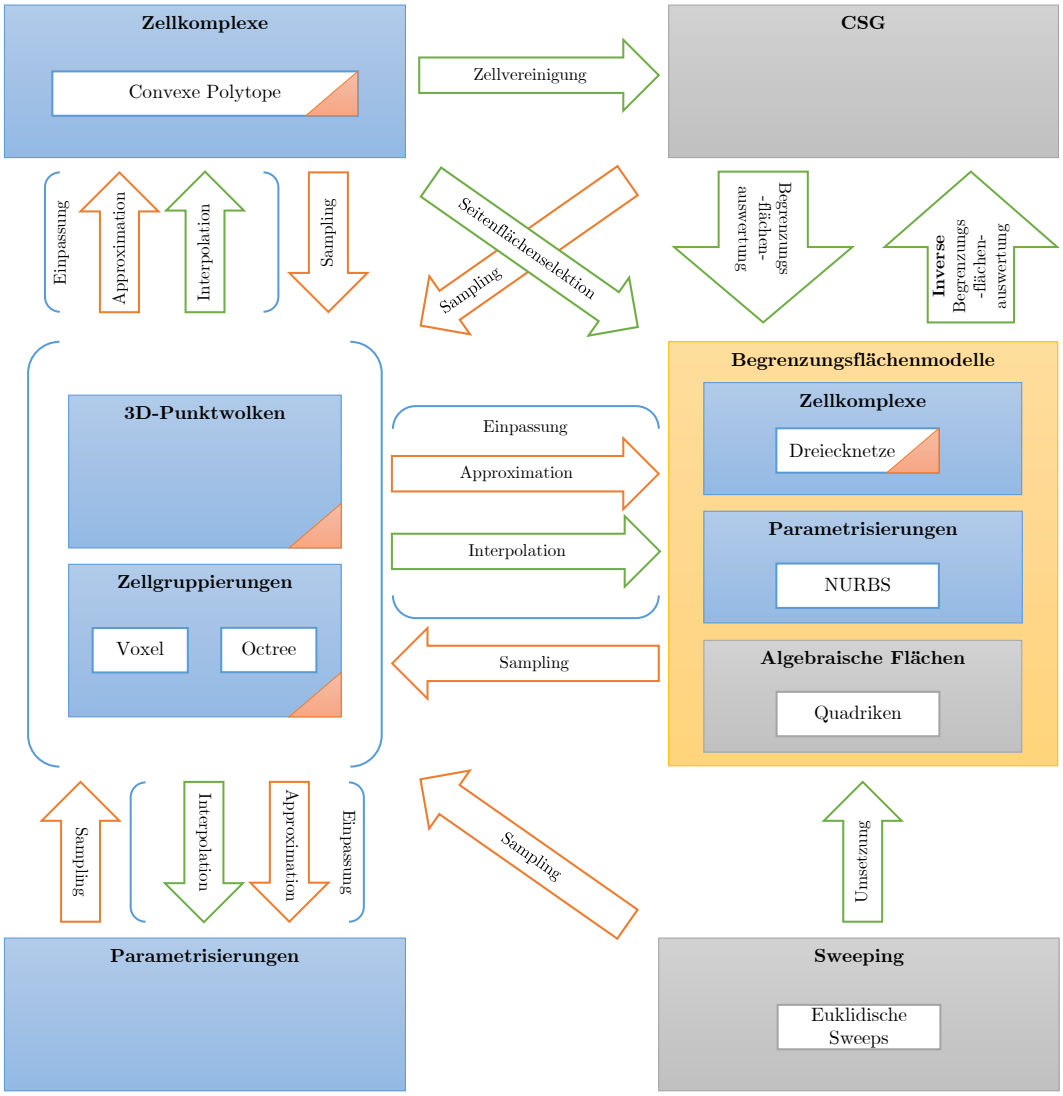


Abbildung 2.12.: Diskutierte Repräsentationsschemas und mögliche Konversionen (aufzählende Repräsentationsschemas in Blau, implizite in Grau und Begrenzungsflächenmodelle in Orange). Pfeile in Rot stellen approximative Konversionen dar, grüne Pfeile solche, die eindeutig sind und somit auch invertiert werden können. Repräsentationsschemas mit einer roten Ecke rechts unten stellen Festkörper im Allgemeinen nur approximativ dar.

2.4. Optimierungsprobleme

Viele in dieser Arbeit behandelten Probleme lassen sich formal als Optimierungsprobleme beschreiben und mit geeigneten Methoden lösen. Dieses Kapitel beinhaltet eine formale Definition des Begriffs der Optimierung sowie eine Ka-

tegorisierung entsprechender Probleme. Darüber hinaus werden für die Arbeit relevante Lösungsverfahren für bestimmte Optimierungsprobleme beschrieben.

2.4.1. Definition und Kategorisierung

Unter der Optimierung eines parametrisierten Systems (Θ, F) versteht man die Suche nach einer Belegung θ der Systemparameter aus dem Suchraum Θ, welche hinsichtlich einer Bewertungs- oder Zielfunktion

$$F\colon \Theta \to \mathbb{R}, \tag{2.31}$$

optimal ist, d.h. F minimal ist (Minimierungsproblem):

$$\underset{\theta\in\Theta}{\operatorname{argmin}} F(\theta). \tag{2.32}$$

Dabei lässt sich aufgrund des im Dualitätsprinzip der Optimierung [71] postulierten Zusammenhangs $\max(F) = -\min(-F)$ ein Minimierungs- in ein Maximierungsproblem umwandeln und vice versa.
Ist Θ endlich groß oder enthält nur eine abzählbar unendliche Menge an Elementen, spricht man von einem diskreten Optimierungsproblem. Ist dies nicht der Fall (z.B. $\Theta = \mathbb{R}^n$) bezeichnet man es als kontinuierlich. Im Fall einer endlich großen Menge Θ spricht man auch von kombinatorischen Optimierungsproblemen, welche in dieser Arbeit von besonderer Bedeutung sind.
Ein Optimierungsproblem ist linear, wenn F linear ist. Ist F nichtlinear, so wird das zugehörige Optimierungsproblem ebenfalls als nichtlinear bezeichnet. Beschreibt die Zielfunktion mehrere Unterziele, die gemeinsam optimiert werden sollen, spricht man von einem multikriteriellen Optimierungsproblem. Kombinatorische Probleme diesen Typs sind in dieser Arbeit ebenfalls im Besonderen relevant.

2.4.1.1. Randbedingungen

Je nach abzubildendem Problem kann die zugrundeliegende Lösungsmenge durch sog. Randbedingungen eingeschränkt werden. Formal sind k Randbedingungen $\Phi_i\colon \Theta \to \{0,1\},\ i = 1,...,k$ Abbildungen, die die Zugehörigkeit von Elementen aus Θ zum eingeschränkten Lösungsraum Φ bestimmen, wobei $\Phi := \{\theta \in \Theta \mid \Phi_i(\theta) = 1\ \forall\ i = 1,...,k\}$ ist. Nun wird eine Lösung nicht mehr in Θ gesucht, sondern in Φ.

2.4.2. Ausgleichsrechnung

Wenn immer eine Menge von m kontinuierlichen Messpunkten $X = \{x_1,...,x_m\}$ und dazugehörige Messwerte $Y = \{y_1,...,y_m\}$ in ein durch $\theta \in \Theta$ parametrisiertes Modell $M_\theta\colon X \to Y$ mit Parametern aus dem Parameterraum Θ eingepasst werden soll, werden dazu Methoden der Ausgleichsrech-

nung bemüht. Dazu wird eine Parameterbelegung $\theta \in \Theta$ gesucht, welche die Abweichung des Modells zu den gemessenen Daten minimiert:

$$\underset{\theta \in \Theta}{\operatorname{argmin}} \sum_{i=1}^{m} d(y_i, M_\theta(x_i)), \tag{2.33}$$

wobei $d(\cdot, \cdot)$ eine Distanzmetrik für Elemente aus Y ist (z.B. die Euklidische Distanz). Es handelt sich bei einem Ausgleichsproblem also um ein kontinuierliches Optimierungsproblem. Dieses wird durch ein überbestimmtes Gleichungssystem, welches damit mehr Gleichungen als Unbekannte, also mehr Datenpunkte als Parameter enthält, beschrieben. Ist das Modell F linear, spricht man von einem linearen Ausgleichsproblem; ist die verwandte Norm zusätzlich noch die Euklidische Norm, lässt sich das Minimierungsproblem mit der Methode der kleinsten Quadrate lösen [79]. Für nichtlineare, jedoch einfach stetig differenzierbare Modelle bietet sich beispielsweise bei Verwendung der Euklidischen Distanz das Levenberg-Marquardt-Verfahren [118] als Optimierungsmethode an.

2.4.3. Lösungsverfahren für ausgewählte Probleme

Im Folgenden werden Lösungsverfahren für drei ausgewählte Problemkategorien beschrieben:

- Kontinuierliche Optimierungsprobleme mit einfach stetig differenzierbarer Zielfunktion: Probleme diesen Typs finden sich z.B. im Trainingsprozess *Künstlicher Neuronaler Netze* (KNNs), ein Verfahren aus dem Bereich des maschinellen Lernens (siehe Kapitel 2.5). Eine zur Lösung oft genutzte Methode wird in Kapitel 2.4.3.1 erläutert.

- Kombinatorische Optimierungsprobleme: Zur Lösung von Problemen dieser Kategorie sind von der Natur inspirierte Metaheuristiken geeignet, wie sie am Beispiel Evolutionärer Algorithmen in Kapitel 2.4.3.2 beschrieben werden.

- Quadratische, Binäre Optimierung ohne Nebenbedingungen (QUBO): Hier ist die Zielfunktion ein quadratisches Polynom definiert auf den binären Zahlen. Derart formulierte Probleme lassen sich mithilfe von sog. Adiabatischen Quantencomputern lösen. Details dazu finden sich in Kapitel 2.4.3.3.

Die oben aufgeführten Problemkategorien wurden für eine genauere Betrachtung ausgewählt, da sie für die in dieser Arbeit beschriebenen Methoden von besonderer Relevanz sind.

2.4.3.1. Methode des steilsten Abstiegs

Für eine einfach stetig differenzierbare Zielfunktion F gibt der Gradient $\nabla F(\theta)$ die Richtung des steilsten Anstiegs von F an der Stelle θ an. Der Gradient kann somit genutzt werden, um F zu minimieren: Von einem Startpunkt θ_0 aus, wird in einem iterativen Prozess der jeweils nächste Punkt θ_{i+1} bestimmt, indem sich in entgegengesetzter Gradientenrichtung an der Stelle θ_i bewegt wird:

$$\theta_{i+1} = \theta_i - w \cdot \nabla F(\theta_i), \tag{2.34}$$

wobei w die sog. Schrittweite darstellt und je nach Problemstellung festgelegt werden muss. Der Vorgang wird wiederholt, bis sich keine Verbesserung (kein niedrigerer Wert für F) mehr einstellt.
Varianten dieses Prinzips, die v.a. hinsichtlich ihres Einsatzes in hochdimensionalen Lösungsräumen optimiert wurden, kommen z.B. im Trainingsprozess von KNNs bevorzugt zum Einsatz [65]. Verfahren, die auch zusätzlich noch das Krümmungsverhalten von F in den Optimierungsprozess miteinbeziehen (z.B. das bereits erwähnte Levenberg-Marquardt Verfahren, falls F die Form eines nichtlinearen Ausgleichsproblems mit Euklidischer Norm hat), sind oft hinsichtlich Konvergenz und Konvergenzgeschwindigkeit überlegen. Jedoch erlaubt die erhöhte Berechnungskomplexität keinen Einsatz bei Problemen mit hochdimensionalen Lösungsräumen [134]. Generell sind gradientenbasierte Verfahren (zu denen die Methode des steilsten Abstieges zählt) auf einen stark strukturierten Suchraum angewiesen, d.h. lokale Optima treten in einem beschränkten Bereich auf, und garantieren keine Konvergenz zu einem globalen Optimum [71].

2.4.3.2. Evolutionäre Algorithmen

Zur Lösung von kombinatorischen Optimierungsproblemen wird in dieser Arbeit auf *Evolutionäre Algorithmen* (EAs) zurückgegriffen. EAs sind populationsbasierte Metaheuristiken, die an den biologischen Prozess der Evolution angelehnt sind. Dabei wird zur Optimierung ein Standardprozess durchlaufen, wie er in Abbildung 2.13 schematisch dargestellt ist:

Optimierungsprozess. Zu Beginn (*Start*) wird eine Population von zufällig zusammengestellten Lösungskandidaten aus Θ, auch Individuen genannt, erzeugt (*Zufällige Population* 0). Damit startet eine iterative Prozessschleife und damit der Optimierungsvorgang. Jedem Kandidaten der aktuellen Population (*Population i*) wird eine Bewertung mithilfe der Zielfunktion F (in diesem Kontext auch Fitnessfunktion genannt) zugeordnet (*Bewertung*). Nun werden Kandidaten für die kommenden Variationsschritte unter Beachtung ihrer Bewertung ausgewählt (*Elternauswahl*). Auf Basis der so ausgewählten Kandidaten werden dann neue Kandidaten erzeugt (*Rekombination*), welche dann zusätzlich noch zufällig verändert werden (*Mutation*). Aus der aktuellen Population und der Menge der neu erzeugten Kandidaten wird dann die

neue Population (*Population* $i + 1$) zusammengestellt (*Gesamtauswahl*). Dabei erfolgt die Auswahl aus der aktuellen Population basierend auf der Bewertung der Kandidaten. Ist ein Abbruchkriterium erfüllt (*Abbruch?*), wird der beste Kandidat der aktuellen Population ausgewählt (*Selektion des besten Individuums*) und als Ergebnis zurückgegeben. Damit ist der Optimierungsprozess abgeschlossen (*Ende*).

Repräsentation. Angelehnt an das biologische Prinzip der Evolution, macht es je nach Problemdomäne Sinn, eine Unterscheidung zwischen dem Genotyp und dem Phänotyp eines Lösungskandidaten vorzunehmen. Dabei ist der Genotyp die Darstellung des Lösungskandidaten, wie sie für die Rekombination und Variation benutzt wird (sein "Genmaterial"), wohingegen der Phänotyp der Repräsentation entspricht, welche für die Bewertung genutzt wird (seine "Ausprägung")[71]. Zur Umwandlung von Genotyp zu Phänotyp muss dann eine bijektive Abbildung definiert werden, welche die Menge der Genotypen auf die Menge der Phänotypen abbildet.

Selektion. Die Selektion in Form der *Elternauswahl* und der *Gesamtauswahl* stellt sicher, dass sich die Lösungskandidaten in der gerade aktuellen Population über den zeitlichen Verlauf hinsichtlich ihrer Bewertung verbessern. Für die Auswahl von Individuen aus der aktuellen Population während der *Gesamtauswahl* wird in dieser Arbeit auf die sog. Plus-Selektion, welche eine deterministische Selektion ist [71], zurückgegriffen. Hier werden die n besten Individuen aus der aktuellen Population der neuen Population hinzufügt. Für die *Elternauswahl* kommt die sog. Turnier-Selektion [71] zum Einsatz. Dabei werden aus der aktuellen Population zufällig k Individuen selektiert und davon dasjenige mit der besten Fitness als Elternteil ausgewählt. Dies wird solange wiederholt, bis die gewünschte Anzahl an Elternteilen erreicht ist.

Variation. Die Variation gliedert sich in zwei Schritte, Rekombination und Mutation, und stellt sicher, dass neu zusammengestellte Lösungskandidaten Teil der neuen Population werden. Die Rekombination erzeugt aus Teilen von zwei oder mehreren Lösungskandidaten in Genotyp-Repräsentation einen neuen Lösungskandidaten. Die exakten Prozessschritte sind dabei abhängig von der Problemdomäne und der verwendeten Genotyp-Repräsentation. Bei der Mutation wird der Genotyp eines Lösungskandidaten zufällig verändert. Auch die Mutation ist im Detail problem- und genotypspezifisch. Sie muss dabei aber möglichst folgende Kriterien erfüllen [71]:

- Um die Konvergenz zu einem globalen Optimum zu garantieren, muss durch eine Mutation jedes Element des Suchraums erreicht werden.
- Keine Mutationsrichtung darf bevorzugt werden, d.h. wahrscheinlicher sein.
- Die Wahrscheinlichkeit einer Mutation und die, der exakt inversen Mu-

tation müssen gleich sein.

- Der Mutationseinfluss muss variierbar sein (z.B. durch eine Abnahme mit zunehmender Prozessdauer), um eine Anpassung an spezifische Suchraumsubstrukturen zu ermöglichen.

Beide Variationsoperatoren, Rekombination und Mutation, sind angelehnt an biologische Reproduktionsprozesse.

Abbruchkriterien. Als Kriterium für den Abbruch des Optimierungsprozesses bietet sich die Erreichung einer maximalen Anzahl von Iterationen an. Eine andere Möglichkeit bietet die Bewertung der Lösungskandidaten. Das Abbruchkriterium ist demnach erfüllt, wenn sich die Bewertung des besten Individuums der aktuellen Population nicht über eine bestimmte Anzahl von Iterationen hinweg verbessert.

Eigenschaften. EAs eignen sich im Besonderen für Kombinatorische Optimierungsprobleme, auch wenn die Lösungsqualität aufgrund ihrer Stochastizität innerhalb mehrerer Ausführungen für dieselbe Probleminstanz variieren kann und je nach Abbruchkriterium eine Konvergenz zum globalen Optimum nicht garantiert werden kann.

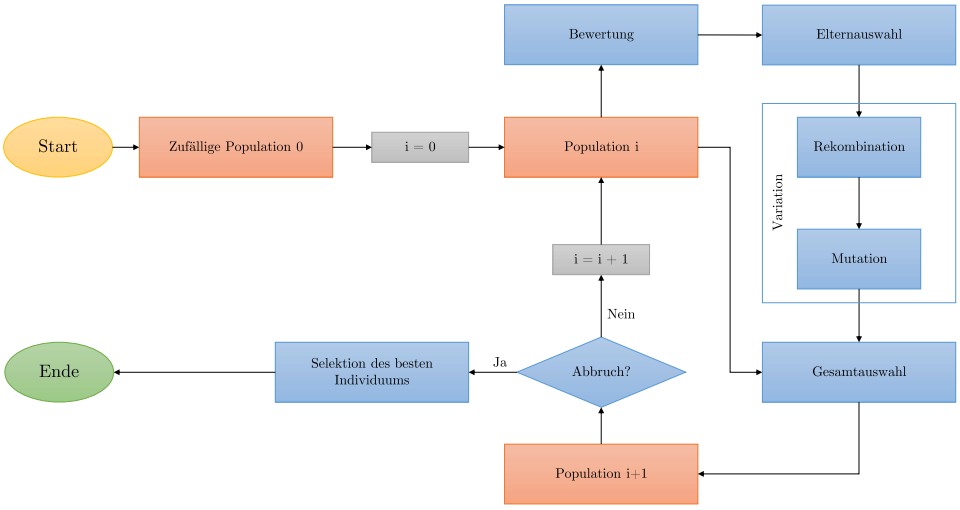


Abbildung 2.13.: Standardprozessdiagramm eines EAs. Populationen zu verschiedenen Zeitpunkten in Rot, Prozessschritte in Blau.

2.4.3.3. Adiabatische Quantencomputer und Quanten-Annealing

Der *Adiabatische Quantencomputer* (AQC) beschreibt in der hier betrachteten nicht-stochastischen Variante ein universelles Modell eines Quantencomputers, also einer Rechenmaschine, die quantenmechanische Effekte für Berechnungen nutzt [2, 46]. Dabei wird die Suche nach der optimalen Lösung eines

Optimierungsproblems per Adiabatischer Evolution, wie sie das Adiabatische Theorem der Quantenmechanik ermöglicht, durchgeführt [4, 46].

Quantenmechanische Systeme. Ein quantenmechanisches System (im Folgenden nur System) wird vollständig durch die Wellenfunktion Ψ als die Änderung des Systemzustands über die Zeit beschrieben. Parameter des Systems (sog. Observablen), wie z.B. Ort, Geschwindigkeit oder Spin eines Teilchens, werden durch Operatoren bestimmt. Möchte man eine Observable messen, wird der entsprechende Operator auf Ψ angewandt und anschließend der Systemzustand gemessen. Dabei ist die Messung im Allgemeinen destruktiv, d.h. der Zustand des Systems wird bei der Messung gestört.

Systemzustände und Operatoren. Der Systemzustand zum Zeitpunkt $t \in [0, T]$ (T ist der betrachtete Zeitraum), wird dabei formal durch einen Vektor $|\Psi(t)\rangle$ in sog. Bra-Ket-Notation in einem Hilbertraum $\mathcal{H}$ (Vektorraum auf den reellen oder komplexen Zahlen mit Skalarprodukt $\circ$ bzgl. dessen induzierter Norm er abgeschlossen ist) beschrieben. Ein Operator ist dabei eine lineare, selbstadjungierte Abbildung A ($A|x\rangle \circ |y\rangle = |x\rangle \circ A|y\rangle$, wobei $|x\rangle, |y\rangle \in \mathcal{H}$) mit Eigenwerten $a_1, a_2, ...$ Nach der Messung einer Observablen entspricht das Messergebnis einem der Eigenwerte des entsprechenden Operators (a_m). Die Messung versetzt das System in den Zustand, der dem zum Eigenwert a_m gehörenden Eigenvektor $|\psi_m\rangle$ entspricht (sog. Eigenzustand). Ist das System vor der Messung im bekannten Eigenzustand eines Operators $|\psi_m\rangle$, kann das Messergebnis vorhergesagt werden und entspricht dann a_m. Das System kann grundsätzlich mehr als einen Eigenzustand derselben Observablen gleichzeitig annehmen (sog. Superposition), wobei Eigenzustände dann als Linearkombinationen der Basisvektoren des Hilbertraums dargestellt werden. Sind dabei alle angenommenen Eigenzustände demselben Eigenwert a_m zugeordnet, ist eine Vorhersage des Messergebnisses möglich und entspricht dann wieder a_m. Ist dies nicht der Fall, kann jedem möglichen Messausgang nur eine Wahrscheinlichkeit zugeordnet werden. Zustände zweier Systeme können zudem auch ortsungebunden verschränkt sein, sodass diese nur noch in einem gemeinsamen Zustand existieren. Eine Messung des einen Systems löst die Verschränkung auf und legt zudem den Zustand des anderen Systems instantan fest.
Ein für das AQC wichtiger Operator ist der zeitabhängige Hamilton-Operator $H(t)$, der die Gesamtenergie des Zustands eines quantenmechanischen Systems $|\Psi(t)\rangle$ zum Zeitpunkt t bestimmt. Dessen Änderung über die Zeit ergibt sich aus der Schrödingergleichung

$$i\hbar \frac{\partial}{\partial t} |\Psi(t)\rangle = H(t)|\Psi(t)\rangle, \tag{2.35}$$

wobei $\hbar$ die Plancksche Konstante und i die imaginäre Einheit ist.

Qubits. Ein Qubit ist ein quantenmechanisches Zweizustandsystem, dessen Hilbertraum durch zwei komplexe Basisvektoren $|0\rangle$ und $|1\rangle$ aufgespannt wird. Damit wird der Systemzustand als Linearkombination der Basisvektoren $|\Psi(t)\rangle = \alpha|0\rangle + \beta|1\rangle$ beschrieben, wobei $\alpha, \beta \in \mathbb{C}$. Ein Zusammenschluss von n Qubits wird über die Bildung des Tensorprodukts der einzelnen Hilberträume erreicht und resultiert in einem Hilbertraum der Größe 2^n. Qubits sind die quantenmechanischen Äquivalente zu den klassischen Bits in digitalen Rechenmaschinen.

Adiabatische Evolution. Beim AQC-Modell werden nun der Schrödingergleichung folgend (siehe Gleichung 2.35) die Zustände der zusammengeschlossenen Qubits über die Zeit kontinuierlich in andere Zustände überführt. Dazu wird das Problem als zeitabhängiger Hamilton-Operator $H(t)$ formuliert:

$$H(t) = (1 - S(t))H_I + S(t)H_P, \tag{2.36}$$

wobei $S\colon [0, T] \to [0, 1]$ eine stetig differenzierbare Funktion mit Randbedingungen $S(0) = 0$ und $S(T) = 1$ ist.
Zu Beginn des Prozesses befindet sich das System im bekannten Eigenzustand des initialen Hamilton-Operators H_I ($S(0) = 0$) mit der niedrigsten Energie (Grundzustand). Der Hamilton-Operator H_P, der die Zielfunktion des Problems kodiert und dessen unbekannter Grundzustand mit der besten Lösung assoziiert ist, hat hingegen keinen Einfluss. Über die Zeit nimmt nun $S(t)$ von 0 ausgehend hin zu 1 und damit der Einfluss von H_P zu. Am Ende des Prozesses ist der Grundzustand von $H(t)$ identisch mit dem Grundzustand von H_P und damit die bestmögliche Lösung des Optimierungsproblems gefunden, wenn die im Adiabatischen Theorem der Quantenmechanik formulierten Bedingungen erfüllt sind. Nach diesem ist die Wahrscheinlichkeit, dass das System trotz der zeitlichen Änderung von $H(t)$ im Grundzustand verbleibt, nahe Eins, wenn die Geschwindigkeit des Abkühlungsprozesses langsam genug ist und zudem keine Interaktion mit der Umgebung stattfindet [4]. Die richtige Wahl der Geschwindigkeit ist dabei problemabhängig.

Ising-Modell und QUBO-Formulierung. Das Ising-Modell ist ein mathematisches Modell zur Beschreibung ferromagnetischer Effekte in Festkörpern, welches auf einer zweidimensionalen Anordnung von Atomen in einem $N \times N$ großen Gitternetz basiert [90]. Der Spin eines Atoms hat dabei nur zwei mögliche Ausrichtungen, parallel und antiparallel bzgl. der Gitternetzebene. Weiterhin wird der Spin des Atoms mit Index i, s_i, von zwei Faktoren beeinflusst: Zum einen von der Stärke des externen Magnetfelds M und zum anderen von den Spins der Nachbaratome mit Index j, s_j, dessen Einfluss durch die Kopplungskonstante J_{ij} festgelegt wird. Damit ergibt sich der Hamilton-Operator zur Beschreibung des zu lösenden Problems, H_P, im

Ising-Modell aus

$$H_P = -\frac{1}{2}\sum_{i,j} J_{ij} s_i s_j + M \sum_{i}^{N} s_i. \tag{2.37}$$

Ein positiver Wert von J_{ij} drückt dabei aus, dass benachbarte Atome nach dem gleichen Spin streben, wohingegen ein negativer Wert einen entgegengesetzten Spin bevorzugt. Das Problem, den Grundzustand eines Systems im Ising-Modell zu finden, ist $\mathcal{NP}$-*schwer* und somit lassen sich alle $\mathcal{NP}$-*schweren* Probleme in diesem Modell beschreiben und lösen [99]. Das zu lösende Optimierungsproblem nimmt die Form

$$\underset{\theta\in\{-1,1\}^N}{\operatorname{argmin}} \sum_{i=0}\sum_{j<i} \theta_i\theta_j Q_{ij} + \sum_{i=0} \theta_i Q_{ii} \tag{2.38}$$

an, wobei Q eine reelle $N \times N$-Matrix und $\theta = \{\theta_1, ..., \theta_N\}$ den gesuchten Lösungsvektor darstellt. Je nach Problemstellung kann eine Lösung dieses Optimierungsproblems effizienter gelingen als bei klassischen Verfahren [115], was v.a. bei Problemen mit vielen lokalen Minima der Fall ist. Der Grund hierfür liegt auch in der Nutzung des quantenmechanischen Tunneleffekts, der eine Überwindung von Energiebarrieren in der Lösungslandschaft ermöglicht und damit das Verweilen in einem lokalen Minimum verhindert.
Eine äquivalente Formulierung, die jedoch oft einfacher zu handhaben ist, ist die als *Quadratic Unconstrained Binary Optimization*-Problem (QUBO-Problem):

$$\underset{\theta\in\{0,1\}^N}{\operatorname{argmin}}\, \theta^T Q \theta, \tag{2.39}$$

wobei nun die gesuchte Lösung θ ein Binärvektor ist. In dieser Arbeit kommt eine QUBO-Formulierung des Mengenüberdeckungsproblems zum Einsatz (siehe Kapitel 6).

Quanten-Annealing. In realen Experimenten können die Anforderungen des Adiabatischen Theorems nicht erfüllt werden. Aus diesem Grund existiert mit dem sog. *Quanten-Annealing* (QA) die Umsetzung des theoretischen Konzepts des AQCs und der darin enthaltenen zeitlichen Evolution des Quantenzustands als Metaheuristik zur Lösung von diskreten Optimierungsproblemen [92]. Entsprechende Hardware nutzt ebenfalls quantenmechanische Effekte (Superposition, Verschränkung, Tunneleffekt), muss jedoch bzgl. Ergebnisoptimalität, Rauschartefakten und Universalität des Berechnungsmodells Kompromisse eingehen [191]. Evolutionsiterationen müssen dabei mehrmals wiederholt und Ergebnisse mittels statistischem Sampling aufbereitet werden. Trotz allem ist nicht garantiert, dass das finale Ergebnis das globale Optimum enthält. QA-Hardware existiert z.B. von der Firma *D-Wave Systems* Inc.[1].

[1] *D-Wave Systems Inc.*, `https://www.dwavesys.com/`

Ein QUBO-Problem wird durch die sog. QUBO-Matrix Q beschrieben (siehe Gleichung 2.39). Da die QUBO-Matrix eine zweidimensionale quadratische Matrix ist, kann sie auch als Adjazenzmatrix eines Graphens G_Q (im folgenden Problemgraph genannt) betrachtet werden. In diesem Graphen bilden die logischen Qubits der Problemformulierung die Knoten. Sämtliche Diagonalwerte der QUBO-Matrix sind dabei als Knotengewichte im Problemgraphen aufzufassen. Wechselwirkungen zwischen logischen Qubits (Werte ungleich Null, die sich nicht auf der Hauptdiagonalen der QUBO-Matrix befinden) werden im Problemgraphen durch gewichtete und ungerichtete Kanten dargestellt. Zur Lösung eines QUBO-Problems mittels QA-Hardware muss die von ihr vorgegebene Topologie betrachtet werden. Bei dem System der Firma *D-Wave Systems Inc.* nennt sich diese Topologie *Chimera* [27], bei späteren Generationen *Pegasus* [19]. Aufgrund von technischen Gegebenheiten kann es vorkommen, dass ein Qubit nicht mit allen anderen Qubits interagieren kann. Daher muss vor dem Annealing-Prozess der durch Q induzierte Problemgraph G_Q zunächst auf die Qubit-Topologie der jeweiligen QA-Hardware abgebildet werden. Dieser Prozess wird durch heuristische Verfahren realisiert [202]. Je nach vorgegebener Topologie, Füllgrad von Q und verwendeter Heuristik ergibt sich durch die Einbettung ein erhöhter Bedarf an physikalischen Qubits im Vergleich zur Anzahl der logischen Qubits in der ursprünglichen Problemformulierung. Qualitätsmerkmale der genutzten Einbettungsheuristik sind dabei die möglichst geringe Anzahl benötigter physikalischer Qubits sowie die Größe der Teilgraphen des Einbettungsgraphen, die ein logisches auf mehrere physikalische Qubits abbilden. Ersteres ermöglicht die Lösung größerer Probleminstanzen auf QA-Hardware mit einer fixen Anzahl physikalischer Qubits, Letzteres hat Auswirkungen auf die Ergebnisqualität [68].

2.5. Maschinelles Lernen

Maschinelles Lernen (ML) ist eine Verfahrensfamilie innerhalb der *Künstlichen Intelligenz* (KI). Darunter werden Algorithmen zusammengefasst, die auf Basis von Daten lernen können, eine bestimmte Aufgabe zu erfüllen. Im Vergleich zum imperativen oder funktionalen Programmierparadigma wird dabei auf die Formulierung expliziter Regeln verzichtet. Formal lässt sich das wie folgt definieren:

> *Ein Computerprogramm lernt von der Erfahrung E hinsichtlich einer Aufgabe T und einem Leistungskriterium P, wenn sich seine Leistung bei der Durchführung von T, wie sie von P gemessen wurde, mit der Erfahrung E verbessert [120].*

Im Folgenden werden typische Aufgaben, die mit Methoden des MLs gelöst werden können sowie eine Übersicht existierender Verfahren im Detail erläutert.

2.5.1. Verfahren

Verfahren des MLs lassen sich grob in zwei Kategorien aufteilen: Überwachtes und unüberwachtes Lernen. Ersteres wird in Kapitel 2.5.2 erläutert, letzteres in Kapitel 2.5.3.
Weitere Kategorien sind beispielsweise Hybride aus den beiden genannten Kategorien oder das sog. Bestärkende Lernen (engl. Reinforcement Learning, für eine Einführung siehe [184]), die aber für diese Arbeit nicht relevant sind und deshalb nur der Vollständigkeit halber aufgeführt sind.
Für alle Verfahren unabhängig ihrer Einordnung gilt jedoch das sog. *No-Free-Lunch*-Theorem der statistischen Inferenz [197], welches besagt, dass kein Modell grundsätzlich einem anderen überlegen ist, wenn keine Vorannahmen bezüglich des zu lernenden Datensatzes getroffen werden. Dies bedeutet dass für jeden Trainingsdatensatz im Prinzip jedes zur Verfügung stehende Modell evaluiert werden müsste. In der Praxis zeigt sich jedoch, dass sich oft aus der Komplexität einer Aufgabe das geeignetste Modell ableiten lässt [65].

2.5.2. Überwachtes Lernen

Gegeben ist mit X der sog. Merkmalsraum und mit Y der sog. Labelraum. Elemente $x \in X$ werden Merkmalsvektoren, Elemente $y \in Y$ Labelvektoren genannt. Ziel des überwachten Lernens ist es nun, eine Abbildung $h\colon X \to Y$ anhand eines m-elementigen Trainingsdatensatzes $S = \{(x_1, y_1), ..., (x_m, y_m)\}$ zu lernen, die für einen Merkmalsvektor x_i den prädizierten Labelvektor $\hat{y}_i$ zurückgibt. Dabei stammt S aus einer unbekannten gemeinsamen Wahrscheinlichkeitsverteilung $P(X, Y)$.

Varianten. [126] folgend, lassen sich grundsätzlich zwei Varianten des überwachten Lernens unterscheiden: Generative Modelle lernen die Parameter $\theta \in \Theta$ eines statistischen Modells, welches die gemeinsame Wahrscheinlichkeitsverteilung $P(X, Y)$ mit $\hat{P}_\theta(X, Y)$ approximiert. Es wird also gelernt, auf welche Weise Trainingsdaten tatsächlich erzeugt wurden, was die Generierung neuer Datensätze ermöglicht ($(x, y) \sim \hat{P}_\theta(X, Y)$). Mithilfe des Satzes von Bayes kann dann die approximierte bedingte Wahrscheinlichkeit $\hat{P}_\theta(Y|X)$ ermittelt werden, welche für die Prädiktion wie folgt verwendet wird:

$$h_\theta(x) = \underset{y \in Y}{\operatorname{argmax}}\, \hat{P}(Y|X = x). \tag{2.40}$$

Es wird also dasjenige $y \in Y$ für ein $x \in X$ gewählt, welches die höchste Wahrscheinlichkeit gegeben durch die Approximation der bedingten Wahrscheinlichkeitsverteilung $\hat{P}(Y|X = x)$ hat.
Bei der zweiten Variante, den sog. diskriminativen Modellen, werden hingegen die Parameter $\theta \in \Theta$ eines statistischen Modells gelernt, welches direkt die bedingte Wahrscheinlichkeit $P(Y|X)$ mit $\hat{P}_\theta(Y|X)$ approximiert (probabilisti-

sche, diskriminative Modelle). Sie sind diskriminativ, weil nur versucht wird, die Zuordnungsgrenzen für Elemente aus dem Labelraum möglichst genau zu ziehen, anstelle die Gesamtverteilung des Datensatzes anzunähern. Manche diskriminativen Modelle lernen h_θ direkt, d.h. ohne statistische Modellbildung, wobei dann θ nicht die Parameter einer Verteilung, sondern die eines anderssgearteten parametrisierten Systems sind (nicht-probabilistische, diskriminative Modelle). Beispiel hierfür sind *Stützvektormaschinen* (SVMs) (siehe Kapitel 2.5.2.1). Für die Quantifizierung des Prädiktionsfehlers zwischen $\hat{y}_i$ und y_i kommt die sog. Loss-Funktion $L\colon Y \times Y \to \mathbb{R}$ zum Einsatz. Dabei wird im Trainingsprozess das Optimierungsproblem

$$\underset{\theta\in\Theta}{\operatorname{argmin}} \frac{1}{m} \sum_{i=1}^{i\leq m} L(y_i, h_\theta(x_i)) \tag{2.41}$$

gelöst. Sind L und h_θ dabei stetig differenzierbar, bieten sich gradientenbasierte Optimierungsmethoden an (siehe Kapitel 2.4.3.1).

Regression und Klassifikation. Ist der Labelraum kontinuierlich, spricht man von einer Regressionsaufgabe, ist er endlich und diskret, von einer Klassifikationsaufgabe. Bei Letzteren muss einem Eingabedatum eine bestimmte Klasse zugeordnet werden. Beispielsweise könnte hier eine Aufgabe lauten, für eine Menge von Tierbildern die korrekte Gattung des abgebildeten Tieres zu bestimmen. Bei Regressionsaufgaben geht es um die Zuordnung (oder die Prädiktion) eines kontinuierlichen Wertes zu einem Eingabedatum. Ein Beispiel wäre die Temperaturvorhersage für den morgigen Tag auf Basis der gemessenen Temperaturen der letzten Tage.

Auswertungsmetriken. Für binäre Klassifikationsprobleme gibt es eine Reihe wichtiger Metriken zur quantitativen Bewertung eines Modells:

- Präzision (engl. accuracy): $\frac{\#\text{ Richtig Positive}}{\#\text{ Anzahl Tests}}$
- Spezifität (engl. specificity): $\frac{\#\text{ Richtig Negative}}{\#\text{ Richtig Negative}+\#\text{ Falsch Positive}}$
- Ausfallrate (engl. fallout): $\frac{\#\text{ Falsch Positive}}{\#\text{ Richtig Negative}+\#\text{ Falsch Positive}}$
- Relevanz (engl. precision): $\frac{\#\text{ Richtig Positive}}{\#\text{Richtig Positive}+\#\text{ Falsch Positive}}$
- Trefferquote (engl. recall): $\frac{\#\text{ Richtig Positive}}{\#\text{Richtig Positive}+\#\text{ Falsch Negative}}$
- F_1-Maß (engl. F_1 score): $2 \times \frac{\text{Relevanz}\times\text{Trefferquote}}{\text{Genauigkeit}+\text{Trefferquote}}$

Das F_1-Maß kombiniert Relevanz und Trefferquote. Wichtig zu beachten ist, dass im Allgemeinen die Erhöhung der Relevanz zu einer Verringerung der Trefferquote führt und umgekehrt [65].

Eine spezielle Metrik für Mehrklassenprobleme ist der *Jaccard Ähnlichkeitskoeffizient* (JÄK) J, der für jede Klasse $i \in [1, n]$ berechnet und dann über alle n Klassen gemittelt wird:

$$J = \frac{1}{n} \sum_{i=1}^{n} \frac{\#\text{Richtig Positive}_i}{\#\text{Falsch Positive}_i + \#\text{Falsch Negative}_i + \#\text{Richtig Positive}_i} \tag{2.42}$$

In dieser Arbeit wird diese Metrik bei der Vorevaluation passender KNN-Architekturen für die Punktwolkensegmentierung verwendet (siehe Kapitel 4.6.3.1).
Eine weitere Darstellung der Güte eines binären Klassifikators ist die *Operationscharakteristik eines Beobachters* (OCB, engl. *Receiver Operating Characteristic* (ROC)). Sie stellt Trefferquote und Ausfallrate gegenüber. Die Fläche unter der Kurve verschiedener Klassifikatoren lässt sich nun vergleichen, wobei ein perfekter Klassifikator die Fläche 1 hat. Die meist in Schwarz eingezeichnete Winkelhalbierende ist dabei die Kurve eines vollständig zufälligen Klassifikators mit einer Fläche unter der Kurve von $0,5$. Zum Einsatz kommt diese Darstellung in Kapitel 4.6.3.2 (siehe Abbildung 4.19b).
Eine anschauliche Darstellung der Ergebnisse eines Mehrklassenklassifikators ist die Wahrheitsmatrix, auch Konfusionsmatrix genannt. Jede Zeile der Matrix stellt die Klasse dar, die laut Trainingsdatensatz vorhergesagt werden sollte, jede Spalte diejenige Klasse, die tatsächlich vorhergesagt wurde. Eine Zelle gibt an, wie viele Elemente des Trainingsdatensatzes aus der, durch die Zeile gegebenen Klasse stammen und für die, die durch die Spalte gegebene Klasse prädiziert wurde. Alle Werte aller Zellen aufaddiert ergeben die Anzahl der Elemente im Trainingsdatensatz. Die Wahrheitsmatrix eines perfekten Modells ist eine Diagonalmatrix. Ein Beispiel für eine normierte Wahrheitsmatrix (alle Felder summieren sich zu 1 auf) findet sich in Kapitel 4.6.3.2 (siehe Abbildung 4.19a).

Unter- und Überanpassung. Ein trainiertes Modell ist an den Trainingsdatensatz überangepasst, wenn es zwar auf dem Trainingsdatensatz gute Prädiktionsergebnisse liefert, jedoch nicht auf ungesehenen Daten generalisieren kann. Dabei passt sich das Modell auf zu detaillierte Muster an, die auch Teil des informationslosen Rauschens innerhalb des Trainingsdatensatzes sein können und lernt denselben dabei quasi auswendig. Gründe dafür können zu komplexe Modelle mit zu vielen Parametern oder ein zu kleiner oder zu verrauschter Trainingsdatensatz sein. Dem entgegensteuern kann man, neben der Verbesserung der Qualität und Quantität des Trainingsdatensatzes, mit einer Vereinfachung des Modells, einer sog. Regularisierung [65].
Bei der Unteranpassung handelt es sich um das genaue Gegenteil der Überanpassung: Das Modell ist nicht komplex genug, um die Struktur des Trainingsdatensatzes vollständig zu erfassen. Gegenmaßnahmen können die Wahl eines leistungsfähigeren Modells oder auch die Auswahl aussagekräfti-

gerer Merkmale sein [65].

Datenaugmentierung. Unter Datenaugmentierung versteht man die synthetische Variation von Elementen eines Trainingsdatensatzes. Besteht der Trainingsdatensatz z.B. aus Punktwolken, wäre die Erzeugung von affin transformierten Kopien der Ausgangspunktwolken mit zufällig gewählten Parametern eine solche Augmentierung. Der resultierende Datensatz ist somit reicher an Variation. Je nach konkreter Ausprägung handelt es sich bei der Datenaugmentierung um eine wirksame Strategie gegen Überanpassung.

Im Weiteren werden Aufgaben und Methoden des überwachten Lernens beschrieben, wobei der Fokus auf diskriminativen Modellen liegt.

2.5.2.1. Klassische Verfahren

Zu den Verfahren, die sich in den letzten Jahren im praktischen Einsatz bewährt haben, zählen die Stützvektormaschinen (engl. *Support Vector Machine*, kurz SVMs) [3, 20, 34] und die Entscheidungsbäume (engl. *Decision Trees*) [23, 65, 82, 88]. Dabei sind SVMs besonders für kleine bis mittelgroße (100-10k) Datensätze mit hoher Dimensionalität geeignet [65]. Wohingegen Entscheidungsbäume den Vorteil haben, menschenlesbare Regeln zu finden und anzuwenden, was Prädiktionen ermöglicht, die leicht nachvollziehbar sind. Als nachteilig gilt ihre Anfälligkeit gegen kleine Veränderungen oder Rotationen des Eingabedatensatzes [65]. Aufgrund ihrer hohen strukturellen Flexibilität im Vergleich zu klassischen Verfahren wurde in dieser Arbeit auf *Künstliche Neuronale Netze* (KNNs) zurückgegriffen, wie sie in folgendem Kapitel beschrieben werden.

2.5.2.2. Künstliche Neuronale Netze (KNNs)

Ein *Künstliches Neuronales Netz* (KNN) ist im ursprünglichen Sinn eine abstrakte, vereinfachte Umsetzung der Verknüpfung biologischer Neuronen, wie sie im Gehirn komplexer Lebensformen vorkommen, zur Anwendung im ML. Im Folgenden sind sog. Perzeptrone als grundlegende KNN-Architektur im Fokus [151]. Diese bestehen aus einem Eingabevektor, einem Ausgabevektor, einem Gewichtsvektor und einer Aktivierungsfunktion (siehe Abbildung 2.14, links). Als Aktivierungsfunktionen kommen meist Funktionen zum Einsatz, die an relevanten Stellen differenzierbar sind, wie z.B. die Softmax-Funktion, der hyperbolische Tangens oder sog. Rectifier [72].
Mehrere Perzeptrone können in Schichten zusammengefasst werden und formen damit *mehrschichtige Perzeptrone* (siehe Abbildung 2.14, rechts, MSPs). Dabei ist jedes Perzeptron der vorhergehenden Schicht mit jedem der folgenden Schicht verbunden. Neben der Eingabe- und der Ausgabeschicht können dabei beliebig viele Schichten hinzugenommen werden, die dann als sog. verdeckte Schichten bezeichnet werden. Zusätzlich zu den Perzeptronen enthält jede außer der Ausgabeschicht ein sog. Bias-Neuron, das immer den

Wert 1 ausgibt. Es kann gezeigt werden, dass MSPs jede beliebige stetige Funktion zwischen zwei Euklidischen Räumen approximieren können [35, 85, 86], was deren Vielseitigkeit unterstreicht.

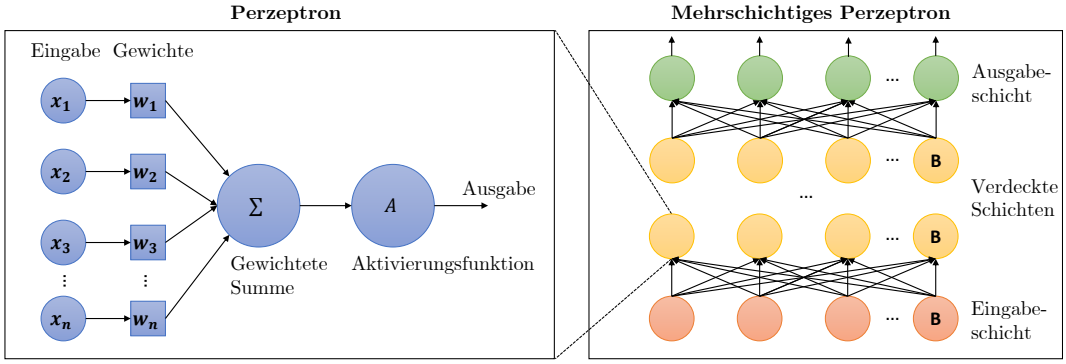


Abbildung 2.14.: Links: Schema eines einfachen Perzeptrons. Rechts: Mehrschichtiges Perzeptron mit einer Eingabeschicht (rot), einer Ausgabeschicht (grün), mehreren verdeckten Schichten (orange) und Bias-Neuronen (**B**). Jedes Perzeptron ist mit jedem anderen der darüberliegenden Schicht verbunden.

Sind alle zum Einsatz kommenden Aktivierungsfunktionen sowie die verwendete Loss-Funktion differenzierbar, lässt sich ein MSP mittels gradientenbasierter Optimierungsmethoden (siehe Kapitel 2.4.3.1) auf einem Trainingsdatensatz S effizient trainieren. Dabei wird ein einzelner Optimierungsschritt als Trainingsepoche bezeichnet. Die Menge der zu ermittelnden Parameter θ ist dabei die Menge aller Gewichte. Die Mächtigkeit des Konzepts erschließt sich, wenn man statt der einfachen Perzeptronstruktur arbiträr verknüpfte Operationen betrachtet, die sich zu einem durch θ parametrisierten, zyklenfreien Berechnungsgraphen h_θ mit Eingabe-, Ausgabe und verdeckten Knoten zusammenfügen. Solange jede Operation des Graphen differenzierbar ist, kann das Modell h_θ effizient trainiert werden, indem das Optimierungsproblem in Gleichung 2.41 gelöst wird.
Eine effiziente Lösung mittels gradientenbasierten Verfahren benötigt dazu entsprechende Gradienten der Loss-Funktion L, also die partiellen Ableitungen von L nach den Parametern θ. Die robuste Ermittlung der Gradienten erfolgt dabei über das automatische Differenzieren im Rückwärtsmodus, das auf der Anwendung der Kettenregel basiert [65]. Die Kombination von gradientenbasierten Verfahren mit der automatischen Differenziation im Rückwärtsmodus zur Optimierung der Loss-Funktion wird auch als Fehlerrückführung (engl. Backpropagation) bezeichnet [65, 155]. Aus der beschriebenen Flexibilität ergibt sich eine Vielzahl von Architekturtypen, die je nach Anwendungszweck zum Einsatz kommen.

Faltende Neuronale Netze. Für Modelle, die auf ein-, zwei- oder mehrdimensionalen Rasterdaten (z.B. Bildern) trainiert werden, bieten sich

Faltende Neuronale Netze an (FNNs, engl. Convolutional Neural Networks (CNNs)) [60]. Diese verknüpfen sog. Faltungsschichten mit dimensionsreduzierenden, sog. Poolingschichten. Eine Faltungsschicht besteht aus Filterkerneln festgelegter Größe und mit lernbaren Gewichten, die auf die Eingabe angewandt werden. Dabei besteht die Eingabe aus mehreren Kanälen, für die in der Faltungsschicht je ein Filterkernel zur Verfügung steht (bei Farbbildern z.B. sind es drei, je die Pixel des Rot-, Grün- bzw. Blaukanals). Die Ausgabe der Faltungsschicht besteht ebenfalls aus mehreren Kanälen, jeder durch eine sog. Merkmalskarte repräsentiert. Dabei ist die Anzahl der Ausgabekanäle frei wählbar und muss nicht der der Eingabekanäle entsprechen. Meist werden Faltungsschichten zudem mit einer Rectifier-Aktivierungsfunktion, die auf die Ausgabe angewandt wird, kombiniert. Eine Poolingschicht führt eine Unterabtastung der Eingabe durch, indem diese in Regionen unterteilt wird und auf jeder Region eine symmetrische Funktion angewandt wird, die die Datenpunkte der Region auf einen einzelnen Datenpunkt abbildet. Für diese Arbeit ist das sog. Max-Pooling relevant, bei dem als symmetrische Funktion die Maximum-Funktion gewählt wird. Als Ausgabeschicht eines FNNs dient ein MSP, das die Ausgabe der letzten Poolingschicht, je nach Aufgabe, auf Klassen (Klassifikation) oder einen kontinuierlichen Wertebereich (Regression) abbildet. Eine FNN-Architektur kann dabei mehrere Faltungs- und Poolingschichten beinhalten, jedoch nur eine Ausgabeschicht.

Rekurrente Neuronale Netze. Soll die Länge des Eingabevektors variabel sein (z.B. bei Zeitserien), werden sog. *Rekurrente Neuronale Netze* (RNNs) eingesetzt [155]. Bei dieser Netzarchitektur werden Rückkopplungen einzelner Knoten zu Knoten vorhergehender Schichten oder zu sich selbst zugelassen, indem die Ausgabe wieder als Eingabe verwendet wird. Der Berechnungsgraph ist damit nicht mehr zyklenfrei. Es existieren jedoch Trainingsmethoden für RNNs, die trotzdem auf die Methode der Fehlerrückführung zurückgreifen können [65]. Eine weitverbreitete rekurrente Architektur ist die *Long Short-Term Memory*-Architektur (LSTM) [83].

Grundsätzlich existieren Architekturen für generative und diskriminative Modelle, was das Anwendungsspektrum von KNNs sehr breit macht. Die Wahl der Aktivierungsfunktion hängt im Allgemeinen von der zu lösenden Aufgabe (Klassifikation (Zwei- oder Mehrklassenprobleme) oder Regression), von der Lage des Neurons (verdeckte Schicht oder Ausgabeschicht) und von der verwendeten Netzarchitektur (z.B. FNNs oder RNNs) ab. Die Eignung einer Loss-Funktion ist ebenfalls aufgabenabhängig. So wird bei Regressionsaufgaben meist auf die quadratische Abweichung und bei Klassifikationsaufgaben auf die Kreuzentropie zurückgegriffen [65].
Aus der Verallgemeinerung klassischer KNN-Architekturen geht die sog. *Differenzierbare Programmierung* (DP) als neues Programmierparadigma hervor [194]. Deren Grundlage bildet die funktionale Programmierung, also

die Algorithmenentwicklung mittels der Kombination von Funktionen ohne Nebeneffekte. Hinzu kommt nun eine Parametrisierung der Funktionen, die anhand von Trainingsdaten gelernt wird. Beschränkt man sich zudem auf differenzierbare Funktionen, ist der Trainingsprozess effizient. Zusammen mit den für KNNs entwickelten Werkzeugen zur automatischen Differenzierung und Optimierung ergibt sich ein leistungsfähiges Paradigma mit Anwendungen im maschinellen Sehen [107] und in der Simulation dynamischer Systeme [81]. In dieser Arbeit kommt eine spezifische KNN-Architektur für die Klassifikation von Punktwolken zum Einsatz, die auf MSPs und FNNs basiert (siehe Kapitel 4.4.3).

2.5.3. Unüberwachtes Lernen

Beim unüberwachten Lernen existieren keine Labelvektoren. Vielmehr wird versucht, generelle Zusammenhänge innerhalb der vorhandenen Merkmalsvektoren $X = \{x_1, ..., x_m\}$ zu finden. Dazu gehören Verfahren, die das Auffinden von Gruppierungen innerhalb einer Gesamtmenge anhand homogener Merkmale ermöglichen (sog. Clustering) sowie Methoden, die die Dimension des Merkmalsraums reduzieren und dabei die Varianz des Datensatzes weitmöglichst erhalten (sog. Dimensionsreduzierung). Für die vorliegende Arbeit relevante Methoden für unüberwachtes Lernen werden im Folgenden beschrieben.

2.5.3.1. Clustering

Clustering-Methoden identifizieren Teilmengen eines Datensatzes, welche hinsichtlich bestimmter Merkmale ähnlich oder gleich sind.

k-Means. Als Vertreter der partitionierenden Clustering-Verfahren, wird beim k-Means-Clustering der Datensatz X in k Teilmengen $\{X_1, ..., X_k\}$ zerlegt, sodass die akkumulierten Distanzen der Datenpunkte einer Teilmenge zum empirischen Mittelwert $\overline{X_i}$ der jeweiligen Teilmenge $X_i, i \in \{1, ..., k\}$ minimal sind. Damit muss

$$\underset{X_1,...,X_k}{\operatorname{argmin}} \sum_{i=1}^{k} \sum_{x \in X_i} d(x, \overline{X_i}) \tag{2.43}$$

minimiert werden, wobei $d(\cdot, \cdot)$ eine Distanzmetrik auf dem Merkmalsraum X ist. Dieses Optimierungsproblem ist $\mathcal{NP}$-*schwer* [64]. Es existieren jedoch effiziente Heuristiken, wie der Algorithmus von Lloyd [112] mit einer Laufzeitkomplexität von $O(tkm)$ (t ist die Anzahl durchgeführter Iterationen). k-Means-Clustering ist dann geeignet, wenn die Anzahl der gewünschten Cluster k bekannt ist und Cluster grob der Form einer Hyperkugel entsprechen. In dieser Arbeit kommt k-Means-Clustering bei der automatischen Erzeugung von Trainingsdatensätzen zum Einsatz (siehe Kapitel 4.4.1.1).

DBSCAN. *Density-Based Spatial Clustering of Applications with Noise* (DBSCAN) gehört zur Gattung der dichtebasierten Clustering-Verfahren, bei denen Cluster als Regionen erhöhter Datenpunktdichte aufgefasst werden [44]. Im Folgenden wird eine abstrakte Beschreibung des Verfahrens angelehnt an [164] vorgenommen. Für DBSCAN zentral ist die Abbildung $\rho\colon X \to \mathbb{R}$. Diese ordnet jedem Datenpunkt aus X einen Dichtewert zu, der sich aus der Anzahl der Punkte, die sich in einem benutzerdefinierten Radius ϵ um den Datenpunkt befinden, ergibt:

$$\rho_\epsilon(x) = \sum_{s \in X \setminus \{x\}} \begin{cases} 1, & \text{falls } d(x,s) \leq \epsilon \\ 0, & \text{sonst,} \end{cases} \tag{2.44}$$

wobei $d(\cdot,\cdot)$ eine Distanzmetrik auf dem Merkmalsraum X ist. Abhängig von seiner Dichte wird nun jedem Datenpunkt eine Kategorie aus $\{\text{Kernpunkt}, \text{Randpunkt}, \text{Rauschpunkt}\}$ zugeordnet:

$$c_{\epsilon,\mu}(x) = \begin{cases} \text{Kernpunkt}, & \text{falls } \rho_\epsilon(x) \geq \mu \\ \text{Randpunkt}, & \text{falls } \exists s \in X : c_{\epsilon,\mu}(s) = \text{Kernpunkt} \wedge d(x,s) \leq \epsilon \\ \text{Rauschpunkt}, & \text{sonst,} \end{cases} \tag{2.45}$$

wobei μ die benutzerdefinierte minimale Anzahl von Punkten pro Cluster darstellt. Rauschpunkte werden direkt entfernt. Ein Graph G wird mit den Kernpunkten als Knoten aufgebaut. Zwei Knoten in G werden mit einer Kante verbunden, falls der Abstand zwischen den zugehörigen Kernpunkten kleiner ϵ ist. Die verbundenen Komponenten von G entsprechen den gefundenen Clustern. Abschließend werden Randpunkte den Clustern ihrer nächstgelegenen Kernpunkte zugeordnet.
Die Laufzeitkomplexität des Verfahrens ist $O(m^2)$, wenn keine speziellen Datenstrukturen zur Beschleunigung der Nachbarschaftssuchen, wie sie zur Berechnung von ρ_ϵ nötig sind, verwendet werden [164]. Interessant ist das Verfahren in Szenarien in denen die Anzahl der Cluster initial unbekannt ist. Unterscheidet sich die Dichte des Datensatzes lokal stark, ist die von DBSCAN vorgenommene Festlegung auf genau einen Dichteschwellwert ϵ hingegen problematisch. In dieser Arbeit kommt DBSCAN bei der Segmentierung von Punktwolken zum Einsatz (siehe Kapitel 3.2.3).

Spektrales Clustering. Beim Spektralen Clustering werden die Merkmalsvektoren aus X als Knoten und die Unähnlichkeit zwischen zwei Merkmalsvektoren als nicht-negatives Kantengewicht eines Graphen G mit Adjazenzmatrix A repräsentiert. Dabei kann G auf unterschiedliche Weise aufgebaut werden [192]:

- Vollständiger Graph: Jeder Knoten in G wird mit jedem anderen Knoten verbunden und die entsprechenden Kantengewichte berechnet.

- k-nächste Nachbarn: Jeder Knoten in G wird mit seinen k, bezogen auf die verwendete Ähnlichkeitsmetrik, nächsten Nachbarn verbunden. Dabei werden doppelte Verbindungen ignoriert.
- ϵ-Nachbarschaft: Zwei Knoten in G werden mit einer Kante verbunden, wenn deren Abstand bezüglich der verwendeten Ähnlichkeitsmetrik kleiner einem ϵ ist.

Ziel ist es, den normalisierten, minimalen Schnitt von G zu finden, also eine Partitionierung von X in k Partitionen $\{X_1, ..., X_k\}$, sodass die Kosten zur Trennung der Cluster minimal sind. Dabei geben die Kosten eines Schnitts die akkumulierten Gewichte der zu entfernenden Kanten an. Das gelingt mit der Lösung des Optimierungsproblems

$$\underset{X_1,...,X_k}{\operatorname{argmin}} \sum_{l=1}^{k} \frac{\sum_{x_i \in X_l} \sum_{x_j \notin X_l} A_{ij}}{\sum_{x_i \in X_l} \sum_{x_j \in X_l} A_{ij}}, \tag{2.46}$$

wobei dieses $\mathcal{NP}$-*schwer* ist [192, 193]. Eine effiziente Approximation der Lösung kann jedoch mit Spektralem Clustering gefunden werden. Dazu werden folgende Schritte durchgeführt:

1. Zuerst werden die Eigenvektoren $\{v_1, ..., v_k\}$ zu den k größten Eigenwerten der Laplace-Matrix $L = D - A$ von G berechnet, wobei D die sog. Gradmatrix mit
$$D_{ij} := \begin{cases} \deg(x_i) \text{ falls } i = j \\ 0 \text{ sonst} \end{cases} \tag{2.47}$$
darstellt. Dabei ist $\deg(x_i)$ die Anzahl der mit dem i-ten Knoten x_i aus G verbundenen Kanten.
2. Die $m \times k$ Matrix U wird mit den Eigenvektoren $\{v_1, ..., v_k\}$ als Spalten befüllt.
3. Ein partitionierendes Clustering-Verfahren, wie z.B. k-Means, wird auf die Zeilenvektoren von U angewandt, was jeder Zeile von U einen von k Clustern zuordnet. Mit der gleichen Zeilenzuordnung lassen sich die Merkmalsvektoren aus X zu Clustern zuordnen: Merkmalsvektor x_i wird dem Cluster des i-ten Zeilenvektors von U zugeordnet.

Spektrales Clustering hat eine Laufzeitkomplexität von $O(m^3)$. Ist A dünn besetzt, reduziert sie sich auf $O(m^2)$ [163]. Das DBSCAN-Clustering-Verfahren kann als Spezialfall des Spektralen Clusterings verstanden werden [163]. Dabei sind die von DBSCAN gefundenen Cluster identisch mit dem Resultat eines minimalen Schnitts von G, wenn als Knoten nur Kernpunkte betrachtet werden. In dieser Arbeit kommt Spektrales Clustering bei der Segmentierung von Punktwolken in schwach-konvexe Teilmengen zum Einsatz (siehe Kapitel 3.2.3).

2.5.3.2. Dimensionsreduzierung

Bei der Dimensionsreduzierung wird versucht, einen hochdimensionalen Datensatz in einen Raum niedrigerer Dimension zu überführen, ihn also zu vereinfachen, und dabei möglichst wenig relevante Information zu verlieren. Grundsätzlich lassen sich zwei Verfahrenskategorien unterscheiden [65]:

- **Projektion:** Für die m n-dimensionalen Merkmalsvektoren von Datensatz X wird ein d-dimensionaler Unterraum (wobei $d < n$) identifiziert, in den Merkmalsvektoren projiziert werden.

- **Lernen von Mannigfaltigkeiten:** Eine d-dimensionale Mannigfaltigkeit ist ein topologischer Raum (wieder: $d < n$), der sich lokal wie ein d-dimensionaler Euklidischer Raum verhält. So ist die Oberfläche einer Kugel eine zweidimensionale Mannigfaltigkeit eingebettet in den 3-dimensionalen Euklidischen Raum. Anstatt nun eine simple Projektion vorzunehmen, wird eine Mannigfaltigkeit M mit $d_M < n$ Dimensionen gelernt, auf der die Merkmalsvektoren aus X möglichst passgenau liegen. Nun kann jeder Merkmalsvektor bezüglich M ausgedrückt werden, d.h. mit nur noch d_M-dimensionalen Merkmalsvektoren. Ein Vertreter dieser Methodenfamilie ist die lokal-lineare Einbettung [154].

Im Folgenden wird die Hauptkomponentenanalyse [137] als ein in dieser Arbeit zur Anwendung kommender Vertreter der Projektionsverfahren vorgestellt. Die Hauptkomponentenanalyse ist ein weitverbreitetes Verfahren zur Dimensionsreduzierung, welches diejenigen d Achsen im Datensatz X identifiziert, in deren Richtung die meiste Varianz zu finden ist. Die Achsen, beschrieben durch die Einheitsvektoren $\{c_1, ..., c_n\}$, bezeichnet man als Hauptkomponenten. Dabei spannen die Vektoren $\{c_1, ..., c_d\}$ einen d-dimensionalen Unterraum des Merkmalsraums auf. Sei X_n eine $m \times n$ Matrix

$$X_n = \begin{bmatrix} (x_1)^T \\ \vdots \\ (x_m)^T \end{bmatrix} \tag{2.48}$$

mit den Merkmalsvektoren als Zeilenvektoren. Dann ergibt sich die Matrix

$$V = \begin{bmatrix} | & & | \\ c_1 & \dots & c_n \\ | & & | \end{bmatrix}, \tag{2.49}$$

welche die Hauptkomponenten als Spalten enthält, aus der Singulärwertzerlegung von X_n in $X_n = U \cdot \Sigma \cdot V^T$, wobei U eine orthonormale $m \times m$ Matrix, V eine orthonormale $n \times n$ Matrix und Σ eine $m \times n$ Diagonalmatrix ist. Dabei ist V eine orthonormale Basis des Zeilenraums von X_n und U eine orthonormale Basis des Spaltenraums. Die Werte auf der Diagonalen von Σ werden

Singulärwerte genannt. Die Projektionsmatrix

$$P = \begin{bmatrix} | & & | \\ c_1 & \dots & c_d \\ | & & | \end{bmatrix}, \tag{2.50}$$

besteht nun aus den ersten d Spalten von V und wird genutzt um X_n in einen d-dimensionalen Raum zu projizieren: $X_d = X_n \cdot P$.
Erweiterungen der Hauptkomponentenanalyse ermöglichen z.B. die inkrementelle Dimensionsreduktion bei großen Datensätzen [152] oder sind in der Lage, auch nichtlineare Projektionen abzubilden [160]. In dieser Arbeit kommt die Hauptkomponentenanalyse indirekt als Verfahren für die Ermittlung der Orientierung einer zentrierten Punktwolke im Raum zum Einsatz. Die Orientierung wird in dem Fall mit einer 3×3 Transformationsmatrix beschrieben, die exakt V ($n = 3$) entspricht.

2.6. Zusammenfassung

In diesem Kapitel wurden die Grundlagen der Arbeit gelegt. Dazu zählen neben der Einführung von wichtigen mathematischen Konzepten auch relevante Bausteine zur Erschließung der Problemdomäne. Den Anfang machten ausgesuchte Repräsentationsformen von Festkörpern sowie mögliche Repräsentationskonversionen als zentrale Bestandteile des Problemkomplexes (siehe Kapitel 2.3). Da sich viele in dieser Arbeit behandelte Teilprobleme als Optimierungsprobleme formulieren lassen, wurden diese mit ausgewählten Lösungsstrategien vorgestellt (siehe Kapitel 2.4). Darauf aufbauend führte Kapitel 2.5 Methoden des MLs ein, welche eigene Optimierungsprobleme in sich bergen. In Form von KNNs und verschiedenen Clustering-Methoden kommen sie im weiteren Verlauf der Arbeit zum Einsatz. Basierend auf den hier eingeführten Konzepten wird im Folgekapitel die in dieser Arbeit behandelte Problemstellung im Detail beleuchtet und eine generische Prozess-Pipeline zu deren Lösung vorgestellt (siehe Kapitel 3).

3. Problemdefinition und Prozess-Pipeline

Dieses Kapitel widmet sich einer genauen Beschreibung des in dieser Arbeit behandelten Problems und beschreibt zudem systematisch existierende Lösungen. Auf Basis einer exakten Problemdefinition (siehe Kapitel 3.1) wird dazu eine, in dieser Tiefe neuartige, Prozess-Pipeline vorgestellt, die als Schablone für existierende sowie neu eingeführte Lösungsstrategien fungiert (siehe Kapitel 3.2). Darauf aufbauend werden in diesem Kapitel bestehende Lösungen für einzelne Pipelineschritte detailliert erläutert. Diese dienen als Grundlage für die in den folgenden Inhaltskapiteln thematisierten neuen Ansätze zur Lösung einzelner Teilprobleme. Das Kapitel schließt mit einer kurzen Zusammenfassung (siehe Kapitel 3.3).

3.1. Problemdefinition

Basierend auf den theoretischen Grundlagen, wie sie in Kapitel 2 gelegt wurden, lässt sich die in dieser Arbeit behandelte Problemstellungen, die Synthese und Optimierung von KBs auf unstrukturierten räumlichen Daten, auf drei Kernaspekte reduzieren:

- **Problem 1:** Die Extraktion geometrischer Primitive, wie Kugeln, Zylinder und konvexe Polytope, aus unstrukturierten räumlichen Daten repräsentiert durch Punktwolken.

- **Problem 2:** Die Synthese von KBs aus Punktwolken und extrahierten geometrischen Primitiven. Bei diesem Problem handelt sich also um eine Repräsentationskonversion (siehe Kapitel 2.3.4), wobei erschwerend hinzukommt, dass die Ursprungsrepräsentation (Punktwolke) hinsichtlich geometrischer und topologischer Information mangelhaft bzw. uneindeutig ist (siehe Kapitel 2.3.2.3).

- **Problem 3:** Die Optimierung von KBs. Je nach Syntheseverfahren können KBs über redundante Strukturen verfügen oder hinsichtlich ihrer manuellen Editierbarkeit Defizite aufweisen. Daher ist es notwendig, geeignete Optimierungsverfahren zu entwickeln, die Redundanzen beseitigen, die Editierbarkeit verbessern oder hinsichtlich anderer Kriterien Verbesserungen erzielen können.

Eine Lösungsstrategie für die drei Probleme lässt sich in Form einer mehrstufigen Prozess-Pipeline formulieren (siehe Abbildung 3.1), wie sie im Rahmen dieser Arbeit entwickelt wird. Diese basiert auf dem vom Autor in [56] vorgestellten Prozessmodell, erweitert dieses jedoch umfassend. Dabei sind die drei großen Inhaltskapitel (Kapitel 4, 5 und 6) neuartigen Lösungen für je eines der drei genannten Probleme gewidmet.

3.2. Prozess-Pipeline

In diesem Kapitel werden die einzelnen Pipeline-Stufen zusammen mit bereits verfügbaren Lösungsverfahren im Detail erläutert. Diese Lösungsstrategien dienen den in dieser Arbeit neu vorgestellten Verfahren als Basis.

3.2.1. Punktwolkenerfassung

Dieses Kapitel behandelt die verschiedenen Quellen, die für die Erzeugung von Punktwolken infrage kommen.

3.2.1.1. Sampling

Wie in Abbildung 2.12 dargestellt, lassen sich Festkörper, die als Zellkomplexe, KBs oder Begrenzungsflächenmodelle repräsentiert werden, in eine Repräsentation als Punktwolke umwandeln. Dazu wird meist ein den Festkörper umschließender Quader angenommen, der, ähnlich einem Voxelgitter, in reguläre Zellen unterteilt wird. Für den Mittelpunkt jeder Zelle wird der Abstand zur Oberfläche des Festkörpers ermittelt. Liegt dieser in einem vorher festgelegten Intervall, wird der Mittelpunkt zur Punktwolke hinzugefügt. Aufgrund der Diskretisierung des umschließenden Volumens geht dieser Umwandlungsprozess dabei mit einem Informationsverlust einher. Ebenfalls möglich ist die Umwandlung von Zellgruppierungen in Punktwolken. Dazu wird beispielsweise jeder Zellmittelpunkt zur Punktwolke hinzugefügt, was mehr einem Einfügestatt einem Abtastvorgang entspricht.

3.2.1.2. 3D-Scanning

3D-Scanning bezeichnet einen Messvorgang, bei dem Objekte aus der physikalischen Welt in unterschiedlicher Granularität abgetastet werden und das Ergebnis z.B. als Punktwolke abgelegt wird. Dabei existiert eine Vielzahl unterschiedlicher Verfahren, die sich hierarchisch kategorisieren lassen, wie Abbildung 3.2 zeigt.
Zuallererst lassen sich Messsysteme unterscheiden, die das zu scannende Objekt während des Messvorgangs berühren (*Mit Kontakt*) und solche, die das nicht tun (*Ohne Kontakt*). Ein Beispiel für ein kontaktbasiertes System sind

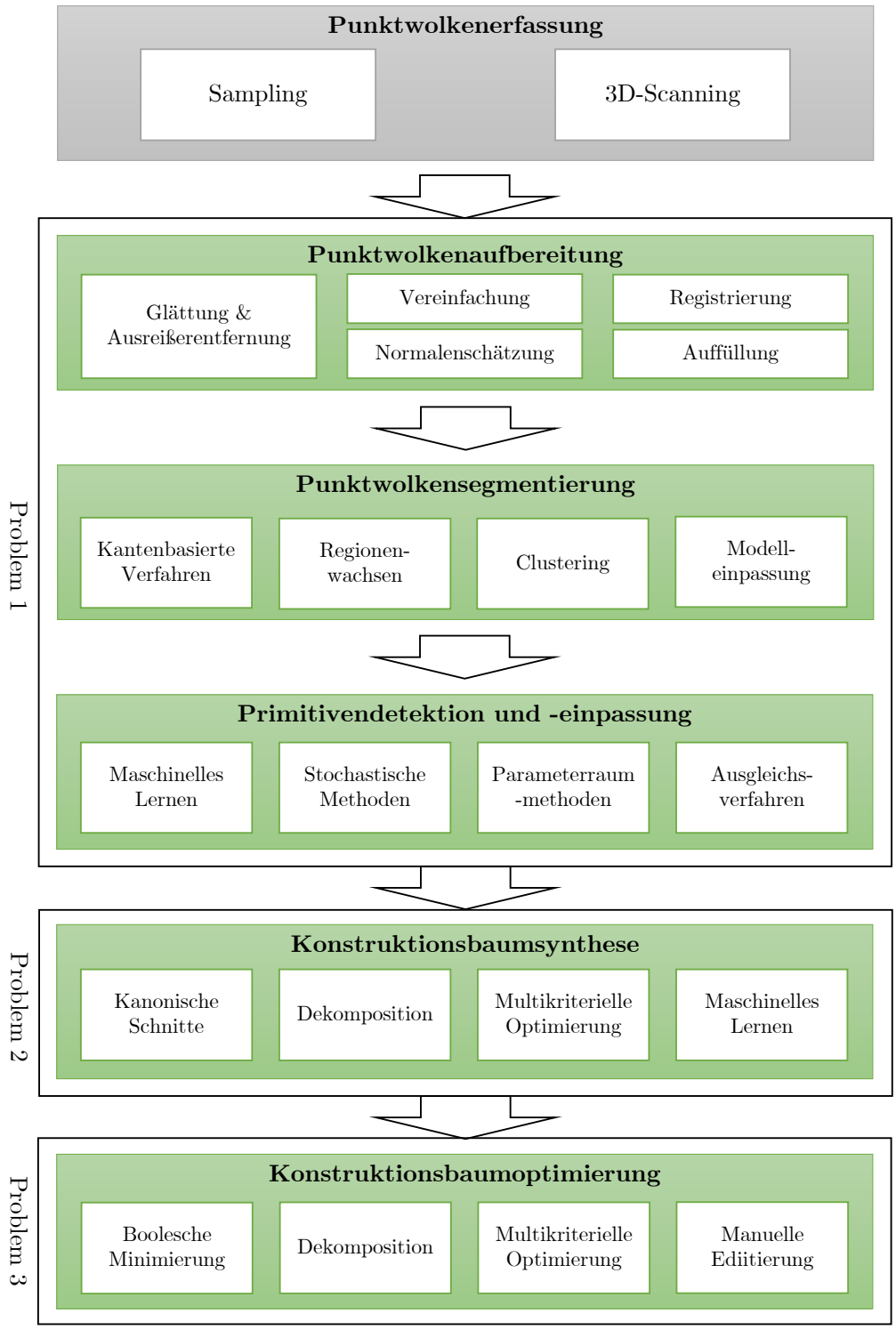


Abbildung 3.1.: Im Rahmen dieser Arbeit entwickelte Prozess-Pipeline für die Wiederherstellung und Optimierung von KBs aus Punktwolken. Prozessschritte im Fokus dieser Arbeit in Grün.

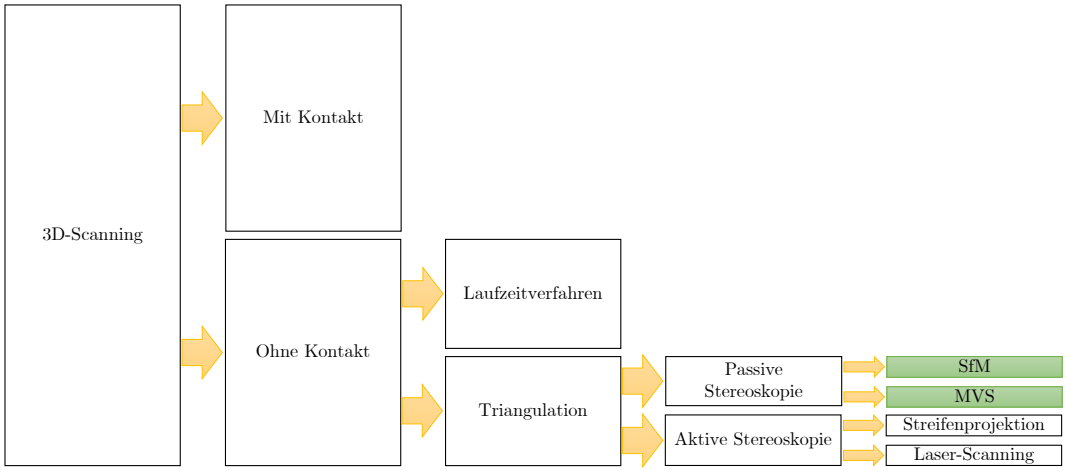
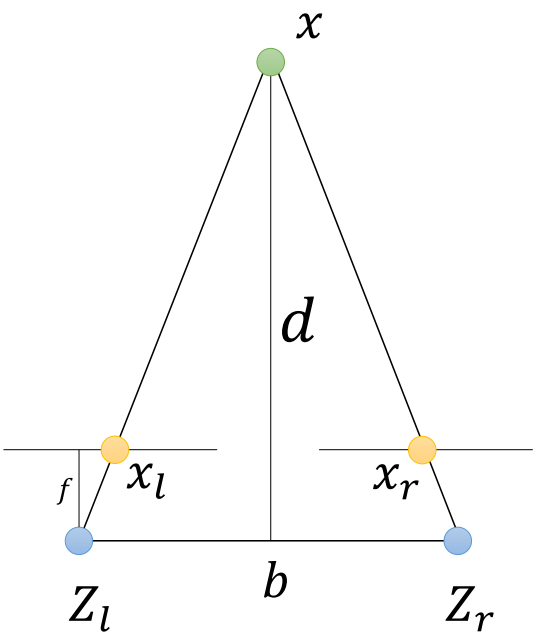


Abbildung 3.2.: 3D-Scanning-Verfahren hierarchisch kategorisiert. Für diese Arbeit relevante Verfahren sind grün markiert. Abbildung basierend auf [37].

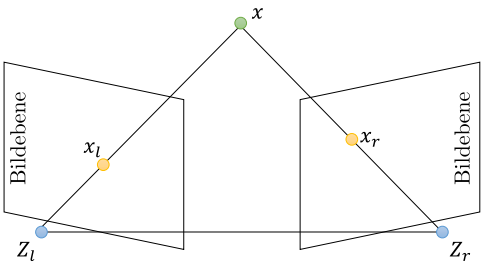
Messmaschinen aus der Koordinatenmesstechnik. Bei Systemen, die ohne Objektberührung auskommen, gibt es solche, die die Laufzeit des Lichts ausnutzen, um Abstände zu messen (*Laufzeitverfahren*) und solche, die auf dem Prinzip der Triangulation basieren.
Per Triangulation lässt sich die Entfernung (oder Tiefe) eines Objekts aus der Sicht zweier Beobachtungspunkte (Kameras) ermitteln. Die Distanz d zu einem Punkt x ergibt sich dabei aus der Gleichung

$$d = \frac{f * b}{x_l - x_r}, \tag{3.1}$$

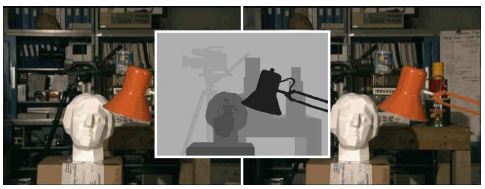
wobei x_l und x_r die Projektionen des dreidimensionalen Punkts x auf die Bildebene der linken (Zentrum Z_l, Brennweite f) respektive der rechten Kamera (Zentrum Z_r, Brennweite f) sind und beide Kameras den Abstand b zueinander haben. Diese vereinfachte Beschreibung von externen (Kamera-Pose) und internen (Objektiveigenschaften) Kameraparametern wird als kanonische Stereogeometrie bezeichnet (siehe Abbildung 3.3a). Im allgemeinen Fall wird dieser Zusammenhang zwischen rechter und linker Kamera über die sog. Epipolargeometrie modelliert (siehe Abbildung 3.3b) [77]. Ein Beispiel für eine Rekonstruktion von Tiefenwerten auf Basis zweier Kamerabilder ist in Abbildung 3.3c dargestellt.
Die für die Triangulation benötigten Punktkorrespondenzen x_l, x_r lassen sich entweder nur auf Basis optischer Sensoren (*Passive Stereoskopie*) oder aus einer Kombination von optischen Sensoren und aktiven Lichtquellen (*Aktive Stereoskopie*) ermitteln. Im ersten Fall werden nahezu gleiche Muster in zwei Bildern desselben Objekts gesucht, von denen ausgegangen werden kann, dass sie dasselbe Objekt abbilden (photogrammetrische Methoden). Dies ist nur bei diffus-reflektierenden Objektoberflächen möglich. Im zweiten Fall wird entwe-



(a) Kanonische Stereogeometrie.



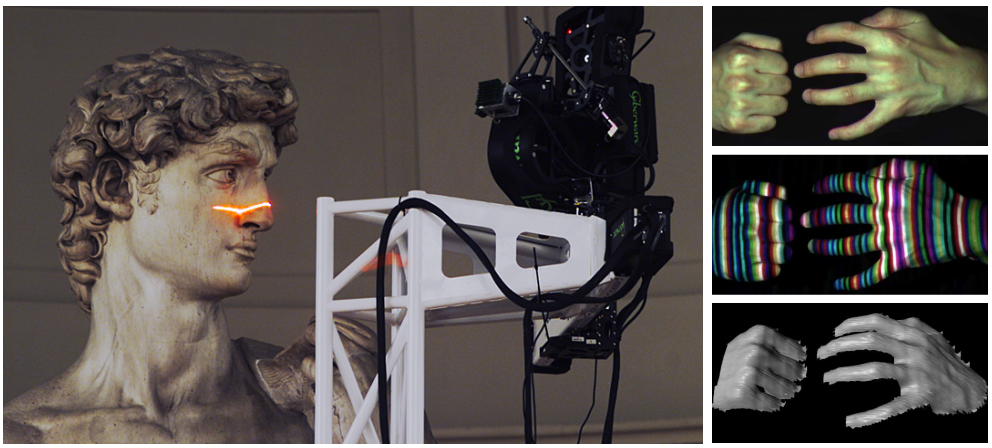
(b) Epipolargeometrie zweier Kameras. Quelle: Angepasst aus [183].



(c) Beispielszene mit linkem (links) und rechtem (rechts) Kamerabild sowie ermittelte Tiefenwerte (Mitte). Quelle: [124].

Abbildung 3.3.

der ein Lichtmuster (*Streifenprojektion*) oder ein Laserstrahl (*Laser-Scanning*) auf das Objekt projiziert, welches die Korrespondenzfindung erleichtert. Für Beispiele siehe Abbildung 3.4.



(a) Quelle: [105].

(b) Quelle: [204].

Abbildung 3.4.: Beispiele für Verfahren aus der aktiven Stereoskopie. a) Laser-Scanning. b) Streifenprojektion.

Bei der passiven Stereoskopie wird im üblichen Gebrauch zusätzlich noch zwischen Verfahren unterschieden, bei denen die externen und internen Kameraparameter der aufgenommenen Bilder bekannt sind (*Multi-View Stereo*

(MVS), z.B. [77, 161]) und bei denen diese Informationen fehlen (*Structure-From-Motion* (SfM), z.B. [89, 162]). Dabei sind MVS-Verfahren als Verallgemeinerungen der klassischen Stereoskopie auf mehr als zwei Kamerabilder zu sehen. Wichtig bei SfM-Strategien ist, dass ohne die Einführung zusätzlicher Nebenbedingungen für die Kameraparameter nur Rekonstruktionen möglich sind, die bis auf eine projektive Transformation mit dem zu rekonstruierenden Objekt übereinstimmen (sog. Projektive Rekonstruktion [89]). Möchte man hier eine Übereinstimmung bis auf eine affine oder gar Euklidische Transformation, muss man dem Rekonstruktionsprozess zusätzliche Information zuführen, was z.B. mit einer manuellen oder automatischen Kalibrierung der in- oder externen Kameraparameter gelingen kann [89]. Oftmals werden beide Verfahren kombiniert, um erst die Kameraparameter zu schätzen (per SfM) und darauf basierend eine dichtgelagerte Rekonstruktion der Oberfläche durchzuführen (per MVS) [161, 162].
Die Ergebnisse der hier aufgeführten Verfahren unterscheiden sich hinsichtlich Messgenauigkeit, Anzahl zurückgegebener Messpunkte sowie Maßstab und Oberflächenbeschaffenheit abtastbarer Objekte. Aus diesem Grund sind Prozessschritte notwendig, die die gemessene Punktwolke für die weitere Verarbeitung aufbereiten. Dazu geeignete Verfahren werden in Kapitel 3.2.2 behandelt.

3.2.2. Punktwolkenaufbereitung

Punktwolken können unterschiedlichste Mängel aufweisen, die eine Weiterverarbeitung erschweren. Aus diesem Grund ist meist ein Aufbereitungsschritt notwendig, der die Eingabepunktwolke O_e in eine aufbereitete Ausgabepunktwolke O_a überführt. Gängige Verfahren zur Beseitigung typischer Mängel werden hierzu im Folgenden beleuchtet, wobei [16] als Grundlage dient.

3.2.2.1. Rauschunterdrückung & Ausreißerentfernung

Hierbei handelt es sich um Methoden, die Messrauschen und -fehler in der Eingabepunktwolke reduzieren oder komplett eliminieren sollen. Ziel ist es dabei, die Punktmenge so zu bearbeiten, dass sie eine möglichst "glatte" Oberfläche beschreibt, ohne dabei geometrische Details zu verlieren. Im Weiteren werden als Rauschen zufällige Abweichungen in den Koordinaten von Punkten nahe der Objektoberfläche bezeichnet, meist aufgrund von sensorspezifischen Ungenauigkeiten. Ausreißer sind Punkte, die weit von der Objektoberfläche entfernt liegen und auf Fehler im Erfassungsprozess hindeuten [16]. Beide Fehlerarten werden in Abbildung 3.5 dargestellt.
Statistische Größen, wie empirischer Mittelwert $\bar{\cdot}$ und Standardabweichung $\tilde{s}(\cdot)$, lassen sich auf ungeordnete dreidimensionale Punktdaten anwenden, indem für jeden Punkt $o \in O_e$ der durchschnittliche Abstand zu den k nächsten Nachbarpunkten betrachtet wird ($\overline{d_o}$). In [156] wird darauf basierend der Schwellwert

$$t = \overline{D_{O_e}} + \alpha \cdot \tilde{s}(D_{O_e}) \tag{3.2}$$

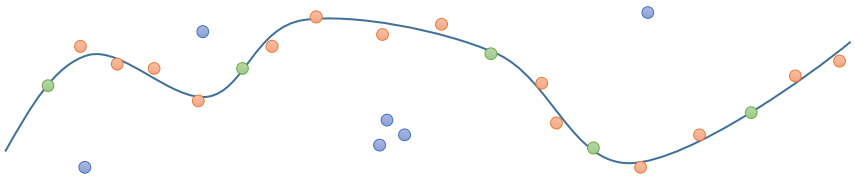


Abbildung 3.5.: Verrauschte Punkte (rot), Ausreißer (blau) und korrekte Punkte (grün). Quelle: Abgeändert aus [16].

definiert, wobei $D_{O_e} = \{\overline{d}_o | o \in O_e\}$ und α ein benutzerdefinierter Parameter ist. Ist nun der durchschnittliche Abstand eines Punktes größer t, ist er ein Ausreißer und damit nicht Teil der Ausgabepunktwolke O_a.
Andere Verfahren approximieren die Punktwolke lokal (in einer k-Punktnachbarschaft) durch eingepasste Flächenstücke und projizieren naheliegende Punkte auf ebendiese. Dabei werden Flächenstücke meist durch Polynome beschrieben, wie z.B. in [29] und [5]. In [103] wird eine Projektion der Punkte auf die Oberfläche vorher eingepasster Primitive durchgeführt. Das Verfahren wird in dieser Arbeit u.a. zur Vorverarbeitung von Eingabepunktwolken verwendet (siehe Kapitel 4.5.1).
Verfahren basierend auf KNNs sind aktuell im Fokus. Dabei existieren sowohl Ansätze aus dem Bereich des unüberwachten Lernens [80] als auch solche, die überwachte Lernstrategien einsetzen [144] oder sogar beide Strategien mit unterschiedlichen Netzzusammenstellungen unterstützen [116].

3.2.2.2. Vereinfachung

Enthält die Punktwolke zu viele Punkte für den gewünschten Einsatzort, müssen Verfahren zur Punktreduktion eingesetzt werden (siehe Abbildung 3.6). Da eine Punktwolke im Allgemeinen unstrukturiert ist und ihre Punkte in einem eindimensionalen Feld abgelegt sind, würde eine Entfernung von Punkten gemäß ihres Feldindexes (z.B. jeder zweite Punkt für eine Reduktion um 50%) potentiell zu unregelmäßig verteilten Abtastlücken auf der approximierten Oberfläche führen. Aus diesem Grund müssen Vereinfachungsverfahren permutationsinvariant bezüglich des Feldindexes sein, d.h. es muss egal sein, ob ein und derselbe Punkt an Feldindex 3 oder 523 liegt.

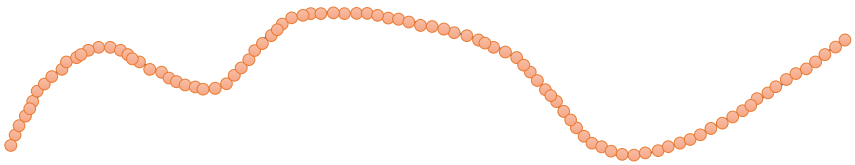


Abbildung 3.6.: Punktwolke mit hoher Punktdichte. Nicht jeder Punkt repräsentiert notwendiges Oberflächendetail.

Ein simples Verfahren beruht auf der Verwendung eines Voxelgitters mit benutzerdefinierter Zellgröße, welches die Eingabepunktwolke komplett einschließt.

Für jede Zelle wird der Zellmittelpunkt Teil der reduzierten Punktwolke, wenn sie mindestens einen Punkt der Eingabepunktwolke beinhaltet. Der Algorithmus stellt Mindestabstände zwischen Punkten sicher, hat jedoch keine direkte Kontrolle über die Größe der ausgedünnten Punktwolke.
Ein Verfahren, bei dem die Größe k der ausgedünnten Punktwolke im Vorfeld festgelegt werden kann, ist das sog. *Farthest Point Sampling* (FPS) [67, 122]. Dabei wird zuerst ein zufälliger Punkt aus der Eingabepunktwolke O_e in die Ausgabepunktwolke O_a überführt. Dann wird derjenige Punkt aus O_e ausgewählt, der die größte Distanz zum nächstliegenden Punkt aus O_a hat. Letzteres wird dann solange wiederholt, bis O_a k Punkte enthält. Im Vergleich zur zufälligen Selektion von k Punkten maximiert FPS die Distanzen zwischen den ausgewählten Punkten in O_a. Abbildung 2.8 zeigt mit FPS ausgedünnte Punktwolken. Das Verfahren wird in dieser Arbeit an mehreren Stellen verwendet (siehe z.B. Kapitel 4.4.3).
Es existiert eine Vielzahl weiterer Methoden zur Vereinfachung von Punktwolken [121, 136, 177], die z.B. zusätzlich in der Lage sind, Objektkanten zu erhalten [75]. Ein Vergleich verschiedener Varianten findet sich in [136].

3.2.2.3. Normalenschätzung

Möchte man die Oberfläche anhand der abgetasteten Punktwolke zusammenhängend rekonstruieren, ist es oft hilfreich, die Oberflächennormale zu kennen. Dabei handelt es sich um denjenigen normierten Vektor, der an einem Punkt der Oberfläche senkrecht zur selben steht. Ist diese Information nicht vorhanden (wie z.B. in Abbildung 3.7 für die orangen Punkte), muss sie aus den Punkten der Punktwolke geschätzt werden. Dies gelingt z.B. mit der Einpassung einer Ebene in die Punktmenge der k nächsten Nachbarn eines jeden Punkts der Punktwolke O_e, wobei die Flächennormale der Oberflächennormale an diesem Punkt annähernd entspricht. Dazu lassen sich prinzipiell alle Verfahren zur Primitiveneinpassung verwenden (siehe Kapitel 3.2.4). In diesem Fall wird aber meist eine Formulierung als lineares Ausgleichsproblem genutzt (siehe Kapitel 2.4.2) oder auf eine Analyse der Hauptkomponenten der Nachbarschaftpunktwolke eines jeden Punkts zurückgegriffen (siehe Kapitel 2.5.3.2). Letztere findet eine orthogonale Basis, die eine Nachbarschaftspunktwolke am besten repräsentiert und damit auch die Flächennormale der gewünschten Ebene als den Eigenvektor, dessen dazugehöriger Eigenwert der kleinste ist [84]. Ist die abgetastete Oberfläche hingegen gekrümmt, bietet es sich an, statt einer planaren, eine gekrümmte Approximationsfläche zu wählen [29]. Diese kann z.B. durch Quadriken repräsentiert werden (siehe Kapitel 2.3.3.2).
Neben der Richtung ist auch die Orientierung der Oberflächennormalen relevant, die möglichst konsistent für Nachbarpunkte der Punktwolke sein sollte. Als Gegenbeispiel bieten sich die grauen Punkte in Abbildung 3.7 an. Eine einfache Methode, dies zu erreichen, wird in [84] vorgeschlagen und funktioniert wie folgt: Für jeden Punkt aus O_e wird ein Knoten zu einem Graphen G hinzugefügt. Zwei Knoten erhalten eine Kante, wenn mindestens einer der

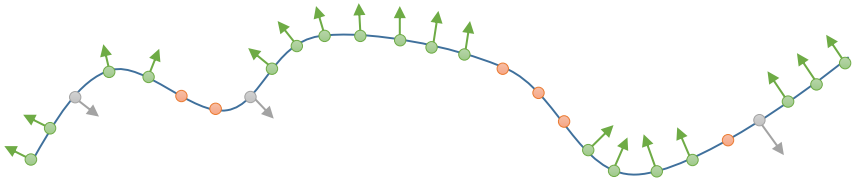


Abbildung 3.7.: Oberflächennormalen können komplett fehlen (orange) oder falsch orientiert sein (grau).

beiden korrespondierenden Punkte den jeweils anderen als einen seiner k Nachbarn hat. Jede Kante zwischen zwei Punkten o_i und o_j erhält das Gewicht $w_{ij} = 1 - |\mathbf{n_i}^T \mathbf{n_j}|$, wobei $\mathbf{n_i}$ und $\mathbf{n_j}$ die Normalen der Punkte o_i und o_j sind. Nun wird der minimale Spannbaum von G berechnet und die Normalenorientierung ausgehend von einem Startpunkt durch diesen propagiert. Dies geschieht, indem für zwei Nachbarpunkte o_i und o_j, die Orientierung von $\mathbf{n_j}$ umgedreht wird, wenn $\mathbf{n_i}^T \mathbf{n_j} < 0$ gilt. Die Navigation ausschließlich auf Kanten des minimalen Spannbaums minimiert Fehler an scharfen Kanten, da nur entlang nahezu paralleler Tangentialebenen navigiert wird.
Neben den klassischen Methoden zur Schätzung von Oberflächennormalen existieren mittlerweile auch Arbeiten, die auf KNNs basieren [15, 104].

3.2.2.4. Registrierung

Wurde die Eingabepunktwolke O_e aus Teilen unterschiedlicher Herkunft zusammengesetzt, besteht die Möglichkeit, dass ungewollte Überlappungen und fehlerhafte Anschlüsse auftreten, wie es Abbildung 3.8 beispielhaft darstellt. Abhilfe schaffen dabei Methoden aus dem Bereich der Punktwolkenregistrierung. Dabei wird je nach Aufgabenstellung eine Euklidische oder eine affine Transformation T gesucht, die korrespondierende Punkte zweier Punktwolken O_0 und O_1 möglichst perfekt aufeinander abbildet. Formal muss das Optimierungsproblem

$$\underset{T}{\operatorname{argmin}} \sum_{(o_0,o_1)\in O_{01}} d(T(o_0), o_1) \tag{3.3}$$

gelöst werden, wobei O_{01} die Menge der Punktkorrespondenzen von Punkten aus O_0 und O_1 ist [91]. Sind die Punktkorrespondenzen nicht bekannt, müssen auch diese im Laufe des Verfahrens ermittelt werden.
Ein robustes Verfahren, das eine Euklidische Transformation für zwei Punktwolken ohne vorher bekannter Punktkorrespondenzen findet, ist das *Iterative Closest Point*-Verfahren (ICP) [9]. Es existieren eine Vielzahl von Varianten dieser Methode [140], unter anderem eine, die auf dem Levenberg-Marquardt-Algorithmus basiert (siehe Kapitel 2.4.2) [50]. Für eine Übersicht weiterer Verfahren zur Punktwolkenregistrierung sei auf [140, 190, 208] verwiesen.

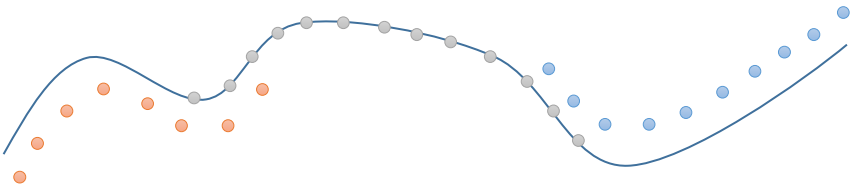


Abbildung 3.8.: Ist die Eingabepunktwolke aus verschiedenen Punktwolken (orange, grau, blau) zusammengesetzt, können Angleichungs- oder Abschlussfehler auftreten. Quelle: Abgeändert aus [16].

3.2.2.5. Auffüllung

Stammt eine Punktwolke aus einem 3D-Scanning-Vorgang, besteht die Möglichkeit, dass Regionen des zu scannenden Objekts nicht abgetastet wurden (siehe Abbildung 3.9). Das kann eine Reihe von Gründen haben: So können Triangulationsverfahren basierend auf passiver Stereoskopie keine spiegelnden Objektoberflächen abtasten (siehe Kapitel 3.2.1.2). Womöglich fehlen auch Objektbereiche, die durch das Scan-Gerät schwer erreichbar waren.

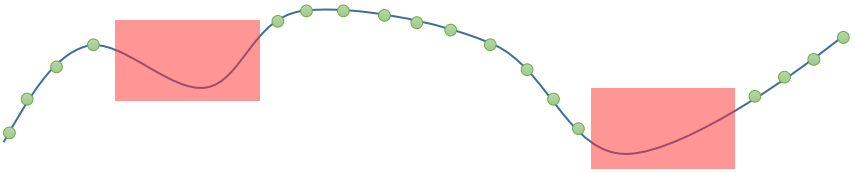


Abbildung 3.9.: Bereiche nicht abgetasteter Oberfläche (rot). Quelle: Abgeändert aus [16].

Um entstandene Abtastlücken automatisch zu füllen, bedarf es Auffüllungsalgorithmen. Eine Möglichkeit ist die Detektion und Einpassung von geometrischen Primitiven (siehe Kapitel 3.2.4) innerhalb der Eingabepunktwolke, deren Oberfläche dann in Bereichen fehlender Punktwolkenabdeckung abgetastet werden (siehe Kapitel 3.2.1.1) [157]. Dies geschieht unter der Annahme, dass das Objekt, dessen Punktwolke Abtastlücken enthält, über eine geschlossene Oberfläche verfügt, d.h. wasserdicht ist und aus geometrischen Primitiven aufgebaut wurde. Andere Verfahren nutzen Datenbanken bekannter Formen oder Formteile, die passend ausgesucht und in Abtastlücken eingepasst werden, was meist als kombinatorisches Optimierungsproblem formuliert und gelöst wird [176, 182]. Grundlegend für die Ergebnisqualität ist dabei die Qualität und Größe der verwendeten Datenbank. Davon abgesehen finden sich auch neuere Ansätze, die das Auffüllungsproblem mit KNNs lösen [76]. Weitere Verfahren werden in [16] gegenübergestellt.

3.2.3. Punktwolkensegmentierung

Je nach Größe und Dichte der Eingabepunktwolke kann eine Segmentierung derselben in einzelne Bereiche für die Robustheit und Effizienz des folgenden

Schritts *Primitivendetektion und -einpassung* (siehe Kapitel 3.2.4) notwendig oder zumindest hilfreich sein. Dabei versteht man in diesem Kontext unter Segmentierung die Aufteilung der Punktwolke in Regionen, die bzgl. einer zu wählenden Metrik homogen sind. Eine Metrik könnte dabei z.B. die Position eines Punktes oder dessen bekannte Oberflächennormale berücksichtigen. Wichtig ist, dass eine Segmentierung an unterschiedlichen Stellen der Prozess-Pipeline, auch wiederholt, durchgeführt werden kann. Im Folgenden wird eine grobe Einteilung verschiedener Verfahren vorgenommen, wie sie auch in [70] getroffen wurde.

3.2.3.1. Kantenbasierte Verfahren

Methoden dieser Kategorie identifizieren Grenzen möglicher Segmente als Kanten innerhalb der Eingabepunktwolke. Kanten sind Punktnachbarschaften, in denen sich Oberflächennormalen oder das abgeleitete Krümmungsverhalten benachbarter Punkte stark voneinander unterscheiden. Punkte werden dann einem durch ermittelte Kanten begrenzten Segment zugeordnet. Kantenbasierte Verfahren sind effizient, können aber meist nicht mit variierenden Punktdichten und häufig auftretenden Rauschartefakten umgehen [70].

3.2.3.2. Regionenwachsen

Regionenwachsen (engl. region growing) beschreibt eine Familie von Ansätzen zur Segmentierung, die sich grob in zwei Kategorien aufteilen lässt: Entweder werden ausgehend von einer Menge an Regionenstartpunkten Nachbarpunkte mit ähnlicher Charakteristik der jeweiligen Region hinzugefügt (unten nach oben) oder einer einzigen Startregion werden zu Beginn alle Punkte hinzugefügt und es folgt eine sukzessive Aufteilung dieser Region in Teilregionen basierend auf einem zu wählenden Kriterium (oben nach unten) [70]. Das zu lösende Problem ist damit entweder die Ermittlung guter Regionenstartpunkte und eines passenden Homogenitätskriteriums oder die Bestimmung optimaler Teilungsgrenzen. Erstmals Erwähnung fand ein solches Verfahren in [17]. Modernere Ansätze werden in [70] ausführlich diskutiert.

3.2.3.3. Clustering

Methoden zur Partitionierung von Datensätzen (siehe Kapitel 2.5.3.1) eignen sich ebenfalls zur Segmentierung von Punktwolken. Dabei stellt jeder Merkmalsvektor des zu clusternden Datensatzes ein Punkt der Punktwolke dar und kann neben Position und Oberflächennormale auch andere, dem Punkt zugeordnete Attribute enthalten.

3.2.3.4. Modelleinpassung

Aus der Beobachtung, dass natürliche Objekte oft aus Teilen einfacherer Struktur aufgebaut sind, lässt sich eine weitere Form der Segmentierung ableiten,

bei der parametrisierte Teilobjekte in die Punktwolke eingepasst werden [70]. Jeder Punkt nahe eines eingepassten Teilobjekts wird diesem zugeordnet, was in einer Segmentierung der Eingabepunktwolke resultiert. Handelt es sich bei den Teilobjekten um geometrische Primitive (Zylinder, Kugeln, Kegel, etc.), lassen sich dazu Methoden aus dem folgenden Schritt *Primitivendetektion und -einpassung* einsetzen (siehe Kapitel 3.2.4).

3.2.4. Primitivendetektion und -einpassung

Dieser Schritt hat zum Ziel, eine Menge von Primitiven H und deren Parameter aus der aufbereiten Eingabepunktwolke O_a oder aus deren Segmente zu extrahieren. Dabei trennen manche Methoden die Detektion des Primitiventyps von der Einpassung explizit, bei anderen ist beides in einem Verfahren implizit vereint. Wieder andere Techniken decken nur die Primitiveneinpassung ab.
Grundsätzlich lassen sich vier Problemklassen unterscheiden (wie z.B. in [12], mit aufsteigendem Schwierigkeitsgrad):

- *Eine-Klasse-Eine-Instanz* (EKEI): Es wird nach genau einem Primitiv bekannten Typs gesucht.
- *Mehrere-Klassen-Eine-Instanz* (MKEI): Es wird nach genau einem Primitiv gesucht, das unterschiedlichen Typs sein kann.
- *Eine-Klasse-Mehrere-Instanzen* (EKMI): Es wird nach mehreren Primitiven gleichen Typs gesucht.
- *Mehrere-Klassen-Mehrere-Instanzen* (MKMI): Es wird nach mehreren Primitiven unterschiedlichen Typs gesucht.

Im Folgenden werden verschiedene Verfahrenskategorien eingeführt, wobei sich eine detaillierte Diskussion entsprechender Verfahren im dafür passenden Kapitel zu verwandten Arbeiten befindet (siehe Kapitel 4.3).

3.2.4.1. Stochastische Methoden

Wie in [94] eingeführt, lassen sich Methoden, die in Teilen auf zufällige Ereignisse oder statistische Größen aufbauen, als stochastische Methoden zusammenfassen. Die für diese Arbeit relevanten stochastischen Methoden verwenden das *Random Sample Consensus* (RANSAC)-Prinzip, welches auf der Ausführung folgender Schritte basiert:

1. Entnehme n Punkte aus der Eingabepunktwolke. Für eine Kugel wäre z.B. $n = 4$, da sich eine Kugel mit vier Punkten eindeutig beschreiben lässt.

2. Zähle alle Punkte aus der Eingabepunktwolke, die innerhalb eines bestimmten Abstands ϵ zur Oberfläche des durch die n gezogenen Punkte beschriebenen Primitivs liegen.

3. Führe Schritt 1 und 2 m-mal (m ist ein benutzerdefinierter Parameter) aus.

4. Wähle jenes Primitiv aus, dass für Schritt 2 die größte Punktmenge auf seiner Oberfläche vereinen kann.

5. Wenn mehrere Primitive erkannt werden sollen: Entferne alle Punkte, die auf dem aktuell eingepassten Primitiv liegen von der Eingabepunktwolke und fahre mit Schritt 1 fort.

Verfahren dieser Kategorie sind robust gegen Ausreißer und Datensätze mit Rauschartefakten, haben jedoch Nachteile impliziert durch die gegebene Stochastizität und der damit verbundenen Varianz in den Ergebnissen für ein und dieselbe Eingabepunktwolke. Primitive unterschiedlichen Typs können detektiert werden, indem Schritt 1-3 für verschiedene Parametrisierungen (z.B. für Kugel, Zylinder oder Kegel) angewandt wird. Gewählt wird dann das beste Ergebnis. Die Detektion des Primitiventyps erfolgt also implizit. Mehrere Primitive können ebenfalls erkannt werden, indem das Verfahren wiederholt wird und Punkte bereits erkannter Primitive aus der Eingabepunktwolke entfernt werden. Damit ist RANSAC in der Lage, das MKMI-Problem zu lösen. Eine in dieser Arbeit verwendete Variante des RANSAC-Prinzips (siehe Kapitel 4.4.4), welche auf den Anwendungsfall der Primitiveneinpassung optimiert ist, findet sich in [158].

3.2.4.2. Parameterraummethoden

Parameterraummethoden arbeiten nicht auf dem Raum der Punkte der Punktwolke, sondern auf dem Raum der Parameter des zu erkennenden Primitiventyps. Für die Kugel ist das beispielsweise der Parameterraum $\mathbb{R}^4$, da eine Kugel mit vier Parametern (Zentrum $z \in \mathbb{E}^3$, Radius $r \in \mathbb{R}$) eindeutig beschrieben werden kann.
Der bekannteste Vertreter der Parameterraum-Methoden ist die sog. *Hough*-Transformation [40]. Folgende Schritte werden dabei durchgeführt:

1. Diskretisiere den für den gewählten Primitiventyp zur eindeutigen Beschreibung notwendigen Parameterraum in Zellen fixer Größe (siehe Voxel-Repräsentation, Kapitel 2.3). Damit wird der diskrete Parameterraum durch eine $m_1 \times ... \times m_n$ Matrix M repräsentiert, wobei $m_i, i \in \{1, .., n\}$ die Zellenanzahl in Parameterdimension i des n-dimensionalen Parameterraums darstellt.

2. Für jeden Punkt der Eingabepunktwolke: Ermittle alle diskreten Parameterkombinationen, für die der Punkt auf der Oberfläche des durch die

jeweilige Parameterkombination beschriebenen Primitivs liegt und inkrementiere den Zähler der dazugehörigen Zelle in M.

3. Ermittle alle Zellen, deren Zähler maximal ist. Die dazugehörigen Parameterkombinationen beschreiben die in der Punktwolke erkannten und eingepassten Primitive.

Die *Hough*-Transformation in ihrer ursprünglichen Form ist einfach zu implementieren und zu parallelisieren, hat jedoch eine Laufzeitkomplexität von $O(m_1 \times ... \times m_n \times |O_a|)$, was sie für einen praktischen Einsatz in höherdimensionalen Parameterräumen ungeeignet macht. Zudem kann die Diskretisierung des Parameterraums mit einem Verlust der Einpassgenauigkeit verbunden sein. Wichtig ist auch, dass der Parameterraum und damit der einzupassende Primitiventyp im Vorfeld bekannt sein muss. Ansonsten muss das Verfahren für mehrere Parameterräume ausgeführt und dann die gesamt-beste Kombination ausgewählt werden. Das Verfahren löst also das EKMI-Problem direkt, wohingegen das MKMI-Problem mit dem Umweg über mehrere, iterativ analysierte Parameterräume zu lösen ist. Für eine Diskussion von Erweiterungen der ursprünglichen Methode siehe [16].

3.2.4.3. Ausgleichsverfahren

Ausgleichsverfahren eignen sich zur Lösung von überbestimmten Optimierungsproblemen, wie sie für das Einpassen von Primitiven in Punktwolken formuliert werden können. Nimmt man einen n-dimensionalen Parameterraum an und Primitive, die durch eine mit den Parametern $\theta \in \Theta$ parametrisierte VDF F beschrieben werden, so muss das Optimierungsproblem

$$\underset{\theta \in \Theta}{\operatorname{argmin}} \sum_{o \in O_a} |F_\theta(o)| \tag{3.4}$$

mit Methoden der Ausgleichsrechnung gelöst werden (siehe Kapitel 2.4.2). Man erhält damit diejenigen Parameter θ des Primitivs, dessen Oberfläche den geringsten akkumulierten Abstand zu den Punkten der Eingabepunktwolke aufweist.
Wichtig zu beachten ist hierbei, dass bereits eine Segmentierung der Eingabepunktwolke vorliegen muss, die für jedes Segment nur noch genau ein Primitiv vorsieht. Ansonsten ist nicht klar, welche Punkte für den Einpassprozess gewählt werden sollen. Zudem muss eine Aufbereitung der Eingabepunktwolke Ausreißer und Rauschartefakte entfernt haben, da sonst das Ergebnis unter der mangelnden Robustheit der Methode leidet. Weiterhin detektieren Einpassmethoden basierend auf Ausgleichsverfahren nicht den Primitiventyp, welcher somit vor Ausführung bekannt sein muss oder iterativ über die Verwendung mehrerer Parametersätze und Auswahl des besten ermittelt werden muss (ähnlich RANSAC). Damit wird in einfachster Ausbaustufe nur das EKEI-Problem gelöst.

3.2.4.4. Maschinelles Lernen

Für die Detektion und Einpassung von Primitiven eignen sich Verfahren des überwachten Lernens (siehe Kapitel 2.5.2). Wird nur das Detektionsproblem betrachtet, also die Zuordnung eines Primitiventyps zu jedem Punkt oder Segment der Eingabepunktwolke, bietet sich die Formulierung als Klassifikationsaufgabe an. Soll hingegen zusätzlich eine Menge von Primitiven und deren Parameter aus einer Eingabepunktwolke prädiziert werden, handelt es sich um eine Regressionsaufgabe. Grundsätzlich ist es damit je nach eingesetztem Verfahren möglich, das MKMI-Problem zu lösen [106].
Wichtig ist die Verfügbarkeit eines angemessen großen und möglichst diversen Datensatzes, was die Parameter der in ihm enthaltenen Primitiven angeht. Zudem kommt der Generalisierbarkeit des gewählten Modells große Bedeutung zu. Eine weitere Schwierigkeit ergibt sich in der Repräsentation des Eingabedatensatzes als Punktwolke. Sie muss permutationsinvariant sein, d.h. die Reihenfolge der Punkte innerhalb der Punktwolke (gespeichert als Liste) darf keine Rolle spielen; jede Permutation der Punktliste soll auf dieselbe Repräsentation abgebildet werden. Zudem muss sie invariant bzgl. affiner Transformationen sein. So sind z.B. Lage, räumliche Ausdehnung und Orientierung der Punktwolke irrelevant. Außerdem müssen lokale Strukturmerkmale, die sich aus Punktnachbarschaften ergeben, Berücksichtigung finden.
Eine ausführliche Diskussion von überwachten Lernmethoden zur Detektion und Einpassung von Primitiven findet sich in Kapitel 4.3.

3.2.5. Konstruktionsbaumsynthese

Mit einer erzeugten und aufbereiteten Punktwolke O_a (siehe Kapitel 3.2.1 und Kapitel 3.2.2) und einer Menge von Primitiven H mit bekannten Parametern (siehe Kapitel 3.2.3 und Kapitel 3.2.4) lässt sich nun ein KB synthetisieren, welcher den durch die Punktwolke O_a approximativ dargestellten Festkörper mit Punktmenge S (auf den Index m wird im weiteren Verlauf zur besseren Lesbarkeit verzichtet) so exakt wie möglich repräsentiert. Im Folgenden werden verschiedene Verfahrenskategorien für diesen Schritt vorgestellt. Eine detaillierte Diskussion verwandter Arbeiten findet sich in Kapitel 5.3.

3.2.5.1. Kanonische Schnitte

Ein KB ist in *Disjunktiver Normalform* (DNF), wenn er aus Gruppen von mit $\cap^*$ verknüpften Primitiven oder deren Komplementen aus H besteht und diese Gruppen untereinander mit $\cup^*$ verknüpft sind. Die Gruppen werden im Weiteren als Implikanten bezeichnet. Beinhaltet jeder Implikant eines Ausdrucks in DNF alle Primitive aus H oder deren Komplement, handelt es sich um eine *Kanonische Disjunktive Normalform* (KDNF). Derartige Implikanten werden als Fundamentalprodukte oder kanonische Schnittterme bezeichnet [172] (siehe Abbildung 3.10).

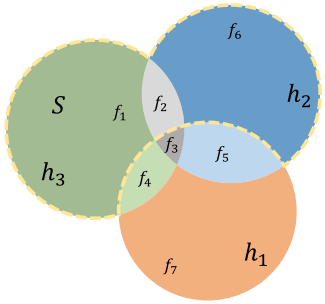


Abbildung 3.10.: Alle für $H = \{h_1, h_2, h_3\}$ möglichen Fundamentalprodukte $F = \{f_1, ..., f_7\}$. Damit lässt sich S (orange gestrichelt) durch den KB $f_1 \cup^* f_2 \cup^* f_6$ darstellen.

Die naheliegendste Methode zur Erzeugung eines KBs aus einer Menge von Primitiven H und einer Punktwolke O_a ist die Erzeugung aller mit den Primitiven aus H möglichen Fundamentalprodukte und die Selektion aller Fundamentalprodukte, die sich räumlich innerhalb der Punktwolke O_a und somit innerhalb des Festkörpers befinden. Die ermittelten Fundamentalprodukte werden mit dem $\cup^*$-Operator verknüpft. Nachteilig bei diesem Verfahren ist die Größe des resultierenden KBs, wie dessen ausführliche Analyse in Kapitel 5.4 zeigt.

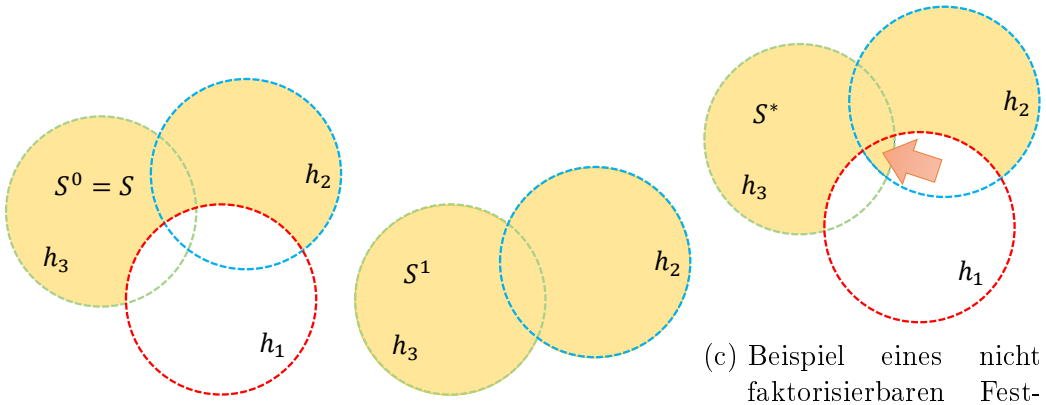
3.2.5.2. Dekomposition

Bei der Dekomposition, wie sie in [172] beschrieben wird, wird davon ausgegangen, dass die meisten oder gar alle Primitive aus H entweder vollständig innerhalb der Punktmenge S des zu repräsentierenden Festkörpers (im Folgenden: innenliegend) oder vollständig außerhalb derselben liegen (im Folgenden: außenliegend). Mit einer Sequenz derartiger Primitive $D = (d_1, ..., d_n)$ aus H der Länge n, die auch dominante Primitive bzw. dominante Halbräume genannt werden, lässt sich nun $S^0 = S$ mit $O^0 = O_a$, $H^0 = H$ und $D^0 = D$ rekursiv faktorisieren:

$$S^i = ((...(S^{i+1} \oplus |d_1^i|) \oplus ...) \oplus |d_{n-1}^i|) \oplus |d_n^i|, \tag{3.5}$$

wobei i die Rekursionstiefe darstellt und $\oplus$ entweder $\cup^*$ ist, wenn das folgende dominante Primitiv vollständig in S^i liegt oder $-^*$, wenn es vollständig außerhalb liegt. S^{i+1} ist die nach dem Dekompositionsschritt übriggebliebene Punktmenge des Restfestkörpers. H^{i+1} ist die Menge der Primitive H^i ohne die faktorisierten dominanten Primitive aus $D^i = (d_1^i, ..., d_n^i)$ (Die Sequenz D^i ist dabei nach Typ des dominanten Primitivs sortiert: Erst alle innenliegenden, dann alle außenliegenden). O^{i+1} besteht aus der Punktwolke O^i ohne die Punkte, die zu den faktorisierten dominanten Primitiven aus D^i gehören.
Dieses Verfahren lässt sich nun rekursiv auf S^{i+1} anwenden. Es werden also alle dominanten Primitive in H^{i+1} bzgl. S^{i+1} gesucht und die Dekomposition auf S^{i+1} angewandt. Der Algorithmus terminiert, wenn entweder S^{i+1} leer ist

oder keine weiteren dominanten Primitive existieren. Ist S^{i+1} leer, so lässt sich aus der Dekomposition direkt ein KB ableiten, der optimal hinsichtlich der Größe ist, da jedes Primitiv aus H exakt einmal als Operand verwendet wird [172]. Abbildung 3.11a und 3.11b zeigen den Ablauf anhand eines Beispiels.
Nicht immer aber ist S^i mit dem gegebenen Primitiven H^i vollständig faktorisierbar, S^{i+1} ist am Ende also nicht leer (siehe Abbildung 3.11c). Dann müssen zusätzliche Verfahren eingesetzt werden, um einen KB für die restlichen Primitive H^i zu finden (siehe Kapitel 3.2.5.1 oder Kapitel 3.2.5.3).
Entscheidend bei der Methode im Einsatzfeld dieser Arbeit ist die robuste und effiziente Bestimmung der dominanten Primitive, also die exakte Beantwortung der Frage, ob ein Primitiv vollständig innerhalb oder vollständig außerhalb des Festkörpers mit Punktmenge S^i liegt. Da S^i nur approximativ durch die Punktwolke O_i beschrieben ist, muss diese Entscheidung auf Basis der enthaltenen Punkte getroffen werden. Je nach Güte der Eingabepunktwolke hinsichtlich Rausch- und Ausreißerartefakten kann dies unterschiedlich robust gelingen. In dieser Arbeit kommen neu konzipierte Varianten der Dekomposition bei der Synthese von KBs aus Punktwolken und bei der Optimierung von KBs zum Einsatz (siehe Kapitel 5 und Kapitel 6).



(a) $i = 0$.

(b) $i = 1$.

(c) Beispiel eines nicht faktorisierbaren Festkörpers.

Abbildung 3.11.: Der Festkörper mit Punktmenge S (orange) soll durch einen KB mit den Primitiven $H = \{h_1, h_2, h_3\}$ (rot, blau, grün) repräsentiert werden. Gestartet wird mit allen Primitiven aus H (a). Primitiv h_1 is dominant, damit lässt es sich zu $S^0 = S^1 -^* |h_1|$ faktorisieren. Nun sind h_2 und h_3 dominant (b) und können ebenfalls faktorisiert werden: $S^1 = (|h_2| \cup^* |h_3|)$. Da nun alle Primitive aus H faktorisiert wurden, terminiert der Algorithmus. Den Gesamtausdruck erhält man durch rekursives Einsetzen (S^1 in S^0): $S = (|h_2| \cup^* |h_3|) -^* |h_1|$. In c) wird ein Festkörper mit Punktmenge S^* dargestellt, der nicht faktorisierbar ist, da kein Primitiv dominant ist.

3.2.5.3. Multikriterielle Optimierung

Das Problem der Synthese von KBs aus Punktwolken lässt sich auch als multikriterielles kombinatorisches Optimierungsproblem formulieren (siehe Kapitel 2.4). Das ist insbesondere dann sinnvoll, wenn die Dekomposition nicht alle Primitive faktorisieren kann (siehe Kapitel 3.2.5.2). Der abzählbar unendlich große Suchraum Θ besteht dabei aus allen mit den Primitiven H und der Menge erlaubter Operatoren (z.B. den regularisierten Mengenoperatoren aus Kapitel 2.3.1.1) konstruierbaren Ausdrücken. Die zu minimierende Zielfunktion

$$F_{O_a,H}(\Phi) = \alpha \cdot Q_{O_a,H}(\Phi) + \beta \cdot T(\Phi) \tag{3.6}$$

bewertet die geometrische Passgenauigkeit (wie gut passt der durch den Ausdruck $\Phi \in \Theta$ und die Primitive H erzeugte Festkörper in die Punktwolke O_a?) mit der Abbildung $Q_{O_a,H} \colon \Theta \to \mathbb{R}$ sowie den Aufbau (z.B. die Anzahl benutzter Knoten oder die Baumtiefe) des KBs mit $T \colon \Theta \to \mathbb{R}$. Dabei sind α und β Gewichtungsparameter aus $\mathbb{R}$, die je nach Domäne zu setzen sind. Zur Wahrung der Konsistenz werden hier die Elemente des Suchraums Θ mit Großbuchstaben bezeichnet (Φ) und nicht mit θ (siehe Kapitel 2.4). Dabei wird dem aus [172] stammenden und in dieser Arbeit verwendeten Formalismus für die Bezeichnung von KBs gefolgt (siehe Kapitel 2.3.1.1).
Minimiert werden kann die Zielfunktion beispielsweise mit EAs (siehe Kapitel 2.4.3.2). Entsprechende Verfahren, wie z.B. die Arbeiten von Fayolle et al. [47], werden in Kapitel 5.3 genauer erläutert. In dieser Arbeit kommt u.a. ein angepasster EA zur Synthese von KBs mittels multikriterieller Optimierung zum Einsatz (siehe Kapitel 5.5).

3.2.5.4. Maschinelles Lernen

Eine weitere Möglichkeit zur Synthese von KBs aus Punktwolken besteht in der Nutzung von Methoden aus dem Bereich des MLs, hier sinnvollerweise überwachtes Lernen mit KNNs. Zu lösen sind dabei die permutations- und transformationsinvariante Kodierung der Punktwolke O_a (siehe Kapitel 3.2.4.4), die passende Darstellung des Ausgabeausdrucks und die Formulierung einer differenzierbaren Loss-Funktion, welche die geometrische Passgenauigkeit des prädizierten Ausdrucks bewertet (siehe Gleichung 3.6). Eine Beschreibung existierender Methoden findet sich in Kapitel 5.3.

3.2.6. Konstruktionsbaumoptimierung

Das CSG-Repräsentationsschema ist eindeutig, jedoch nicht einzigartig (siehe Kapitel 2.3.1.1). Damit ergibt sich die Möglichkeit, für einen Festkörper existierende KB-Ausdrücke hinsichtlich festgelegter Kriterien zu optimieren. Ein Kriterium kann hierbei z.B. die möglichst geringe Anzahl an Operationsknoten sein, um eine hohe Effizienz bezüglich Editierbarkeit und Verarbeitung zu ermöglichen. Kapitel 6 widmet sich diesem Problembereich ausführlich. Im

Folgenden werden daher verschiedene Verfahrenskategorien zur Optimierung eines KBs Φ mit Primitiven H nur grundlegend eingeführt.

3.2.6.1. Boolesche Minimierung

Auf Festkörpern lässt sich zusammen mit den regularisierten Mengenoperatoren $\cup^*$, $\cap^*$ und $\setminus^*$ eine Boolesche Algebra definieren (siehe Kapitel 2.3). Dies ist jedoch nur möglich, wenn man Festkörper unendlicher Ausdehnung zulässt, da endliche Festkörper nicht abgeschlossen bzgl. des $\setminus^*$-Operators sind. In Tabelle 3.1 werden die Booleschen Algebren von KB- und logischen Ausdrücken dargestellt. Dieser Zusammenhang lässt eine Nutzung von Algorithmen zur Minimierung logischer Ausdrücke für das Problem der Größenoptimierung von KB-Ausdrücken zu. Dafür geeignete Methoden sind beispielsweise der *Quine-*

Boolesche Algebra	**Festkörper**	**Logischer Ausdruck**
Element	Festkörperinstanz	$x \in \{0, 1\}$
Addition	$\cup^*$	Disjunktion ($\vee$)
Multiplikation	$\cap^*$	Konjunktion ($\wedge$)
Komplement	$\setminus^*$	Negation ($\neg$)
Inklusion	$\subset$	Implikation ($\rightarrow$)
0-Element	$\emptyset$	0
1-Element	W ($\mathbb{E}^3$)	1

Tabelle 3.1.: Boolesche Algebren für Festkörper und logische Ausdrücke im direkten Vergleich. Quelle: [172].

McCluskey-Algorithmus [119] oder das *Espresso*-System [22]. Das in Kapitel 6 vorgestellte System nutzt die genannten Algorithmen. Wichtig ist, dass es sich hierbei um DNF-Optimierer handelt, die Eingabe also in DNF-Form erfolgt. Damit ist keine absolut minimale Ausgabe garantiert [172].

3.2.6.2. Dekomposition

Die Technik zur Synthese von KBs per Dekomposition (siehe Kapitel 3.2.5.2) lässt sich auch auf das Problem der Optimierung von KBs hinsichtlich der Anzahl verwendeter Knoten übertragen. Da der Festkörper direkt und exakt über den Eingabeausdruck Φ definiert und nicht über eine Punktwolke approximiert wird, vereinfacht sich die Ermittlung der dominanten Primitive D^i sowie die Bestimmung des Restfestkörpers S^{i+1}. Die im Rahmen dieser Arbeit entstandene Optimierungs-Pipeline nutzt die Dekomposition als einen ihrer Prozessschritte (siehe Kapitel 6.4).

3.2.6.3. Multikriterielle Optimierung

Wie für die Dekomposition gilt auch für Verfahren basierend auf kombinatorischer Optimierung, dass sich Methoden aus der Synthese (siehe Kapitel 3.2.5.3)

auch zur Optimierung von KBs eignen. Dazu werden die Gewichte α und β in Gleichung 3.6 so angepasst, dass der resultierende Baum immer eine perfekte Passgenauigkeit aufweist. So wird gewährleistet, dass der optimierte KB den exakt gleichen Festkörper wie der Eingabebaum repräsentiert. Weiterhin muss die geometrische Passgenauigkeit nicht anhand der Eingabepunktwolke gemessen werden, sondern kann direkt mit der durch Φ induzierten Punktmenge verglichen werden. Zudem wird das Lösungsverfahren (z.B. ein EA) mit dem Eingabeausdruck Φ und Primitiven H initialisiert, was meist eine schnellere Konvergenz mit sich bringt.
Gegenüber der Dekomposition ergibt sich hierbei der Vorteil, dass nicht nur auf die Anzahl von Baumknoten optimiert wird, sondern jede Eigenschaft des Baumes Ziel der Optimierung sein kann. Zudem gilt der für die Dekomposition aufgeführte Nachteil, dass nicht für alle Festkörper mit einer bestimmten Menge von Primitiven eine vollständige Dekomposition möglich ist, nicht für die multikriterielle Optimierung (siehe Kapitel 3.2.5.2 und Abbildung 3.11c).

3.2.6.4. Manuelle Editierung

Neben den bereits aufgeführten automatischen Verfahren zur Optimierung von KB-Ausdrücken ist es auch möglich, menschliche Eingaben zu erlauben, um mithilfe von Expertenwissen manuelle Anpassungen am Ausdruck vorzunehmen. Dies ist besonders interessant, da sich die CSG-Repräsentation grundsätzlich als sehr intuitiv und verständlich darstellt und damit eine effiziente manuelle Editierung vereinfacht wird. Beispielsweise wird in [207] ein *Virtual Reality* (VR)-System vorgestellt, welches die Manipulation von KB-Festkörpern mithilfe von Handgesten ermöglicht. Auch in dieser Arbeit wird ein solches System vorgestellt (siehe Kapitel 6.6), das zusätzlich Sprachkommandos zur Interaktion nutzt.

3.3. Zusammenfassung

In diesem Kapitel wurde die Problemstellung detailliert beschrieben sowie das für die gesamte Arbeit strukturgebende Pipeline-Modell entwickelt. Diese Prozess-Pipeline dient dabei als grobe Lösungsskizze für das in dieser Arbeit behandelte Problem. Jedes der drei identifizierten Teilprobleme wurde detailliert beleuchtet und dabei vorhandene Lösungsstrategien kategorisiert und beschrieben. Die nun folgenden drei Inhaltskapitel 4, 5 und 6 orientieren sich an diesem Modell. Sie beschreiben dabei im Gesamten eine neuartige Lösung aller drei Teilprobleme mittels eines Systems zur Synthese und Optimierung von KBs aus unstrukturierten räumlichen Daten repräsentiert durch Punktwolken.

4. Extraktion Geometrischer Primitive

Alles in der Natur modelliert sich nach Kugel, Kegel und Zylinder. Man muss aufgrund dieser einfachen Formen Malen lernen, dann wird man alles machen können, was man will.

—Paul Cézanne, 1904

Dieses Kapitel widmet sich dem automatischen Auffinden geometrischer Primitive, wie sie unter anderem im Eingangszitat Paul Cézannes Erwähnung finden, in einer Menge von räumlichen Messpunkten. Mit Auffinden ist, präziser formuliert, die Ermittlung von Anzahl, Art und Parameter der in einer Punktwolke enthaltenen speziellen Festkörper gemeint. Zu diesem Zweck wird ein System vorgestellt, welches Festkörper, wie Kugeln, Zylinder, Quader und sogar allgemeine konvexe Polytope aus Punktwolken extrahieren kann.
Dabei ist das Kapitel wie folgt aufgebaut: Zuerst werden in Kapitel 4.1 bereits vorveröffentlichte Inhalte diskutiert. Darauffolgend wird die zu lösende Problemstellung in Kapitel 4.2 motiviert und präzisiert. Kapitel 4.3 widmet sich dann einer detaillierten Analyse mit der Problemstellung verwandter Arbeiten. Das Herzstück des Kapitels bildet die umfassende Erläuterung der Prozess-Pipeline des vorgestellten Systems (siehe Kapitel 4.4). Da das dort beschriebene Basissystem noch keine effiziente Unterstützung für die Extraktion arbiträrer konvexer Polytope enthält, ist die Beschreibung der entsprechenden Erweiterung Teil eines separaten Kapitels (siehe Kapitel 4.5). In Kapitel 4.6 werden umfassende Evaluationsergebnisse zusammengetragen, diskutiert und interpretiert. Eine Zusammenfassung der Kerninhalte des Kapitels findet sich in Kapitel 4.7.
Zur Einordnung dieses Kapitels in den Gesamtkontext der Arbeit ist wichtig zu erwähnen, dass das hier vorgestellte System grundlegend für die Lösung des übergeordneten Problems der Synthese von Konstruktionsbäumen aus Punktwolken ist. Damit wird eine Strategie zur Lösung von *Problem 1* beschrieben, wie es in Abbildung 3.1 dargestellt wird.

4.1. Vorveröffentlichungen

Die in diesem Kapitel vorgestellten Inhalte wurden in mehreren Teilen bereits vom Autor veröffentlicht. Die Evaluation passender Netzarchitekturen für die Segmentierung von Punktwolken nach Primitiventyp wurde bereits in [54] ver-

öffentlicht. Gleiches gilt für das Konzept zur automatischen Generierung von Trainingsdatensätzen bestehend aus Punktwolken. Die Idee der Punktwolkensegmentierung basierend auf DBSCAN in Kombination mit prädizierten Primitiventypzuordnungen zur Verbesserung der Robustheit klassischer Primitiveneinpassungsverfahren stammt vom Autor und wurde bereits in [55] veröffentlicht. Ebenfalls vom Autor stammt das Konzept zur Zusammenfassung von eingepassten Ebenen zu konvexen Polytopen, welches zuerst in [55] veröffentlicht wurde. Die Erweiterung dieser Technik (siehe Kapitel 4.5) mittels einer zusätzlichen Segmentierung der Eingabepunktwolke in schwach-konvexe Cluster, wurde bereits in [51] veröffentlicht. Für die grundlegende Idee ist ebenfalls der Autor verantwortlich. Weiterhin ist das Verfahren zur Ermittlung von Zylinderdeckflächen (siehe Kapitel 4.4.4.3) aus [56] entnommen und basiert dabei auf einem Konzept des Autors. Die Evaluation in Kapitel 4.6 wurde für diese Arbeit komplett neu durchgeführt und enthält einen zusätzlichen, bisher nicht veröffentlichten Datensatz (Modell M12), welcher auf Basis photogrammetrischer Methoden erzeugt wurde (siehe Kapitel 3.2.1.2). Eine Ausnahme bildet hier die Evaluation des trainierten KNNs zur Punktwolkensegmentierung (siehe Kapitel 4.6.3.2), welche [56] entnommen ist. Die Abbildungen 4.5 und 4.28 a)-e) stammen aus [55], die Abbildungen 4.13, 4.15a, 4.15c und 4.19 aus [51]. Aus [56] wurden die Abbildungen 4.19a, 4.19b, 4.2a, 4.2b, 4.6c und 4.6d übernommen.

4.2. Motivation und Zielsetzung

Das generelle Motiv für eine Extraktion von geometrischen Primitiven aus einer Punktwolke ist die Gewinnung von topologischer und geometrischer Information aus einem Messdatensatz, wie er z.B. durch Verfahren, die in Kapitel 3.2.1.2 beschrieben werden, gewonnen wurde. Dieser Prozess kann Teil eines geometrischen Produktnachbaus sein, wie er in alternativen Ansätzen zur rechnergestützten Produktentwicklung zum Einsatz kommt [180]. Im Kontext dieser Arbeit ist das hier vorgestellte System jedoch eher als eine Art Zwischenschritt hin zur Synthese einer vollständigen KB-Repräsentation zu sehen (siehe Kapitel 5).
Die Zielsetzung des Systems beschränkt sich dabei auf die Rekonstruktion von Primitiven, deren Oberfläche durch Quadriken beschrieben werden können (siehe Kapitel 2.3.3.2). Diese Beschränkung wird von der Tatsache relativiert, dass 85% aller industriell gefertigten, mechanischen Bauteile mithilfe von Ebenen, Kegeln, Kugeln und Zylinder beschreibbar sind [129]. Berücksichtigt man zusätzlich allgemeine Tori, die ebenfalls durch Quadriken beschrieben werden können, steigt die Zahl auf 95% [149]. Auch in der Architektur findet sich in vielen Stilrichtungen eine Dominanz einfacher Formen, wie es zum Beispiel der bereits in der Einleitung erwähnte Bauhaus-Stil kultiviert. Möchte man noch mehr Ausdrucksstärke und arbiträr gekrümmte Oberflächen zulassen, muss z.B. auf eine NURBS-Repräsentation zurückgegriffen werden (siehe Kapitel

2.3.3.3). Deren Einpassung in eine Punktwolke ist im allgemeinen Fall jedoch äußerst komplex [25, 117].
Quadriken beschreiben unbeschränkte Festkörper, also Festkörper mit potentiell unendlichem Volumen. Hier haben z.B. Zylinder und Kegel entsprechend unendliche Höhe. Gleiches gilt für Ebenen, welche einen unendlich großen, linear-separierten Halbraum des $\mathbb{E}^3$ beschreiben. Das entspricht nicht der für diese Arbeit gewählten Definition eines Festkörpers (siehe Kapitel 2.2). Im Unterschied zu Methoden, die durch Quadriken beschriebene Begrenzungsflächenmodelle aus Punktwolken erzeugen, geht das hier vorgestellte System einen Schritt weiter, indem erkannte Primitive vollständig durch beschränkte Festkörper beschrieben werden. Das System schätzt dazu die passende Höhe eines Zylinders und setzt einzelne Ebenen zu Quadern bzw. zu arbiträren konvexen Polytopen zusammen.
Wie bereits erwähnt, kann nur durch die ausschließliche Verwendung von beschränkten Festkörpern als Primitive garantiert werden, dass ein KB, der diese Primitive zusammen mit den regularisierten Mengenoperatoren verwendet, ein physikalisch herstellbares Modell beschreibt. Damit ist ein sinnvoller Einsatz des Systems im oben erwähnten Einsatzgebiet innerhalb der Produktentwicklung gewährleistet. Um den Implementierungsaufwand zu beschränken, wurde auf die Unterstützung von Kegeln und Tori zugunsten einer Erkennung von konvexen Polytopen verzichtet. An geeigneter Stelle in der Konzeptbeschreibung wird jedoch auf die für eine Unterstützung notwendigen Schritte eingegangen.

4.3. Verwandte Arbeiten

Im Folgenden werden verwandte Arbeiten zur Rekonstruktion von Primitiven aus Punktwolken diskutiert. Dabei wird auf der Problemeinführung in Kapitel 3.2.4 inhaltlich aufgebaut und dementsprechend weiterführende Methoden kategorisiert nach Ergebnisgeometrie behandelt.

4.3.1. Einfache Primitive

Zu den einfachen Primitiven zählen die durch Quadriken darstellbaren, wie in Kapitel 2.3.3.1 beschrieben. Methoden zur Einpassung werden in Kapitel 3.2.4 erläutert. Hier relevant sind v.a. RANSAC-basierte Verfahren, wie z.B. das in [158] eingeführte, da ein Teil des hier vorgestellten Systems darauf beruht. Möchte man neben lokalen Einpassungen einzelner Primitive auch deren globale Ausrichtung zueinander optimieren (z.B. zur Entfernung kleiner Spalten zwischen eingepassten Primitiven), existieren Ansätze, wie der in [109] vorgestellte. Auch im Bereich des MLs gibt es Verfahren zur Einpassung einfacher Primitive, wie z.B. das in [106] beschriebene. Dabei wird ein KNN trainiert, das zur Kodierung von Eingabepunktwolken auf die *PointNet++*-Architektur

zurückgreift [142]. Unter den möglichen Primitiventypen finden sich Ebenen, Kugeln, Zylinder und Kegel. Im Vergleich zu dem in dieser Arbeit vorgestellten System werden keine konvexen Polytope unterstützt. Was beide Methoden jedoch eint ist die Verwendung von *PointNet++* zur Verarbeitung von Punktwolken.
Dabei ist zu beachten, dass einige Verfahren meist nur unbeschränkte Primitive einpassen können und damit z.B. ein Zylinder unendliche Höhe besitzt (z.B [158]). Das in dieser Arbeit vorgestellte System geht einen Schritt weiter, da alle resultierenden Primitive in ihrem Volumen beschränkt sind.

4.3.2. Polyeder

Einige Vorarbeiten extrahieren vereinfachte Polyeder aus Punktwolken oder detaillierten Dreiecksnetzen [13, 21, 45, 125]. Dabei sind derartige Methoden meist auf bestimmte Objekttypen spezialisiert, wie z.B. Gebäudemodelle [21, 131] oder mechanische Bauteile [13, 125]. Im Vergleich zu dem in dieser Arbeit vorgestellten System unterstützt keine dieser Arbeiten andere Primitiventypen als Ebenen bzw. Polyeder. Auch ist es nicht möglich, konvexe Polytope zu extrahieren, die das Modell in wiederverwendbare Subelemente unterteilen.
Zhong et al. stellt in [206] eine laufzeiteffiziente Methode vor, um ein konvexes Polytop direkt aus einer Punktwolke zu erzeugen. Ziel ist es jedoch nicht, das konvexe Polytop zu finden, welches sich möglichst genau in die Eingabepunktwolke einpasst, sondern dasjenige, welches möglichst keine Punkte der Punktwolke enthält. Ebenfalls verschieden zu dem in diesem Kapitel beschriebenen Ansatz ist die Beschränkung auf ein einzelnes konvexes Polytop, das so gefunden werden kann.
Im Bereich des MLs finden sich Arbeiten zur Extraktion von einfachen Quadern [181, 209] oder konvexen Polytopen [36]. Andere lösen nur das Problem der konvexen Segmentierung und nicht das der Einpassung entsprechender Polytope [61]. Damit sind diese Ansätze eher mit den in dieser Arbeit verwendeten Methoden zur konvexen Segmentierung vergleichbar [10, 93]. Systeme basierend auf ML unterliegen jedoch prinzipbedingten Restriktionen. So gestaltet sich die Generalisierbarkeit auf Modelle, die dem Trainingsdatensatz nicht ähnlich sind, meist als schwierig.

4.3.3. Arbiträre Begrenzungsflächenmodelle

In [175] wird ein Ansatz vorgestellt, mit dem neben einfachen Primitiven auch arbiträr gekrümmte Oberflächen, ähnlich den NURBS (siehe Kapitel 2.3.3.3), eingepasst werden können. Der Ansatz basiert auf KNNs, hat jedoch den Nachteil, dass resultierende Objekte nicht wasserdicht sind und damit nicht direkt für die Fertigung eines Echtweltobjekts geeignet sind. Zudem werden im Vergleich mit der hier vorgestellten Methode keine konvexen Polytope unterstützt. Gleiches gilt für die von Brujic et al. eingeführten Methode zur Einpassung von

NURBS-Oberflächen in Punktwolken, die auf einer effizienten Lösung der Formulierung des Problems als lineares Ausgleichsproblem beruht [25]. Auch in [117] wird zu diesem Zweck ein lineares Ausgleichsproblem gelöst, jedoch erfolgt in einem ersten Schritt die Ermittlung der optimalen Gewichte über eine symmetrische Eigenwertdekomposition.

4.3.4. Schablonen

Einige Ansätze passen vorgefertigte, parametrisierte CAD-Modelle in Punktwolken ein [11, 100]. Dabei ist die offensichtliche Beschränkung der Ausdrucksstärke auf eine initiale Menge derartiger Schablonenobjekte der größte Nachteil dieser Methodenfamilie. Vorteilhaft sind Methoden basierend auf Schablonen in Szenarien, in denen aus einer festen Menge von Geometrien ausgewählt werden kann. Die obere Schranke für die Modellkomplexität ist dabei abhängig von der Anzahl der Parameter, die für eine Schablone gefunden werden müssen sowie von der Anzahl unterschiedlicher Schablonen. Im einfachsten Fall zählen zu den zu findenden Parametern ausschließlich die Pose des Objekts und dessen Skalierung. Im Vergleich zu dem hier vorgestellten System, können Methoden basierend auf Schablonen insgesamt komplexere Objekte zusammenfügen, sind aber in ihrer Ausdrucksstärke beschränkt.

4.3.5. Zusammenfassung

Bei der Betrachtung verwandter Arbeiten wurde nach Primitivenart kategorisiert. Im Gesamten bleibt festzuhalten, dass der folgend vorgestellte Ansatz in einigen Aspekten einzigartig ist: Anders als Methoden zur Rekonstruktion einfacher Primitive werden zusätzlich auch konvexe Polytope unterstützt und zudem Strategien implementiert, die ausschließlich beschränkte Primitive als Ergebnis haben. Verglichen mit Verfahren zur Polyederdetektion und -einpassung ist es auch möglich, Kugeln und Zylinder zu rekonstruieren. Jedoch werden arbiträr gekrümmte Flächen nicht unterstützt. Aufgrund des hier vorgestellten erweiterbaren Prozessmodells ist es aber jederzeit möglich, neue Primitiventypen hinzuzufügen. Dies wird auch in der Zusammenfassung dieses Kapitels thematisiert (siehe Kapitel 4.7). Im Vergleich zu Techniken, die auf Schablonen basieren, ist das hier vorgestellte System flexibler bezüglich der Vielfalt rekonstruierbarer Objekte.

4.4. Konzept eines Systems zur Extraktion geometrischer Primitive

Im Folgenden wird ein System zur Extraktion geometrischer Primitive beschrieben, das die in Kapitel 4.2 definierten Zielsetzungen erfüllt. Kurz zusammengefasst sind diese:

- Als Eingabe dient eine Punktwolke, wie sie mit in Kapitel 3.2.1 beschriebenen Verfahren gewonnen werden kann. Dies umfasst explizit auch Punktwolken, die auf Basis von photogrammetrischen Methoden gewonnen wurden und damit systematische Fehler enthalten können.

- Zylinder, Kugeln und Ebenen sollen in der Punktwolke detektiert und deren Parameter geschätzt werden.

- Detektierte Primitive müssen als beschränkte Festkörper, d.h. als Festkörper mit endlich großen Volumen, repräsentierbar sein. Das bedeutet insbesondere, dass extrahierte Ebenen zu Quadern oder allgemein zu konvexen Polytopen kombiniert werden sollen.

- Insgesamt steht die Robustheit und die Rekonstruktionsqualität des Systems im Vordergrund. Das Laufzeitverhalten wird zwar ebenfalls betrachtet, spielt jedoch eine untergeordnete Rolle.

Die grundsätzliche Idee ist, ein bestehendes Verfahren zur Detektion- und Einpassung von Primitiven zu adaptieren (hier: RANSAC, siehe Kapitel 3.2.4.1) und durch einen zusätzlichen Segmentierungsschritt durchgeführt mit Techniken aus dem Bereich des MLs hinsichtlich Robustheit zu optimieren. Dies wird mit einem Verfahren verknüpft, welches aus extrahierten Ebenen Quader und allgemeine konvexe Polytope zusammensetzt. Die Zusammensetzung gelingt über die Formulierung des zugrundeliegenden Problems als ein kombinatorisches Optimierungsproblem mit der Menge der kombinatorisch möglichen Ebenenzusammensetzungen als Lösungsraum (siehe Kapitel 2.4). Gelöst wird das Problem durch die Anwendung eines EAs (siehe Kapitel 2.4.3.2). In einem abschließenden Schritt wird die Höhe der Zylinder über die Ausdehnung des jeweils assoziierten Punktwolkensegments approximiert.
Zentral für das Gesamtsystem ist dabei eine Prozess-Pipeline (siehe Abbildung 4.1), die die notwendigen Verarbeitungsschritte von der Eingabepunktwolke bis zur ausgegebenen Menge extrahierter Primitive umfasst und sich am allgemeinen Prozessmodell orientiert (siehe Kapitel 3.2 und Abbildung 3.1). Die folgenden Kapitel beschreiben die skizzierte Lösungsidee aufgeteilt in einzelne Prozessschritte.

4.4.1. Punktwolkenerfassung

Bei der Erfassung der vom System genutzten Punktwolken lassen sich zwei spezifische Anwendungsfälle unterscheiden. Zum einen werden gelabelte Punktwolken in Form von Trainingsdaten für den auf ML basierenden Segmentierungsschritt benötigt (siehe Kapitel 4.4.1.1). Zum anderen müssen für die Evaluation des Gesamtsystems möglichst diverse Datensätze zusammengestellt werden, die dann eine profunde Aussage über die Leistungsparameter des Systems zulassen (siehe Kapitel 4.4.1.2).

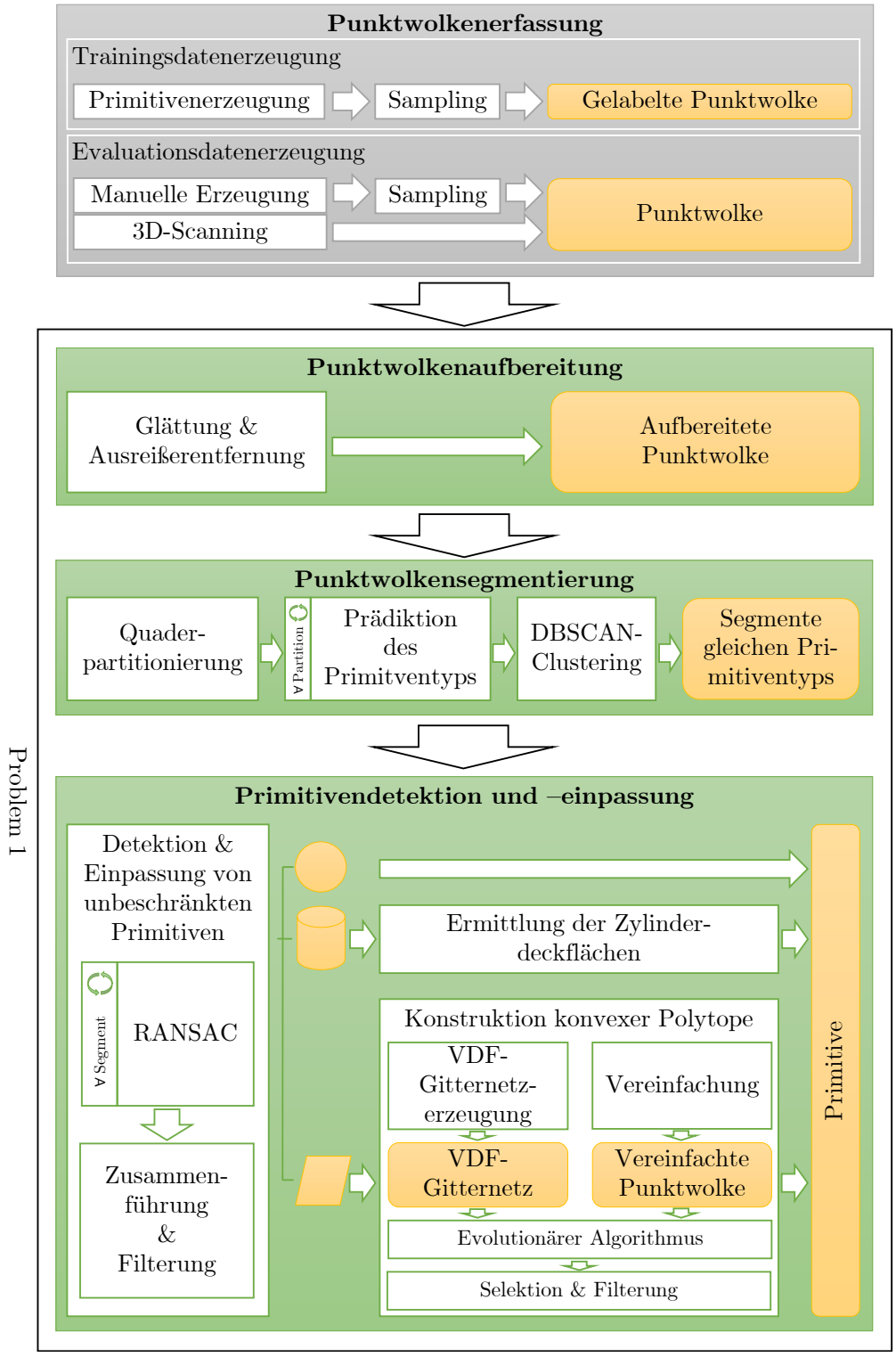


Abbildung 4.1.: Schematische Darstellung der Gesamt-Pipeline zur Extraktion geometrischer Primitive.

4.4.1.1. Trainingsdatenerzeugung

Die Punktwolkensegmentierung zur Erhöhung der Robustheit des Einpassungsverfahrens basiert auf der Prädiktion von Primitiventypen (Kugel, Fläche, Zylinder) für jeden Punkt der Eingabepunktwolke mittels eines KNNs. Die dafür notwendigen Trainingsdaten bestehen aus künstlich erzeugten Punktwolken, die jeweils durch Oberflächenabtastung einer Menge von generierten Primitiven gewonnen werden (siehe Abbildung 4.1, *Sampling*). Dabei ist für die spätere Prädiktionsrobustheit das Hinzufügen von künstlichem, normalverteiltem Rauschen zur durch den Abtastvorgang erzeugten Punktwolke wichtig.
Der entwickelte Generator zur Erzeugung von Primitiven (siehe Abbildung 4.1, *Primitivenerzeugung*) durchläuft dazu folgende Schritte:

1. 3D-Modelle in Form von Dreiecksnetzen, die z.B. öffentlich verfügbaren Datenbanken entstammen, werden abgetastet.

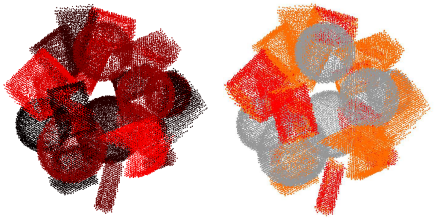
2. Die resultierende Punktwolke wird dann mit einem k-Means-Clustering (siehe Kapitel 2.5.3.1) in k Segmente zerlegt. Für jedes der k Punktwolkensegmente wird dessen Orientierung im Raum bestimmt. Dies wird mit der Durchführung einer Hauptkomponentenanalyse auf jedem Segment erreicht (siehe Kapitel 2.5.3.2).

3. In jedes Segment wird ein Primitiv zufälligen Typs (Kugel, konvexes Polytop, Zylinder) eingepasst. Wobei für das Zentrum des Primitivs der Mittelwert aller Segmentpunkte gewählt wird. Die Orientierung des Primitivs im Raum wird durch die ermittelte Segmentorientierung festgelegt. Die Ausdehnung des Primitivs (z.B. die Höhe eines Zylinders) ergibt sich aus der Varianz der Segmentpunkte entlang der durch die Orientierung gegebenen Koordinatenachsen.

Das vorgestellte Verfahren nimmt nur eine grobe Einpassung von Primitiven in eine Punktwolke vor. Dies ist an dieser Stelle jedoch erwünscht, da das Ziel keine exakte Einpassung, sondern ein möglichst diverser Datensatz gelabelter Punktwolken ist (siehe Abbildung 4.1, *Gelabelte Punktwolke*). Zu erwähnen ist, dass zwar Quader und allgemein konvexe Polytope generiert werden, der dazugehörige Primitiventyp jedoch immer die Ebene ist. Abbildung 4.2 zeigt ein Beispiel für eine auf diese Weise generierte Punktwolke. Für eine Unterstützung von Kegeln und Tori müsste auch der Generator entsprechende Primitive bzw. dessen Punktwolken erzeugen können.

4.4.1.2. Evaluationsdatenerzeugung

Passende Punktwolken für die Evaluation des Gesamtsystems werden auf zwei Arten erzeugt: Entweder wird die Oberfläche eines CAD-Modells abgetastet (siehe Abbildung 4.1, *Manuelle Erzeugung* bzw. *Sampling*) oder es wird ein real existierendes Objekt digitalisiert (siehe Abbildung 4.1, *3D-Scanning* und Kapitel 3.2.1.2).



(a) (b)

Abbildung 4.2.: a) Eine durch den Generator erzeugte Punktwolke. Die Einfärbung dient der Sichtbarmachung der verschiedenen Primitive. b) Dieselbe Punktwolke mit Einfärbung nach Primitiventyp (grau: Kugel, orange: Ebene, rot: Zylinder).

Konkret wird für die Digitalisierung eines Objekts auf ein quelloffenes System zurückgegriffen, welches auf passiver Stereoskopie (siehe Kapitel 3.2.1.2) zur Erzeugung eines 3D-Modells aus einer Serie von Fotografien basiert[1]. Dieses folgt grob folgenden Prozessschritten:

1. **Bilderzeugung:** Das Echtweltobjekt wird mit einer Digitalkamera aus verschiedenen Perspektiven fotografiert. Für ein gutes Endresultat ist darauf zu achten, dass Lichtverhältnisse sowie interne Kameraparameter konstant bleiben und dass die Oberflächen der zu rekonstruierenden Objekts diffus reflektierend und ausreichend texturiert sind.

2. **Merkmalsextraktion:** Mithilfe des SIFT-Algorithmus werden auf den Eingabebildern Merkmale detektiert [114].

3. **Bildabgleich:** Eingabebilder, die gleiche Regionen der aufgenommenen Szene enthalten, werden zu Bildpaaren gruppiert. Dabei kommt das Verfahren von Nister et al. [128] zum Einsatz.

4. **Merkmalsabgleich**: Ziel ist es, Merkmale in Bildpaaren zu finden, die den gleichen Punkt der Szene abbilden, sog. Korrespondenzpunkte. Dazu wird für jedes Bildpaar folgende Prozedur durchlaufen: Für jedes Merkmal in Bild 1 werden die zwei am besten übereinstimmenden Merkmale in Bild 2 gesucht. Um den Vorgang zu beschleunigen, werden effiziente Datenstrukturen zur Nächste-Nachbarn-Suche verwendet [123]. Abschließend werden die besten Korrespondenzpunkte per RANSAC-Verfahren ausgewählt.

5. **Structure-from-Motion:** Beginnend mit dem Bildpaar, welches die meisten robusten Korrespondenzen enthält, werden die internen und externen Parameter beider Kameras geschätzt und dann die entsprechenden dreidimensionalen Punkte zu den Korrespondenzpunkten per

[1] *Meshroom* Version 2020.1.1 entwickelt unter der Schirmherrschaft der *AliceVision Association*, `https://alicevision.org/`.

Triangulation erzeugt. Als nächstes werden alle Eingabebilder gewählt, dessen Menge an Korrespondenzpunkten, zu denen bereits dreidimensionale Punkte existieren, am größten ist. Für diese werden die zugehörigen externen Kameraparameter mit dem *Perspektivischen n-Punkt* (PNP)-Algorithmus [49] geschätzt und per nichtlinearer Optimierung verbessert. Nach einem abschließenden Optimierungsschritt (sog. Bündelausgleichung, engl. bundle adjustment [77]), der interne und externe Parameter aller Kameras verfeinert und ggf. Kameras aussortiert, wird das Verfahren für die nächstbesten Eingabebilder wiederholt.

6. **Multi-View-Stereo:** Basierend auf den rekonstruierten Kameraparametern wird nun ein Verfahren angewandt, das dichtgelagerte Rekonstruktionen der Szene aus verschiedenen Perspektiven zum Ergebnis hat und auf einer Idee von Collins et al. basiert [32]. Die resultierenden Rekonstruktionen werden im Anschluss geglättet und zu einem Dreiecksnetz zusammengeführt.

Aus dem auf diese Weise erzeugten Dreiecksnetzmodell wird eine Teilmenge der Eckpunkte mit assoziierten Normalenvektoren in eine Punktwolke überführt. Unabhängig von der Art der Erzeugung resultieren alle Ansätze damit in einer Punktwolke, die Punkte mit dreidimensionaler Koordinate und normiertem Normalenvektor enthält (siehe Abbildung 4.1, *Punktwolke*). Eine Normalenschätzung (siehe Kapitel 3.2.2.3) ist daher in keinem Fall notwendig.

4.4.2. Punktwolkenaufbereitung

Je nach Quelle können Punktwolken unterschiedliche Fehler oder Unvollständigkeiten aufweisen (siehe Kapitel 3.2.2). Alle erzeugten Eingabepunktwolken werden per Translation und Skalierung in den Einheitswürfel (Kantenlänge: 1, Würfelmittelpunkt im Ursprung) transformiert. Eine weitere Glättung findet nicht statt, da nachfolgende Prozessschritte hinsichtlich Rauschartefakten robust sind. Nur die per *3D-Scanning* erzeugten Punktwolken werden zusätzlich manuell zurechtgeschnitten und per FPS initial vereinfacht, um die ebenfalls rekonstruierten Stellflächen der Modelle sowie die Punktwolkengrößen zu reduzieren. Dies ist jedoch der einzige Bearbeitungsschritt, der in den Bereich der *Glättung & Ausreißerentfernung* fällt und zusammen mit den anderen Schritten der Aufbereitung in einer aufbereiteten Punktwolke O_a (siehe Abbildung 4.1, *Aufbereitete Punktwolke*) resultiert.

4.4.3. Punktwolkensegmentierung

Wie einführend erwähnt ist die Idee, die Robustheit eines klassischen Primitiveneinpassungsverfahrens wie RANSAC zu verbessern, indem die Eingabepunktwolke entsprechend segmentiert und damit das Einpassungsproblem vereinfacht wird. Dazu wird für jeden Punkt, neben den Punktkoordinaten und

Normalenvektoren, auch der Primitiventyp betrachtet und ein entsprechendes Clustering durchgeführt. Dabei wird der Primitiventyp eines Punkts mithilfe einer geeigneten KNN-Architektur prädiziert. Auf Basis einer Vorevaluation möglicher Kandidaten (siehe Kapitel 4.6.3.1) wurde zu diesem Zweck die *PointNet++*-Architektur ausgewählt [142].

PointNet++ wendet die Vorgängerarchitektur *PointNet*[141] rekursiv auf Punktpartitionen an. Die hierarchische Struktur ermöglicht eine bessere Erfassung von lokalen Punktwolkendetails sowie einen robusteren Umgang mit Dichtevariationen innerhalb der Punktwolke verglichen mit *PointNet*. Im Folgenden soll zuerst die *PointNet*-Architektur näher erläutert werden (siehe Abbildung 4.3).
Die wichtigsten Aspekte, die es bei der Anwendung von ML-Techniken auf Punktwolken zu berücksichtigen gilt, sind (siehe Kapitel 3.2.4.4): 1) Permutationsinvarianz, 2) Transformationsinvarianz und 3) die Erfassung lokaler Merkmale, die sich für jeden Punkt aus seinen Nachbarpunkten ergeben.
In der *PointNet*-Architektur wird Permutationsinvarianz über eine Max-Pooling Schicht erreicht (siehe Kapitel 2.5.2.2 und Abbildung 4.3). Das Problem der Transformationsinvarianz wird mittels zweier Strategien gelöst. Für die Transformationsinvarianz bzgl. Translation und Skalierung werden in einem Vorverarbeitungsschritt Eingabepunktwolken in den Einheitswürfel verschoben und entsprechend skaliert. Für die Transformationsinvarianz bzgl. Rotation wird ein Teilnetz T_x eingeführt, das für jede Eingabepunktwolke eine $x \times x$ Rotationsmatrix lernt (*T-Netz*, siehe Abbildung 4.3) und diese auf die Punkte der Eingabepunktwolke anwendet (*Matrixmultiplikation*, siehe Abbildung 4.3). Lokale Merkmale werden mittels mehrerer *Faltungsschichtgruppen* (*FSGs*, siehe Abbildung 4.3) pro Punkt aggregiert. In den FSGs werden dazu eindimensionale Faltungsschichten (siehe Kapitel 2.5.2.2) mit Filterkernelgröße 1×1 hintereinandergeschaltet. Dabei nutzen alle bis auf die letzte Faltungsschicht eine Rectifier-Aktivierungsfunktion.
PointNet ermöglicht die Klassifikation von n Punkten bezogen auf s Klassen. Im hier betrachteten Fall entspricht s der Anzahl der zu detektierenden Primitiventypen (3). Punktwolken unterschiedlicher Größen werden unterstützt, indem diese per FPS auf maximal n Punkte reduziert werden. Eine Darstellung der wichtigsten Elemente des KNNs findet sich Abbildung 4.3. Das darauf aufbauende *PointNet++* nutzt eine rekursive Partitionierungsstrategie:

FPS & Gruppierung. Zuerst werden per FPS n_{i+1} Punkte aus der Eingabepunktwolke extrahiert, die als Zentren der Partitionen fungieren, wobei alle n_i bis auf n_0 benutzerdefiniert sind. i ist die aktuelle Rekursionstiefe startend bei 0. $n = n_0$ ist die Größe der Eingabepunktwolke. Für jedes Zentrum werden dann maximal k Punkte der Eingabepunktwolke in einem benutzerdefinierten Radius selektiert. k ist ebenfalls frei wählbar.

PointNet. Für jede der n_{i+1} Nachbarschaftspunktmengen wird mittels eines *PointNets* ein Merkmalsvektor der Länge 1024 extrahiert (*globale Merkmale*, siehe Abbildung 4.3) und mit den aufsummierten Merkmalsvektoren der Länge c_i der vorhergehenden Rekursionstiefe konkateniert ($c_{i+1} = c_i + 1024$, wobei c_0 eine Länge von 0 hat). Wichtig hierbei ist, dass die Punkte der Nachbarschaftspunktmengen vor der Eingabe in das *PointNet* in ein lokales Koordinatensystem überführt werden, in dem die korrespondierenden Zentrumspunkte abgezogen werden. Als Resultat ergibt sich die Menge der n_{i+1} d-dimensionalen Zentrumspunkte, wobei für jeden Punkt ein Merkmalsvektor der Länge c_{i+1} existiert.

Die Schritte *FPS & Gruppierung* und *PointNet* werden mit der Menge der Zentrumspunkte als Eingabepunktmenge rekursiv wiederholt, wobei die Anzahl der Wiederholungen benutzerdefiniert ist. Dieser Teil der Netzarchitektur wird als *Abstraktionsmodul* bezeichnet (siehe Abbildung 4.4). Um für jeden Punkt der Eingabepunktwolke eine Klasse prädizieren zu können, muss für jeden die Wahrscheinlichkeitsverteilung der s möglichen Klassen ermittelt werden. Dies geschieht wieder rekursiv, ausgehend von den Punkten der letzten Rekursionsebene und deren Merkmalsvektoren.

Interpolation & Konkatenation. Für die Punkte der aktuellen Ebene werden die Merkmalsvektoren aus denen der vorhergehenden Ebene interpoliert. Dazu werden für jeden Punkt die k nächsten Nachbarpunkte aus der vorhergehenden Ebene bestimmt und deren Merkmalsvektoren gewichtet, um sie anschließend nach ihren Abständen aufzusummieren. Dabei erhält der Merkmalsvektor des am weitesten entfernten Punkts das geringste Gewicht. Zuletzt wird der Ergebnisvektor mit dem Merkmalsvektor desselben Punkts aus dem *Abstraktionsmodul* konkateniert.

FSG. Eine FSG mit eindimensionalen Faltungsschichten und 1×1 Filterkerneln wird auf die Punkte mit interpolierten Merkmalsvektoren angewendet, um deren Länge sukzessive auf s zu reduzieren. Die FSG der letzten Ebene nutzt eine Softmax-Aktivierungsfunktion, alle anderen FSGs verwenden Rectifier-Aktivierungsfunktionen.

Die Schritte *FSG* und *Interpolation & Konkatenation* werden wiederholt, bis die Ebene der Eingabepunktwolke erreicht ist und damit für n_0 Punkte Merkmalsvektoren der Länge s existieren. Dieser Teil der Netzarchitektur wird als *Segmentierungsmodul* bezeichnet (siehe Abbildung 4.4).

Ziel der Segmentierung mithilfe von *PointNet++* ist es, dass jedes Segment nur Primitive eines Typs und weiterhin so wenige Primitive wie möglich enthält. Damit können Primitive pro Segment eingepasst werden, was das Problem von einem MKMI- auf ein EKMI- und bestenfalls auf ein EKEI-

Problem reduziert. Im Folgenden werden die zur Segmentierung notwendigen Schritte detailliert erläutert.

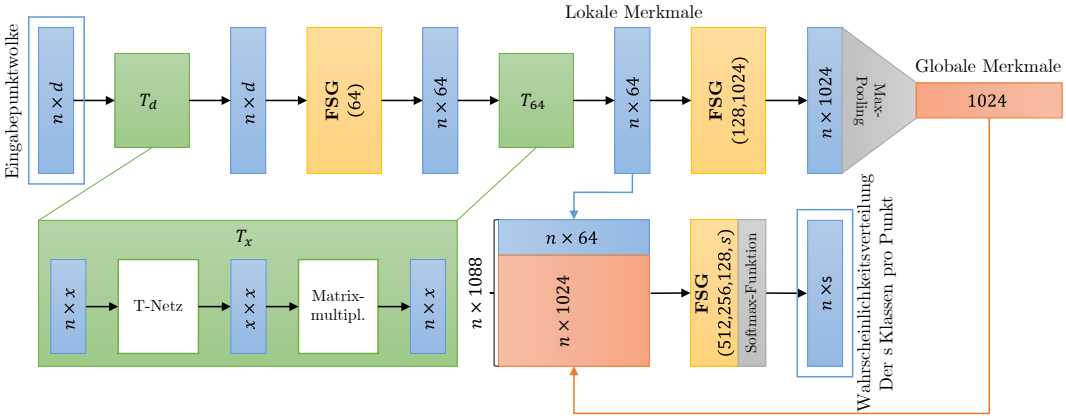


Abbildung 4.3.: *PointNet*-Architektur. Die n d-dimensionalen Eingabepunkte werden über mehrere Transformationsschritte (grün) und FSGs (orange, Anzahl der Ausgabekanäle pro Schicht in Klammern) auf einen globalen Merkmalsvektor der Größe 1024 (rot, *Globale Merkmale*) abgebildet. Zur Klassifikation pro Punkt werden die *globalen Merkmale* mit den *lokalen Merkmalen* kombiniert, indem an jedes der n lokalen Merkmalsvektoren der globale Merkmalsvektor angehängt wird. Mittels einer FSG wird das Resultat auf s Klassenlabels abgebildet (ein Label pro Punkt). Abbildung angepasst aus [141].

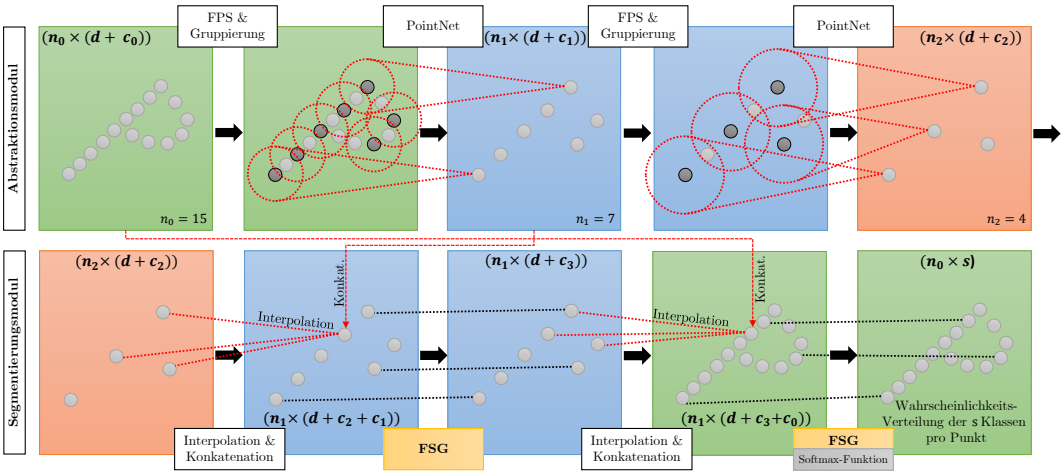


Abbildung 4.4.: *PointNet++*-Architektur. Farbige Rechtecke symbolisieren die Rekursionsebenen. Punktwolke der aktuellen Ebene in Grau, Partitionszentren in Schwarz. In Klammern wird die Dimension der in der jeweiligen Ebene verarbeiteten Ergebnisdaten angegeben. Abbildung angepasst aus [142].

4.4.3.1. Quaderpartitionierung

Frühe Tests auf der gewählten Netzarchitektur *PointNet++* zur *Prädiktion des Primitiventyps* haben ergeben, dass sich das Laufzeitverhalten verbessert, wenn O_a in Partitionen aufgeteilt wird. Dies geschieht, indem die Punkte aus O_a in Quader partitioniert werden, die nach den Koordinatenachsen ausgerichtet sind (siehe Abbildung 4.5a für ein Beispiel).

4.4.3.2. Prädiktion des Primitiventyps

Der Prädiktionsschritt, der jedem Punkt einer Punktwolke einen Primitiventyp zuordnet, wird für jede der quaderförmigen Partitionen durchgeführt. Im Anschluss werden die Punkte der einzelnen Partitionen wieder zu einer Gesamtpunktwolke O_t zusammengeführt, wobei jeder Punkt nun zusätzlich zu seiner Koordinate und seiner Oberflächennormalen ein Primitiventypattribut besitzt (siehe Abbildung 4.5b für ein Beispiel).
Wichtig hierbei ist, dass sich der erzeugte Trainingsdatensatz für das genutzte KNN stark von den an CAD-Modelle erinnernden Datensätzen für die Evaluation des Gesamtsystems unterscheidet (siehe Kapitel 4.4.1.1, Abbildung 4.2). Das trainierte Modell ist in diesem Fall jedoch in der Lage, auf die entsprechend andersartige geometrische Struktur zu generalisieren. Dies gelingt u.a. durch eine Partitionierung der Punktwolken des Trainingsdatensatzes in Quader während des Trainingsprozesses. Des Weiteren haben Experimente ergeben, dass die Oberflächennormale eines jeden Punkts signifikante geometrische Information enthält und damit die Prädiktionsqualität steigert. Aus diesem Grund wird neben der Koordinate eines Punkts auch dessen Oberflächennormale im Trainingsprozess verwendet.

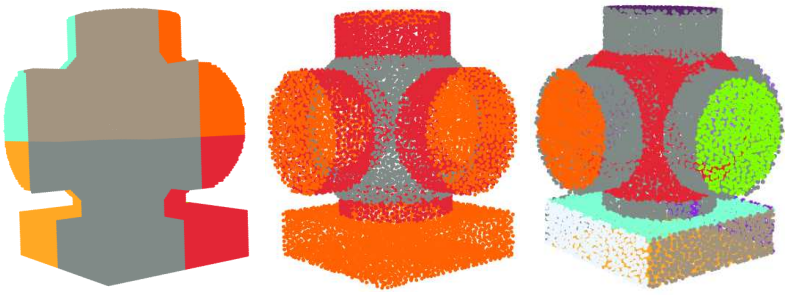
4.4.3.3. DBSCAN-Clustering

Die um den Primitiventyp erweiterte Punktwolke O_t wird nun mit dem DBSCAN-Clusteringverfahren (siehe Kapitel 2.5.3.1) in *Segmente gleichen Primitiventyps* zerlegt. Dabei fiel die Wahl auf dieses Verfahren, da es keine manuelle Festlegung der Anzahl gewünschter Cluster erfordert.
Als Distanzmetrik für DBSCAN wurde aus Effizienzgründen die Euklidische Distanz gewählt. Dies hat zur Folge, dass der Primitiventyp eines Punkts $o \in O_t$ als 1-aus-n-Code [159] repräsentiert werden muss. Dabei handelt es sich um eine Kodierung, die jeder von n Elementen einer Menge ein Bitmuster der Länge n zuordnet, indem genau ein Bit gesetzt ist. Für die Elemente der Menge der Primitiventypen $\{\text{Kugel}, \text{Zylinder}, \text{Ebene}\}$, also $n = 3$, wären das die Kodierungen $\{(1,0,0),(0,1,0),(0,0,1)\}$. Anstelle der 1 wird ein Gewichtungsfaktor μ zur Kontrolle des Einflusses des Primitiventyps auf das Clustering eingeführt. Der Vorteil dieser Kodierung liegt darin, dass die Euklidische Distanz zwischen zwei Primitiventypen entweder 0 (gleich) oder μ (ungleich)

ist. Im Gesamten ergibt sich damit die 9-dimensionale Punktbeschreibung

$$o = (p^x, p^y, p^z, \mathbf{n}^x, \mathbf{n}^y, \mathbf{n}^z, t^0, t^1, t^2), \tag{4.1}$$

wobei $o_p = (p^x, p^y, p^z)$ die Punktposition im $\mathbb{E}^3$, $o_\mathbf{n} = (\mathbf{n^x}, \mathbf{n^y}, \mathbf{n^z})$ den normierten Normalenvektor und $o_t = (t^0, t^1, t^2) \in \{\mu, 0\}^3$ den 1-aus-3-kodierten Primitiventyp darstellt. DBSCAN wird in zwei Schritten angewandt: Zuerst nur auf die Punktattribute o_p und $o_\mathbf{n}$ und dann für jedes entstehende Cluster noch einmal mit den Punktattributen o_p und o_t. Der Clustering-Schritt resultiert in einer Aufteilung der Punktwolke O_t in *Segmente gleichen Primitiventyps* (siehe Abbildung 4.1). Abbildung 4.5c enthält ein Beispiel.



(a) (b) (c)

Abbildung 4.5.: Ergebnisse der Punktwolkensegmentierung: a) *Quaderpartitionierung* (Farben: Partitionen). b) *Prädiktion des Primitiventyps* (Zylinder in Rot, Ebenen in Orange, Kugeln in Grau). c) *DBSCAN-Clustering* (Farben: Segmente).

4.4.4. Primitivendetektion und -einpassung

Für jedes ermittelte Punktwolkensegment sollen die Parameter der in ihm enthaltenen *Primitive* (siehe Abbildung 4.1) ermittelt werden. Dies geschieht in zwei Schritten: Zuerst erfolgt eine Einpassung von unbeschränkten Primitiven in die Eingabepunktwolke (siehe Kapitel 4.4.4.1). Darauf folgt eine Umwandlung der ermittelten unbeschränkten Festkörper, wie Zylinder und Ebenen, in beschränkte Festkörper (siehe Kapitel 4.4.4.2 und 4.4.4.3).

4.4.4.1. Detektion & Einpassung von unbeschränkten Primitiven

Die Einpassung von unbeschränkten Primitiven in jedes Segment der Eingabepunktwolke erfolgt in den folgenden zwei Schritten:

RANSAC. Die Schätzung der Parameter der Primitive eines Segments erfolgt über das RANSAC-Verfahren (siehe Kapitel 3.2.4.1). Die Laufzeitkomplexität zur Auffindung eines Primitivs, dem in einer Menge von n Punkten m Punkte zugeordnet werden können, beträgt $O(\frac{1}{(\frac{m}{n})^k})$. Dabei ist k die Anzahl

der Punkte, aus denen die Parameter des Primitivs eindeutig bestimmt werden können [158]. Dies erklärt auch, warum aus Laufzeitperspektive ein möglichst kleines k, also eine möglichst kleine Anzahl an nötigen Parametern zur Beschreibung eines Primitivs wünschenswert ist. Daraus ergibt sich ein weiterer Vorteil der Beschränkung auf durch Quadriken beschreibbare, unbeschränkte Primitive. Eine Erweiterung der unterstützten Primitive um Tori und Kegel ist mit dem RANSAC-Verfahren leicht zu bewerkstelligen, da die meisten Implementierungen dies bereits anbieten [158].
Da die vorhergehende Segmentierung garantiert, dass jedes Segment nur Primitive desselben Typs enthält, gewinnt das Verfahren an Robustheit. Grund dafür ist, dass sich das Problem der Einpassung von einem MKMI- auf ein EKMI-Problem - im besten Fall sogar auf ein EKEI-Problem reduziert, falls das entsprechende Segment nur ein einziges Primitiv enthält.

Zusammenführung & Filterung. Die Primitive eines jeden Segments werden auf die Primitivenmengen E (Ebenen), K (Kugeln) und Z (Zylinder) aufgeteilt. In manchen Fällen befinden sich in den zusammengeführten Primitivenmengen Primitive mit sich bis auf ein festzulegendes ϵ gleichenden Parametern. Diese Duplikate werden in diesem Schritt erkannt und aus der entsprechenden Menge entfernt.

4.4.4.2. Konstruktion konvexer Polytope

Die im vorangegangenen Schritt ermittelten Ebenen E dienen nun der Konstruktion konvexer Polytope. Dazu wird ein kombinatorisches Optimierungsproblem mit Lösungsraum Θ formuliert, wobei jedes Element in Θ eine Menge von Teilmengen von E darstellt, also eine Menge von Ebenenkombinationen mit Ebenen aus E. Die Größe des Lösungsraums ergibt sich damit aus

$$|\Theta| = \frac{n(n+1)}{2} \cdot \sum_{i=k}^{m} \binom{|E|}{i} \tag{4.2}$$

für 1 bis n Kombinationen mit k bis m Ebenen. Die Anzahl aller Mengen mit 1 bis 3 ($n = 3$) Ebenenkombinationen bestehend aus genau 4 ($k = m = 4$) Ebenen ist für 6 Ebenen ($|E| = 6$) beispielsweise gleich 90.
Konvexe Polytope sind beschränkte konvexe Polyeder (siehe Kapitel 2.3.2.4). Somit ist nach der Definition von Kapitel 2.2 auch jedes konvexe Polytop ein valider Festkörper. Jedoch müssen die mit den Ebenen in E konstruierbaren konvexen Polyeder nicht zwangsweise beschränkt sein, d.h. nicht jeder auf diese Weise erzeugbare konvexe Polyeder ist ein konvexes Polytop. Aus diesem Grund werden folgende Randbedingungen eingeführt, um den Lösungsraum Θ auf konvexe Polytope, bzw. sogar auf eine Untermenge möglicher konvexer Polytope, zu beschränken:

1. **Anzahl Ebenen**: Ebenenkombinationen müssen mindestens vier Ebe-

nen enthalten, da das der minimal möglichen Anzahl an Seitenflächen eines konvexen Polytops entspricht.

2. **Gleichheit**: Identische Ebenenkombinationen führen zu identischen konvexen Polytopen, sind deswegen innerhalb einer Lösung redundant und können entfernt werden.

3. **Volumen**: Konvexe Polytope sind beschränkt und haben damit endliches Volumen. Mit einer benutzerdefinierten oberen Schranke V_{max} für das Volumen lassen sich potentiell unbeschränkte konvexe Polyeder erkennen und entfernen.

4. **Quader**: Ein oft anzutreffendes konvexes Polytop ist der Quader. Ebenenkombinationen mit sechs Ebenen, die paarweise annähernd parallel zueinander sind, können bevorzugt ausgewählt werden, um quaderförmige Strukturen leichter rekonstruieren zu können.

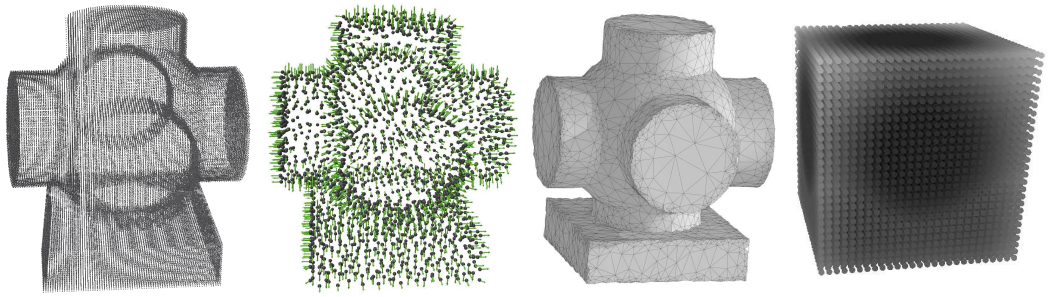
Die Randbedingungen werden dabei an unterschiedlichen Stellen durchgesetzt. So lässt sich Randbedingung 1 und 4 bei der Zusammensetzung einer einzelnen Ebenenkombination überprüfen, wohingegen Randbedingung 2 am einfachsten durch einen Filterprozess nach der Erzeugung eines kompletten Lösungskandidaten durchgesetzt werden kann. Für Randbedingung 3 hingegen muss eine Ebenenkombination bereits als konvexes Polytop in V-Darstellung vorliegen, da nur mit bekannten Extremalpunkten dessen Volumen bestimmt werden kann.
Das beschriebene Optimierungsproblem wird mit einem EA gelöst. Für die Bewertung von Lösungskandidaten muss ein geometrisches Maß verwendet werden, das möglichst exakt und laufzeiteffizient quantifizieren kann, wie gut Lösungskandidaten in die Eingabepunktwolke "passen". Dazu wird die in der Punktwolke enthaltene, räumliche Information in zweierlei Hinsicht restrukturiert:

Vereinfachung. Die *aufbereitete Punktwolke* O_a wird per FPS auf eine festgelegte Anzahl von Punkten reduziert (siehe Abbildung 4.1, *Vereinfachung*). Dieser Vereinfachungsschritt erhöht v.a. die Laufzeiteffizienz der folgenden Verarbeitungsschritte und resultiert in einer *vereinfachten Punktwolke* O_v (siehe Abbildung 4.1).

VDF-Gitternetzerzeugung. Zur Berechnung eines Maßes für die geometrische Passgenauigkeit von Primitiven bzgl. einer Punktwolke, wie es in der Zielfunktion für die Lösung des Optimierungsproblems benötigt wird, muss für eine Vielzahl von Raumpunkten deren Distanz zur Oberfläche berechnet werden (wobei die Oberfläche durch die Punktwolke approximiert wird). Um diesen Abfragevorgang zu beschleunigen, wird die *aufbereitete Punktwolke* O_a in ein *VDF-Gitternetz* G (siehe Abbildung 4.1) konvertiert. Dabei handelt es sich um ein aufzählendes Repräsentationsschema, das der

Voxel-Repräsentation sehr ähnlich ist (siehe Kapitel 2.3.2.2). Für jede Zelle speichert es dabei die vorzeichenbehaftete Distanz zur von der Punktwolke approximierten Oberfläche. Um diese Distanz zuverlässig ermitteln zu können, wird die Punktwolke zuvor in ein wasserdichtes Dreiecksnetz überführt. Dies wird mit der *Poisson*-Oberflächenrekonstruktion [98] robust bewerkstelligt.

Abbildung 4.6 zeigt beispielhafte Ergebnisse der Restrukturierung.



(a) (b) (c) (d)

Abbildung 4.6.: Ergebnisse der Punktwolkenrestrukturierung: a) *Aufbereitete Punktwolke*, b) *Vereinfachte Punktwolke* mit eingezeichneten Normalen (in Grün), c) Dreiecksnetz als Zwischenschritt der *VDF-Gitternetzerzeugung*, d) *VDF-Gitternetz*. Helligkeitswert einer Zelle entspricht der gespeicherten Distanz zur Oberfläche.

Evolutionärer Algorithmus. Im Folgenden werden die einzelnen Schritte des eingesetzten EAs detailliert beschrieben (siehe Abbildung 4.7).

Start. Zu Beginn wird die Startpopulation (siehe Abbildung 4.7, *Zufällige Population* 0) mit 1 bis n_{pop} zufällig erzeugten Polytopmengen befüllt. Die bis zu $n_{P\,max}$ konvexen Polytope für jede Polytopmenge werden dabei wie folgt erzeugt: Mit einer Wahrscheinlichkeit von $1 - p_{qu}$ wird für jedes zu erzeugende konvexe Polytop eine Menge von mindestens vier (Randbedingung 1) und höchstens $n_{e\,max}$ Ebenen aus E zufällig ausgewählt. Alternativ werden mit einer Wahrscheinlichkeit von p_{qu} sechs paarweise parallele Ebenen gesucht und selektiert (Randbedingung 4). Für jedes so entstandene konvexe Polytop wird die dazugehörige V-Repräsentation mithilfe der Doppelbeschreibungsmethode (siehe Kapitel 2.3.2.4) erzeugt, u.a. um dessen Volumen zu berechnen. Ist dieses über dem Maximalwert V_{max}, wird das konvexe Polytop verworfen und ein neues erzeugt (Randbedingung 3).
Mit der so erzeugten Population bestehend aus randomisiert konstruierten Polytopmengen beginnt der iterative Optimierungsprozess.

Filterung. In diesem Schritt wird überprüft, ob einzelne Individuen der aktuellen Population konvexe Polytope enthalten, die mit exakt den gleichen

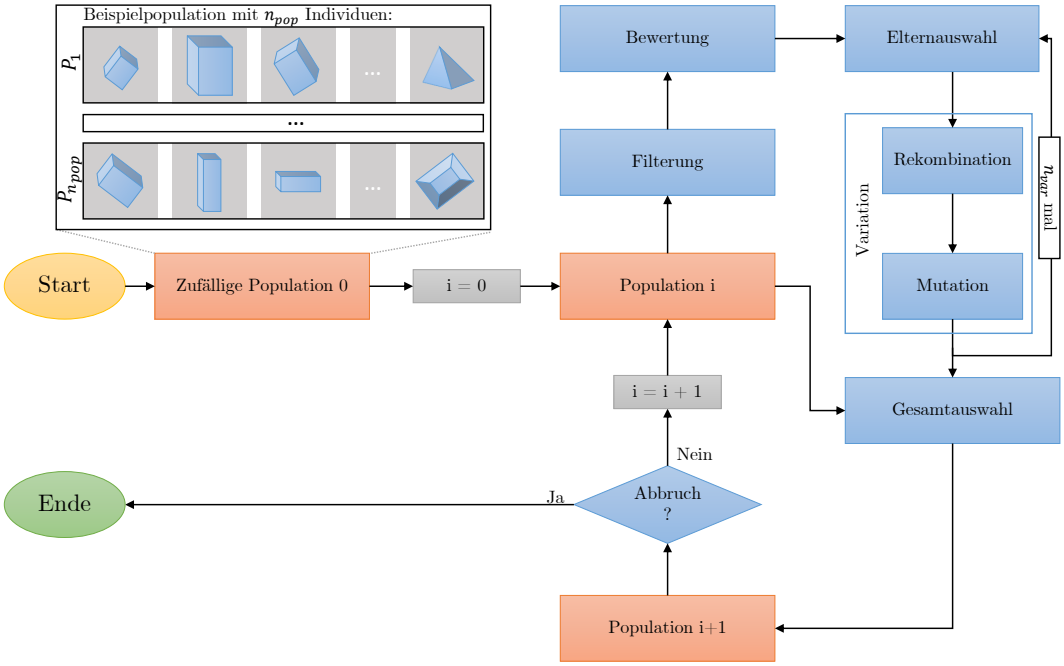


Abbildung 4.7.: Schaubild des EAs zur *Konstruktion konvexer Polytope* aus Ebenen mit Beispielpopulation.

Ebenen zusammengesetzt wurden. Derartige Duplikate innerhalb einzelner Individuen werden entsprechend entfernt. Damit wird Randbedingung 2 durchgesetzt.

Bewertung. In diesem Schritt erfolgt die Bewertung von Individuen $P \in \Theta$ aus der aktuellen Population bezüglich der zu maximierenden Zielfunktion

$$F_{O_v,G}(P) = \alpha \cdot G_{O_v}(P) + \beta \cdot V_G(P) - \gamma \cdot \frac{|P|}{n_{P\ max}}, \tag{4.3}$$

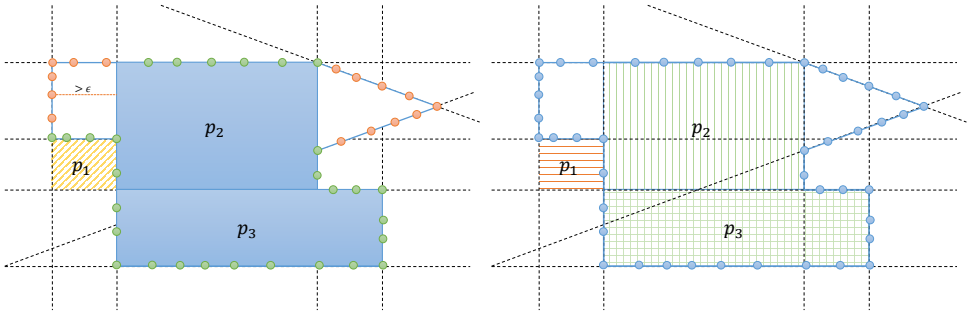
wobei $G_{O_v}(\cdot)$ der globale Geometrieterm und $V_G(\cdot)$ der Geometrieterm zur Bewertung einzelner Polytope darstellt. Der letzte Term bestraft Individuen mit zu großen Polytopmengen, wobei $n_{P\ max}$ die benutzerdefinierte, maximale Polytopmengengröße darstellt. Die Parameter α, β und γ sind ebenfalls benutzerdefiniert und haben einen Wertebereich von $[0, 1]$. Zur besseren Übersicht werden Elemente des Lösungsraums hier mit $P \in \Theta$ und nicht mit θ bezeichnet (siehe Kapitel 2.4). Aus der Definition der Zielfunktion wird weiterhin ersichtlich, dass eine Unterscheidung zwischen Geno- und Phänotyp nicht stattfindet (siehe Kapitel 2.4.3.2). Der globale Geometrieterm wird durch

$$G_{O_v}(P) = \frac{1}{|O_v|} \sum_{o \in O_v} \begin{cases} 1, & \text{falls } \min_{p \in P} |F_p(o)| < \epsilon \\ 0, & \text{sonst} \end{cases}, \tag{4.4}$$

beschrieben, wobei $F_p(\cdot)$ die VDF des konvexen Polytops p mit Ebenen $H_p \subset H$ in H-Darstellung ist, wie sie in Gleichung 2.28 definiert wird. Weiterhin ist ϵ ein benutzerdefinierter Wert. Anschaulich gesprochen bewertet dieser Term, zu welchem Grad die Oberflächen der konvexen Polytope eines Individuums die durch die Punktwolke O_v beschriebene Oberfläche abdecken. Abbildung 4.8a zeigt diesen Zusammenhang anschaulich.
Der Geometrieterm zur Bewertung der Passgenauigkeit einzelner konvexer Polytope ergibt sich aus

$$V_G(P) = \sum_{p \in P} \frac{1}{|G_p|} \sum_{g \in G_p} \begin{cases} 1, & \text{falls } F_G(o_g) < \epsilon \\ 0, & \text{sonst} \end{cases}, \tag{4.5}$$

wobei G_p die Menge der Zellen des diskretisierten Volumens von Polytop p und o_g der Mittelpunkt einer Zelle $g \in G_p$ darstellt. $F_G(\cdot)$ ist die VDF, welche zu einem Punkt die nächstliegende vorzeichenbehaftete Distanz aus G zurückgibt und ϵ ist der benutzerdefinierte Parameter aus Gleichung 4.4. Dieser Term dient der Bewertung der Passgenauigkeit eines Polytops p bezüglich des Zielmodells. Die Bewertung basiert dabei auf einem Festkörpervergleich, wobei der Festkörper des Polytops durch die zellbasierte Diskretisierung G_p dargestellt wird und der des Zielmodells analog durch G. Abbildung 4.8b veranschaulicht diesen Zusammenhang anhand eines Beispiels.



(a) (b)

Abbildung 4.8.: Beispielindividuum P mit drei konvexen Polytopen $P = \{p_1, p_2, p_3\}$ und Ebenen aus E als schwarz gestrichelte Linien. a) $G_{O_v}(P)$ mit eingefärbten Punkten aus O_v. Punkte, die von jeder Polytopoberfläche mehr als ein ϵ entfernt liegen, werden nicht gezählt (Punkte in Rot). Punkte, innerhalb des ϵ-Randes werden gezählt (grün). Wichtig: Polytop p_1 ist komplett außerhalb des Zielmodells, trägt aber trotzdem positiv zur Bewertung bei. (b) $V_G(P)$ zählt die Zellen des diskretisierten Volumens V_p eines jeden $p \in P$. Die Polytope p_2 und p_3 liegen vollständig innerhalb des Zielvolumens beschrieben durch G (grün), p_1 hingegen vollständig außerhalb (rot).

Das diskretisierte Volumen G_p eines Polytops p wird erzeugt, indem die

Extremalpunkte $v \in V_p$ dessen V-Darstellung genutzt werden, um einen orientierten Begrenzungsrahmen um p zu erzeugen, der dann in Zellen gleicher Größe zerlegt wird und somit das diskretisierte Volumen G_p ergibt. Wichtig ist, dass die Genauigkeit der späteren Berechnung von $V_G(\cdot)$ stark von der Wahl der für die Diskretisierung des Volumens G resp. G_p benutzten Zellgröße abhängt. Abbildung 4.9 stellt die Erzeugung von G_p anschaulich dar.

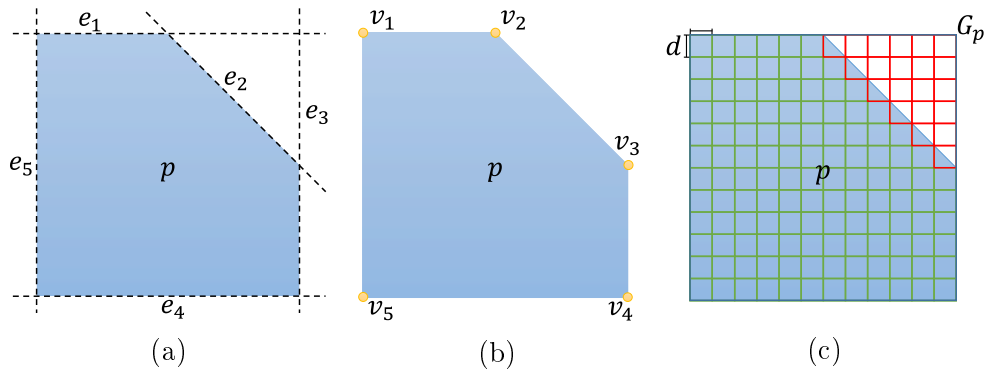


Abbildung 4.9.: Erzeugung des diskretisierten Volumens für Polytop p, G_p. a) H-Darstellung des Polytops p (blau) über Ebenen $H_p = \{e_1, ..., e_5\}$ (schwarz). b) Umwandlung in V-Darstellung mit Extremalpunkten $V_p = \{v_1, ..., v_5\}$ (orange). c) Diskretisierung des Polytopvolumens innerhalb des durch die Punkte V_p definierten Begrenzungsrahmens, der in Dunkelblau eingezeichnet ist. Zellen der Höhe und Breite d innerhalb von p in Grün, Zellen außerhalb in Rot. Zellen, die außerhalb liegen, werden bei der Berechnung der Passgenauigkeit ignoriert.

Elternauswahl. Für die Auswahl der Elternindividuen für die *Variation* wird auf die Turnier-Selektion zurückgegriffen (siehe Kapitel 2.4.3.2). Der Auswahlprozess und die folgende *Variation* durch *Rekombination* und *Mutation* werden dabei für eine festgelegte Anzahl von $n_{var} \leq n_{pop}$ Iterationen wiederholt.

Rekombination. Zwei selektierte Elternindividuen werden mit einer benutzerdefinierten Wahrscheinlichkeit von p_{rek} rekombiniert, indem zufällig gewählte Polytopsequenzen zwischen beiden ausgetauscht werden. Somit resultiert dieser Schritt in zwei modifizierten Polytopmengen.

Mutation. Ein bereits rekombiniertes Individuum wird zusätzlich mit einer benutzerdefinierten Wahrscheinlichkeit von p_{mut} einer Mutationsoperation unterzogen. Diese existiert in fünf verschiedenen Varianten, die zufällig ausgewählt werden:

- **Ersetzen:** Ein zufällig aus der Polytopmenge gewähltes Polytop wird

mit einem neuen, zufällig erstellten Polytop ersetzt.

- **Modifizieren:** Eine zufällig gewählte Ebene eines zufällig gewählten Polytops wird durch eine zufällig gewählte neue Ebene aus E ersetzt.
- **Hinzufügen:** Ein zufällig erstelltes Polytop wird der Menge der Polytope hinzugefügt.
- **Entfernen:** Ein zufällig gewähltes Polytop wird aus der Menge der Polytope entfernt.
- **Neu:** Die komplette Polytopmenge wird mit zufällig erzeugten Polytopen neu aufgebaut.

Für jede Mutationsvariante kann eine benutzerdefinierte Wahrscheinlichkeit festgelegt werden, wobei sich die Wahrscheinlichkeiten zu 1 addieren: $p_{ers} + p_{mod} + p_{hin} + p_{ent} + p_{neu} = 1$.

Gesamtauswahl. Die n_{var} durch *Variation* modifizierten Individuen werden zusammen mit den $n_{pop} - n_{var}$ besten Individuen der aktuellen Population (siehe Abbildung 4.7, *Population* i) zur neuen Population (siehe Abbildung 4.7, *Population* $i + 1$) der Größe n_{pop} zusammengefügt.

Abbruch. Wenn eine benutzerdefinierte Anzahl von n_{iter} Iterationen erreicht ist oder sich der beste Wert der Zielfunktion über m_{iter} Iterationen nicht verbessert, terminiert die Optimierungsschleife.

Zusammengefasst wird der EA zur *Konstruktion konvexer Polytope* durch die in Tabelle 4.1 dargestellten, benutzerdefinierten Parameter konfiguriert.

Parameter	Beschreibung
n_{pop}	Größe der Population
n_{var}	Anzahl der Wiederholungen der Variationsoperatoren
$n_{e\,max}$	Maximale Anzahl von Ebenen pro Polytop
$n_{p\,max}$	Maximale Anzahl von Polytopen pro Individuum
n_{iter}	Maximale Anzahl an Optimierungsiterationen
m_{iter}	Maximale Anzahl an Iterationen ohne Verbesserung der Zielfunktion
V_{max}	Maximales Volumen eines konvexen Polytops
p_{qu}	Wahrscheinlichkeit, dass bei der Erzeugung eines neuen konvexen Polytops eine Quaderform angestrebt wird
α,β, γ	Gewichtungsfaktoren der Terme der Zielfunktion (Gleichung 4.3)
ϵ	Maximal tolerierbarer Abstand in Gleichung 4.4 und 4.5
p_{mut}	Mutationswahrscheinlichkeit
p_{rek}	Rekombinationswahrscheinlichkeit
p_{ers}, p_{mod}, p_{hin}, p_{ent}, p_{neu}	Wahrscheinlichkeiten der verschiedenen Mutationsvarianten
$V_{G\,min}$	Benutzerdefinierter Mindestwert für die geometrische Passgenauigkeit

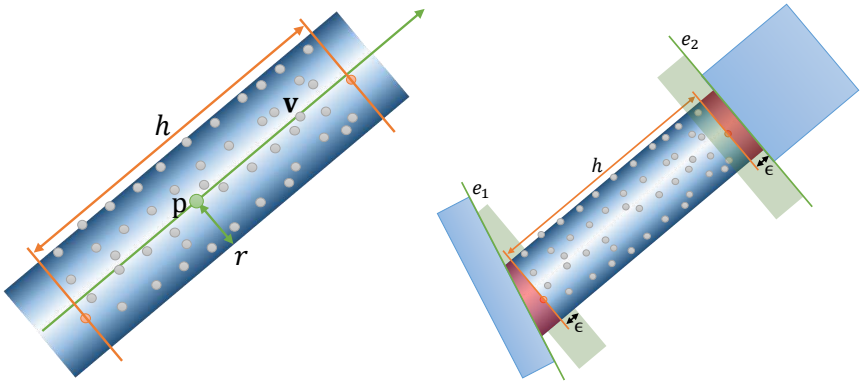
Tabelle 4.1.: Benutzerdefinierte Parameter des EAs zur *Konstruktion konvexer Polytope*.

Selektion & Filterung. Aus der Population der letzten Iteration des EAs wird das Individuum (Polytopmenge) mit der besten Bewertung als Endergebnis ausgewählt. Zudem werden in diesem abschließenden Schritt abermals Duplikate entfernt. Dies geschieht durch den paarweisen Vergleich der VDF-Gitternetze der resultierenden konvexen Polytope. Außerdem werden konvexe Polytope entfernt, die einen benutzerdefinierten Mindestwert $V_{G\,min}$ für die geometrische Passgenauigkeit (Gleichung 4.5) nicht erreichen. Damit ist der Optimierungsprozess am *Ende* angekommen und das Resultat kann verwendet werden.

4.4.4.3. Ermittlung der Zylinderdeckflächen

Alle Zylinder in Z sind unbeschränkte Festkörper. Um diese entsprechend zu beschränken, werden Zylinderdeckflächen ermittelt. Grundsätzlich lässt sich die Zylinderhöhe anhand der dem Zylinder zugeordneten Punktwolke bestimmen. Dazu wird die maximale Ausdehnung der Punktwolke entlang der Hauptachse des Zylinders gemessen (siehe Abbildung 4.10a). Setzt man voraus, dass

das Gesamtobjekt, welches den Zylinder als Teilobjekt enthält, zusammenhängend ist, dann lässt sich der beschränkte Zylinder als eine Kombination von Begrenzungsebenen darstellen (siehe Abbildung 4.10b). Diese Darstellung hat den Vorteil, dass etwaige Lücken zwischen Zylinder und anschließenden Primitiven vermieden werden können. Für eine Unterstützung von Kegeln als Primitiventyp müsste das System um ein ähnliches Verfahren zur Ermittlung der Kegeldeckfläche erweitert werden.



(a) (b)

Abbildung 4.10.: *Ermittlung der Zylinderdeckflächen*: a) Schätzung der Zylinderhöhe h (orange) basierend auf der Punktwolke (grau) und der Zylinderhauptachse $(p, \mathbf{v})$ (grün), wobei r der Zylinderradius ist. b) Entstehende Lücken (rot) können geschlossen werden, indem bereits eingepasste Ebenen, die in einem Bereich (grün, durch ϵ definiert) um die Zylinderenden liegen, als Deckflächen verwendet werden (e_1 und e_2).

4.5. Unterstützung komplexer konvexer Polytope

Die in Gleichung 4.2 zum Ausdruck gebrachte Komplexität des Optimierungsproblems zur *Konstruktion konvexer Polytope* beschränkt die Anzahl der mit dieser Methode gleichzeitig extrahierbaren konvexen Polytope beträchtlich. Daher stellt sich die Frage, wie das Problem vereinfacht werden kann, um komplexere Modelle rekonstruieren zu können.
Ein dabei oft gewählter Weg ist *Divide et Impera*, also das Aufteilen des Problems in einfacher zu lösende Teilprobleme und die darauffolgende Zusammenführung der Teillösungen zu einer Gesamtlösung. Diese Lösungsstrategie, die auch als *Dynamische Programmierung* bekannt ist, funktioniert für Probleme, die dem Optimalitätsprinzip von Bellmann genügen. Dieses fordert, dass sich jede optimale Gesamtlösung aus optimalen Teillösungen zusammensetzt [14].

Angewandt auf das hier betrachtete Problem der Extraktion konvexer Polytope bringt die Aufteilung der Eingabepunktwolke in konvexe Segmente in vielen Fällen die gewünschte Komplexitätsreduktion (siehe Kapitel 4.5.2). Die in Kapitel 4.4.4.2 beschriebene Methode zur Extraktion von Polytopen wird dann auf jedes Segment angewandt und die so erhaltenen konvexen Polytope zu einem Gesamtresultat vereint.
Darüber hinaus lässt sich der Suchraum durch eine weitere Randbedingung beschränken, indem ein Nachbarschaftsgraph $G_N = (E, N)$ für die zu verwendenden Ebenen E berechnet wird. G_N stellt dazu Ebenen als Knoten und Nachbarschaften zwischen zwei Ebenen als Kanten dar. Bei der zufälligen Konstruktion neuer Polytope kann jetzt G_N verwendet werden, um gut geeignete Ebenenkombinationen ausgehend von einer zufällig gewählten Ebene zu ermitteln.
Die für beide Vereinfachungsstrategien notwendigen Modifikationen finden im Pipeline-Schritt *Primitivendetektion und -einpassung* (siehe Abbildung 4.1) statt und werden im Folgenden beschrieben. Eine entsprechende Übersicht ist in Abbildung 4.11 dargestellt.

4.5.1. Ermittlung von Ebenennachbarschaften

Um Ebenennachbarschaften zu identifizieren, wird zuerst eine Punktwolkenrestrukturierung vorgenommen. Dazu wird auf ein in [103] beschriebenes Verfahren zurückgegriffen. Jede Ebene $e \in E$ lässt sich dabei als ein Tupel $e = \{P_e, O_e\}$ aus Ebenenparameter P_e und zugeordneter Punktwolke O_e beschreiben. In einem ersten Schritt zur Strukturierung der Ebenenpunktwolken wird ein paralleles Viereckснetz auf jeder Ebene $e \in E$ platziert (Zellgröße: $\sqrt{2}\epsilon$, ϵ ist benutzerdefiniert) und die Punkte der Ebene O_e hineinprojiziert. Für jede mit einem Punkt ausgefüllte Viereckszelle wird deren Mittelpunkt Teil der neuen strukturierten Punktwolke der Ebene (siehe Abbildung 4.12a). Damit wird garantiert, dass Ebenenpunkte äquidistant auf der Ebenenoberfläche verteilt liegen.
Um Kanten zwischen benachbarten Ebenen zu detektieren und entsprechende Kantenpunkte zu erhalten, werden zuerst Schnittgeraden zwischen je zwei Nachbarebenen berechnet. Neue Punkte werden dann entlang der Schnittgeraden im Abstand von 2ϵ erzeugt, solange deren Umgebung (Kugel mit Radius 2ϵ um einen Punkt) Punkte aus den Punktwolken beider Nachbarebenen enthält. Zuletzt werden noch Eckpunkte identifiziert, indem der Nachbarschaftsgraph G_N auf Zyklen der Länge 3 abgesucht wird und die jeweils zugehörigen drei Ebenen miteinander geschnitten werden (siehe Abbildung 4.12b).
Das Resultat der Punktwolkenrestrukturierung ist eine Punktmenge O_s, welche die strukturierten Punktwolken aller Ebenen in E zusammen mit allen erzeugten Kanten- und Eckpunkten enthält. Diese kann nun für die Ermittlung der Ebenennachbarschaften in E verwendet werden, welche durch den Nachbarschaftsgraph G_N repräsentiert wird. Dazu wird eine Idee aus [103]

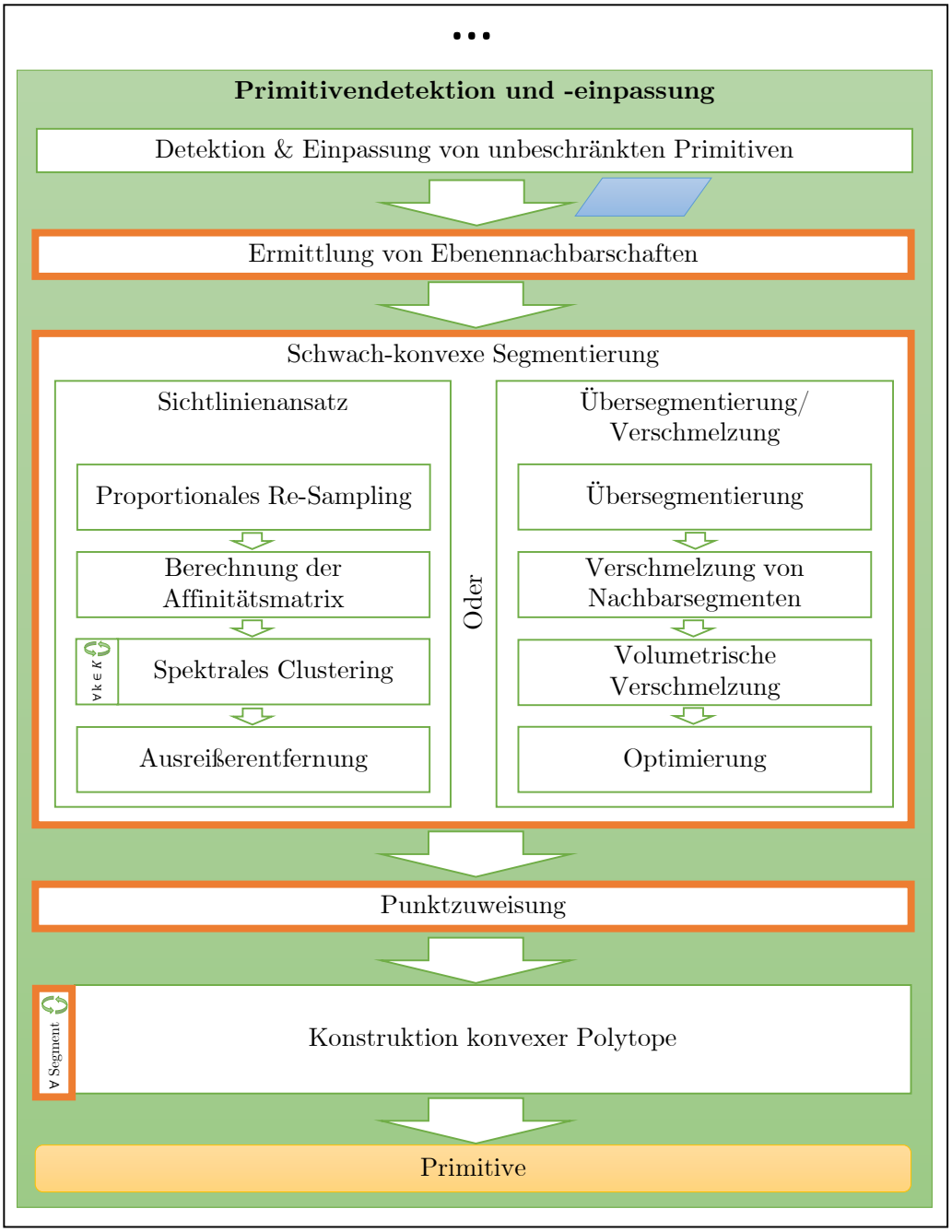
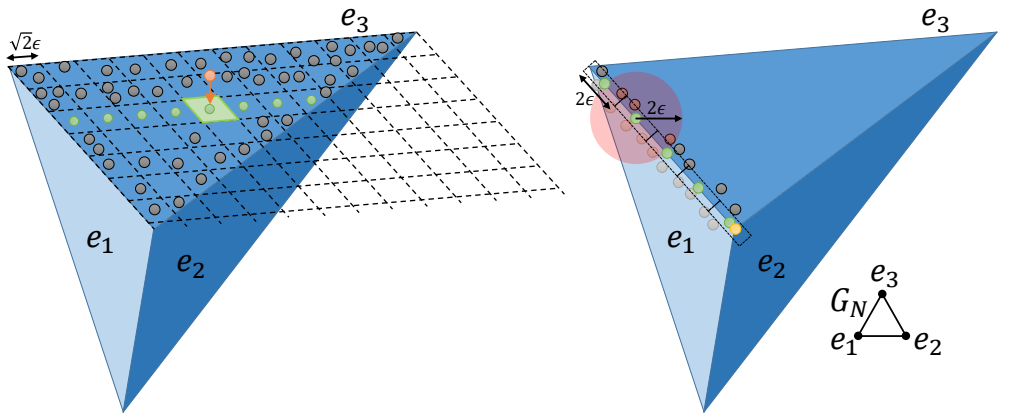


Abbildung 4.11.: Erweiterung des Pipeline-Schritts *Primitivendetektion und -einpassung* aus Abbildung 4.1 zur effizienteren Einpassung von konvexen Polytopen. Neu hinzugekommene Elemente in Dunkelorange.

aufgegriffen. Derzufolge sind zwei Ebenen $e_1, e_2 \in E$ benachbart, wenn mindestens ein Punktepaar existiert, bei dem ein Punkt Ebene e_1 und ein Punkt Ebene e_2 zugeordnet ist und die im k-Nächste-Nachbarn-Graph der strukturierten Punktwolke O_s benachbart sind.



(a) (b)

Abbildung 4.12.: Punktwolkenstrukturierung für Ebenen $\{e_1, e_2, e_3\}$: a) Unstrukturierte Ebenenpunkte in Dunkelgrau, bereits strukturierte Punkte in Grün. Der Punkt in Rot wird beispielhaft auf die Zelle in Grün projiziert. Der resultierende strukturierte Punkt liegt im Zentrum der Zelle. b) Abtastung entlang der Kante zwischen e_1 und e_3 mit äquidistantem Abstand 2ϵ. Resultierende Kantenpunkte sind in Grün dargestellt. Die Punkte in Hellgrau sind die zu Ebene e_1 gehörigen. Ein Kantenpunkt ist valide, wenn sich innerhalb einer Kugel mit Radius 2ϵ (rot) mindestens jeweils ein Punkt aus e_1 und e_3 befindet. In Orange dargestellt ist der Eckpunkt als Schnittpunkt der Ebenen des 3-Zyklus in G_N, (e_1, e_2, e_3).

4.5.2. Schwach-konvexe Segmentierung

Ziel ist es, die Eingabepunktwolke (in dem Fall O_s) in konvexe Segmente C zu zerlegen, um dann aus jedem Segment $c \in C$ entsprechende konvexe Polytope zu extrahieren. Dabei ist jedes Segment durch ein Tupel $c = (O_c, E_c)$ aus zugeordneter Punktmenge O_c und Ebenen E_c charakterisiert. Da jedoch die Aufteilung eines dreidimensionalen Polyeders in eine minimale Anzahl nicht überlappender, konvexer Segmente ein $\mathcal{NP}$-*schweres* Problem ist [30], wird auf eine vereinfachte Form der Konvexität zurückgegriffen, die sog. schwache Konvexität [10]. Für diese lässt sich die Kennzahl

$$KR(O) = \frac{|LoS(O)|}{|O|^2} \tag{4.6}$$

berechnen, der sog. Konvexitätsrang, wobei $LoS(O)$ die Menge der Punktpaare in O enthält, für die ein direkter Sichtstrahl existiert, das Punktpaar sich also "sehen" kann [10]. Ist z.B $KR(O) = 1$ handelt es sich bei der Punktmenge O um die Approximation der Oberfläche eines konvexen Festkörpers. Damit lässt sich das Segmentierungsproblem spezifischer ausdrücken: Gesucht ist eine Segmentierung, die die Anzahl der Segmente minimiert und dabei den Konvexitätsrang eines jeden Segments maximiert [93].
Es handelt sich bei diesem Schritt also um eine weitere *Punktwolkensegmentierung* vom Typ *Modelleinpassung* (siehe Kapitel 3.2.3). Dabei findet das Einpassen des Ebenenmodells im Teilschritt *Detektion & Einpassung von unbeschränkten Primitiven* statt (siehe Abbildung 4.1). Da sie aber zwischen zwei Schritten der Stufe *Primitivendetektion und -einpassung* ausgeführt wird, ist sie in Abbildung 4.11 auch dort verortet.
Für den Segmentierungsprozess wurden zwei unterschiedliche Verfahren implementiert und evaluiert (siehe Kapitel 4.5.2.1 bzw. Kapitel 4.5.2.2).

4.5.2.1. Sichtlinienansatz

Zentrales Element des *Sichtlinienansatzes* für die schwach-konvexe Segmentierung von Punktwolken, wie er in [10] eingeführt wurde, ist der Graph G_V. Dieser enthält jeden Punkt der Punktwolke als Knoten. Kanten zwischen zwei Knoten sind immer dann vorhanden und mit einem Einheitsgewicht von 1 belegt, wenn sich das zugehörige Punktepaar "sieht", d.h. das Verbindungsliniensegment zwischen beiden Punkten nicht die Oberfläche des zu rekonstruierenden Objekts schneidet. Nun haben Punkte innerhalb schwach-konvexer Segmente Sichtlinien zu allen anderen Punkten des Segments, jedoch wenig bis gar keine zu Punkten außerhalb des Segments (siehe Abbildung 4.13).
Möchte man die Zuordnung zwischen Punkten und schwach-konvexen Segmenten bestimmen, lässt sich dies als Suche nach dem normalisierten, minimalen Schnitt von G_V interpretieren (siehe Kapitel 2.5.3.1). Eine approximative Lösung dieses $\mathcal{NP}$*-schweren* Problems lässt sich über ein Spektrales Clustering von G_V ermitteln. Im Folgenden werden die einzelnen Schritte des *Sichtlinienansatzes* beschrieben.

Proportionales Re-Sampling. Die Laufzeitkomplexität der Sichtstrahlenberechnung, wie sie hier im Unterschied zu [10] verwendet wird, verhält sich quadratisch zur Anzahl an Punkten der Punktwolke, die des Spektralen Clusterings sogar kubisch (siehe Kapitel 2.5.3.1). Aus diesem Grund wird eine Vereinfachung der strukturierten Punktwolke O_s vorgenommen, um die Anzahl der Punkte zu reduzieren. Dazu wird ein FPS auf jede, mit einer Ebene e assoziierten Teilmenge von O_s, $O_{s\,e}$, durchgeführt. Die jeweilige Anzahl der Punkte pro Ebene, ergibt sich dabei aus

$$k_i = k\frac{|O_{s\,e_i}|}{|O_s|},\ i \in \{1, ..., |E|\}, \tag{4.7}$$

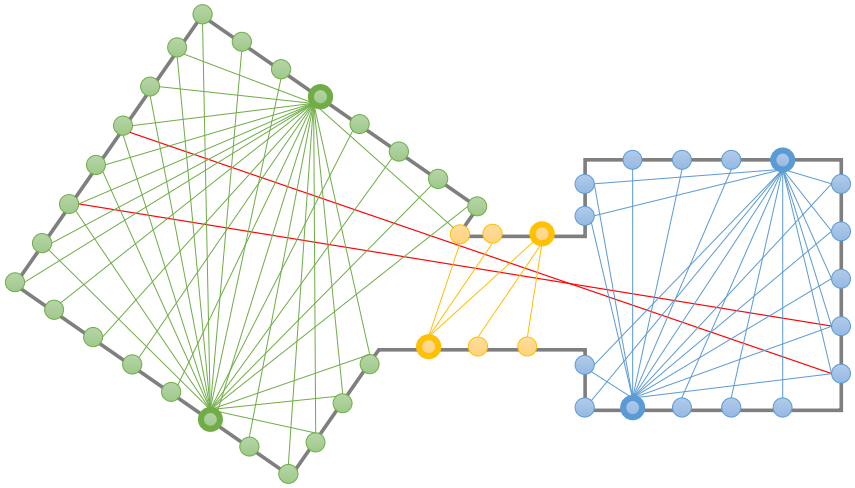


Abbildung 4.13.: Darstellung der segmentierten Eingabepunktwolke mit konvexen Clustern in Grün, Orange und Blau. Die approximierte Oberfläche ist in Grau dargestellt. Sichtlinien sind für zwei Beispiele je Cluster (umrandete Kreise) eingezeichnet. Sichtlinien zwischen Punkten verschiedener Cluster sind rot markiert.

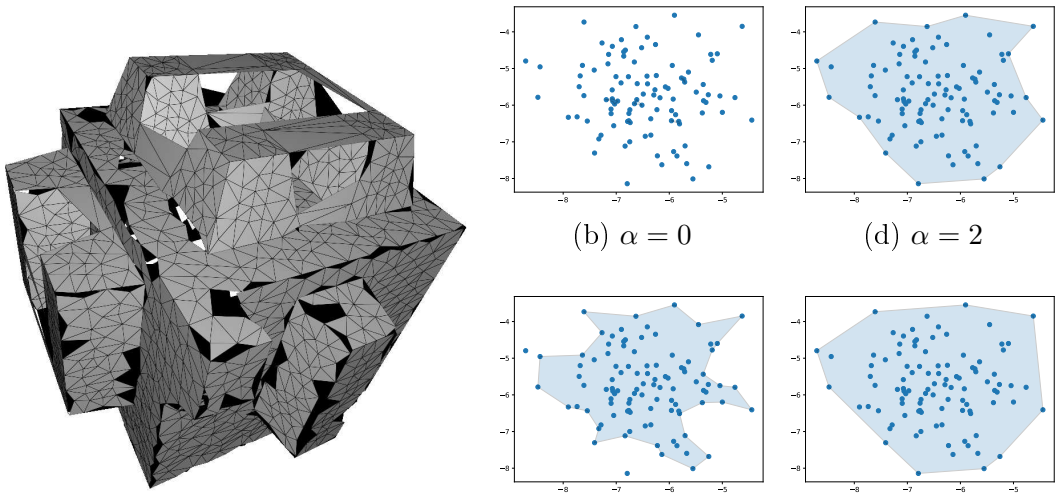
wobei k benutzerdefiniert ist und die gewünschte Größe der aus diesem Schritt resultierenden Punktwolke O_{sr} angibt. Das in dieser Arbeit neu eingeführte, separate Re-Sampling pro Ebene erhält die Punktdichte der Ebenenpunktwolken relativ zueinander und ist deshalb geeigneter als ein einmaliges Re-Sampling auf der Gesamtpunktwolke O_s.

Berechnung der Affinitätsmatrix. Für die Punktwolke O_{sr} wird nun der Sichtbarkeitsgraph $G_V = (O_{sr}, LoS(O_{sr}))$ berechnet, wobei $LoS(O_{sr})$ hier der Kantenmenge von G_V entspricht. Die dazugehörige Adjazenzmatrix A_V wird auch Affinitätsmatrix genannt. Deren Koeffizienten ergeben sich aus

$$A_{Vij} = \begin{cases} 1, & \text{falls } (v_i, v_j) \in LoS(O_{sr}) \\ 0, & \text{sonst} \end{cases}, \tag{4.8}$$

wobei v_i bzw. v_j der i-te bzw. j-te Punkt in O_{sr} darstellt. Eine Sichtlinie zwischen zwei Punkten existiert, wenn das entsprechende Liniensegment keine Schnitt- oder Berührpunkte mit der Oberfläche des zu rekonstruierenden Objekts hat. Da die Objektoberfläche nur als Punktwolke vorliegt, muss eine geeignete Oberflächenschätzung vorgenommen werden. Die bei der *VDF-Gitternetzerzeugung* (siehe Kapitel 4.4.4.2) erzeugte Oberflächenapproximation ist dabei v.a. an Objektkanten nicht ausreichend genau. Die in [10] verwendete Methode, die auf einer Approximation der Oberfläche mit kleinen Oberflächenstücken für jeden Punkt beruht, ist zu sehr von der richtigen Wahl des Größenparameters für die Oberflächenstücke abhängig.

Aus diesem Grund wird in dieser Arbeit ein anderer Ansatz verfolgt, der auf sog. α-Formen (engl. α-Shapes) beruht [42]. Mit α-Formen lässt sich eine Art von konkaver Hülle einer n-dimensionalen Punktwolke beschreiben, parametrisiert durch die reelle Zahl α. Für $\alpha = 0$ degeneriert die Form zur Eingabepunktwolke. Für $\alpha = \infty$ ergibt sich deren konvexe Hülle (siehe Abbildungen 4.14b und 4.14e für zweidimensionale Beispiele sowie Abbildungen 4.14c und 4.14d für weitere). Eine automatische Bestimmung von α ist möglich. Dazu werden verschiedene Werte von Null aus ansteigend angenommen und derjenige gewählt, bei dem alle Datenpunkte entweder innerhalb der regularisierten α-Form oder auf dessen Oberfläche liegen.
Wie in [42] angemerkt, lassen sich dreidimensionale α-Formen intuitiv wie folgt verstehen: Aus einem mit Eiscreme gefüllten Raum werden mit einem kugelförmigen Löffel (Radius $r = \sqrt{\alpha}$) iterativ Bereiche entfernt. Die Eiscreme enthält jedoch Stücke aus Schokolade (die Punkte der Punktwolke), welche nicht entfernt werden können. Die α-Form ist nun diejenige Form, die übrig bleibt, wenn alle mit dem Löffel entfernbaren Eiscremebereiche entfernt wurden.
In dem hier beschriebenen Anwendungsfall werden für jede Ebene zweidimensionale α-Formen aus den auf die Ebene projizierten Ebenenpunkten erzeugt und in ein Dreiecksnetz umgewandelt. Diese Dreiecksnetzflächen approximieren die Oberfläche des zu rekonstruierenden Objekts für jede Ebene stückweise (siehe Abbildung 4.14a). Eine solche Oberflächendarstellung hat den Vorteil, dass v.a. Kanten erhalten bleiben. Ein Nachteil ist, dass das resultierende Dreiecksnetz der Oberfläche im Allgemeinen nicht mehr wasserdicht ist, was sich aber für den hier betrachteten Anwendungsfall als vernachlässigbares Problem herausgestellt hat. Um festzustellen, ob sich zwei Punkte der Punktwolke "sehen" können, wird geprüft, ob sich deren Verbindungssegment mit einem der α-Formen schneidet. Ist dies nicht der Fall, besteht ein direkter Sichtstrahl zwischen beiden Punkten.
Die hier verwendete Methode zur Berechnung von A_V unterscheidet sich von der in [10] gewählten Strategie, die wie folgt funktioniert: Grundsätzlich werden dabei keine Vorverarbeitungsschritte, wie das *Proportionale Re-Sampling*, durchgeführt, sondern direkt auf der aufbereiteten Eingabepunktwolke O_a gearbeitet. Effizient ist das Verfahren trotzdem, weil dessen Laufzeit nicht mehr quadratisch, sondern linear von der Anzahl der Punkte abhängt. Im ersten Schritt werden für jeden Punkt $o \in O_a$ Strahlen erzeugt, die in entgegengesetzter Richtung der Normale von o mit einer zufälligen Richtungsabweichung von bis zu 30 Grad verlaufen und in o ihren Ursprung haben. Nun werden die ersten Schnittpunkte der Strahlen mit der Oberfläche untersucht und derjenige Punkt als von o sichtbar markiert, der dem jeweiligen Schnittpunkt am nächsten liegt. Diese Strategie hat sich in ersten Tests als unbrauchbar erwiesen, da die resultierende Affinitätsmatrix A_V in allen Testfällen zu dünn besetzt war und damit zu einem Clustering von schlechter Qualität führten.



(a) α automatisch bestimmt. (b) $\alpha = 0$ (c) $\alpha = 1$ (d) $\alpha = 2$ (e) $\alpha = \infty$

Abbildung 4.14.: a) Dreiecksnetz zusammengesetzter α-Formen. b), c), d), e): α-Formen auf Basis einer synthetischen Punktwolke für verschiedene Belegungen des Parameters α.

Spektrales Clustering. Spektrales Clustering funktioniert über ein k-Means-Clustering der größten k Eigenvektoren der Laplace-Matrix L von G_V bzw. A_V (siehe Kapitel 2.5.3.1). Die Anzahl der Cluster k muss vorweg bekannt sein, was nicht immer möglich ist. Frühe Tests haben zudem ergeben, dass sich für diesen Anwendungsfall andere Clusteringverfahren, wie z.B. DBSCAN, die ohne eine Festlegung von k auskommen, nicht eignen. Aus diesem Grund werden Mechanismen zur automatischen Schätzung der optimalen Cluster- und damit der optimalen Segmentanzahl verwendet.
Ebenfalls vorevaluiert wurde eine Methode, die über das Auffinden von Lücken innerhalb der Sequenz der Eigenwerte der Laplace-Matrix funktioniert [203]. Dabei signalisiert jede Lücke den Übergang in ein neues Cluster. Diese Strategie führte jedoch in frühen Tests zu einer Übersegmentierung der Eingabepunktwolke, sodass auf eine alternative Methode zurückgegriffen wurde [10]. Diese basiert auf einer einfachen linearen Suche nach dem besten Wert für k in einem Intervall von ganzzahlig positiven Werten $K = [k_{\min}, k_{\max}]$. Die Quantifizierung der Qualität eines Clusterings für bestimmte k und C_k, erfolgt dabei über die Bewertungsfunktion

$$Q(C_k) = \frac{1}{|O_{sr}|^2} \sum_{c \in C_k} |LoS(O_c)| + |\overline{LoS}(O_c, O_{sr} \setminus O_c)|, \tag{4.9}$$

wobei die Menge $\overline{LoS}(O_1, O_2)$ alle Punktpaare $(o_1 \in O_1, o_2 \in O_2)$ enthält, zwischen denen keine ununterbrochene Sichtlinie besteht und O_c die Punkt-

menge des Clusters c darstellt. Spektrales Clustering wird nun für alle $k \in K$ durchgeführt und dasjenige Clustering-Ergebnis gewählt, dessen Bewertung bzgl. der Bewertungsfunktion aus Gleichung 4.9 am besten ist [7, 145, 205]. Im Unterschied zur einfachsten Version des Spektralen Clusterings, wie es in Kapitel 2.5.3.1 beschrieben wird, nutzt die hier gewählte Variante für verbesserte Robustheit nicht die Laplace-Matrix L von G_V, sondern die normalisierte Variante $L_{sym} = D^{-1/2}LD^{1/2}$, die sog. *Symmetrische, Normalisierte Laplace-Matrix* [192].

Ausreißerentfernung. Die aus dem Schritt *Spektrales Clustering* gewonnenen Cluster enthalten bisweilen Ausreißer. Um diese zu entfernen, wird ein einfacher Detektor benutzt, der die maximale Distanz eines jeden Punkts zu seinen k nächsten Nachbarn betrachtet und gegen einen benutzerdefinierten Schwellwert testet.

4.5.2.2. Übersegmentierung/Verschmelzung

Als Alternative zum bereits vorgestellten *Sichtlinienansatz* wird in dieser Arbeit noch ein weiteres Verfahren zur Segmentierung von Punktwolken in schwach-konvexe Segmente betrachtet, welches von Kaick et al. in [93] vorgestellt wurde. Im Folgenden wird dessen Funktionsweise anhand vierer zentraler Prozessschritte erläutert. Eine vollständige Beschreibung des Verfahrens findet sich in [93].

Übersegmentierung. Im ersten Schritt wird die Eingabepunktwolke O_a in Teilmengen $O = \{O_1, ..., O_n\}$ aufgeteilt. Frühe Tests haben ergeben, dass diese Segmentierungsvariante besser auf der *aufbereiteten Punktwolke* O_a funktioniert als auf der strukturierten Punktwolke O_s. Grund dafür ist die höhere Punktwolkendichte in O_a. Die Teilmengen werden dazu über ein Spektrales Clustering des sog. Punktkonvexitätsgraphen G_{pk} erzeugt. G_{pk} enthält die Punkte aus O_a als Knoten, die immer dann mit einer Kante verbunden sind, wenn die Punkte des zugehörigen Punktpaars Nachbarn im k-Nächste-Nachbarn-Graph G_a sind und sich deren Oberflächennormalen bis auf ein benutzerdefiniertes Winkel-δ gleichen. Das Resultat dieses Schritts ist in Abbildung 4.15a beispielhaft dargestellt. Es ist erkennbar, dass es sich um eine Übersegmentierung handelt, da einzelne konvexe Segmente nicht größtmöglich sind.

Verschmelzung von Nachbarsegmenten. Die Teilmengen O werden mithilfe eines Ansatzes, der auf dem Prinzip des Regionenwachsens (siehe Kapitel 3.2.3.2) beruht zu Segmenten C verschmolzen. Dazu wird initial jede Teilmenge aus O genau einem Segment aus C zugeordnet. Dann werden die Segmente in C iterativ verschmolzen, wobei in jeder Iteration folgendes passiert: Paare benachbarter Segmente aus C werden gemäß ihres Konvexitätsrangs (siehe Term innerhalb der Summe in Gleichung 4.9) absteigend

sortiert und in dieser Reihenfolge schrittweise verschmolzen, sofern deren Konvexitätsrang einen Schwellwert θ nicht unterschreitet. Dies wird für eine benutzerdefinierte Anzahl an Iterationen mit unterschiedlichen Schwellwerten wiederholt.

Volumetrische Verschmelzung. Ein weiterer Schritt zur Verbesserung der Segmentierung ist die *Volumetrische Verschmelzung*. Dazu wird für jedes Segment in $c \in C$ eine volumetrische Signatur in Form eines Histogramms h_c erstellt. Dieses Histogramm enthält Werte der *Formdurchmesserfunktion* (FDF) (engl. Shape Diameter Function, eingeführt in [167]), die an einer Menge von Abtastpunkten auf der Oberfläche von c bestimmt wird. Die FDF ist eine reellwertige Funktion, die für einen Punkt o auf der Objektoberfläche den Durchmesser der lokalen Nachbarschaft bestimmt. Damit ist sie eine lokale Volumenbeschreibung eines Festkörpers. Approximiert wird sie meist, indem Strahlen ausgehend von o innerhalb und in Richtung eines nach innen zeigenden Kegels mit den Objektoberflächen geschnitten und das arithmetische Mittel der Längen der daraus entstehenden Strahlensegmente gebildet wird [62]. Abbildung 4.15b zeigt eine Beispiel-FDF.
Mittels der Histogramme können Segmente mit dem Distanzmaß

$$d(c_1, c_2) = WM(h_{c_1}, h_{c_2}) \tag{4.10}$$

gemäß ihrer volumetrischen Signatur miteinander verglichen werden, wobei WM die *Wasserstein-Metrik* [132], auch *Earth-Mover's-Metrik* genannt, darstellt. Nun werden benachbarte Segmente, deren normalisierte Distanz unterhalb eines Schwellwerts liegt, miteinander verschmolzen.

Optimierung. Abschließend werden die Segmenträndер optimiert, damit diese ausreißerfrei entlang von konkaven Oberflächenteilen verlaufen. Dies erfolgt über die Formulierung eines minimalen Schnittproblems auf Basis des bereits erwähnten k-Nächste-Nachbarn-Graphen G_a mit Kanten E_a und Knoten O_a und dem zu minimierenden Energiefunktional

$$E(\theta) = \frac{1}{10} \cdot \sum_{o \in O_a} D(\theta_o) + \sum_{(o_1, o_2) \in E_a} S(\theta_{o_1}, \theta_{o_2}), \tag{4.11}$$

wobei θ die gesuchte optimale Zuordnung jedes Punkts $o \in O_a$ zu einem Segment und θ_o die gesuchte optimale Zuordnung eines einzelnen Punkts o zu einem Segment darstellt. Der zugehörige Datenterm ist

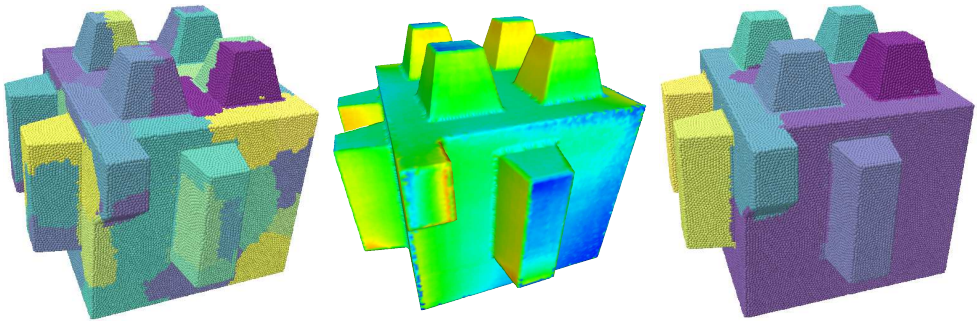
$$D(\theta_o) = \begin{cases} 0, & \text{falls } l_o = \theta_o \\ d_{\theta_o}(o)^{1.5}, & \text{sonst} \end{cases}, \tag{4.12}$$

wobei l_o das aktuell zu Punkt o zugeordnete Segment und $d_{\theta_o}(o)$ die Euklidischen Distanz zwischen Punkt o und dem Rand des gesuchten optimalen

Segments θ_o darstellt. Der Datenterm bewirkt, dass Punkte ihre Segmentzuordnung behalten. Es sei denn, sie befinden sich innerhalb eines schmalen Bands zwischen Segmentgrenzen. Der Glättungsterm

$$S(\theta_{o_1}, \theta_{o_2}) = \begin{cases} 0, & \text{falls } \theta_{o_1} = \theta_{o_2} \\ \frac{\alpha_o}{\pi} & \text{sonst} \end{cases}, \tag{4.13}$$

mit dem Winkel α_o zwischen den Oberflächennormalen an den Punkten o_1 und o_2 sorgt für eine Verringerung der Schnittkosten in konkaven Bereichen.



(a) (b) (c)

Abbildung 4.15.: Ergebnisse der Prozessschritte: a) *Übersegmentierung*, b) Berechnung der FDF abgebildet auf das Rot/Blau-Farbspektrum (Rot: kleinster Wert, Blau: größter) und c) *Volumetrische Verschmelzung*.

4.5.3. Punktzuweisung

In diesem Schritt wird die strukturierte Punktwolke O_s neu auf die gewonnenen Segmente C verteilt, sodass $\bigcup_{c \in C} O_c$ der Punktwolke O_s entspricht. Dies ist bei der Segmentierung mit dem *Sichtlinienansatz* nötig, da durch den Pipeline-Schritt *Proportionales Re-Sampling* die Eingangspunktwolke stark ausgedünnt wurde. Auch bei der *Übersegmentierung/Verschmelzung* ist eine Neuzuweisung der Punkte aus O_s nötig, da Segmente Punkte aus O_a und nicht aus der strukturierten Punktwolke O_s enthalten. Die Zuweisung wird für jeden Punkt $o \in O_s$ vorgenommen, indem o dem Cluster zugeordnet wird, welches den Punkt enthält, der o von allen Punkten aller Cluster am nächsten liegt.

4.5.4. Konstruktion konvexer Polytope

Die *Konstruktion konvexer Polytope* funktioniert wie in Kapitel 4.4.4.2 beschrieben. Jedoch wird der Schritt für jedes schwach-konvexe Segment $c \in C$ ausgeführt und die resultierenden konvexen Polytope zu einer Gesamtmenge vereinigt. Zudem kommt eine weitere Randbedingung hinzu (siehe Kapitel 4.4.4.2):

5. **Ebenennachbarschaft:** Bei der Zusammensetzung von konvexen Polytopen aus Ebenen sollen Ebenen selektiert werden, die im Ebenennachbarschaftsgraphen G_N (siehe Kapitel 4.5.1) verbunden sind.

Damit reduzieren sich die möglichen Ebenenkombinationen signifikant. Umgesetzt wird Randbedingung 5 bei der zufälligen Erzeugung neuer konvexer Polytope, indem bei der Ebenenauswahl nur benachbarte Ebenen in Betracht gezogen werden. Frühe Tests haben jedoch ergeben, dass eine vollständige Durchsetzung dieser Randbedingung zu nicht auffindbaren Polytopen führen kann, falls G_N unvollständig ist. Das kann bei imperfekten Eingabepunktwolken immer der Fall sein. Aus diesem Grund wird bei der Erzeugung konvexer Polytope zuerst eine Anzahl zufällig ausgewählter Ebenen als Basis verwendet, für die dann Nachbarebenen gesucht werden. Wichtig zu erwähnen ist, dass Segmente mehrere konvexe Polytope enthalten können, da deren Punktwolken in einigen Fällen nicht strikt-konvex, sondern schwach-konvex sind.

4.6. Evaluation

Das hier beschriebene Gesamtsystem wurde im Rahmen der Arbeit ausgiebig evaluiert. Die Ergebnisse der Evaluation sind in diesem Kapitel aufbereitet. Dabei wird sich hinsichtlich Struktur und Reihenfolge an den einzelnen Teilschritten der Systempipeline orientiert. Wenn nicht anders angegeben, wurde als Hardware eine Workstation mit einem *AMD Ryzen5 3600* Prozessor, 12 logischen 3,6GHz Kernen sowie 32GB RAM und einer *NVIDIA GeForce RTX 2060* Grafikkarte verwendet. Gewählte EA-Parameter für die *Konstruktion konvexer Polytope* finden sich in Anhang A.

4.6.1. Punktwolkenerfassung

In diesem Kapitel werden die für die Evaluation verwendeten Datensätze beschrieben, wie sie im Rahmen der *Punktwolkenerfassung* erzeugt wurden.

Trainingsdatenerzeugung. Der Datensatz zum Training des KNNs, welches für die Prädiktion von Primitiventypen in der *Punktwolkensegmentierung* verwendet wird, umfasst 1024 Punktwolken. Jede besteht aus ca. 50.000 Punkten (siehe Abbildung 4.2). Erzeugt wurde der Datensatz mit dem in Kapitel 4.4.1.1 beschriebenen Generator.

Evaluationsdatenerzeugung. Die für die Evaluation des Gesamtsystems verwendeten Modelle sind in Abbildung 4.17 dargestellt. Dabei haben die Modelle M9 und M10 ihren Ursprung in [125]. Da die Punktwolken durch einen synthetischen Abtastprozess eines manuell erstellten CAD-Modells entstanden sind (Ausnahme: M12), wurden diese vereinfacht und mit künstlichem, normalverteiltem Rauschen belegt ($\mu = 0, \sigma = 0,005$, siehe Abbildung

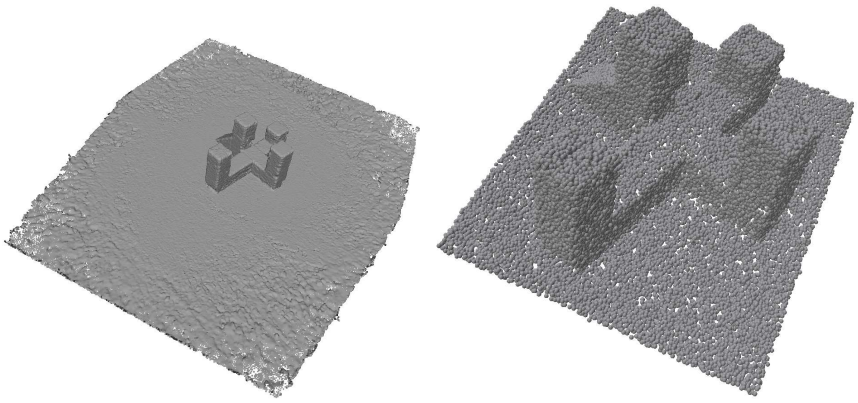
4.21a). M12 wurde mit einem photogrammetrischen Verfahren erzeugt (siehe Kapitel 4.4.1.2). Als Kamerasystem wurde das Mittelklasse-Smartphone *Mi 9 SE* der Firma *Xiaomi* verwendet und damit 60 Bilder eines Objekts bestehend aus gestapelten Holzbauklötzen aufgenommen. Die Bilder haben dabei eine Auflösung von 4000×3000 Pixel. Die Wahl fiel auf genau diese Holzbausteine, da sie über ausreichend diffus reflektierende Oberflächen verfügen, die zudem spezifisch texturiert sind. Damit erhöht sich die Robustheit des photogrammetrischen Rekonstruktionsprozesses. Die Berechnungszeit aller Prozessschritte auf der verwendeten Hardware betrug $12,528$ Minuten. Ergebnisse für Modell M12 sind in Abbildung 4.18 dargestellt. Die Größen aller Modellpunktwolken finden sich in Tabelle 4.2.

M1	**M2**	**M3**	**M4**	**M5**	**M6**
48k	56k	27k	45k	27k	27k
M7	**M8**	**M9**	**M10**	**M11**	**M12**
27k	23k	45k	38k	22k	57k

Tabelle 4.2.: Punktwolkengrößen aller Modelle.

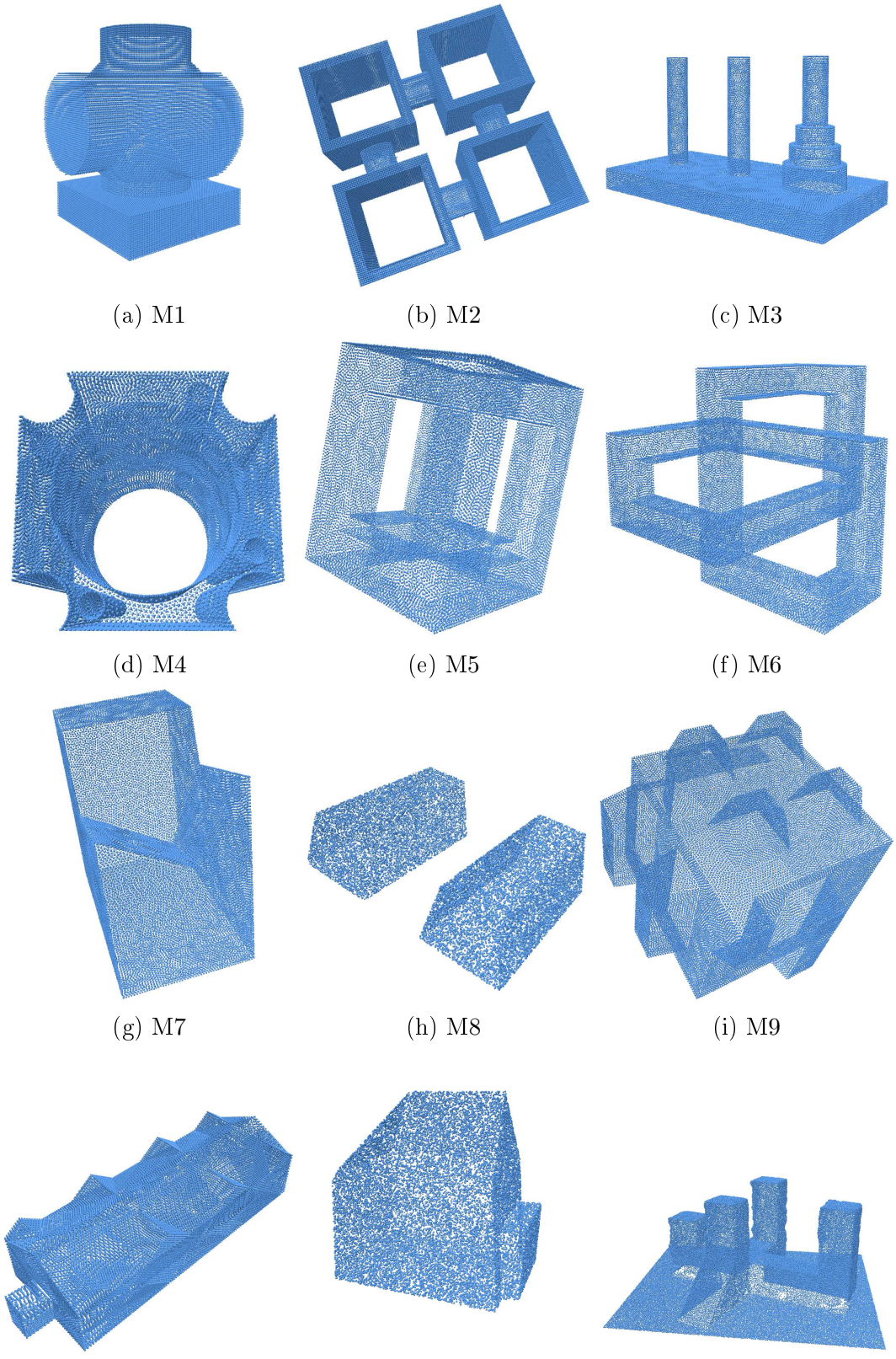
4.6.2. Punktwolkenaufbereitung

Hier relevant ist ausschließlich das Ergebnis der Aufbereitung der Punktwolke von Modell M12, da diese als einzige per *3D-Scanning* erzeugt wurde. Abbildung 4.16 zeigt die Eingabepunktwolke sowie das Ergebnis der Anpassungen (siehe Kapitel 4.4.2).



(a) (b)

Abbildung 4.16.: a) Eingabepunktwolke von Modell M12 mit $7,03$M Punkten.
b) Manuell aufbereitete Punktwolke mit 57k Punkten.



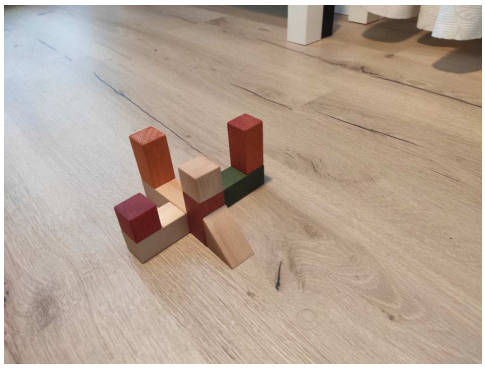
(a) M1 (b) M2 (c) M3

(d) M4 (e) M5 (f) M6

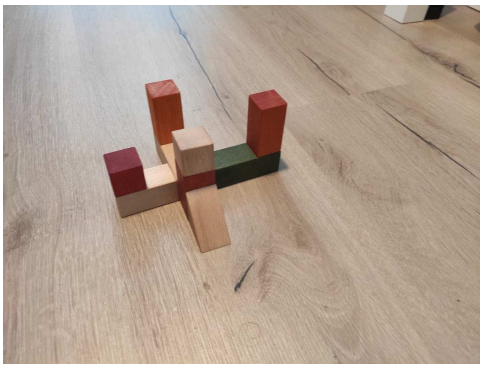
(g) M7 (h) M8 (i) M9

(j) M10 (k) M11 (l) M12

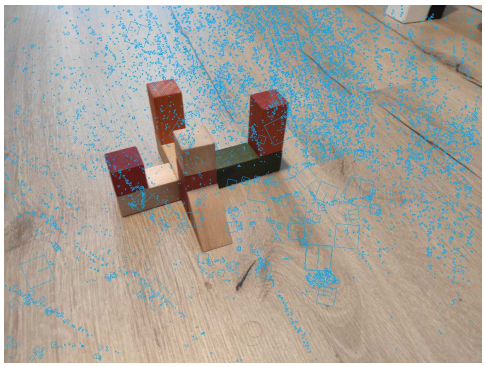
Abbildung 4.17.: Darstellung aller Modelle mittels feingranularer Punktabtastung der Oberfläche.



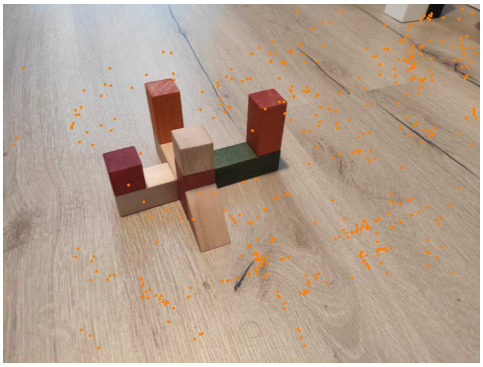
(a) Schritt 1: Bild der roten Kamera aus Abbildung e).



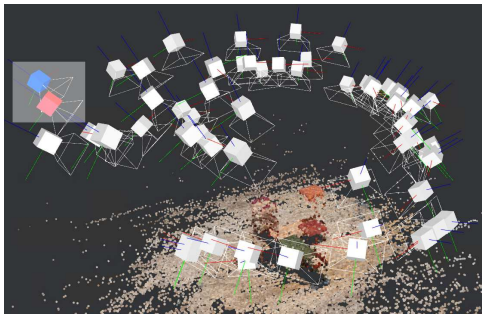
(b) Schritt 1: Bild der blauen Kamera aus Abbildung e).



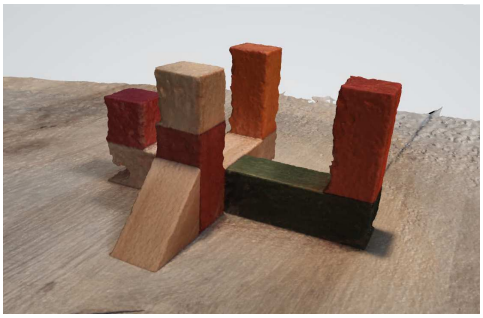
(c) Schritt 2: Extrahierte SIFT-Features aus dem Bild der blauen Kamera (hellblau).



(d) Schritt 3 und 4: Korrespondenzpunkte von roter und blauer Kamera (orange).



(e) Schritt 5: SfM-Ergebnisse mit 58 rekonstruierten Kameras und 46,2k Punkten.



(f) Schritt 6: Texturiertes Dreiecksnetz als Ergebnis des SVM-Schritts mit 7,03M Eckpunkten.

Abbildung 4.18.: Ergebnisse des in Kapitel 4.4.1.2 vorgestellten Systems zur Punktwolkenerzeugung mittels Photogrammetrie.

4.6.3. Punktwolkensegmentierung

Zentrales Element der Punktwolkensegmentierung ist die Prädiktion der Primitiventypen mittels eines KNNs. Dieses Kapitel widmet sich daher der initialen Evaluation geeigneter Netzarchitekturen (siehe Kapitel 4.6.3.1) sowie dem anschließend durchgeführten Trainingsprozess (siehe Kapitel 4.6.3.2). Zudem werden Ergebnisse diskutiert (siehe Kapitel 4.6.3.3).

4.6.3.1. Vorevaluation geeigneter Netzarchitekturen

Zur *Prädiktion des Primitiventyps* für jeden Punkt der Eingabepunktwolke, wie sie bei der Segmentierung der Punktwolke zum Einsatz kommt, wurde auf die Netzarchitektur *PointNet++*[142] zurückgegriffen (siehe Kapitel 4.4.3). Grundlage für diese Entscheidung bildet eine Evaluation dreier verschiedener Netzarchitekturen: *PointNet*[141], *PointNet++* sowie *PointCNN*[108]. Dazu wurde mit einer frühen Version des Generators, wie er in Kapitel 4.4.1.1 beschrieben wird, ein Datensatz mit 15.000 Punktwolken erstellt. Jede Punktwolke repräsentiert 20-80 Primitive (Zylinder, Kugeln und Quader zu gleichen Teilen) und umfasst 2048 Punkte. Robustheit und Generalisierbarkeit der Architekturen wurde dabei mit den Kenngrößen *Jaccard Ähnlichkeitskoeffizient* (JÄK) und *Präzision* auf den folgenden fünf Testdatensätzen mit jeweils 500 Punktwolken evaluiert:

1. **Keine Rotation:** Auf gleiche Weise generiert wie der Trainingsdatensatz, d.h. ohne Primitivenrotation.

2. **Rotation:** Wie *Keine Rotation* nur mit zusätzlich angewandter zufälliger Primitivenrotation.

3. **Weniger Punkte:** Wie *Keine Rotation*, jedoch mit nur der Hälfte der Punkte.

4. **Rauschen (schwach, stark):** Wie *Keine Rotation*, jedoch mit leichtem ($\mu = 0, \sigma = 0.5$) und starkem normalverteiltem Rauschen ($\mu = 0, \sigma = 1.5$).

5. **Mehr Primitive:** Wie *Keine Rotation*, jedoch mit einer erhöhten minimalen Anzahl von Primitiven (40).

Wie in Tabelle 4.3 aufgeführt, ist PointNet++ die mit Abstand beste Netzarchitektur für das gestellte Problem.

4.6.3.2. Trainingsprozess

Jede Punktwolke des Trainingsdatensatzes (1024 Punktwolken mit je ca. 50.000 Punkten) wird in $3 \times 3 \times 3$ Blöcke zerlegt. In jeder Trainingsepoche werden 1024 Punkte aus den Blöcken zufällig gezogen. Weiterhin werden Punkte in

Datensatz	Metrik	PointNet	PointNet++	PointCNN
Keine Rotation	Präzision	0,4291	0,9946	0,3632
	JÄK	0,1866	0,9873	0,1528
Rotation	Präzision	0,3915	0,9341	0,3441
	JÄK	0,1627	0,8038	0,1455
Weniger Punkte	Präzision	0,4283	0,9945	0,3425
	JÄK	0,1859	0,9834	0,1351
Rauschen schwach	Präzision	0,4288	0,9947	0,3523
	JÄK	0,1870	0,9865	0,1480
Rauschen stark	Präzision	0,4272	0,9930	0,3523
	JÄK	0,1867	0,9754	0,1506
Mehr Primitive	Präzision	0,4321	0,9960	0,4216
	JÄK	0,1932	0,9943	0,1988

Tabelle 4.3.: Evaluationsergebnisse für die Netzarchitekturen *PointNet*, *PointNet++* und *PointCNN* für die gewählten Testdatensätze.

ihrer Reihenfolge vertauscht und zusätzlich mit bis zu 0,1% der Gesamtblockgröße zufällig verschoben. Diese Schritte vergrößern den Ursprungsdatensatz um synthetisch erzeugte Variationen der Ausgangsdaten. Diese Datenaugmentierung verhindert eine Überanpassung des Modells (siehe Kapitel 2.5.2).
Wie Abbildung 4.19a und 4.19b zeigen, liefert das Modell nahezu perfekte Resultate bzgl. Wahrheitsmatrix und OCB-Kurven auf einem zufällig generierten Testdatensatz.

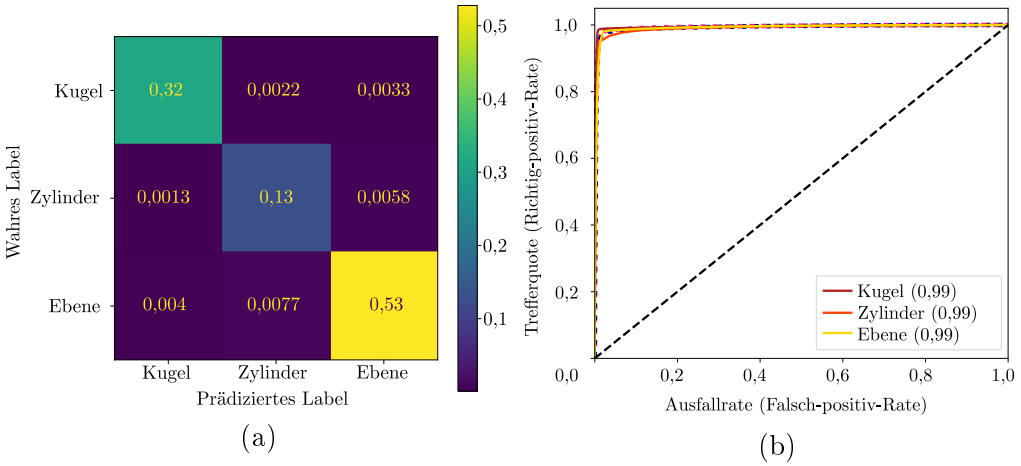


Abbildung 4.19.: a) Aus der Wahrheitsmatrix geht hervor, dass 98,0% der Punkte aus dem Testdatensatz richtig klassifiziert worden sind (Präzision: 0,98). b) Klassenspezifische OCB-Kurven.

4.6.3.3. Qualität und Robustheit

Im Folgenden werden Ergebnisse der *Punktwolkensegmentierung* für die Modelle M1-M12 hinsichtlich der Qualität und allgemeinen Verfahrensrobustheit präsentiert. In den Abbildungen 4.21 und 4.22 finden sich entsprechende Visualisierungen, aus denen hervorgeht, dass die *Prädiktion des Primitiventyps* auf allen Modellen gute Ergebnisse liefert. Quantitativ wird dieser Eindruck durch hohe Präzisionswerte untermauert (siehe Tabelle 4.4).

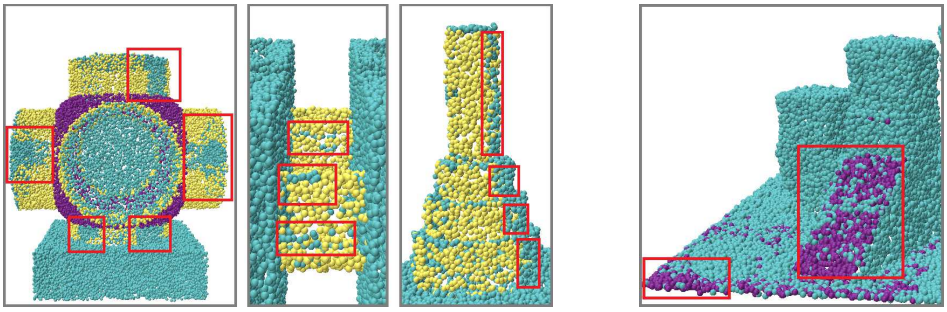
M1	M2	M3	M4	M5	M6
0,9415	0,9506	0,9344	0,9746	1,0	1,0
M7	**M8**	**M9**	**M10**	**M11**	**M12**
0,9961	1,0	1,0	0,9992	1,0	0,9545

Tabelle 4.4.: Präzision der *Prädiktion des Primitiventyps* für alle Modelle.

Auftretende Fehler lassen sich in zwei Kategorien aufteilen:

1. Zylindersegmente, die fälschlicherweise als Ebenen klassifiziert werden (siehe Abbildung 4.20a, Farbkodierung wie in Abbildung 4.21b).
2. Ebenen, die fälschlicherweise als Kugelsegmente klassifiziert werden (siehe Abbildung 4.20b, Farbkodierung wie in Abbildung 4.21b).

Beide Fehlerkategorien lassen sich auf ein grundlegendes Problem der Oberflächenabtastung und -rekonstruktion zurückführen. So ist bei schwach gekrümmten und grob abgetasteten Flächen die Koordinaten- und Normalenvariation innerhalb von Punktnachbarschaften oft nicht von Rauschartefakten auf abgetasteten Ebenen zu unterscheiden und vice versa. Eine weitere Ursache für auftretende Fehler könnte die Unterrepräsentation von Zylindern und Kugeln im Trainingsdatensatz sein (siehe Abbildung 4.19a). Die beschriebenen Fehlklassifikationen führen zu zusätzlichen Segmenten und weiterhin zu falsch eingepassten Primitiven (siehe Kapitel 4.6.4).



(a) Fehlerkategorie 1, M1, M2 und M3. (b) Fehlerkategorie 2, M12.

Abbildung 4.20.: Prädiktionsfehler im Detail.

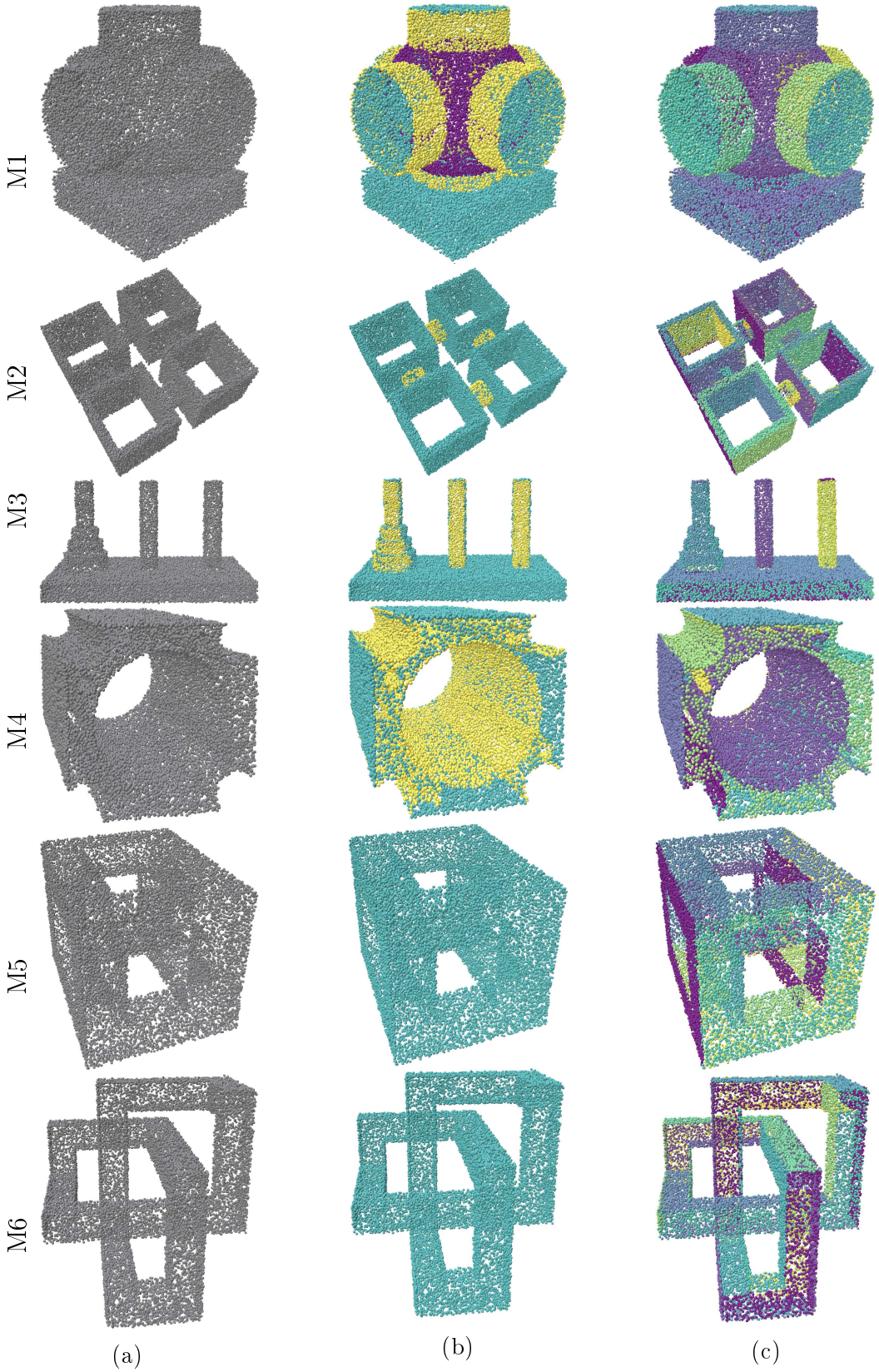


Abbildung 4.21.: a) Eingabepunktwolke, b) Prädizierte Primitiventypen (Ebenen in Türkis, Zylinder in Gelb, Kugeln in Lila) und c) Segmente gleichen Typs für die Modelle M1-M6.

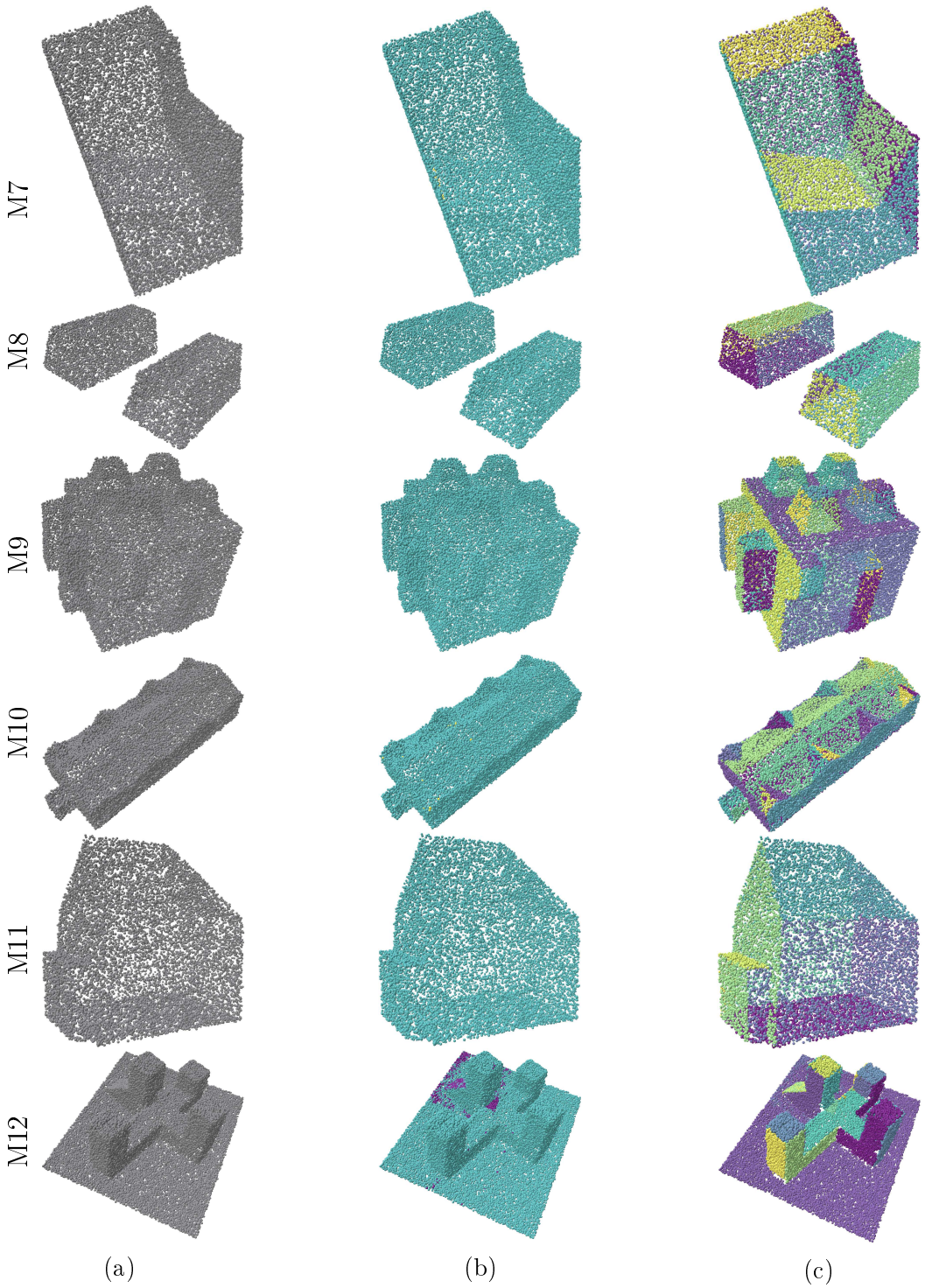


Abbildung 4.22.: a) Eingabepunktwolke, b) Prädizierte Primitiventypen (Ebenen in Türkis, Zylinder in Gelb, Kugeln in Lila) und c) Segmente gleichen Typs für die Modelle M7-M12.

4.6.3.4. Laufzeitbetrachtungen

Die Ausführungslaufzeiten für diesen Prozessschritt finden sich in Abbildung 4.23. Die *Prädiktion des Primitiventyps* ist dabei der dominante Faktor, was die Laufzeit betrifft. Dabei orientiert sich das Laufzeitverhalten erwartungsgemäß an der Größe der Punktwolke (siehe Tabelle 4.2). Interessant ist die relativ hohe Laufzeit für Modell M1, M3 und M4. Dies lässt sich damit erklären, dass für diese Modelle eine *Quaderpartitionierung* in $2 \times 2 \times 2$ Quader zu qualitativ schlechteren Prädiktionsergebnissen führt. Ohne Partitionierung ist die Prädiktion jedoch weniger laufzeiteffizient.

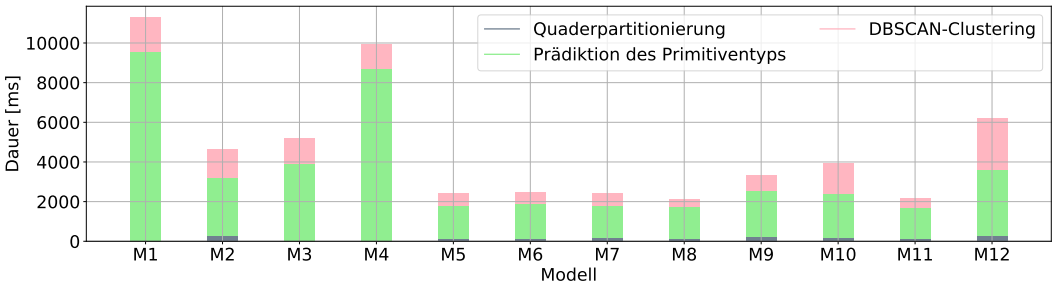


Abbildung 4.23.: Laufzeiten des Prozessschritts *Punktwolkensegmentierung*.

4.6.4. Primitivendetektion und -einpassung

In diesem Kapitel wird der Prozessschritt *Primitivendetektion und -einpassung* genauer beleuchtet und hinsichtlich Ergebnisqualität (siehe Kapitel 4.6.4.1) und Laufzeit (siehe Kapitel 4.6.4.2) diskutiert. Wichtig ist dabei u.a. die Fragestellung, ob der vorangegangene Segmentierungsschritt zu einer Erhöhung der Ergebnisqualität und Robustheit des Schritts *Detektion & Einpassung von unbeschränkten Primitiven* führt und damit das Gesamtergebnis positiv beeinflusst.

4.6.4.1. Qualität und Robustheit

In diesem Kapitel wird Ergebnisqualität und Robustheit dieses Prozessschritts diskutiert. Dazu kommen vor allem Visualisierungen der Ergebnisse zum Einsatz, um deren Qualität bewerten zu können.

Detektion & Einpassung von unbeschränkten Primitiven. Der wichtigste Aspekt für die Beurteilung der Ergebnisqualität ist die Frage, wie genau Primitive eingepasst werden können und wie robust dies über mehrere Iterationen ist. In den Abbildungen 4.25, 4.26 und 4.27 werden die jeweils besten Ergebnisse dieses Schritts aus drei Durchläufen mit und ohne vorangegangene Segmentierung dargestellt. Für die Modelle M1, M2, M4, M5, M6, M7, M8, M9 und M12 gibt es keine signifikanten Qualitätsunterschiede,

sieht man von unterschiedlichen Ebenenaufteilungen ab. Ohne Segmentierung werden Ebenen eher zu einer einzelnen Ebene zusammengefasst (siehe z.B. M12 in Abbildung 4.27). Für die restlichen Modelle zeigen sich klare Vorteile des Segmentierungsschritts. So erfolgt die korrekte Einpassung aller Zylinder in M3 nur bei aktivierter Segmentierung. Für M10 werden mit aktivierter Segmentierung alle Ebenen in der Dachkonstruktion vollständig erkannt. Ist diese nicht aktiv, werden in diesem Bereich nur zwei Ebenen erkannt. Ohne aktive Segmentierung beinhaltet M11 mehrere Ebenen in räumlich zusammenhängenden Bereichen.
Weiterhin lässt sich die Verbesserung der Robustheit des Verfahrens durch die Segmentierung messen. Dazu wird für Modelle, die nur Ebenen enthalten sollten (M5-M12), überprüft, ob auch andere Primitive in der Ergebnismenge enthalten sind. Abbildung 4.24 zeigt das Ergebnis dieser Betrachtung für insgesamt drei Durchläufe. Für die Modelle M9 bis M12 werden ohne vorhergehende Segmentierung fälschlicherweise Primitive erkannt, die keine Ebenen sind. Bei den Modellen M5 bis M8 ist dies nicht der Fall. Mit vorher ausgeführter *Punktwolkensegmentierung* wird nur in Modell M12 ein einzelnes falsches Primitiv erkannt. Der Grund dafür liegt in der falschen Klassifizierung einiger Ebenenpunkte als Kugeln im Schritt *Prädiktion des Primitiventyps* während der Segmentierung (siehe Abbildung 4.20b).

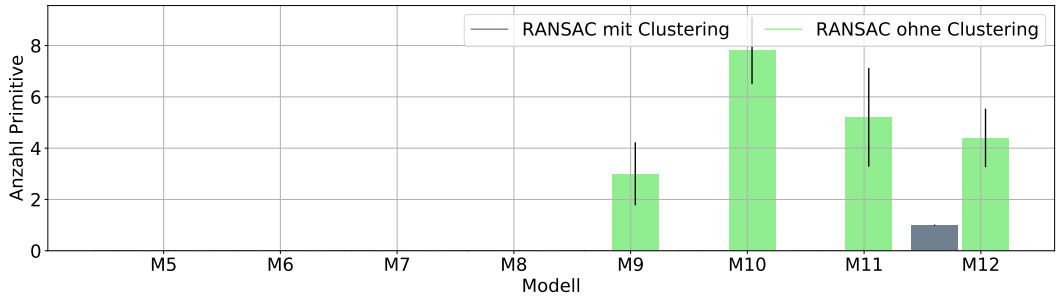


Abbildung 4.24.: Anzahl der Primitive, die keine Ebenen sind und fälschlicherweise der Ergebnismenge hinzugefügt wurden.

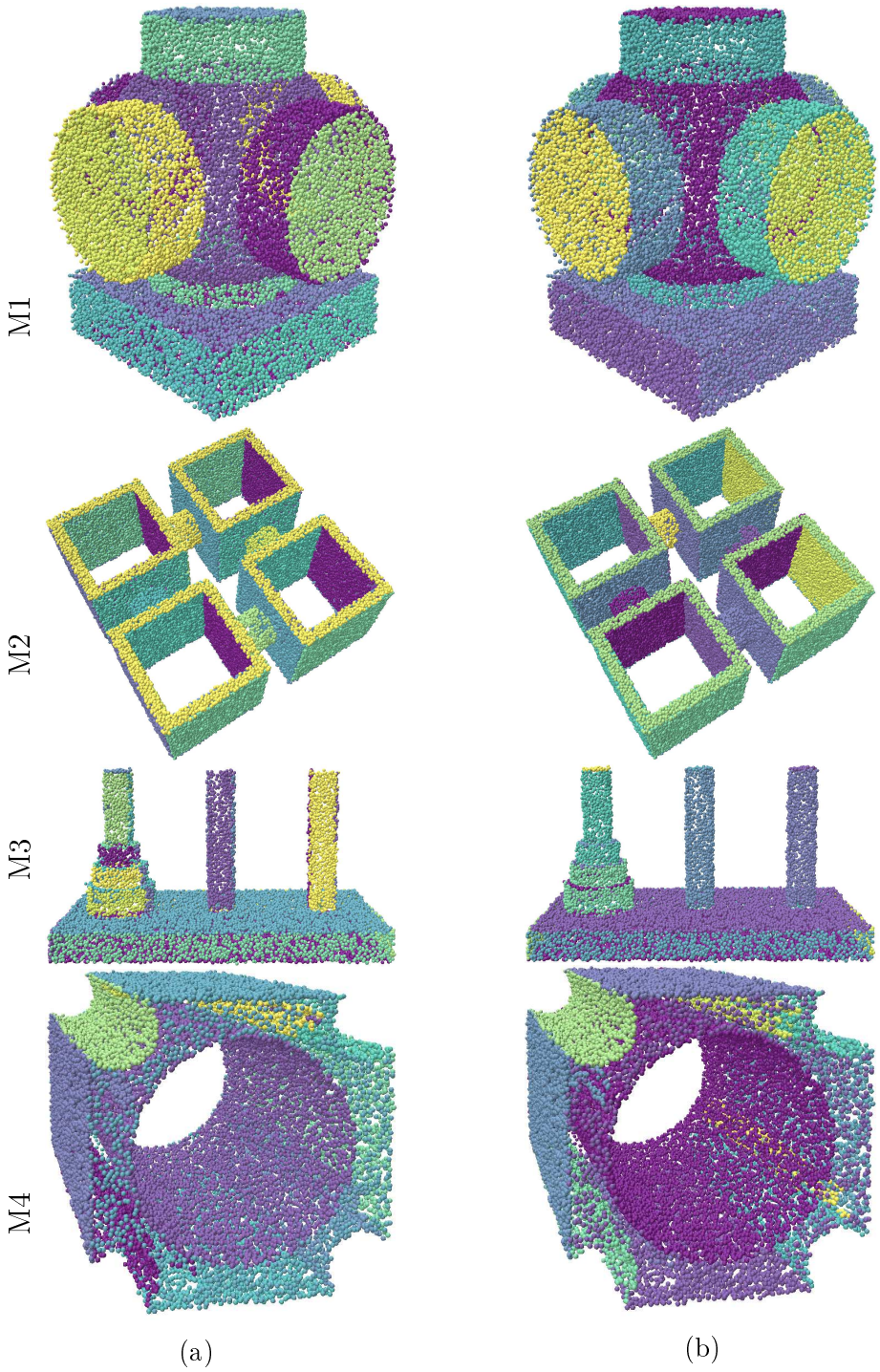


Abbildung 4.25.: Punkte zugeordnet zu eingepassten unbeschränkten Primitiven für Modelle M1 bis M4: a) mit Segmentierung, b) ohne. Farben repräsentieren Punktwolkensegmente, die einem Primitiv zugeordnet sind.

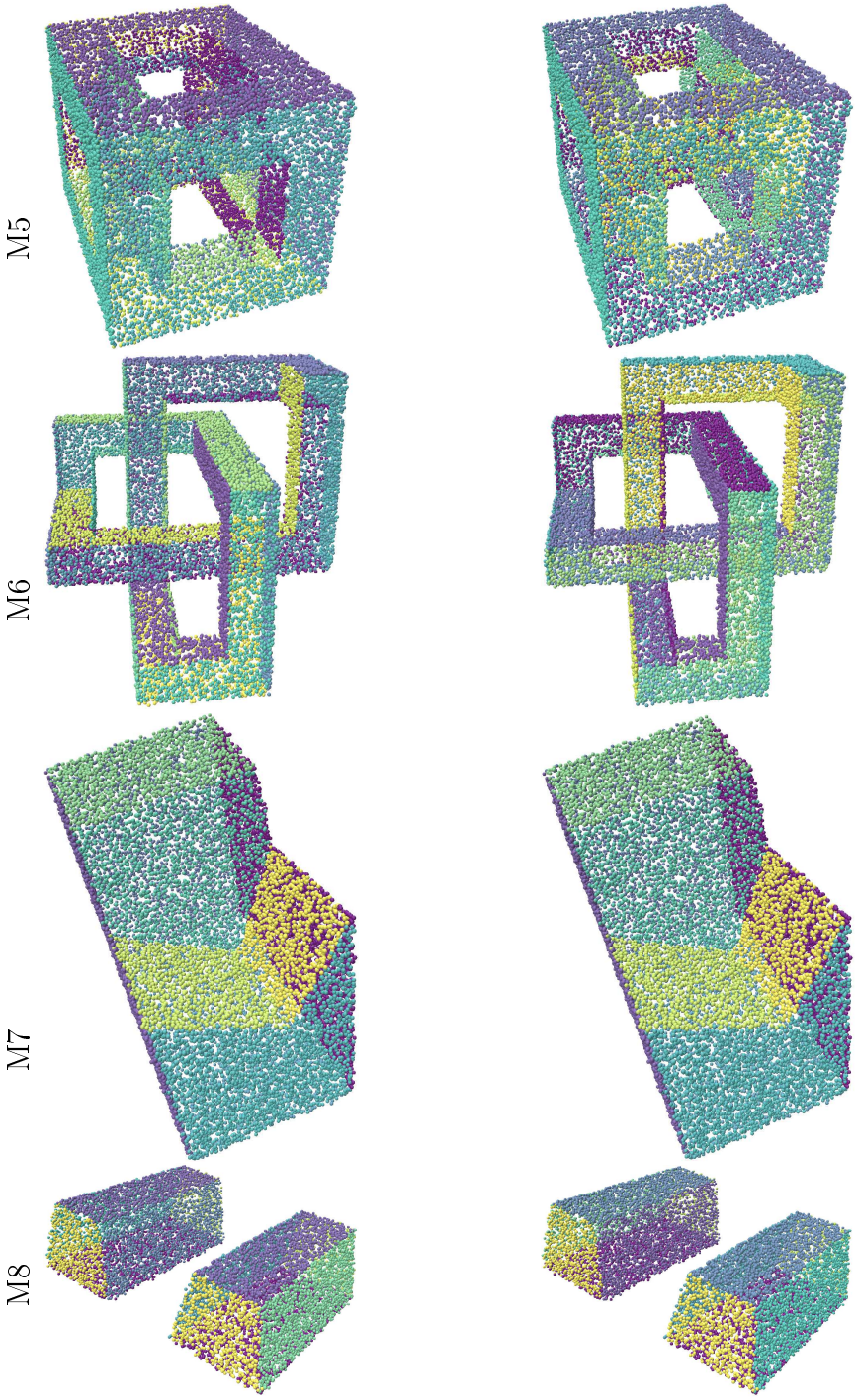


Abbildung 4.26.: Punkte zugeordnet zu eingepassten unbeschränkten Primitiven für Modelle M5 bis M8: a) mit Segmentierung, b) ohne. Farben repräsentieren Punktwolkensegmente, die einem Primitiv zugeordnet sind.

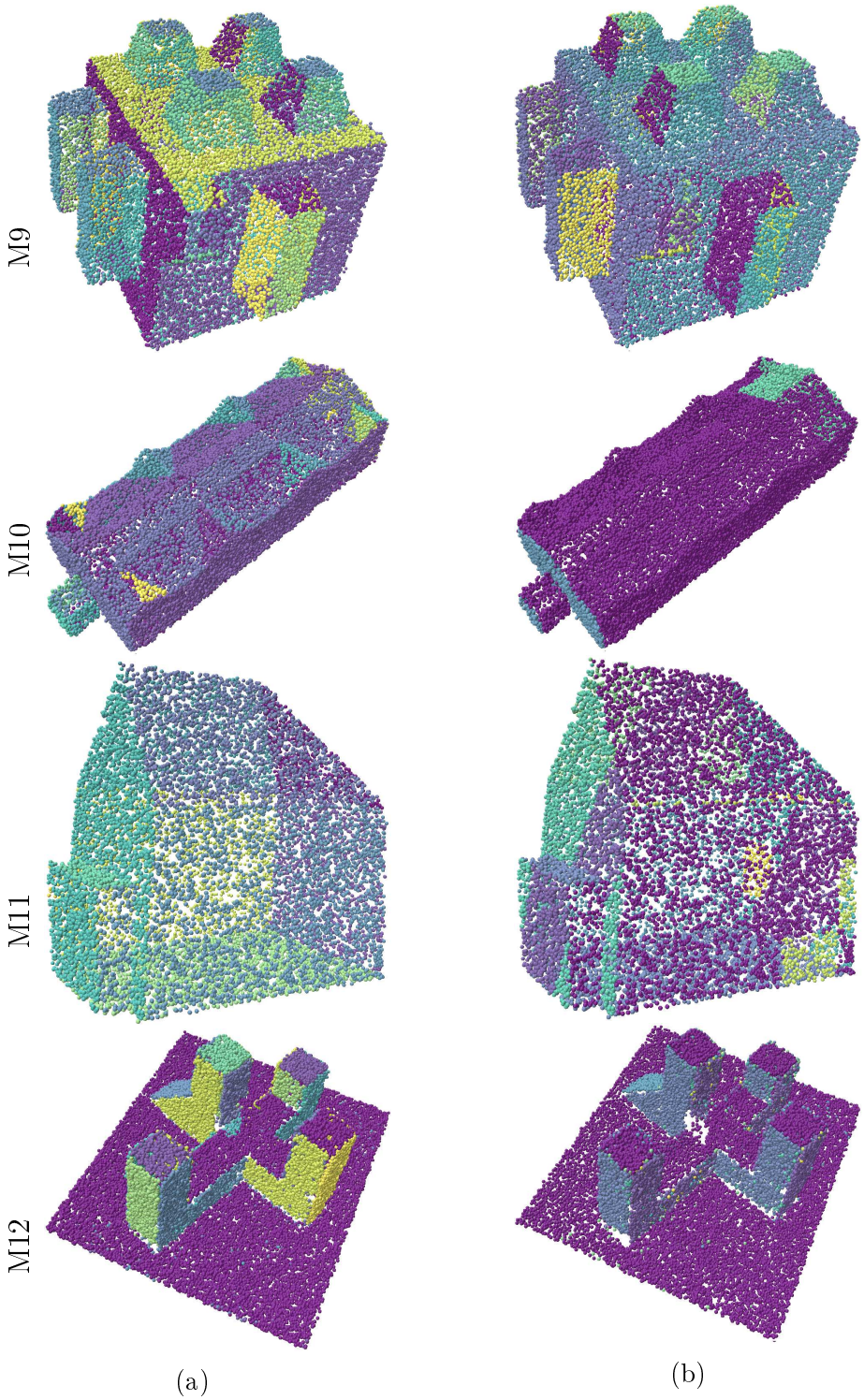
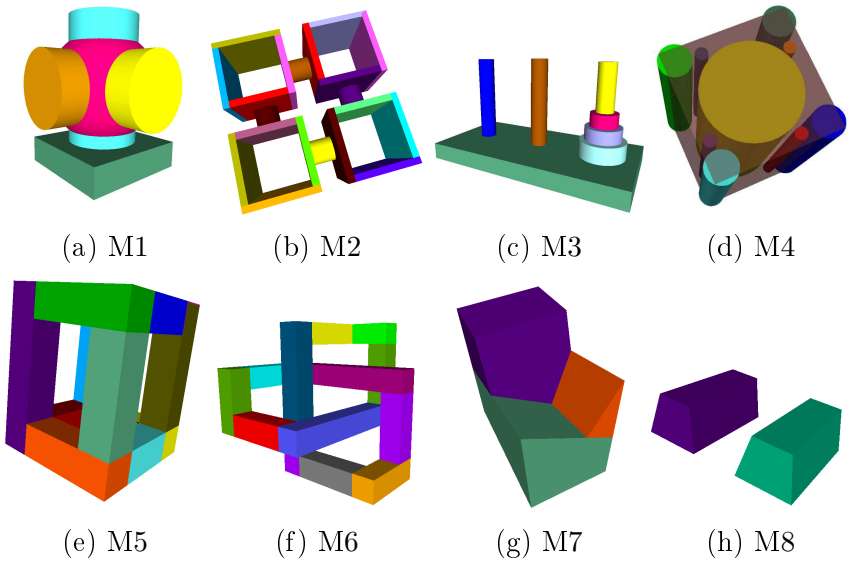


Abbildung 4.27.: Punkte zugeordnet zu eingepassten unbeschränkten Primitiven für Modelle M9 bis M12: a) mit Segmentierung, b) ohne. Farben repräsentieren Punktwolkensegmente, die einem Primitiv zugeordnet sind.

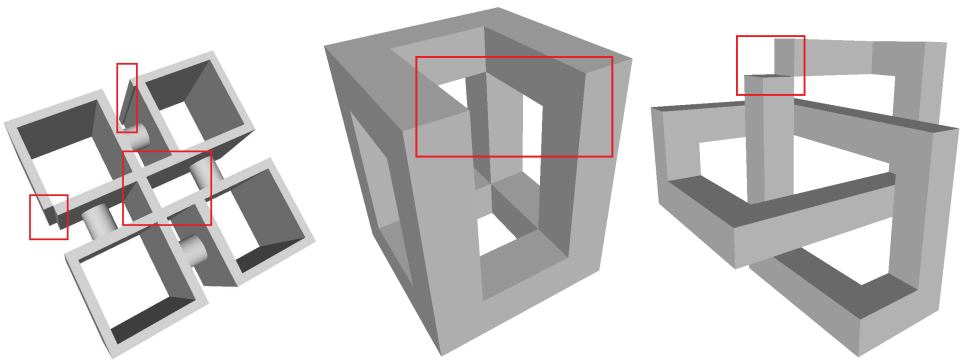
Konstruktion konvexer Polytope. In Abbildung 4.28 sind die besten Ergebnisse dieses Prozessschritts für die Modelle M1 bis M8 basierend auf drei Durchläufen aufgeführt. Daraus ist ersichtlich, dass das System alle in den Modellen vorkommenden konvexen Polytope rekonstruieren kann. Zu beachten ist jedoch das nicht-deterministische Verhalten des zum Einsatz kommenden EAs, welches unterschiedliche Ergebnisse bei mehrmaliger Ausführung zur Folge haben kann. Abbildung 4.29 zeigt dieses Phänomen anhand einiger Negativbeispiele, bei denen nur lokale Maxima gefunden werden.



(a) M1 (b) M2 (c) M3 (d) M4

(e) M5 (f) M6 (g) M7 (h) M8

Abbildung 4.28.: Eingepasste Primitive als Ergebnis von Prozessschritt *Primitivendetektion und -einpassung.*



(a) M2 (b) M5 (c) M6

Abbildung 4.29.: Auftretende Artefakte innerhalb der vom EA erzeugten Menge konvexer Polytope.

4.6.4.2. Laufzeitbetrachtungen

In diesem Kapitel wird das Laufzeitverhalten dieses Prozessschritts analysiert.

Detektion & Einpassung von unbeschränkten Primitiven. Für diesen Schritt finden sich Laufzeitmessungen in Abbildung 4.30. Da es sich bei dem eingesetzten RANSAC-Verfahren zur Einpassung von Primitiven um eine nicht-deterministische Methode handelt, wurde jeweils das Ergebnis mit der höchsten Qualität für die Laufzeitmessung herangezogen. Die Aufteilung des Problems mittels Segmentierung, wie es im Schritt *Punktwolkensegmentierung* durchgeführt wird, hat in einigen Fällen negative Auswirkungen auf die Laufzeit (M2, M3, M4, M5, M6, M9 und M12). Dies ist jedoch bei Laufzeiten unter 1s und der generellen Zielsetzung hin zu mehr Robustheit und Ergebnisqualität weniger relevant. Bei den Modellen M1, M8, M10 und M11 ergeben sich mit Segmentierung Geschwindigkeitsvorteile. Wichtig ist hier, dass der vorhergehende Segmentierungsschritt um zwei Größenordnungen mehr Zeit in Anspruch nimmt (siehe Abbildung 4.23) und sich somit die Zeitersparnis relativiert.

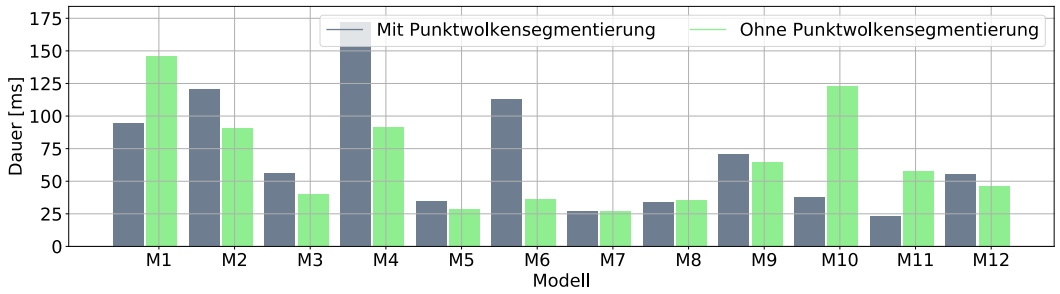


Abbildung 4.30.: Laufzeiten des Prozessschritts *Detektion & Einpassung von unbeschränkten Primitiven.*

Konstruktion konvexer Polytope. Alle Laufzeiten der Teilschritte dieses Schritts sind in Abbildung 4.31 dargestellt. Die Laufzeit des EAs wird in Abbildung 4.32 separat betrachtet. In beiden Fällen wurde die Laufzeit der Durchläufe mit den qualitativ besten Ergebnissen gewählt. Einfachere Modelle mit nur einem oder zwei Polytopen (M1, M3, M4, M8) benötigen signifikant weniger Berechnungszeit. Komplexere Modelle, wie M2, M5 und M6, haben hingegen verhältnismäßig lange Laufzeiten. Zu beachten sind zudem die EA-Laufzeiten, die z.B. für Modell M5 bei über 700s liegen.

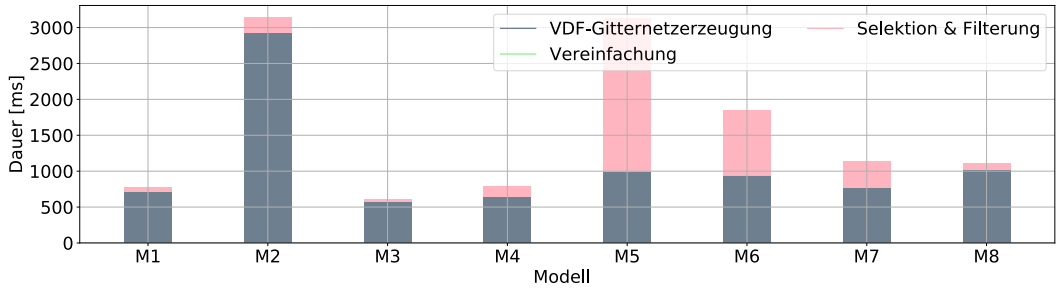


Abbildung 4.31.: Laufzeiten des Prozessschritts *Konstruktion konvexer Polytope*, alle Teilschritte bis auf EA.

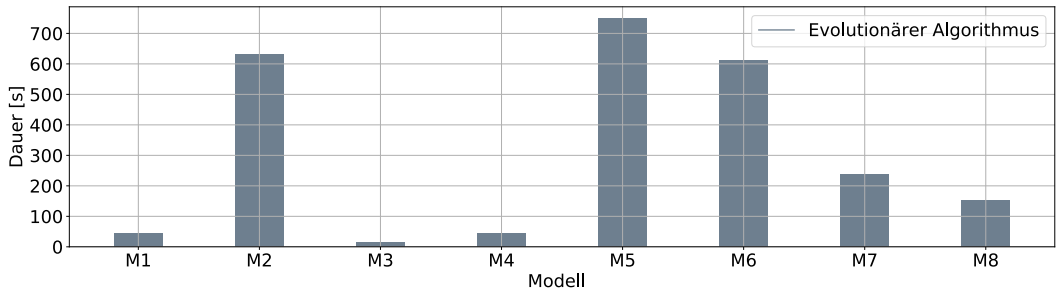


Abbildung 4.32.: Laufzeiten des Prozessschritts *Konstruktion konvexer Polytope*, EA.

4.6.5. Unterstützung komplexer konvexer Polytope

Das Gesamtsystem wurde durch eine zusätzliche Segmentierung der Eingabepunktwolke in schwach-konvexe Elemente erweitert, um zum einen gute Ergebnisse für komplexere Modelle (M9, M10, M11, M12) zu erzielen und zum anderen die Robustheit bei bereits lösbaren Modellen (M5, M6, M7) zu erhöhen (siehe Kapitel 4.5). Im Folgenden werden Ergebnisse hinsichtlich ihrer Qualität bzw. Robustheit (siehe Kapitel 4.6.5.1) sowie hinsichtlich des Laufzeitverhaltens analysiert (siehe Kapitel 4.6.5.2).

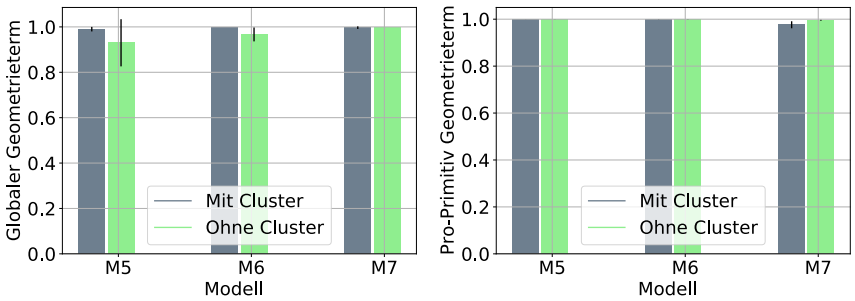
4.6.5.1. Qualität und Robustheit

Im Folgenden werden Qualität und Robustheit der einzelnen Prozessschritte innerhalb der erweiterten Pipeline diskutiert.

Ermittlung von Ebenennachbarschaften. Die sich ergebende strukturierte Punktwolke ist ein guter Indikator für die zu erwartende Qualität der erzeugten Ebenennachbarschaftsgraphen (siehe Abbildung 4.35a). Punkte in Lila markieren detektierte Kantenlinien, Punkte in Gelb Ecken. Generell sind die Resultate der Detektion bis auf vereinzelt fehlende Eckpunkte gut. Eine Ausnahme bildet M12, was auf die Herkunft der Eingabepunktwolke zurückzuführen ist (siehe Kapitel 4.6.1).

Schwach-konvexe Segmentierung. Um eine bessere Vergleichbarkeit beider Ansätze zu erreichen, werden in den Abbildungen 4.35b und 4.35c bereits die jeweiligen Ergebnisse nach der durchgeführten *Punktzuweisung* dargestellt. Für die Modelle M5 und M12 ergeben sich bessere Segmentierungen mit dem *Sichtlinienansatz*. Für die Modelle M7, M9, M10 und M11 hingegen ist die *Übersegmentierung/Verschmelzung* überlegen. Bei Modell M6 erzielen beide Methoden sehr ähnliche Ergebnisse. Interessant sind v.a. die Ergebnisse für Modell M10. Hier ist keine der beiden Methoden in der Lage, die Details der abgebildeten Dachkonstruktion korrekt zu partitionieren.

Konstruktion konvexer Polytope. Für diesen Schritt wurde die jeweils beste konvexe Segmentierung aus dem vorangegangenen Schritt gewählt. Abbildung 4.33 zeigt, dass das Ziel, die Robustheit bereits lösbarer Modelle durch die schwach-konvexe Segmentierung zu verbessern, erreicht werden kann. Dort werden Mittelwert und Standardabweichung der Werte der Geometrieterme mit und ohne Segmentierung dargestellt. Zur Darstellung wurden entsprechende Werte aus jeweils drei Durchläufen herangezogen. Vor allem für die Modelle M5 und M6 ergeben sich Vorteile für die schwach-konvexe Segmentierung im globalen Geometrieterm G_{O_v}. Dies wirkt sich auf die Qualität der Ergebnisse sichtbar aus, da sich genau dieser Unterschied im Ausbleiben der in Abbildung 4.29 gezeigten Artefaktbildung für diese Modelle zeigt.

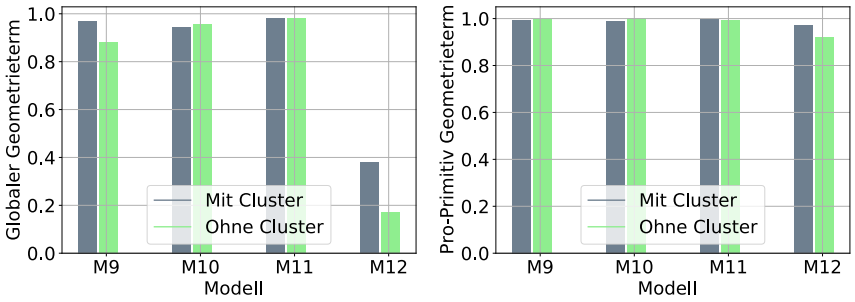


(a) G_{O_v} (b) V_G

Abbildung 4.33.: Robustheit der Geometrieterme für die Modelle M5, M6 und M7 mit und ohne Nutzung der schwach-konvexen Segmentierung.

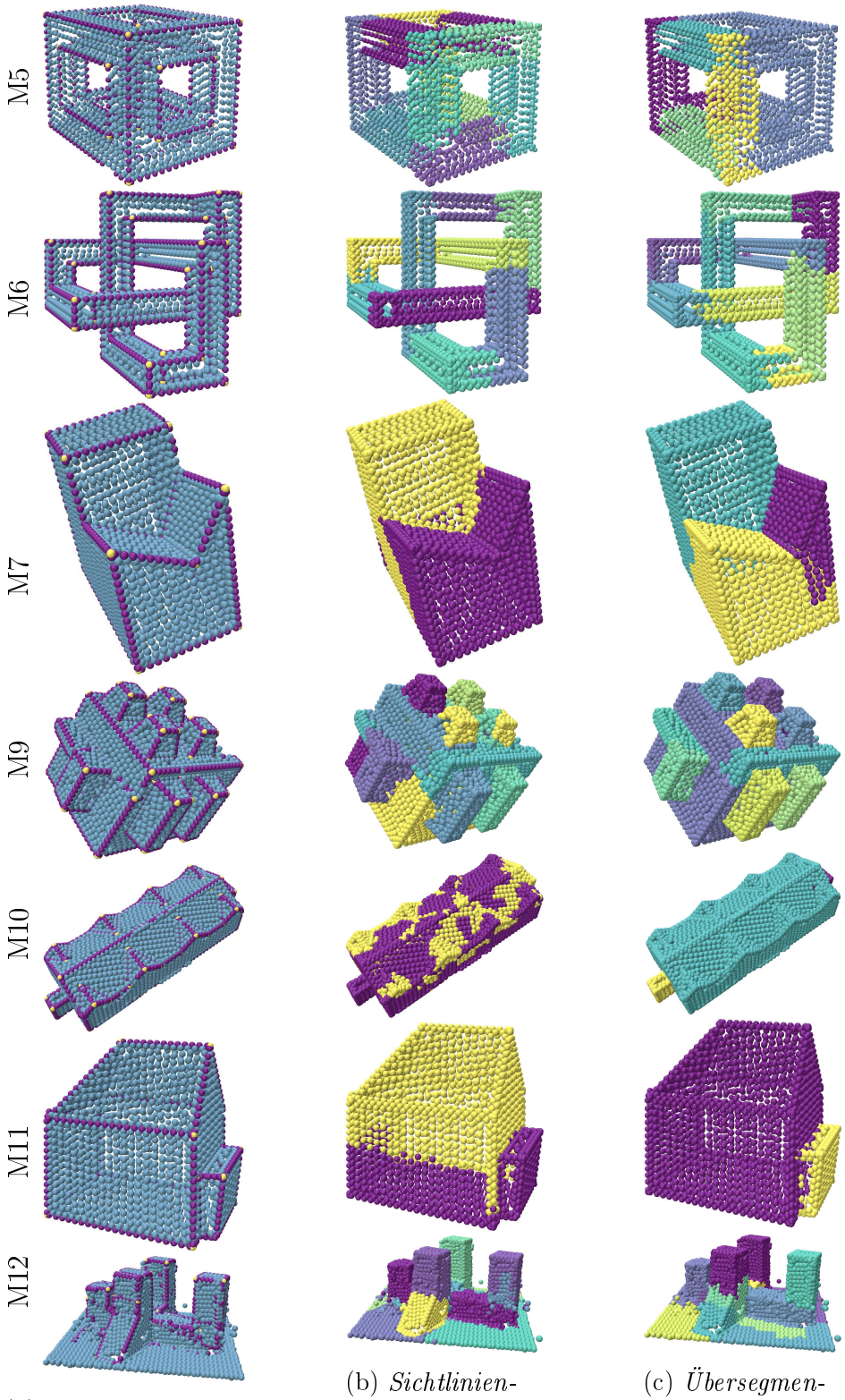
Ein weiteres Ziel ist die Unterstützung komplexer Modelle, wie z.B. die Modelle M9-M12. Dies konnte ebenfalls erreicht werden, wie die Abbildungen 4.34 und 4.36 anhand der besten Ergebnisse aus drei Durchläufen zeigen. Im Vergleich zur Rekonstruktion ohne schwach-konvexe Segmentierung können in den Modellen bzgl. des globalen Geometrieterms G_{O_v} Verbesserungen ausgemacht werden (siehe Abbildung 4.34a). Ebenfalls zu erkennen ist, wie sehr die schwach-konvexe Segmentierung zur Problemvereinfachung beiträgt

und, wie stark die Ergebnisqualität von einer guten Segmentierung abhängig ist. Sind die Qualitätsunterschiede bei Modell M9 und M12 deutlich, ist das nur im Detail bei M11 sichtbar, da für letzteres die Segmentierung nicht relevant ist. Bei Modell M10 resultiert die Variante ohne Segmentierung in einem marginal besseren Ergebnis. Der Grund dafür ist die mangelhafte Segmentierung, die das komplexe Objekt in nur drei Segmente unterteilt.
Der vergleichsweise niedrige Wert für G_{O_v} für Modell M12 lässt sich mit der Herkunft des Datensatzes erklären. So führt die im Erzeugungsvorgang (siehe Kapitel 4.4.1.2) rekonstruierte Bodenebene bei der *VDF-Gitternetzerzeugung* zu fälschlicherweise angenommenen Volumenteilen, die jedoch bei der Einpassung von konvexen Polytopen weitestgehend ignoriert werden.
Der Pro-Primitiv-Geometrieterm V_G verbessert sich nicht signifikant (siehe Abbildung 4.34b), da diese Metrik ausschließlich misst, wie weit ein Primitiv sich innerhalb der Oberfläche des gesuchten Objekts befindet. Sie misst damit nicht, wie vollständig das gesuchte Objekt im Gesamten rekonstruiert wurde.



(a) G_{O_v} (b) V_G

Abbildung 4.34.: Robustheit der Geometrieterme für die Modelle M9-M12 mit und ohne Nutzung der schwach-konvexen Segmentierung.

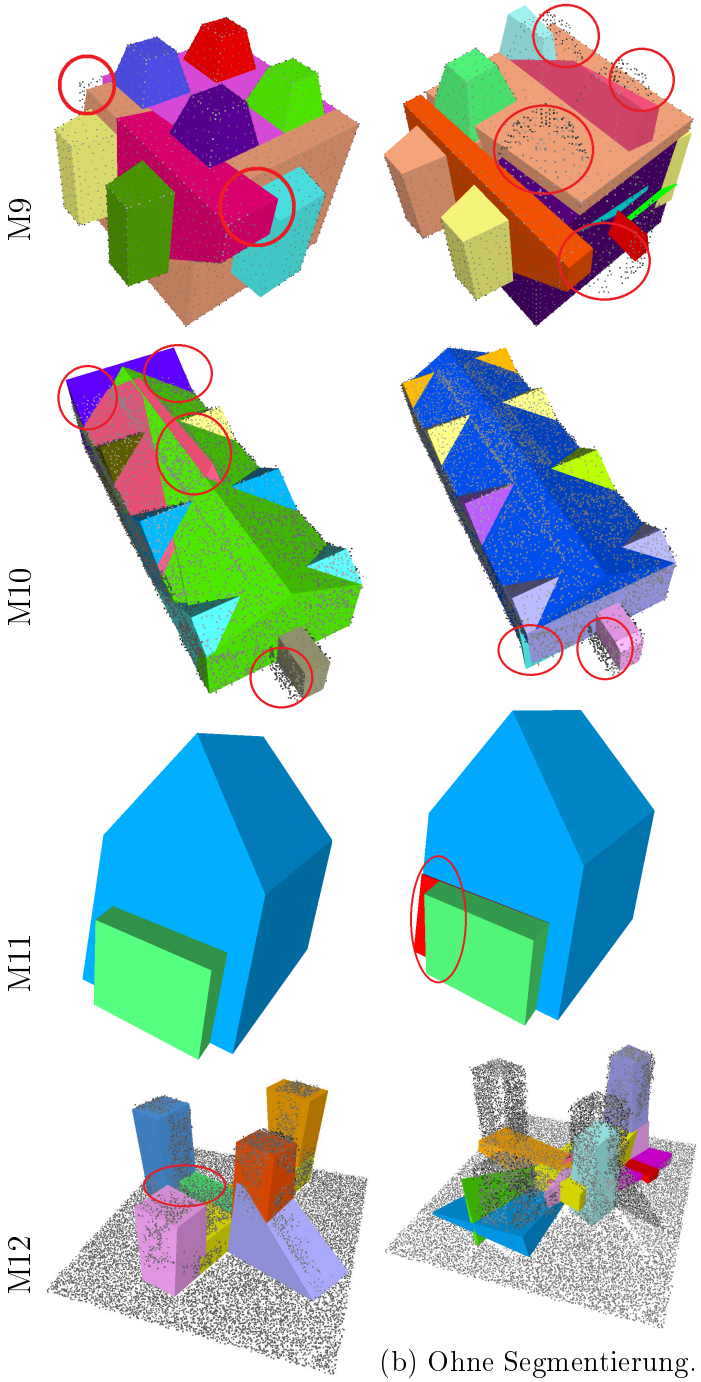


(a) Vorverarbeitung.

(b) *Sichtlinien-ansatz.*

(c) *Übersegmentierung/Verschmelzung.*

Abbildung 4.35.: Ergebnisse der *Ermittlung von Ebenennachbarschaften* a) sowie der *schwach-konvexen Segmentierung* b), c).



(a) Mit Segmentierung.

(b) Ohne Segmentierung.

Abbildung 4.36.: Ergebnisse für die *Konstruktion konvexer Polytope* mit den Modellen M9-M12. Nicht offensichtliche Fehler sind mit roten Kreisen markiert. Zur besseren Darstellung fehlerhafter Geometrie wurden bei Bedarf zugehörige Zielpunktwolken eingeblendet (grau).

4.6.5.2. Laufzeitbetrachtungen

Im Folgenden werden die gemessenen Laufzeiten für die erweiterte Pipeline aufgeführt und diskutiert.

Ermittlung von Ebenennachbarschaften. Die Laufzeiten zu diesem Schritt finden sich in Abbildung 4.37. Im Vergleich zu den folgenden Verarbeitungsschritten, fallen diese nicht ins Gewicht.

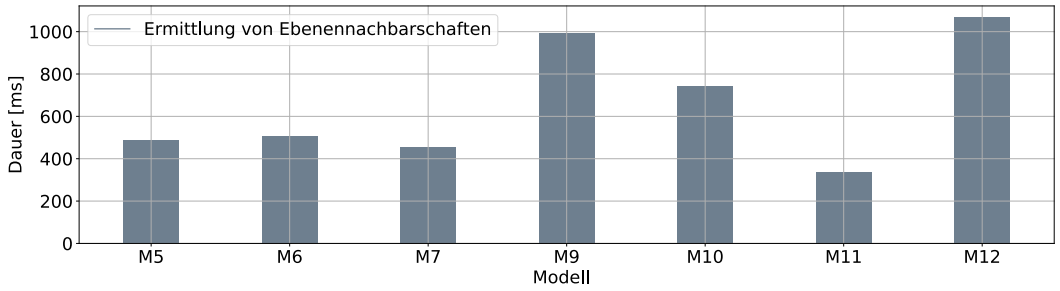


Abbildung 4.37.: Laufzeiten der *Ermittlung von Ebenennachbarschaften*.

Schwach-konvexe Segmentierung. Für den *Sichtlinienansatz* finden sich die gemessenen Laufzeiten in den Abbildungen 4.38 und 4.39. Zusammen mit dem *Spektralen Clustering* ist die *Berechnung der Affinitätsmatrix* der aufwendigste Prozessschritt. Für die Menge K der zu testenden Clusteranzahl wurden immer fünf der aussichtsreichsten Kandidaten ausgewählt. Für die *Übersegmentierung/Verschmelzung* sind die Laufzeiten in Abbildung 4.40 dargestellt. Auffällig ist die unterschiedliche Laufzeit beider Segmentierungsansätze. So benötigt die *Übersegmentierung/Verschmelzung* bis zu einer Größenordnung mehr Zeit für die Ausführung verglichen mit dem *Sichtlinienansatz*. Dies lässt sich mit der erhöhten Komplexität der durchzuführenden Prozessschritte erklären (siehe Kapitel 4.5.2.2).

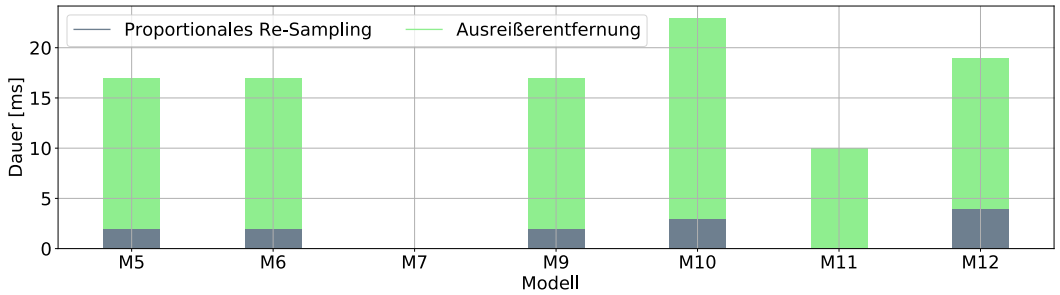


Abbildung 4.38.: Laufzeiten des *Sichtlinienansatzes*: *Proportionales Re-Sampling* und *Ausreißerentfernung*.

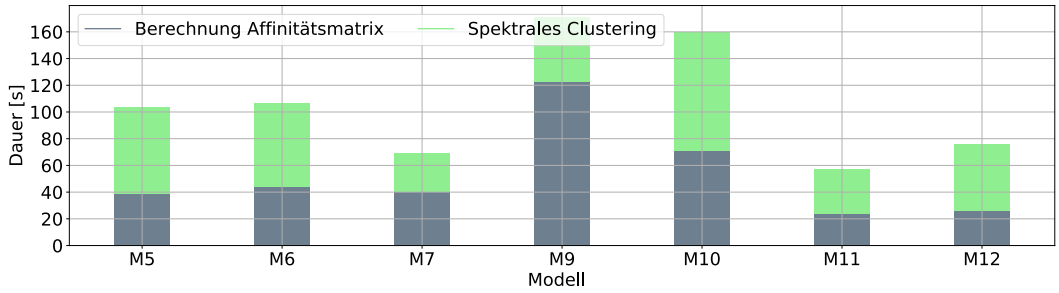


Abbildung 4.39.: Laufzeiten des *Sichtlinienansatzes*: *Berechnung Affinitätsmatrix* und *Spektrales Clustering*.

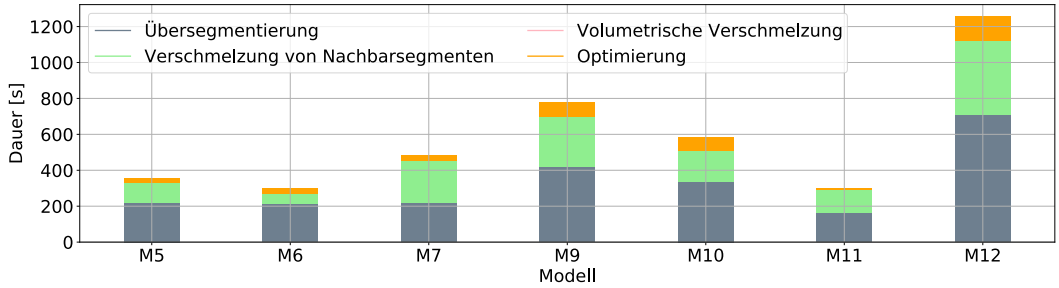


Abbildung 4.40.: Laufzeiten des Segmentierungsansatzes *Übersegmentierung/-Verschmelzung*.

Punktzuweisung. Für diesen Schritt finden sich die Ergebnisse der Laufzeitmessungen in Abbildung 4.41. Dabei wurden Messungen für den *Sichtlinienansatz* sowie für die *Übersegmentierung/Verschmelzung* separat durchgeführt, da sie sich bzgl. der resultierenden Cluster-Punktwolken als Eingabe für die *Punktzuweisung* unterscheiden.

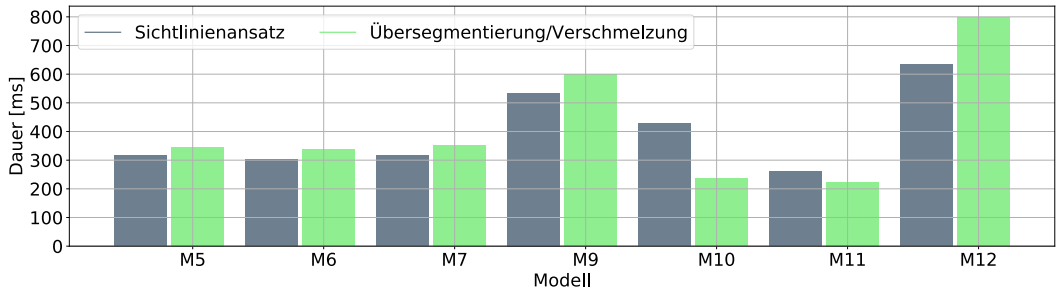
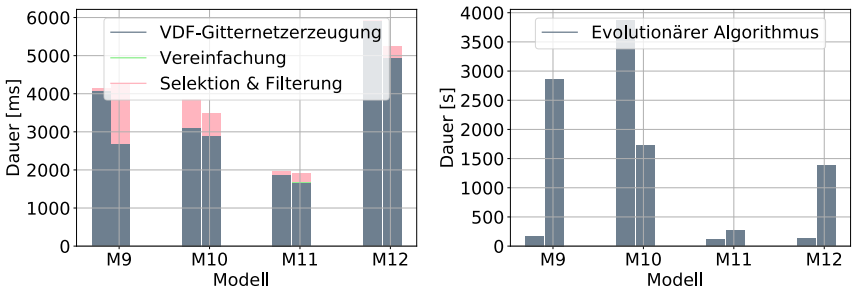


Abbildung 4.41.: Laufzeiten der *Punktzuweisung*.

Konstruktion konvexer Polytope. Die gemessenen Laufzeiten für die Teilschritte der *Konstruktion konvexer Polytope* sind in Abbildung 4.42a und für den EA separat in Abbildung 4.42b aufgeführt. Als Basis dient der Durchlauf mit dem jeweils qualitativ besten Ergebnis. Dabei werden immer beide

Fälle betrachtet, mit und ohne *schwach-konvexe Segmentierung*. Auffällig ist die meist verkürzte Laufzeit des EAs, wenn die schwach-konvexe Segmentierung zum Einsatz kommt. Das ist relevant, da dieser Schritt der aufwändigste ist. Damit wirken sich Effizienzsteigerungen besonders positiv auf die Gesamtlaufzeit aus. Auch unter Beachtung der zusätzlich benötigten Laufzeit für die Segmentierung, ergibt sich daraus für komplexe Modelle, die gut segmentiert werden können (M9, M12), ein Effizienzgewinn. Eine Ausnahme bildet hier M10. Durch die vergleichsweise schlechte Segmentierung muss der für jedes Segment aufgerufene EA ein vergleichsweise komplexes Problem lösen. Da sich die maximale Anzahl an Iterationen nicht für jedes Segment separat einstellen lässt, ist die Laufzeit im Gesamten höher.
Zu beachten ist auch die meist höhere Laufzeit für den Schritt *Selektion & Filterung*, wenn keine Segmentierung verwendet wird (Ausnahme: M10). Dies ist damit zu erklären, dass meist eine größere Anzahl von Primitiven gleichzeitig gefiltert wird und der Filter zur Entfernung redundanter Primitive quadratisches Laufzeitverhalten bzgl. der Primitivenanzahl aufweist.



(a) Pipeline-Schritte ohne EA. (b) EA.

Abbildung 4.42.: Laufzeiten für die Modelle M9-M12 mit (jeweils Balken links) und ohne (jeweils Balken rechts) Nutzung der schwach-konvexen Segmentierung.

4.6.6. Schlüsselergebnisse

Möchte man die Ergebnisse der Evaluation auf wenige Kernpunkte verdichten, wären das die folgenden: Die Grundidee, konvexe Polytope aus einer Kombination von detektierten Ebenen zu extrahieren, funktioniert überzeugend. Die vorgeschlagenen Segmentierungsschritte in der *Punktwolkensegmentierung* sowie in der *Primitivendetektion und -einpassung* verbessern die Gesamtergebnisse. Beide tragen an unterschiedlichen Stellen signifikant zur Komplexitätsreduktion bei. Dabei sorgen sie nachweislich für eine erhöhte Ergebnisqualität bei der *Detektion & Einpassung unbeschränkter Primitive* sowie für eine überlegene Unterstützung komplexer Datensätze bei der *Konstruktion konvexer Polytope*. Weiterhin ist ein Vorteil der vorgestellten Architektur die Fähigkeit, auch für nicht-synthetische Datensätze überzeugende Ergebnisse zu liefern. Dies ist

auch hinsichtlich verwandter Arbeiten ein Alleinstellungsmerkmal. Auch wenn nicht im Fokus der Zielsetzung, bietet der schwach-konvexe Segmentierungsschritt bei der *Konstruktion konvexer Polytope* bzgl. der Laufzeit Vorteile.

4.7. Zusammenfassung und Ausblick

Das in diesem Kapitel vorgestellte System zur Rekonstruktion von Primitiven aus einer Punktwolke besteht aus einer Vielzahl von parametrisierten Prozessschritten. Es kombiniert dabei bestehende sowie vollständig neu entwickelte Verfahren, um in einer Punktwolke auf robuste und effiziente Weise einfache Primitive sowie konvexe Polytope zu detektieren und deren Parameter zu ermitteln. Anforderungen an das System wurden prägnant dargestellt und mit den Eigenschaften verwandter Arbeiten verglichen. Das grundlegende Konzept sowie dessen Erweiterung um einen zusätzlichen Segmentierungsschritt für die Unterstützung komplexerer Modelle wurde detailliert erläutert und anhand von Illustrationen anschaulich erklärt. Die Evaluation des Systems wurde in einer tiefgehenden Analyse der Ergebnisqualität einzelner Prozessschritte sowie hinsichtlich deren Laufzeitverhalten diskutiert. Dabei konnte ein umfassender Einblick in die Leistungsfähigkeit des Gesamtsystems gegeben werden. Zudem wurde gezeigt, dass die Vereinfachung des Gesamtproblems durch Partitionierung das System hinsichtlich Robustheit und Ergebnisqualität verbessert.
Zukünftige Entwicklungen und Forschungsrichtungen basierend auf dem hier vorgestellten System sind vielfältig. So könnte die Erweiterung der Menge unterstützter Primitive um Kegel die Ausdrucksstärke rekonstruierbarer Festkörper weiter erhöhen. Gleiches gilt für die Unterstützung arbiträr gekrümmter Oberflächen, wie sie zum Beispiel durch NURBS beschrieben werden können (siehe Kapitel 2.3.3.3). Dies wäre ein schwierigeres Problem bezüglich Laufzeiteffizienz sowie hinsichtlich der Anforderung, weiterhin wasserdichte Festkörper zu erzeugen. Das liegt darin begründet, dass die Schnittkurve zweier NURBS-Oberflächen im Allgemeinen nicht durch eine NURBS-Beschreibung repräsentiert werden kann und somit nur approximativ darstellbar ist.
In die gleiche Richtung geht eine Aufhebung der momentanen Restriktion auf konvexe Polytope, die nahezu vollständig innerhalb des durch die Punktwolke beschriebenen Festkörpers liegen. Nötig wäre dies, um auch konvexe Polytope, die aus dem Festkörper "ausgeschnitten" werden, berücksichtigen zu können und damit komplett außerhalb desselben liegen. Dazu muss der Geometrieterm zur Bewertung der Passgenauigkeit einzelner konvexer Polytope V_G (siehe Gleichung 4.5) entsprechend modifiziert werden. Dort muss zusätzlich geprüft werden, ob Oberflächennormalen an zu Polytopebenen assoziierten Punkten vollständig in entgegengesetzter Richtung zu den Gradienten des Festkörpers verlaufen. Ist dies der Fall, müssen in V_G nicht alle Zellmittelpunkte innerhalb des approximierten Festkörpers gezählt werden, sondern alle außerhalb liegenden.
Weiterhin herausfordernd ist die Einführung einer zusätzlichen Nebenbedin-

gung, die eine Überlappung von gefundenen konvexen Polytopen verbietet. Dies hat als positiven Aspekt eine klarere Trennung konvexer Subelemente zur Folge, erschwert aber das zugrundeliegende Optimierungsproblem signifikant. Dazu muss die in Gleichung 4.3 angegebene Zielfunktion um geeignete Terme erweitert oder der Suchraum durch eine angepasste Polytoperzeugungsmethode eingeschränkt werden. Letztere Strategie würde jedoch die Komplexität nicht verringern, sondern nur in Richtung Erzeugungsmethode verschieben.
Bei der Betrachtung der Parametervielfalt des Gesamtsystems drängt sich zudem die Frage auf, ob die Möglichkeit besteht, entweder Parameter automatisch abzuleiten oder teilweise in Gänze einzusparen. Dies muss im Einzelfall geprüft und umgesetzt werden, wäre jedoch sinnvoll im Hinblick auf die für die Evaluation nötigen Parametersuchen und Feinjustierungen.
Diese Beispiele möglicher Weiterentwicklungen stellen dabei nur eine kleine Auswahl an möglichen Forschungsrichtungen dar und können nahezu in alle Richtungen ergänzt werden. Im folgenden Kapitel 5 soll hingegen die Synthese von Konstruktionsbäumen aus einer Punktwolke sowie von bekannten Primitiven im Fokus stehen. Das hier vorgestellte System wird dabei zur Erzeugung der dafür notwendigen Primitive verwendet.

5. Synthese von Konstruktionsbäumen

[Wir] nehmen [...] an, dass die Objekte unserer Anschauung aus Teilen konstruiert werden könnten, mit denen wir vertraut sind.

—L.G. Roberts, 1963

Der Annahme im Eingangszitat folgend, lassen sich in vielen Fällen komplexe Objekte aus einfachen Grundformen zusammenfügen. In Kapitel 4 wurde dazu in einem ersten Schritt ein System vorgestellt, das die dazu nötigen Primitive in einer Punktwolke auffinden kann. Was fehlt, ist die Information, wie einzelne Primitive zur vollständigen Beschreibung des Festkörpers kombiniert werden müssen. Wie im CSG-Repräsentationsschema vorgesehen, wird diese Information in Form einer Baumstruktur geordnet (siehe Kapitel 2.3.1.1). Daraus ergibt sich eine mächtige und ausdrucksstarke Sprache zur Beschreibung von Festkörpern.
In diesem Kapitel wird ein System vorgestellt, das einen KB automatisiert aus einer Menge von Primitiven und einer Punktwolke synthetisieren kann und damit *Problem 2* (siehe Abbildung 3.1) löst. Es bewältigt also ein Repräsentationskonversionsproblem zu dem erschwerend hinzukommt, dass die Qualität der Eingaberepräsentation in Form einer Punktwolke durch systemische Messfehler beeinträchtigt sein kann. Der Begriff der Synthese wurde dabei dem Begriff der Rekonstruktion vorgezogen, da letzterer suggeriert, dass der durch die Eingabepunktwolke beschriebene Festkörper bereits ursprünglich als KB modelliert wurde. Da dies augenscheinlich nicht immer der Fall sein muss, wurde die allgemeinere Bezeichnung "Synthese" gewählt.
Dieses Kapitel ist dabei wie folgt strukturiert: Zu Beginn werden Inhalte aufgeführt, die Teil dieser Arbeit sind, jedoch bereits anderweitig publiziert wurden (siehe Kapitel 5.1). Darauf folgt eine detaillierte Beschreibung der für das System gesteckten Ziele sowie eine Motivation der Problemstellung (siehe Kapitel 5.2). In Kapitel 5.3 werden verwandte Forschungsarbeiten aufgeführt und vergleichend diskutiert. Um eine überzeugende Lösung für das behandelte Problem erarbeiten zu können, muss dieses zuerst im Detail verstanden werden. Aus diesem Grund werden in Kapitel 5.4 Fragen zur Problemkomplexität sowie mögliche Lösungs- und Vereinfachungsstrategien erläutert. Im Anschluss wird das Konzept des hier vorgestellten Systems beschrieben (siehe Kapitel 5.5). Eine Diskussion der Systemevaluation findet sich im folgenden Kapitel (siehe Kapitel 5.6). Abschließend werden in Kapitel 5.7 Kerninhalte zusammengefasst und ein Ausblick auf zukünftige Forschungsrichtungen gegeben.

5.1. Vorveröffentlichungen

Die hier vorgestellten Inhalte wurden bereits in mehreren Teilen veröffentlicht. Die Charakterisierung des Suchraums des KB-Syntheseproblems (siehe Kapitel 5.4.1 und 5.4.2) wurde bereits in [56] publiziert. Ebenfalls bereits veröffentlicht wurden die beiden Methoden zur Suchraumpartitionierung (siehe Kapitel 5.4.3.2): Die *cliquenbasierte Partitionierung* wurde in [53] beschrieben, die *hybride Partitionierung* in [52]. Konzept (siehe Kapitel 5.5), Implementierung und Evaluation (siehe Kapitel 5.6) des Gesamtsystems sind hingegen für diese Arbeit neu entstanden, wobei der genutzte EA eine stark modifizierte Variante des EAs aus [56] darstellt. Abbildung 5.2 stammt aus [56]. Die Abbildungen 5.6, 5.5 und 5.4 wurden aus [53] übernommen, jedoch bzgl. Farbgebung und Namensgebung einzelner Elemente modifiziert. Gleiches gilt für Abbildung 5.7, die aus [52] stammt.

5.2. Motivation und Zielsetzung

Die CSG-Repräsentation ist im Bereich des Entwurfs von industriellen Bauteilen aufgrund ihrer intuitiven und übersichtlichen Anordnung von Modellierungsoperationen eine weitverbreitete Darstellungsform [149]. Obwohl durch sie eine effiziente Modellierung möglich ist, stellt sich die Frage, wie deren manuelle Erstellung rechnergestützt beschleunigt werden kann.
Dies ist v.a. für eine kostengünstige Entwicklung relevant. Grundsätzlich erfolgt dabei eine Aufteilung in zwei unterschiedliche Ansätze, den *konventionellen* und den *alternativen Ansatz*, die sich hinsichtlich der Strategie für die Erarbeitung eines *konzeptionellen Modells* des zu entwickelnden Bauteils unterscheiden [180] (siehe Abbildung 5.1). Dabei versteht man unter einem *kon-*

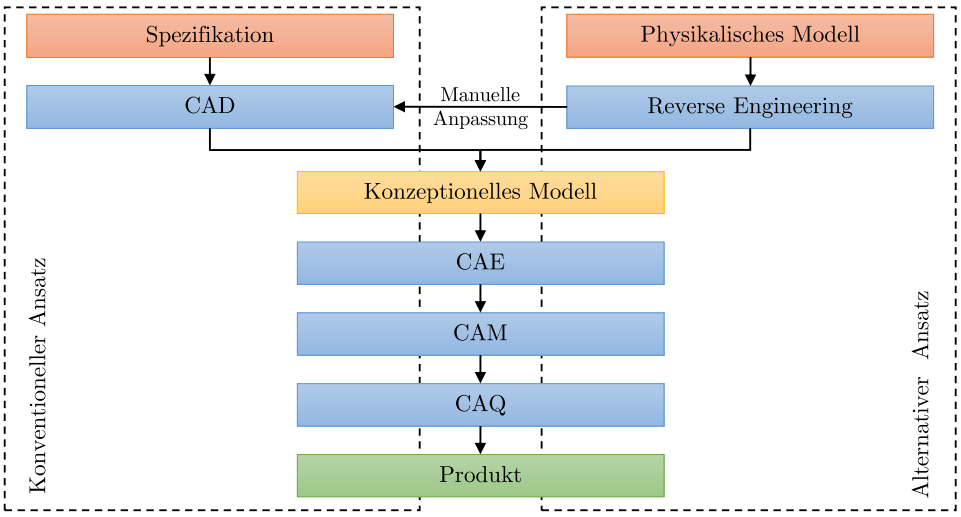


Abbildung 5.1.: Schematische Darstellung des Produktentwicklungsprozess angelehnt an [180].

zeptionellen Modell die digitale Gesamtheit der Information des entwickelten Bauteilmodells, also z.B. seine Form, Abmessungen, erforderliche Toleranzen und Materialbeschaffenheit. Es handelt sich also um Information, die auch bei der Erstellung eines virtuellen Modells des Bauteils (*Digital Twin*) nützlich ist [69, 165]. Die erstgenannten geometrischen Informationen repräsentiert durch KBs sind dabei für diese Arbeit im Fokus.
Der *konventionelle Ansatz* beginnt mit der Ausarbeitung einer Spezifikation des zu entwickelnden Bauteils, die Aussehen sowie Funktion festlegt. Auf Basis der Spezifikation werden dann mit Methoden des rechnergestützten Konstruierens (engl. *Computer-Aided Design* (CAD))[159] geometrische Formen und Abmessungen festgelegt und im *konzeptionellen Modell*, das sich als Konstruktionsplan verstehen lässt, zusammengeführt.
Das *konzeptionelle Modell* dient dem nächsten Schritt, der rechnergestützten Entwicklung (engl. *Computer-Aided Engineering* (CAE))[159] als Eingabe, die physikalische Simulationen zur Validierung von Konstruktionen oder Prozesse zu deren Optimierung beinhaltet. Im nächsten Schritt wird das validierte und optimierte Modell in eine Reihe von Befehlen für automatisierte Fertigungsmaschinen umgesetzt, die das digitale Modell endgültig in seine physikalische Repräsentation und damit in das *Produkt* verwandeln. Dieser Prozessschritt ist im Bereich der rechnergestützten Fertigung (engl. *Computer-Aided Manufacturing* (CAM))[159] verortet. Hier ist die Validitätseigenschaft der CSG-Repräsentation von Vorteil, die garantiert, dass syntaktisch korrekte KBs auch valide und damit physikalisch herstellbare Festkörper repräsentieren (siehe Kapitel 2.3.1.1).
Meist erfolgt noch ein zusätzlicher Schritt zur Qualitätskontrolle, der ebenfalls rechnergestützt abläuft (engl. *Computer-Aided Quality Assurance* (CAQ))[95]. Dabei werden z.B. im Konstruktionsplan festgelegte Toleranzen am hergestellten *Produkt* oder mögliche Strukturfehler in dessen Oberfläche per Messung automatisch überprüft, was aber für diese Arbeit nicht im Detail relevant ist.
Anstatt einer Spezifikation zu folgen, wird nun beim *alternativen Ansatz* die Ableitung des Konstruktionsplans auf Basis der Analyse des Aufbaus und der Funktionsweise eines physikalisch vorhandenen Exemplars des zu fertigenden Bauteils durchgeführt. Dieser Prozess wird im Allgemeinen als Produktnachbau (engl. *Reverse Engineering* (RE)) bezeichnet [8]. Bezieht man sich nur auf die Ableitung geometrischer Information, wird auch der Begriff geometrisches RE verwendet [8]. Beim geometrischen RE wird meist die Oberfläche des physisch vorhandenen Produkts mit 3D-Scannern (siehe Kapitel 3.2.1.2) abgetastet und daraus dessen Festkörperrepräsentation abgeleitet. Das macht das geometrische RE zu einem wichtigen Anwendungsgebiet und zur Hauptmotivation der KB-Synthese aus Punktwolken, wie sie hier betrachtet wird.
Ziel ist also der Entwurf sowie die Umsetzung und Evaluation eines Systems zur KB-Synthese aus einer Menge von Primitiven und einer Punktwolke. Dabei soll der so erzeugte KB einen Festkörper bestmöglich repräsentieren, der approximativ durch eine Punktwolke beschrieben wird. Der Synthesevorgang

soll dabei vollautomatisch sein und zudem berücksichtigen, dass die Eingabepunktwolke Messartefakte aufweist.
Bereits vor dem Entwurf des Systems soll das zu lösende KB-Syntheseproblem formalisiert und charakterisiert werden. Dies dient auch der Konzeption möglicher Problemvereinfachungen, die sich dann positiv auf die Laufzeiteffizienz des Gesamtsystems auswirken. Für die in Kapitel 4 erzeugten Primitive gilt, dass der resultierende Gesamtfestkörper oft nur aus besonders einfachen Primitivenkombinationen aufgebaut ist. Dazu zählen v.a. Vereinigungen aller Primitive oder das komplette Ausschneiden einzelner Primitive aus einer Kombination vereinigter Primitive. Da dies eine häufig anzutreffende Konstellation ist, sollte ein Algorithmus zur KB-Synthese diese erkennen und das entsprechende Problem effizient lösen können.

5.3. Verwandte Arbeiten

Die zu dem hier diskutierten Ansatz in Bezug stehenden Arbeiten werden im Folgenden aufgeführt. Alle haben dabei gemeinsam, dass sie Repräsentationskonversionsprobleme lösen (siehe Kapitel 2.3.4), was eine Einteilung nach Eingaberepräsentation sinnvoll macht.

5.3.1. Konstruktionsbaumsynthese aus Begrenzungsflächenmodellen

Das Problem der Extraktion von KBs aus bestehenden Begrenzungsflächenmodellen (siehe Kapitel 2.3.3) wurde historisch betrachtet als erstes in Angriff genommen. Dabei standen zuerst zweidimensionale Konstruktionen aus Polygonen mit geraden Kanten im Vordergrund [172]. Später erfolgte dann die Erweiterung auf den dreidimensionalen Raum [171, 172] und effizienzverbessernde Variationen der Basisalgorithmen [26]. Die Unterstützung von Polygonen mit gekrümmter Oberfläche wurde in [169] beschrieben. Aus [173] stammt die Erkenntnis, dass die aus einem Begrenzungsflächenmodell direkt gewonnenen Primitive nicht immer ausreichen, um ein Modell vollständig zu beschreiben und deswegen weitere, sog. separierende Primitive eingefügt werden müssen (siehe Kapitel 2.3.4). In derselben Publikation wird auch die Dekomposition als Methode zur Vereinfachung des Konversionsproblems eingeführt (siehe Kapitel 3.2.5.2). Du et al. stellen in [39] ein System vor, das KB-Ausdrücke aus CAD-Modellen repräsentiert durch Dreiecksnetze extrahiert. Dazu wird das Problem zuerst geeignet partitioniert und dann als Programmsyntheseproblem formuliert. Dies wird mit entsprechenden Techniken aus diesem Bereich gelöst. Die genannten Ansätze haben alle gemein, dass das Eingabemodell exakt bzw. als Dreiecksnetz-Approximation vorliegen muss. Damit ist kein Spielraum für etwaige Messfehler und Ungenauigkeiten gegeben, wie er in der Problemstellung dieser Arbeit gefordert und vom in diesem Kapitel beschriebenen System gewährt wird.

5.3.2. Konstruktionsbaumsynthese aus Punktwolken- oder Voxel-Repräsentationen

Im Folgenden werden Verfahren betrachtet, die Punktwolken als Eingabe verwenden.

5.3.2.1. Evolutionäre Algorithmen

In [178] werden einfache KBs aus Punktwolken mittels evolutionärer Methoden extrahiert. Im Gegensatz zu der in dieser Arbeit beschriebenen Prozess-Pipeline wird dabei auch die Primitivenextraktion mittels EA durchgeführt. Die Modellkomplexität ist dabei gering, wobei resultierende KBs eine Vielzahl redundanter Strukturen aufweisen. In [47] wird eine Trennung von Primitivenextraktion und KB-Synthese vorgenommen, wobei letztere durch die Minimierung einer Zielfunktion mittels EA geschieht. Das in dieser Arbeit vorgestellte Verfahren basiert konzeptionell auf diesem System, erweitert dies aber u.a. durch geeignete Methoden zur Problemvereinfachung und -partitionierung. Zudem kommt im hier behandelten System eine Zielfunktion zum Einsatz, die robuster bezüglich fehlerbehafteter Punktwolken ist.

5.3.2.2. Maschinelles Lernen

Im Bereich des MLs wurden Verfahren zur KB-Synthese vorgestellt, die auf KNNs beruhen [96, 174, 188]. So wurde in [174] ein System beschrieben, das KBs auf Basis von zweidimensionalen Rastergrafiken oder dreidimensionalen Voxelgittern erzeugen kann. Als Eingabe dienen also reguläre Gitternetze und keine Punktwolken, was aber durch den Austausch der FNN-basierten Eingabekodierstufe mit einer Punktwolkenkodierstufe (z.B. der aus der PointNet++-Architektur [142]) behoben werden kann. Die Netzarchitektur nutzt ein RNN zur Erzeugung der Primitive und Operationen des KBs. Da die Loss-Funktion nur die Literale des Ausdrucks berücksichtigt und nicht die repräsentierte Geometrie, ist sie differenzierbar. Ebenfalls enthalten ist eine Methode basierend auf Bestärkendem Lernen [184], die ohne Trainingsbeispiele in Form von KBs auskommt und die geometrische Passgenauigkeit erzeugter KBs berücksichtigen kann. Diese wird zur Verbesserung der Ergebnisqualität des KNNs eingesetzt. Im Gesamten ist das System jedoch hinsichtlich Ergebnisqualität und möglicher Baumgröße limitiert. Zudem lassen sich nur KBs erzeugen, die Primitive nur genau einmal enthalten. Eine Netzarchitektur, die vollständig auf KBs im Trainingsdatensatz verzichtet und in Benchmarks besser als [174] abschneidet, wird in [96] beschrieben. Diese formuliert eine differenzierbare Loss-Funktion zur Bewertung der geometrischen Einpassgenauigkeit und nutzt eine RNN-Architektur zum sukzessiven Aufbau des KBs.
Einen anderen Weg schlägt die Netzarchitektur in [188] ein. Um die geometrische Einpassgenauigkeit als differenzierbare Loss-Funktion formulieren zu können, wird die Konversion eines KBs in eine Voxel-Repräsentation durch ein

separates KNN übernommen. Damit ist diese Umsetzung differenzierbar und kann innerhalb der Loss-Funktion verwendet werden. Zudem werden komplexere Programme mit Schleifen anstatt simpler KBs unterstützt.
Alle vorgestellten Netzarchitekturen haben gemein, dass sie sowohl das Primitivenextraktions- als auch das KB-Syntheseproblem lösen. Generell gilt auch, dass sie eher simple Modelle mit vergleichsweise wenig Primitiven verarbeiten können. Zudem ist die Prädiktion auf Modelle begrenzt, die ähnlich zu denen des Trainingsdatensatzes sind, wobei fehlerhafte Eingabedaten keine Berücksichtigung finden.

5.3.2.3. Andere Lösungsverfahren

In [198] wird eine Prozess-Pipeline vorgestellt, die KBs aus Punktwolken synthetisieren kann. Dazu wird ein bekanntes Verfahren zur Extraktion von Primitiven genutzt und in einem mehrstufigen Verfahren Teilbäume mittels binärer Optimierung extrahiert, die dann zu einem Gesamtbaum zusammengefügt werden. Im Vergleich zu der hier vorgestellten Methode können nur KB-Ausdrücke erzeugt werden, die jedes Primitiv nicht mehr als einmal enthalten.
Xiao et al. stellt in [200] ein Verfahren vor, welches dreidimensionale Gebäudepläne in abstrahierter Form aus Punktwolken extrahiert. Diese Abstraktionen werden in Form von KBs mit Quadern als Primitive repräsentiert. Die Prozess-Pipeline sieht dabei erst eine vertikale Segmentierung der Eingabe in zweidimensionale Scheiben vor, für die eine ebenfalls zweidimensionale CSG-Repräsentation ermittelt wird. Abschließend werden dann alle zweidimensionalen CSG-Repräsentationen zu einer finalen dreidimensionalen Variante zusammengeführt. Im Vergleich zu der in dieser Arbeit vorgestellten Methode ist dieses Verfahren auf einfache Gebäudegeometrien beschränkt.

5.3.3. Zusammenfassung

Zu den Anforderungen, die im Rahmen dieser Arbeit an ein KB-Synthesesystem gestellt werden, gehören neben der Robustheit gegen fehlerhafte Eingabepunktwolken auch die Unterstützung von komplexen Modellen mit mehr als zehn Primitiven. Zudem müssen alle mit einer Primitivenmenge möglichen KBs Teil des erfassten Suchraums sein. Keines der hier diskutierten Verfahren erfüllt jedoch alle diese Anforderungen. Dies bleibt Alleinstellungsmerkmal des hier vorgestellten Systems, wie im Folgenden dargelegt wird.

5.4. Das Problem der Konstruktionsbaumsynthese

In diesem Kapitel soll das Problem der KB-Synthese bezüglich seiner Komplexität im Detail durchleuchtet werden. Dabei wird auf die in Kapitel 3.2.5 eingeführten Konzepte aufgebaut. Das betrachtete Problem lässt sich dabei wie

folgt formalisieren: Gegeben eine Menge an Punkten O (z.B. die aufbereitete Punktwolke O_a aus Kapitel 3) sowie eine Menge an eingepassten Primitiven H. Es soll ein KB Φ gefunden werden, sodass die Punktmenge $|\Phi(H)|$ möglichst der Punktmenge S des durch O approximativ beschriebenen Festkörpers entspricht, d.h. also bestenfalls $|\Phi(H)| = S$ gilt.

5.4.1. Kanonische Schnitte

Eine erste Lösung für das Problem lässt sich über die Menge der Fundamentalprodukte ermitteln, die sich aus den Primitiven in H ergeben (siehe Kapitel 3.2.5.1). Dabei muss für jedes Fundamentalprodukt festgestellt werden, ob es vollständig in S liegt. Die Menge der so ermittelten, in S liegenden Fundamentalprodukte mittels regularisiertem Vereinigungsoperator $\cup^*$ verknüpft ergibt dann eine korrekte Lösung als KDNF:

$$\Phi(H) = \bigcup_{k=1}^{2^{|H|}-1} \epsilon_k \left(g_1 \cap^* g_2 \cdots \cap^* g_{|H|} \right), \qquad g_i \in \{h_i, \setminus^* h_i\}, \tag{5.1}$$

wobei ϵ_k Eins ist, wenn die durch das k-te Fundamentalprodukt repräsentierte Punktmenge vollständig in S ist, ansonsten Null.
Je nach geometrischer Lage der Primitive liegt die Anzahl der zu betrachtenden, nichtleeren Fundamentalprodukte F im Intervall $[|H|, 2^{|H|} - 1]$. Schneiden sich keine der Primitive in H untereinander, ergeben sich $|H|$ Fundamentalprodukte - eines für jedes Primitiv. Hat hingegen jedes Primitiv mit jedem anderen eine nichtleere Schnittmenge, erhält man $2^{|H|} - 1$ Fundamentalprodukte. Damit hat die Ermittlung der in S liegenden Fundamentalprodukte eine Laufzeitkomplexität von $O(2^{|H|})$. Das vorgestellte Verfahren hat folgende Nachteile:

1. Der resultierende Ausdruck ist nicht optimal hinsichtlich seiner Größe.
2. Die Anzahl der zu prüfenden Fundamentalprodukte kann, je nach geometrischer Lage der Primitive, exponentiell mit $|H|$ ansteigen.
3. Es lässt sich nicht immer robust und zweifelsfrei prüfen, ob ein Fundamentalprodukt vollständig innerhalb oder vollständig außerhalb des Festkörpers liegt, da dieser nur über eine potentiell fehlerbehaftete Punktwolke O approximativ repräsentiert wird.

Dabei lässt sich das erste Problem mit einer Vergrößerung des Suchraums auf arbiträre binäre Ausdrücke (und damit auf arbiträre binäre Baumstrukturen) beheben, wie im folgenden Kapitel ausgeführt wird.

5.4.2. Binäre Baumstrukturen

Nimmt man die Menge der regularisierten Mengenoperatoren $R = \{\cup^*, \cap^*, -^*\}$ zusammen mit den Primitiven H, lassen sich daraus arbiträre binäre Baum-

strukturen aufbauen. Deren Anzahl ist dabei potentiell unendlich, da Operationen beliebig wiederholt werden können. Möchte man zusätzlich auch das Komplement $\backslash^*$ berücksichtigen, gelingt dies, indem die Universalmenge W als Primitiv zu H hinzugefügt wird. Damit lässt sich die durch das Komplement eines Teilbaums Φ_t beschriebene Punktmenge über $W -^* |\Phi_t|$ ausdrücken (siehe Kapitel 2.2).
Jeder erzeugte Ausdruck lässt sich einer Äquivalenzklasse zuordnen, wobei jede Äquivalenzklasse alle (also unendlich viele) Bäume enthält, die dieselbe Punktmenge repräsentieren. Die Anzahl der Äquivalenzklassen lässt sich dabei von der Anzahl der durch H induzierten Fundamentalprodukte ermitteln und ist $2^{|F|}$ (Jedes der $|F|$ Fundamentalprodukte kann Teil der Punktmenge des gesuchten Festkörpers sein oder nicht). Gesucht wird also einer derjenigen Bäume einer Äquivalenzklasse, die bezüglich ihrer Größe minimal sind. Die richtige Äquivalenzklasse ist dabei diejenige, welche alle Ausdrücke enthält, die den durch O approximativ beschriebenen Festkörper korrekt repräsentieren.

Um nun die Problemkomplexität der Suche nach dem optimalen Baum quantitativ erfassen zu können, muss ein Weg gefunden werden, alle möglichen Baumstrukturen für eine gegebene Anzahl von inneren Knoten n aufzuzählen. Die Festlegung auf ein fixes n ist dabei nötig, da, wie oben beschrieben, unendlich viele Bäume existieren, wenn keine obere Schranke für deren Größe festgelegt wird. Da die betrachteten Operationen alle zweistellig sind, stellt die resultierende Baumstruktur immer einen Binärbaum dar. Die Anzahl der Binärbäume für ein fixes n ergibt sich aus der Catalan-Zahl [101]:

$$C(n) = \frac{1}{n+1}\binom{2n}{n}. \tag{5.2}$$

Die $n+1$ Blattknoten sind dabei Primitive aus H. Eine Darstellung der Bäume für $n = 0, 1, 2, 3$ findet sich in Abbildung 5.2.

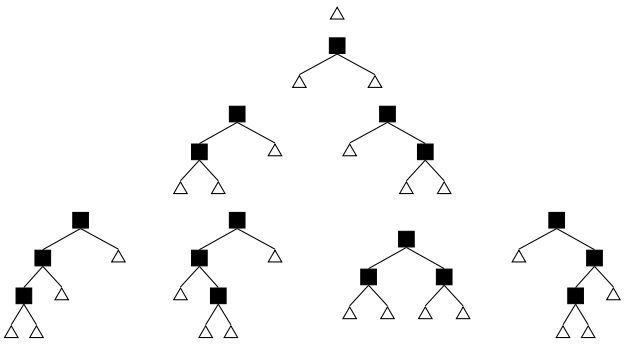


Abbildung 5.2.: Binärbäume für $n \in \{0, 1, 2, 3\}$ innere Knoten (erste Zeile $n = 0$, zweite Zeile $n = 1$, dritte Zeile $n = 2$, vierte Zeile $n = 3$). Schwarze Rechtecke symbolisieren innere Knoten, weiße Dreiecke Blattknoten.

Da jedes Primitiv beliebig oft verwendet werden kann, ergeben sich für die Blattknoten $|H|^{n+1}$ Primitivenkombinationen. Die n inneren Knoten können Operationen aus R enthalten, woraus sich $|R|^n$ Operationenkombinationen ergeben. Für einen KB bestehend aus $2n+1$ Knoten (n innere Knoten, $n+1$ Blattknoten) gibt es somit

$$B(n) = |H|^{n+1} \cdot |R|^n \cdot C(n) \tag{5.3}$$

Kombinationen. Da n unbekannt ist, muss eine möglichst optimale obere und untere Schranke (n_{min} und n_{max}) gefunden werden. Als Wert für die untere Schranke n_{min} bietet sich dabei $|H|-1$ an, da im Ausdruck zumindest jedes Primitiv einmal als Operand vorkommen sollte. Dies ist nur dann der Fall, wenn keine redundanten Primitive in H existieren, was jedoch durch geeignete Filter vermieden werden kann (siehe Kapitel 4.4.4.1). Die Ermittlung einer passenden oberen Schranke n_{max} gestaltet sich hingegen als schwierig, da Bäume z.B. durch das Hinzufügen redundanter Teilausdrücke beliebig groß sein können. Hier hilft in der Regel nur Ausprobieren und damit die empirische Ermittlung des Wertes für eine bestimmte Probleminstanz. Für bekannte obere und untere Schranken ergibt sich die Anzahl möglicher KBs aus

$$A = \sum_{i=n_{min}}^{n_{max}} B(i). \tag{5.4}$$

Damit ist A auch die Größe des Suchraums Θ, der für die Ermittlung des optimalen Baums durchlaufen werden muss (siehe Kapitel 3.2.5.3). Abbildung 5.3 zeigt den Suchraum schematisch. Wie eine Suche in diesem sinnvoll vereinfacht werden kann, ohne das Ziel der Größenoptimalität aufzugeben, wird im folgenden Kapitel erläutert.

5.4.3. Problemvereinfachung

Um das Problem der KB-Synthese effizient lösen zu können, müssen Verfahren zu dessen Vereinfachung gefunden werden. Dazu werden im Folgenden zwei Strategien beschrieben.

5.4.3.1. Dekomposition

Mithilfe der Dekomposition lassen sich Primitive rekursiv aus dem Zielfestkörper faktorisieren, was den Suchraum sukzessive vereinfacht. Die faktorisierbaren Primitive sind dabei sog. dominante Primitive. Je nach Festkörper und Primitivenmenge, ist in manchen Fällen eine vollständige Dekomposition möglich, d.h. alle Primitive können faktorisiert werden. In diesem Fall ist das Resultat ein größenoptimaler KB. Eine detaillierte Beschreibung des Verfahrens findet sich in Kapitel 3.2.5.2.
Die Laufzeitkomplexität der Dekomposition ist im schlechtesten Fall $O(|H|^2)$,

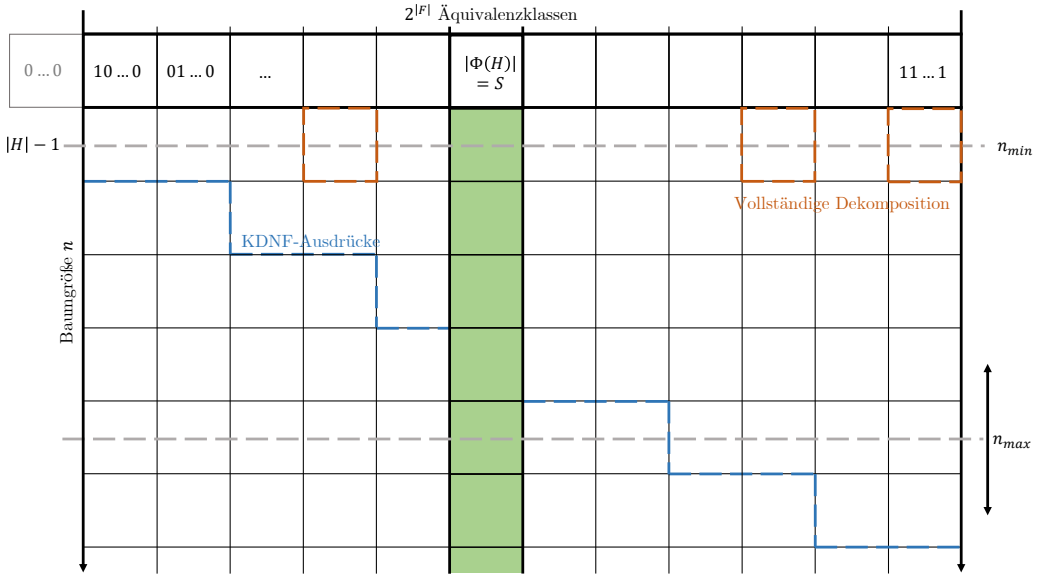


Abbildung 5.3.: Darstellung der KBs des Lösungsraums (ein Quadrat pro Baum Φ) mit einer Anzahl innerer Knoten n zwischen $n_{min} = |H| - 1$ und n_{max} geordnet nach Äquivalenzklasse und Anzahl innerer Knoten. Jeder Baum lässt sich einer der $2^{|F|}$ Äquivalenzklassen zuordnen, wobei der gesuchte Festkörper mit Punktmenge S durch Bäume genau einer Klasse repräsentiert wird (grün). Damit ist jeder Baum dieser Klasse eine valide Lösung, die sich nur hinsichtlich ihrer Anzahl innerer Knoten von anderen Bäumen derselben Klasse unterscheidet. Äquivalenzklassen werden mit einem binären $|F|$-Tupel $(\epsilon_1, ..., \epsilon_{|F|})$ beschrieben, wobei die Anzahl der genutzten Fundamentalprodukte von links nach recht ansteigt. Bäume, die mit der Methode der *kanonischen Schnitte* (siehe Kapitel 5.4.1) ermittelt werden können, sind als gestrichelte Linien in Blau eingezeichnet (*KDNF-Ausdrücke*). Bäume, die Ergebnis einer *vollständigen Dekomposition* (siehe Kapitel 5.4.3.1) sind und damit immer optimale Größe besitzen, sind orange umrandet. Wichtig zu beachten ist, dass die dargestellte Anzahl innerer Knoten für KDNF-Ausdrücke sowie die Anzahl der vollständig faktorisierbaren Zielfestkörper nur schematisch zu verstehen sind und nicht die tatsächlichen Verhältnisse widerspiegeln.

also quadratisch mit der Anzahl der Primitiven. Dieser Fall tritt ein, wenn in jeder Iteration genau ein primitiv faktorisiert wird und damit jedes Primitiv pro Iteration einmal besucht wird (erste Iteration: $|H|$ Besuche, zweite Iteration $|H| - 1$ Besuche, ...). Zusammengenommen resultiert dies bei einer vollständigen Dekomposition im schlechtesten Fall in $\sum_1^{k=|H|} k = \frac{|H|^2+|H|}{2}$ Besuchen. Aus der Dekomposition ergeben sich für den in dieser Arbeit betrachteten Anwendungsfall folgende Probleme:

1. Nicht jeder Festkörper lässt sich für eine Menge von Primitiven vollständig faktorisieren (siehe Abbildung 3.11c).

2. Es lässt sich nicht immer robust und zweifelsfrei prüfen, ob ein Primitiv vollständig innerhalb oder vollständig außerhalb des (Rest-)Festkörpers liegt, da dieser nur über eine potentiell fehlerbehaftete Punktwolke O approximativ repräsentiert wird. Dies macht die Bestimmung dominanter Primitive in diesem Szenario fehleranfällig.

Aus dem ersten Nachteil folgt die Notwendigkeit zur Konzeption von Verfahren, die einen geeigneten Ausdruck für den Restfestkörper finden können. Der zweite Nachteil beschreibt hingegen ein gewichtiges Problem der Dekomposition, das im Rahmen des hier vorgestellten Lösungsansatzes adressiert wird (siehe Kapitel 5.5).

5.4.3.2. Partitionierung

Am Beginn der Überlegungen zur Problempartitionierung steht die Feststellung, dass Primitive, deren Punktmengen keine Schnittmengen aufweisen, bei der Synthese eines passenden KBs separat voneinander behandelt werden können. Dieser Zusammenhang lässt sich als Graph darstellen, dem sog. Intersektionsgraphen $G = (H, E)$. Dabei enthält G für jedes Primitiv aus H einen Knoten und immer dann eine Kante $e \in E$, wenn sich die zu Knoten assoziierten Primitive überschneiden (siehe Abbildung 5.6b). Auf dem Intersektionsgraphen lassen sich nun aus der Graphentheorie bekannte Partitionierungsverfahren anwenden. Für jeden Teilgraphen wird dann das Teilproblem gelöst und Ergebnisse abschließend wieder zu einer Gesamtlösung zusammengefasst. Im Folgenden werden zwei Strategien vorgestellt.

Cliquenbasierte Partitionierung. Zentrale Idee dieser Strategie ist das Auffinden maximaler Cliquen im Intersektionsgraphen. Cliquen sind dabei vollständige Teilgraphen in G. Eine Clique ist maximal, wenn das Hinzufügen eines weiteren Knotens einen nicht mehr vollständigen Teilgraphen zur Folge hat. Das Problem der Enumeration aller maximalen Cliquen ist $\mathcal{NP}$-schwer [41]. Gelöst werden kann es mit dem *Bron-Kerbosch*-Algorithmus [24] (siehe Abbildung 5.6c).
Für jede Clique wird nun das KB-Syntheseproblem separat gelöst (siehe Abbildung 5.6d). Problematisch jedoch ist das Zusammenführen von

Teilergebnissen: Diese können nicht durch das simple Einfügen von Vereinigungsoperatoren verknüpft werden (siehe Abbildung 5.5). Vielmehr muss auf eine komplexe Zusammenführungsstrategie zurückgegriffen werden, die folgende Schritte enthält:

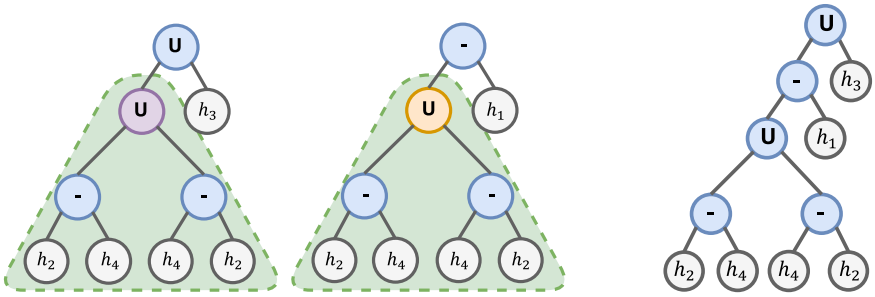
1. Finde ein Teilergebnispaar, dessen Bäume den größten gemeinsamen Teilbaum aller Paare besitzen (siehe Abbildung 5.4a). Dabei sind zwei Teilbäume gleich, wenn sie die selbe Teilpunktmenge von S beschreiben. Die Blattknotenmenge der Teilbäume beider Bäume darf dabei ausschließlich Primitive enthalten. Zudem muss der Wurzelknoten eines jeden Teilbaums einen Pfad entlang von Vereinigungsoperatoren oder entlang des ersten Operanden von Differenzoperatoren bis zum Wurzelknoten des jeweiligen Baums enthalten.

2. Ersetze den Teilbaum des ersten Baums durch den kompletten zweiten Baum (siehe Abbildung 5.4b).

3. Entferne beide Bäume aus der Menge der zu vereinigenden Bäume und füge den zusammengeführten Baum hinzu.

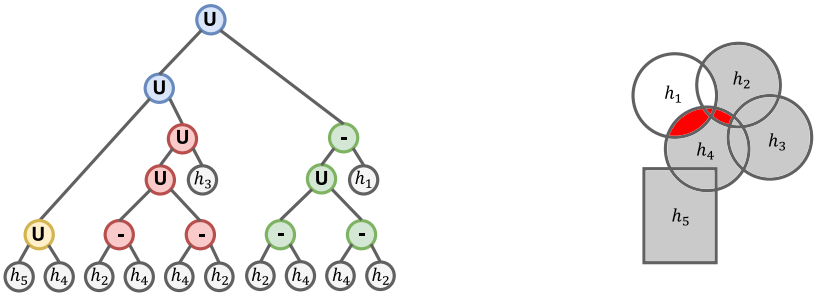
4. Gehe zu Schritt 1, sofern mindestens zwei zu vereinigende Bäume existieren. Ansonsten beende die Prozedur mit dem letzten verbleibenden Teilbaum als Ergebnis (siehe Abbildung 5.6e).

Bei der *cliquenbasierten Partitionierung* verschiebt sich die Komplexität von der KB-Synthese hin zur Zusammenführung von Teilergebnissen. Dabei ist zudem der Teilbaumäquivalenztest (Schritt 1) in dem hier behandelten Szenario nicht immer robust durchführbar, da der Festkörper nur approximativ über die Punktmenge O beschrieben wird. Weiterhin existiert kein formaler Beweis, dass die hier vorgestellte heuristische Methode zur Zusammenführung für jede Eingabe funktioniert und dabei größenoptimale Ergebnisse erzielt.



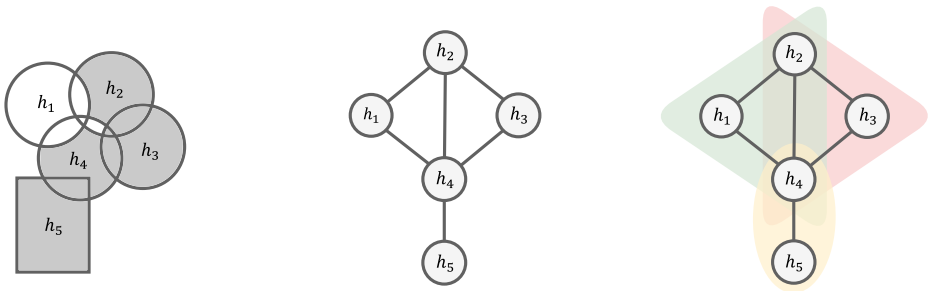
(a) Größte gemeinsame Teilbäume (grün). (b) Resultat.

Abbildung 5.4.: Die Zusammenführungsprozedur für zwei Teilergebnisse.

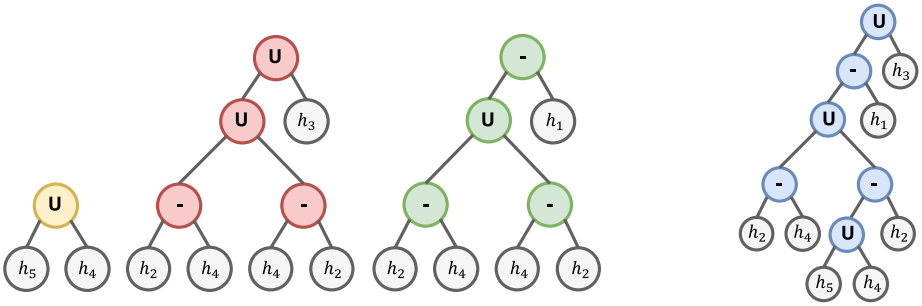


(a) (b)

Abbildung 5.5.: Fehlerhafte Geometrie (rot) beim Zusammenfügen von Teilergebnissen mithilfe von Vereinigungsoperatoren.



(a) $H = \{h_1, ..., h_5\}$ und S in Grau. (b) Intersektionsgraph G. (c) Maximale Cliquen als Partitionen.



(d) Ermittelte Teillösungen.

(e) Ergebnis.

Abbildung 5.6.: Die Schritte der *cliquenbasierten Partitionierung*.

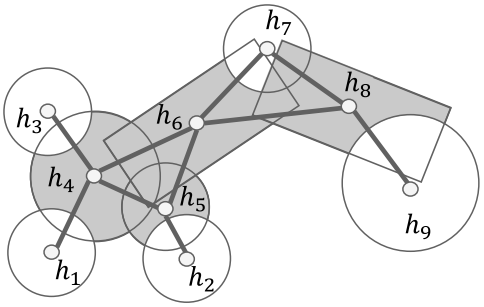
Hybride Partitionierung. Bei dieser Partitionierungsstrategie handelt es sich um ein hybrides Verfahren, das Strukturinformation aus dem Intersektionsgraphen mit dem Konzept der dominanten Primitive verbindet. Die Methode gliedert sich in die folgenden Schritte:

1. Erzeuge den Intersektionsgraph G (siehe Abbildung 5.7a).
2. Finde Knoten, die nur durch eine Kante mit dem Restgraphen verbunden sind (siehe Abbildung 5.7b, rote Kreise).
3. Entferne die in Schritt 2 gefundenen Knoten und ermittle im verbliebenen Graphen alle dominanten Primitive (siehe Abbildung 5.7c). Die Schritte 2 und 3 werden im Folgenden auch als *Pruning* bezeichnet.
4. Trenne alle Verbindungen zwischen dominanten Primitiven und Restgraph (siehe Abbildung 5.7d).
5. Ermittle alle verbundenen Komponenten in G (siehe Abbildung 5.7e, verbundene Komponenten in Gelb).
6. Füge die in Schritt 3 entfernten Knoten und Kanten wieder zu den einzelnen Komponenten hinzu. Die so ermittelten verbundenen Komponenten sind die gesuchten Partitionen (siehe Abbildung 5.7f, Partitionen in Braun).

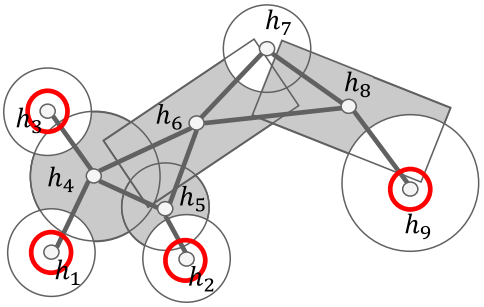
Die auf diese Weise ermittelten Partitionen haben im Vergleich zu denen der *cliquenbasierten Partitionierung* den Vorteil, dass sich die Bäume der korrespondierenden Teilergebnisse mit einfachen Vereinigungsoperatoren zu einem Endergebnis zusammenführen lassen. Das Gesamtergebnis ist dabei immer korrekt und optimal bezüglich der Baumgröße (sofern die Teilergebnisse ebenfalls optimal sind).
Nachteile ergeben sich aus dem Problem, dass dominante Primitive ermittelt werden müssen und dies im bestehenden Szenario nicht immer robust möglich ist. Außerdem erlaubt die gewählte Prozessreihenfolge keine rekursive Anwendung des Verfahrens, wie es z.B. bei der Dekomposition möglich ist. Das in diesem Kapitel vorgestellte System nutzt die Ideen dieser Partitionierungsstrategie, nutzt jedoch Lösungen, die deren Nachteile ausgleichen können.

5.4.4. Zusammenfassung

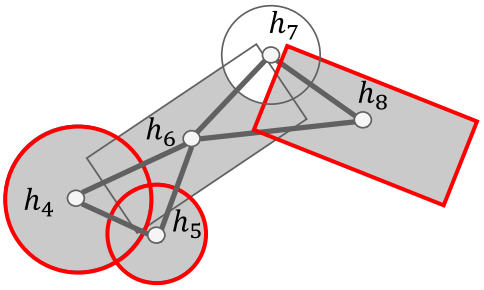
In diesem Kapitel wurde das Problem der KB-Synthese mit bekannten Primitiven H und Punktwolke O im Detail theoretisch beleuchtet. Dabei wurde vor allem Wert auf eine genaue Beschreibung der Beschaffenheit des Suchraums gelegt (siehe Abbildung 5.3). Im Folgenden wird nun ein System vorgestellt, welches das beschriebene Problem auf Basis der theoretischen Überlegungen robust und effizient löst.



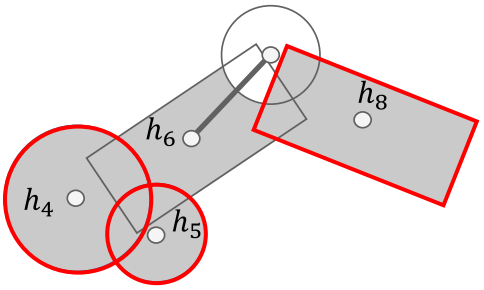
(a) Intersektionsgraph G für Primitive $H = \{h_1, ..., h_9\}$.



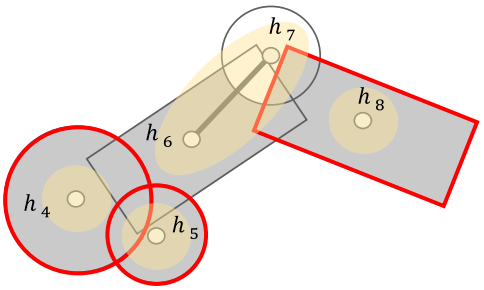
(b) Knoten mit nur einer Kante (rot).



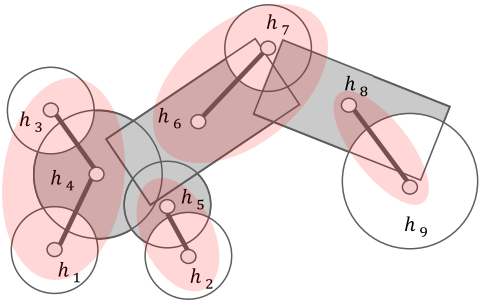
(c) Dominante Primitive (rot).



(d) Entbundene dominante Primitive (rot).



(e) Verbundene Komponenten (gelb).



(f) Zusammengefügte Partitionen (rosa).

Abbildung 5.7.: Prozessschritte der *hybriden Partitionierung* anhand eines Beispiels erläutert.

5.5. Konzept eines Systems zur Konstruktionsbaumsynthese

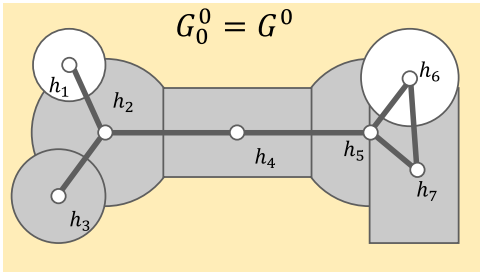
Die grundlegende Lösungsidee ist dabei der Dekomposition entlehnt: Dominante Primitive werden rekursiv detektiert und aus dem gesuchten Festkörper faktorisiert, was das Problem in vielen Fällen stark vereinfacht. Da die Erkennung aller dominanter Primitive in dem hier behandelten Szenario nicht robust möglich ist, fokussiert sich der Ansatz auf diejenigen, die fast ausschließlich mithilfe des Intersektionsgraphen gefunden werden können. Als grundlegende Partitionierungsstrategie kommt zusätzlich die Ermittlung von Zusammenhangskomponenten zum Einsatz, wie sie aus der *hybriden Partitionierung* bekannt ist (siehe Kapitel 5.4.3.2).
Im Folgenden werden die einzelnen Teilschritte im Detail erläutert und deren Verknüpfung in Kapitel 5.5.8 beschrieben. Zur besseren Übersicht wird der Gesamtprozessablauf in Abbildung 5.10 schematisch dargestellt. Die folgenden Unterkapitel sind dabei mit den einzelnen Prozessschritten assoziiert. Abbildung 5.8 zeigt die Resultate einzelner Prozessschritte anhand eines Beispiels.

5.5.1. Erzeugung des Intersektionsgraphen

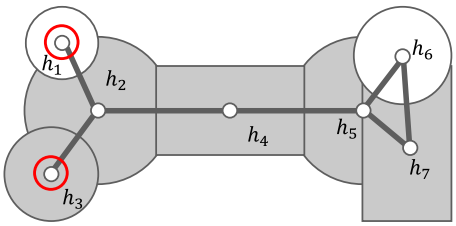
Aus den Eingabeprimitiven H wird der *Intersektionsgraph* G^i der Iterationstiefe $i = 0$ ermittelt. Dazu wird überprüft, ob sich je zwei Primitive schneiden. Bei den hier betrachteten Primitiventypen (Zylinder, Kugeln, konvexe Polytope) existieren dazu numerisch robuste Algorithmen [135]. Ohne Optimierung mittels einer räumlichen Datenstruktur, wie zum Beispiel eines Octrees (siehe Kapitel 2.3.2.2), ist die Laufzeitkomplexität dieses Schritts $O(|H^i|^2)$.

5.5.2. Erzeugung des VDF-Gitternetzes

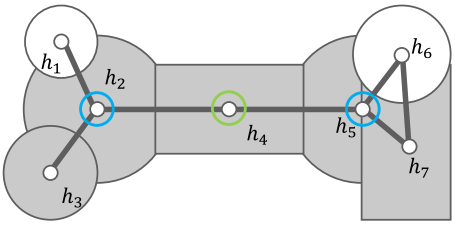
Aus der Eingabepunktwolke O wird ein *VDF-Gitternetz* N^i zur approximativen Repräsentation des Festkörpervolumens erzeugt. Dieses wird in Folgeschritten für die Beschleunigung der komponentenweisen Bestimmung des Typs ermittelter Fundamentalprodukte sowie für die effiziente Quantifizierung der geometrischen Passgenauigkeit eines Lösungskandidaten in der *multikriteriellen Optimierung* verwendet. Dazu wird zuerst ein Dreiecksnetz mittels Poisson-Oberflächenrekonstruktion [98] aus der Punktwolke O erzeugt. Daraufhin wird für jeden Gitternetzpunkt in N^i die vorzeichenbehaftete Distanz zur per Dreiecksnetz approximierten Festkörperoberfläche ermittelt.
Besteht aus vorhergehenden Rekursionsschritten bereits ein *VDF-Gitternetz* N^{i-1}, wird dies wiederverwendet. Dazu müssen bereits faktorisierte dominante Primitive D^{i-1} per Vereinigungsoperation $\cup^*$ mit N^{i-1} kombiniert werden, um N^i zu erhalten.



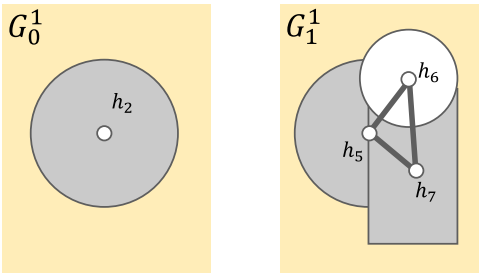
(a) G^0 für Primitive $H = \{h_1, ..., h_7\}$. Komponente $K^0 = \{G_0^0\}$ in Gelb.



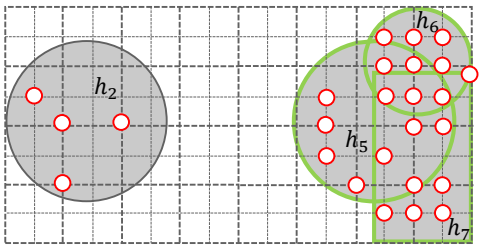
(b) *Pruning*. Entfernbare Primitive in Rot.



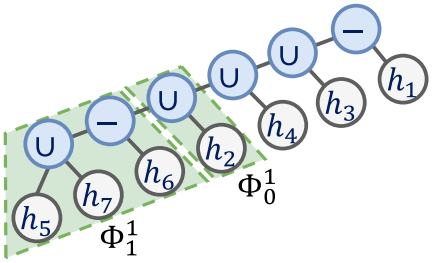
(c) *Artikulationspunktanalyse*. Blau: Artikulationspunkte, grün: Entfernbare Artikulationspunkte.



(d) *Ermittlung zusammenhängender Komponenten* in G_0^1. Komponenten K^1 in Gelb.



(e) *Erzeugung der Fundamentalproduktpunktwolke*. Produkte hellgrün umrandet. Punkte O_f^1 rot umrandet. Gitternetz N^1 in Grau.



(f) KB Φ als Ergebnis von *Teilbaumfusionen*.

Abbildung 5.8.: Ergebnisse der einzelnen Prozessschritte. a) In der Rekursionstiefe $i = 0$ existiert genau eine Komponente in $K^0 = \{G_0^0\}$ mit Primitiven $H_0^0 = H$ und $G_0^0 = G^0$. b) In G_0^0 finden sich durch *Pruning* die dominanten Primitive $\{h_1, h_3\}$. c) Die *Artikulationspunktanalyse* identifiziert mit h_4 ein weiteres dominantes Primitiv. Damit ergibt sich $D_0^0 = \{h_1, h_3, h_4\}$, $H_0^1 = \{h_2, h_5, h_6, h_7\}$ sowie G_0^1 aus G_0^0 ohne die Knoten in D_0^0. d) *Ermittlung zusammenhängender Komponenten* von G_0^1, $K^1 = \{G_0^1, G_1^1\}$. G_0^1 enthält nur das Primitiv h_2. Damit ist $D_0^1 = \{h_2\}$ woraus sich direkt Φ_0^1 ergibt. Für G_1^1 ist D_1^1 leer und $H_1^1 = \{h_5, h_6, h_7\}$. Somit muss für die Ermittlung des Teilbaums Φ_1^1 die *multikriterielle Optimierung* angewandt werden. e) Auf N^1 basierende Punktmenge O_f^1. f) Fusionierter KB.

5.5.3. Erzeugung der Fundamentalproduktpunktwolke

Die Bewertung der geometrischen Passgenauigkeit von Lösungskandidaten in der *multikriteriellen Optimierung* erfolgt normalerweise folgendermaßen: Für jeden Gitternetzpunkt aus N^i wird die absolute Differenz zwischen der gespeicherten vorzeichenbehafteten Distanz zur Festkörperoberfläche und der vorzeichenbehafteten Distanz zwischen Gitternetzpunkt und Oberfläche des durch den Lösungskandidaten repräsentierten Festkörpers betrachtet (siehe Kapitel 5.5.7). Da die Anzahl der Gitternetzpunkte mit der Größe und der Auflösung des Gitternetzes kubisch ansteigt, muss eine effizientere Lösung gefunden werden.
Die Idee ist hierbei, nur relevante Gitternetzpunkte auszuwählen und diese in einer *Punktwolke* O_f^i für die spätere Verwendung abzulegen. Dazu wird für jeden Punkt aus N^i das ihn umgebende Fundamentalprodukt bestimmt. Die zugrundeliegende Menge der Fundamentalprodukte ist dabei die von den Primitiven H^i induzierte. Im Anschluss werden für jedes Fundamentalprodukt n (hier: $n = 100$) Punkte per FPS ausgewählt und O_f^i hinzugefügt. Der Vorteil dieser Vorgehensweise ist die Reduktion der zu prüfenden Gesamtpunktmenge, da nur Punkte innerhalb von Primitiven und damit innerhalb von relevanten Regionen verwendet werden. Die Fokussierung auf Fundamentalprodukte erlaubt zudem eine hohe Abtastrate in Regionen mit hohem Geometriedetail.

5.5.4. Ermittlung zusammenhängender Komponenten

Wie bei der *hybriden Partitionierung* werden im Intersektionsgraphen $G^i = (H^i, E^i)$ der aktuellen Rekursionstiefe i die verbundenen *Komponenten* K^i ermittelt. Dies erfolgt über eine Tiefensuche mit Laufzeitkomplexität $O(|H^i| + |E^i|)$ [33].

5.5.5. Pruning

Für jede Komponente in K^i mit Index k wird der enthaltene Intersektionsgraph G_k^i auf das Vorhandensein von Knoten mit nur einer Kante überprüft, wie es bereits bei der Methode der *hybriden Partitionierung* beschrieben wurde. Dabei sind die assoziierten Primitive in jedem Fall dominante Primitive, weil sie nur die Möglichkeit haben, mit genau einem Primitiv zu interagieren und jede Operation außer $-^*$ und $\cup^*$ zu nicht verbundener Geometrie führen würde. Gleiches gilt für das Primitiv eines Intersektionsgraphen mit nur einem Primitiv. Für die folgende *Ausdrucksfusion* muss jedoch zusätzlich auch dessen Typ (komplett innerhalb liegend, komplett außerhalb liegend) bestimmt werden, falls der Intersektionsgraph mehr als ein Primitiv enthält. Dazu wird für das Primitiv die geometrische Passgenauigkeit $V_{N^i}(\cdot)$ (siehe Gleichung 4.5) ausgewertet. Ist der ermittelte Wert größer als $0,5$, wird angenommen, dass das Primitiv innerhalb des durch N^i repräsentierten Festkörpers liegt (komplett innerhalb liegend), ist er kleiner, nicht (komplett außerhalb liegend). Die

Überprüfung ist robust, da bereits bekannt ist, dass es sich bei dem überprüften Primitiv um ein dominantes Primitiv handelt, es also entweder vollständig innerhalb oder außerhalb liegt.

5.5.6. Artikulationspunktanalyse

Eine weitere Methode, um innere dominante Primitive direkt aus einer Komponente des Intersektionsgraphen abzuleiten, ist die *Artikulationspunktanalyse*. Artikulationspunkte eines zusammenhängenden Graphen sind diejenigen Knoten, deren Entfernung den Graphen in mindestens zwei zusammenhängende Komponenten teilen würde. Die Ermittlung von Artikulationspunkten erfolgt über eine Tiefensuche mit Laufzeitkomplexität $O(|H^i| + |E^i|)$ [185].
Alle Artikulationspunkte in G_k^i werden bestimmt und diejenigen ausgewählt, die die beiden folgenden Kriterien erfüllen:

1. Der Artikulationspunkt hat nur Nachbarknoten, die ebenfalls Artikulationspunkte sind.

2. Für alle benachbarten Artikulationspunkte gilt, dass sich nicht mehr als ein Nachbar in einer maximalen Clique mit dem Artikulationspunkt befinden darf.

Die zu den ausgewählten Artikulationspunkten gehörenden Primitive müssen innere dominante Primitive sein, wenn davon ausgegangen wird, dass die innerhalb einer Komponente beschriebene Geometrie zusammenhängend ist. Die bei den Schritten *Pruning* und *Artikulationspunktanalyse* ermittelten, dominanten Primitive D_k^i werden aus G_k^i entfernt und ein weiterer Rekursionsschritt $i = i+1$ ausgeführt. Die Idee der Suche nach maximalen Cliquen im Intersektionsgraphen ist der *cliquenbasierten Partitionierung* entnommen (siehe Kapitel 5.4.3.2). Jedoch werden hier maximale Cliquen nicht zur Partitionierung, sondern zur Identifikation faktorisierbarer Artikulationspunkte verwendet.

5.5.7. Multikriterielle Optimierung

Wenn keine dominanten Primitive mehr gefunden werden können ($D_k^i = \emptyset$), ist die maximale Rekursionstiefe erreicht. Ist die aktuelle Primitivenmenge dabei nichtleer ($H_k^i \neq \emptyset$), muss ein Teilbaum für die restlichen Primitive H_k^i gefunden werden. Diese Suche nach dem optimalen Baum Φ_k^i wird dazu als multikriterielles kombinatorisches Optimierungsproblem formuliert und per EA gelöst. Beinhaltet H_k^i weniger als drei Primitive, wird Φ_k^i durch einfaches Ausprobieren aller Lösungsmöglichkeiten ermittelt. Die Struktur des EAs folgt dabei der in Kapitel 2.4.3.2 beschriebenen.

Start. Zu Beginn wird die Startpopulation (siehe Abbildung 2.13, *Zufällige Population* 0) mit n_{pop} zufällig erzeugten KBs befüllt. Für die Baumerzeugung werden Primitive aus H_k^i verwendet, um Bäume mit $n_{min} = |H_k^i| - 1$ bis n_{max}

inneren Knoten randomisiert zu erzeugen.

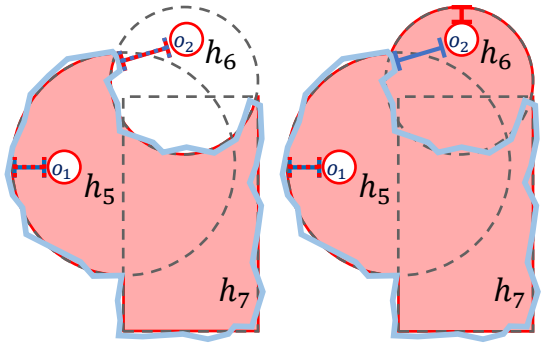
Bewertung. Die zu maximierende Zielfunktion

$$F_{O_f^i,N^i}(\Phi_c) = \alpha \cdot G_{O_f^i,N^i}(\Phi_c) - \beta \cdot \#(\Phi_c) \tag{5.5}$$

bewertet ein einzelnes Individuum Φ_c der aktuellen Population. Dabei ist $G_{O_f^i,N^i}(\cdot)$ der Geometrieterm und α und β sind benutzerdefinierte Parameter. $\#(\cdot)$ zählt die Knoten eines KBs und dient der Bestrafung unnötig großer Individuen. Der Geometrieterm

$$G_{O_f^i,N^i}(\Phi_c) = -\frac{1}{|O_f^i|} \sum_{o \in O_f^i} |F_{\Phi_c}(o) - F_{N^i}(o)| \tag{5.6}$$

misst die geometrische Passgenauigkeit des KBs Φ_c und stellt die negierte, in [179] vorgestellte Approximation der generalisierten Chamfer-Distanz dar. Dabei ist $F_{\Phi_c}(\cdot)$ die VDF des KBs und F_{N^i} die des *VDF-Gitternetzes* N^i (siehe Abbildung 5.9). Um die Wahl der Parameter α und β zu vereinfachen, werden die Terme $G_{O_f^i,N^i}(\cdot)$ und $\#(\cdot)$ normiert, indem die jeweiligen Maximal- und Minimalwerte innerhalb der aktuellen Population herangezogen werden.



(a) (b)

Abbildung 5.9.: Darstellung des Geometrieterms $G_{O_f^i,N^i}(\cdot)$ anhand zweier Beispiele (aufbauend auf Abbildung 5.8). Punktmengen der KBs in Hellrot. Durch N^i beschriebene Approximation der zu repräsentierenden Festkörpers in Hellblau. Punkte aus $O_f^i = \{o_1, o_2\}$ in Weiß mit rotem Rand. a) Lösungskandidat Φ_c hat für beide Punkte aus O_f^i VDF-Werte, die nahezu identisch mit denen von N^i sind, also $|F_{\Phi_c}(o_j) - F_{N^i}(o_j)| \approx 0, j \in \{1, 2\}$. b) Lösungskandidat Φ_c hat für Punkt o_2 von stark N^i abweichende VDF-Werte (rot und blau linierte Intervalle) und ist damit ein schlechterer Lösungskandidat verglichen mit dem aus a), weil $|F_{\Phi_c}(o_2) - F_{N^i}(o_2)| \gg 0$.

Elternauswahl. Die Auswahl der Elternindividuen für die *Variation* erfolgt mittels Turnier-Selektion (siehe Kapitel 2.4.3.2). Der Auswahlprozess und die folgende *Variation* durch *Rekombination* und *Mutation* werden dabei für eine festgelegte Anzahl von $n_{var} \leq n_{pop}$ Iterationen wiederholt.

Rekombination. Zwei selektierte KBs (Elternbäume) werden mit einer benutzerdefinierten Wahrscheinlichkeit von p_{rek} rekombiniert, indem zufällig ausgewählte Teilbäume ausgetauscht werden.

Mutation. Ein bereits rekombinierter KB wird zusätzlich mit einer benutzerdefinierten Wahrscheinlichkeit von p_{mut} einer Mutationsoperation unterzogen.

- **Modifizieren:** Ein zufällig gewählter Teilbaum wird durch einen zufällig neu erstellten Teilbaum ersetzt.
- **Neu:** Der komplette KB wird durch einen zufällig neu erstellten KB ersetzt.

Dabei hat die Mutationsvariante *Neu* die benutzerdefinierte Wahrscheinlichkeit p_{neu}. Die Wahrscheinlichkeit der Mutationsvariante *Modifizieren* ergibt sich dabei aus $1 - p_{neu}$.

Gesamtauswahl. Die neue Population besteht aus den n_{var} per *Variation* veränderten KBs. Zusätzlich werden ihr die $n_{pop} - n_{var}$ besten KBs der aktuellen Population hinzugefügt.

Abbruch. Bei Erreichen einer benutzerdefinierten Anzahl von n_{iter} Iterationen terminiert der EA. Dies geschieht auch, wenn sich innerhalb der Population der beste KBs für m_{iter} Iterationen nicht weiter verbessert.

Der EA hat die in Tabelle 5.1 dargestellten Parameter.

Parameter	Beschreibung
n_{pop}	Größe der Population
n_{var}	Anzahl der Wiederholungen der Variationsoperatoren
n_{iter}	Max. Anzahl an Optimierungsiterationen
m_{iter}	Max. Anzahl an Iterationen ohne Verbesserung
α,β	Gewichte der Terme der Zielfunktion (siehe Gleichung 5.5)
p_{mut}	Mutationswahrscheinlichkeit
p_{rek}	Rekombinationswahrscheinlichkeit
p_{neu}	Wahrscheinlichkeiten der Mutationsvariante *Neu*

Tabelle 5.1.: Benutzerdefinierte Parameter für die *multikriterielle Optimierung.*

5.5.8. Prozessablauf

Die in den bisherigen Kapiteln beschriebenen Prozessteilschritte sind zu einem Ablauf verknüpft (siehe Abbildung 5.10). Im Folgenden werden Details dazu erläutert.
Zu Beginn (Rekursionstiefe $i = 0$), wird der *Intersektionsgraph* G^i aus den Primitiven $H^i = H$ erzeugt. Darauf folgt die Erstellung des *VDF-Gitternetzes* N^i, aus dem dann die reduzierte *Punktwolke* O_f^i auf Basis der aus H^i ermittelten Fundamentalprodukte generiert wird. Im Anschluss wird der Graph G^i in zusammenhängende Komponenten K^i zerlegt.
Für jede Komponente folgt nun die Ermittlung der in ihr enthaltenen dominanten Primitive D_k^i durch das *Pruning* und die *Artikulationspunktanalyse*. Daraus resultieren auch die für jede Komponente separat vorhandenen *Intersektionsgraphen* G_k^{i+1} und Primitive H_k^{i+1} für die jeweils nächste Rekursionstiefe $i = i+1$. Nun wird für jede Komponente überprüft, ob dominante Primitive gefunden wurden. Ist dies der Fall, wird das Verfahren mit G_k^{i+1} und H_k^{i+1} rekursiv wiederholt, um den Teilbaum Φ_k^{i+1} zu erhalten. Der rekursive Aufruf wird jedoch nur dann ausgeführt, wenn H_k^{i+1} Primitive enthält, ansonsten ist Φ_k^{i+1} leer. Φ_k^{i+1} wird dann zusammen mit den dominanten Primitiven D_k^i zum *Teilbaum* Φ_k^i kombiniert (*Teilbaumfusion* $\Phi_k^{i+1}, D_k^i \Rightarrow \Phi_k^i$). Dazu werden zuerst alle innenliegenden dominanten Primitive in D_k^i per $\cup^*$-Operation kombiniert und mit Φ_k^{i+1} fusioniert. Alle außenliegenden dominanten Primitive werden per $-^*$-Operation mit dem Resultat kombiniert, was in dem *Teilbaum* Φ_k^i resultiert. Enthält die Komponente hingegen keine dominanten Primitive ($D_k^i = \emptyset$), wird die *multikriterielle Optimierung* angewandt, um den *Teilbaum* Φ_k^i zu ermitteln, sofern H_k^i noch Primitive enthält. Enthält H_k^i keine Primitive mehr ($H_k^i = \emptyset$), ist der *Teilbaum* Φ_k^i leer.
Wurden für alle Komponenten Teilbäume ermittelt, werden diese mit $\cup^*$ kombiniert als Φ^i zurückgegeben (*Teilbaumfusion* $\Phi_k^i \Rightarrow \Phi^i$). Das zurückgegebene Gesamtergebnis ist dann das der ersten Rekursionstiefe $i = 0$, also $\Phi = \Phi^0$.

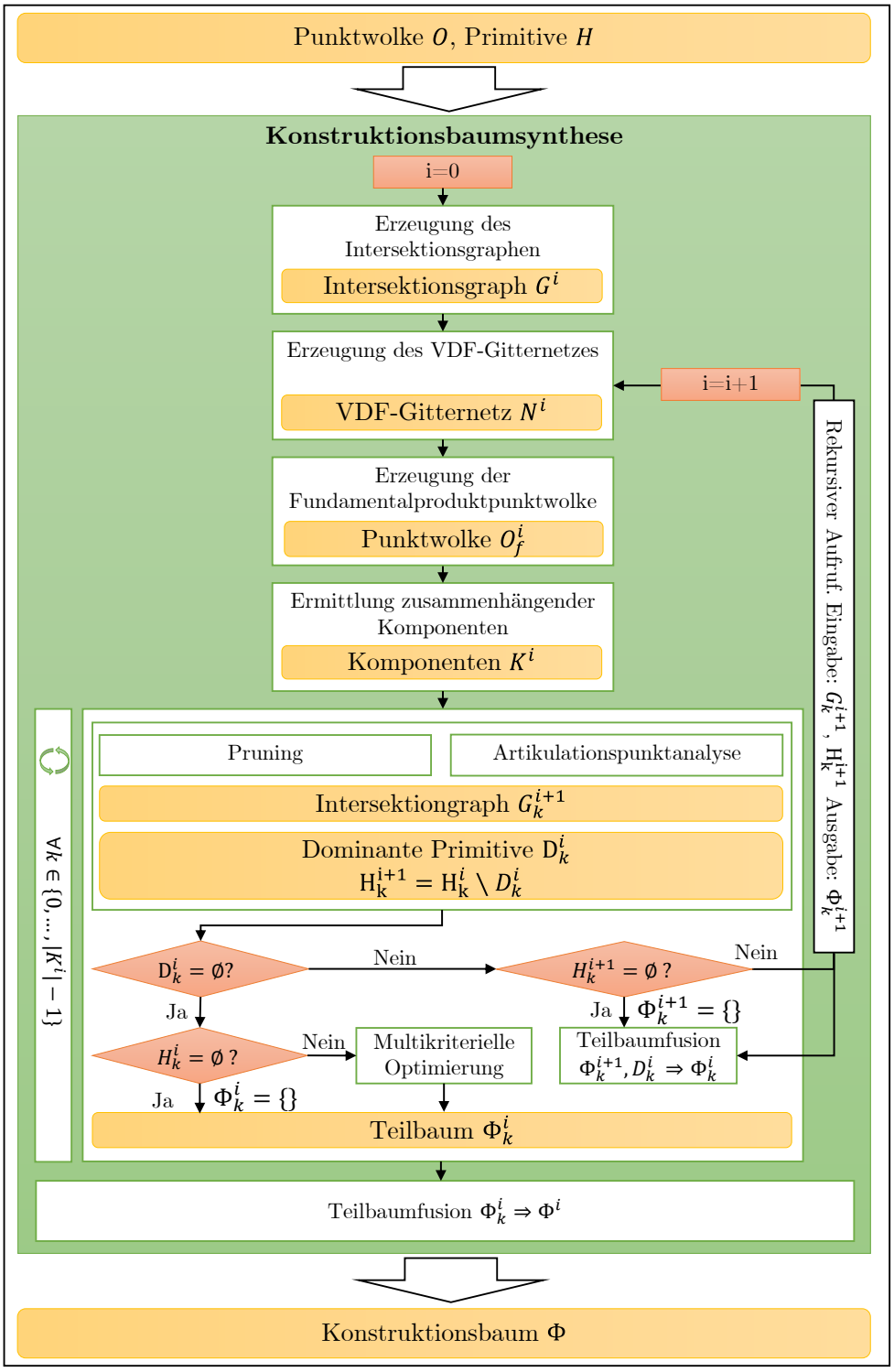


Abbildung 5.10.: Prozess-Pipeline zur Synthese von KBs.

5.6. Evaluation

Die Evaluation des vorgestellten Systems wurde anhand von 12 Modellen durchgeführt (siehe Abbildungen 5.11, 5.12 und 5.13 sowie Tabelle 5.4). Die Modelle M1, M4, M6, M9, M12 sind dabei aus den gleichnamigen Ergebnissen der Primitivenextraktion entnommen (siehe Kapitel 4). Die restlichen Datensätze wurden von Hand erzeugt. Um der Stochastizität des eingesetzten EAs Rechnung zu tragen, wurden Experimente dreimal wiederholt. Als Evaluationshardware kam ein Laptop mit Vierkernprozessor (2,6GHz) und 16GB RAM zum Einsatz. Die Parameter des verwendeten EAs finden sich in den Tabellen 5.2 und 5.3 wieder. Im Fokus der Betrachtung stehen die zu erwartenden Vorteile bzgl. Laufzeiteffizienz und Ergebnisqualität, die durch das eingeführte System (im Folgenden *System* genannt) zu erwarten sind (siehe Kapitel 5.5). Dazu dient ein Verfahren ohne Partitionierungs- und Vereinfachungsmethoden als Vergleichsgröße, das ausschließlich die *multikriterielle Optimierung* nutzt und im Folgenden *nur EA* genannt wird. Damit ähnelt das Vergleichssystem Verfahren, die nur einen EA zur KB-Synthese einsetzen, wie z.B. [47].

Parameter	n_{pop}	n_{var}	n_{iter}	p_{mut}	p_{rek}	p_{neu}	α
Wert	150	1	500	0,3	0,4	0,5	1

Tabelle 5.2.: Für die Evaluation gewählte Parameter für die *multikriterielle Optimierung*. Siehe auch Tabelle 5.3.

Param.	**M1**	**M2**	**M3**	**M4**	**M5**	**M6**
m_{iter}	50	100	100	100	100	100
β	0,001	0,0005	0,001	0,000001	0,0001	0,0001
	M7	**M8**	**M9**	**M10**	**M11**	**M12**
m_{iter}	100	100	100	300	300	100
β	0,001	0,0005	0,00001	0,00001	0,000001	0.0001

Tabelle 5.3.: Für die Evaluation gewählte modellspezifische Parameter für die *multikriterielle Optimierung*.

Eingabe	**M1**	**M2**	**M3**	**M4**	**M5**	**M6**	**M7**	**M8**	**M9**	**M10**	**M11**	**M12**
$\|H\|$	5	7	20	10	10	17	14	37	14	9	17	7
$\|O\|$	48k	64k	14k	45k	21k	27k	6k	28k	45k	6k	55k	57k

Tabelle 5.4.: Daten zu den in der Evaluation verwendeten Modellen.

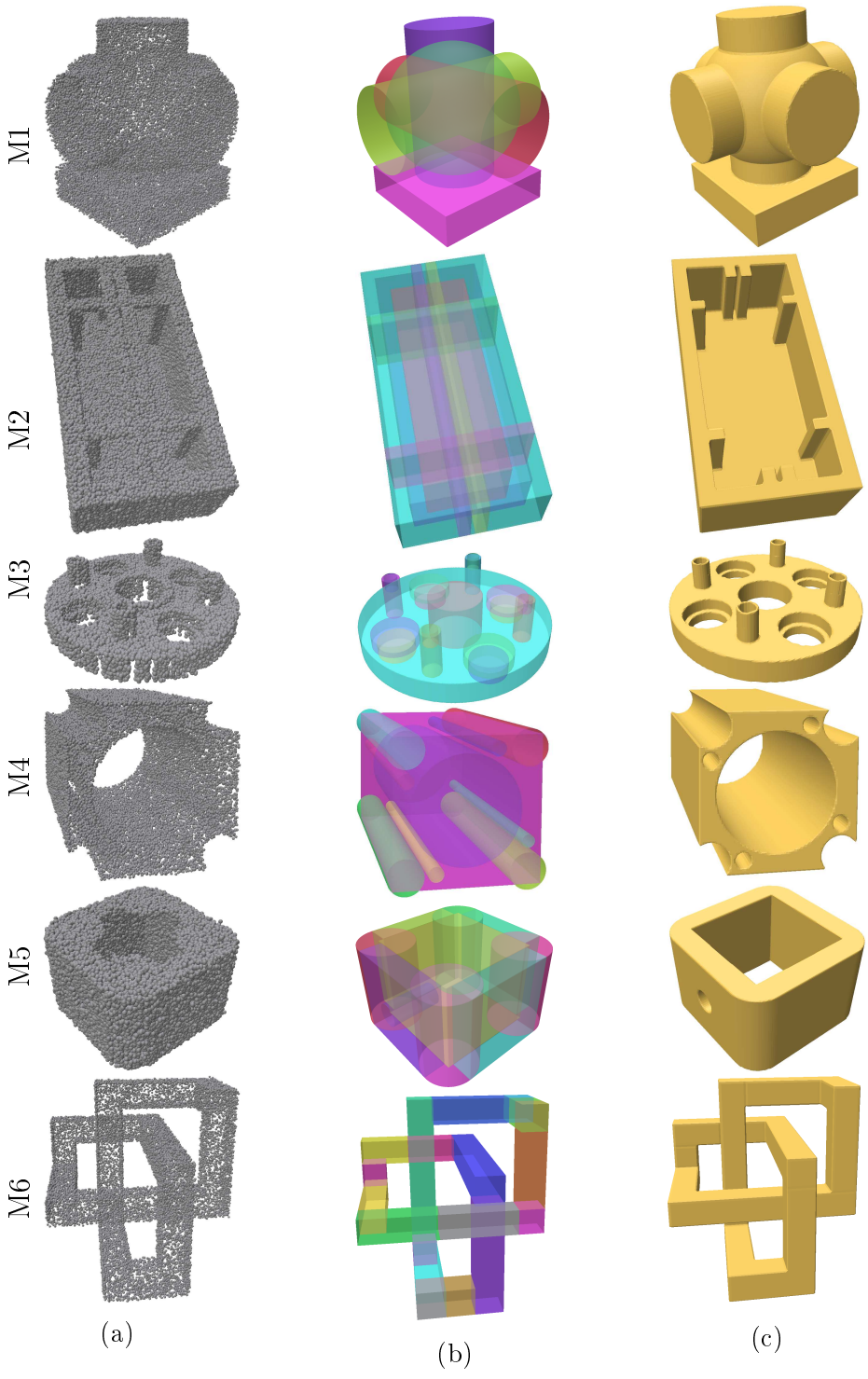


Abbildung 5.11.: a) Eingabepunktwolke O, b) Eingabeprimitive H und c) Ergebnisdarstellung für die Modelle M1-M6 für das *System*.

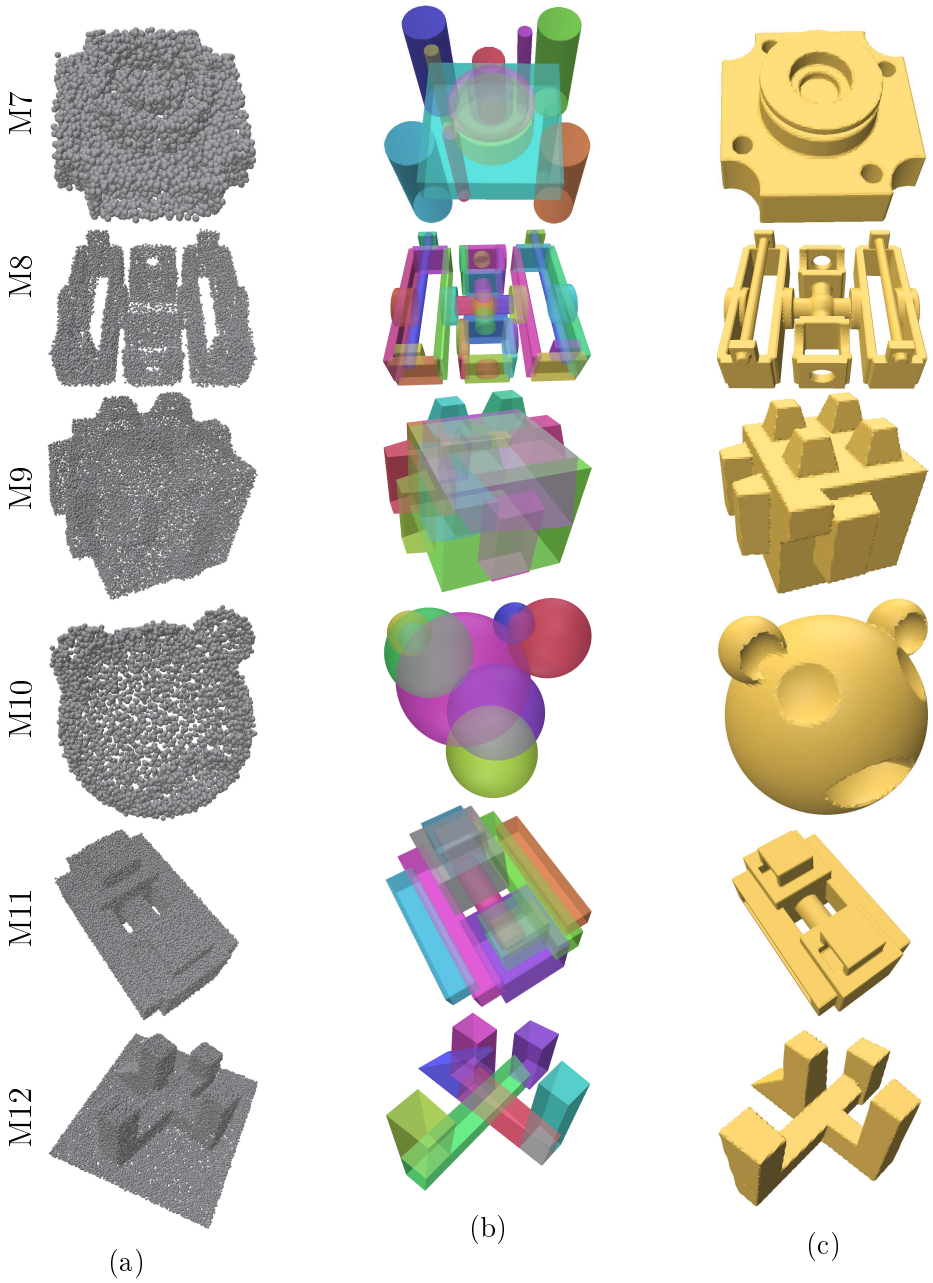


Abbildung 5.12.: a) Eingabepunktwolke O, b) Eingabeprimitive H und c) Ergebnisdarstellung für die Modelle M7-M12 für das *System*.

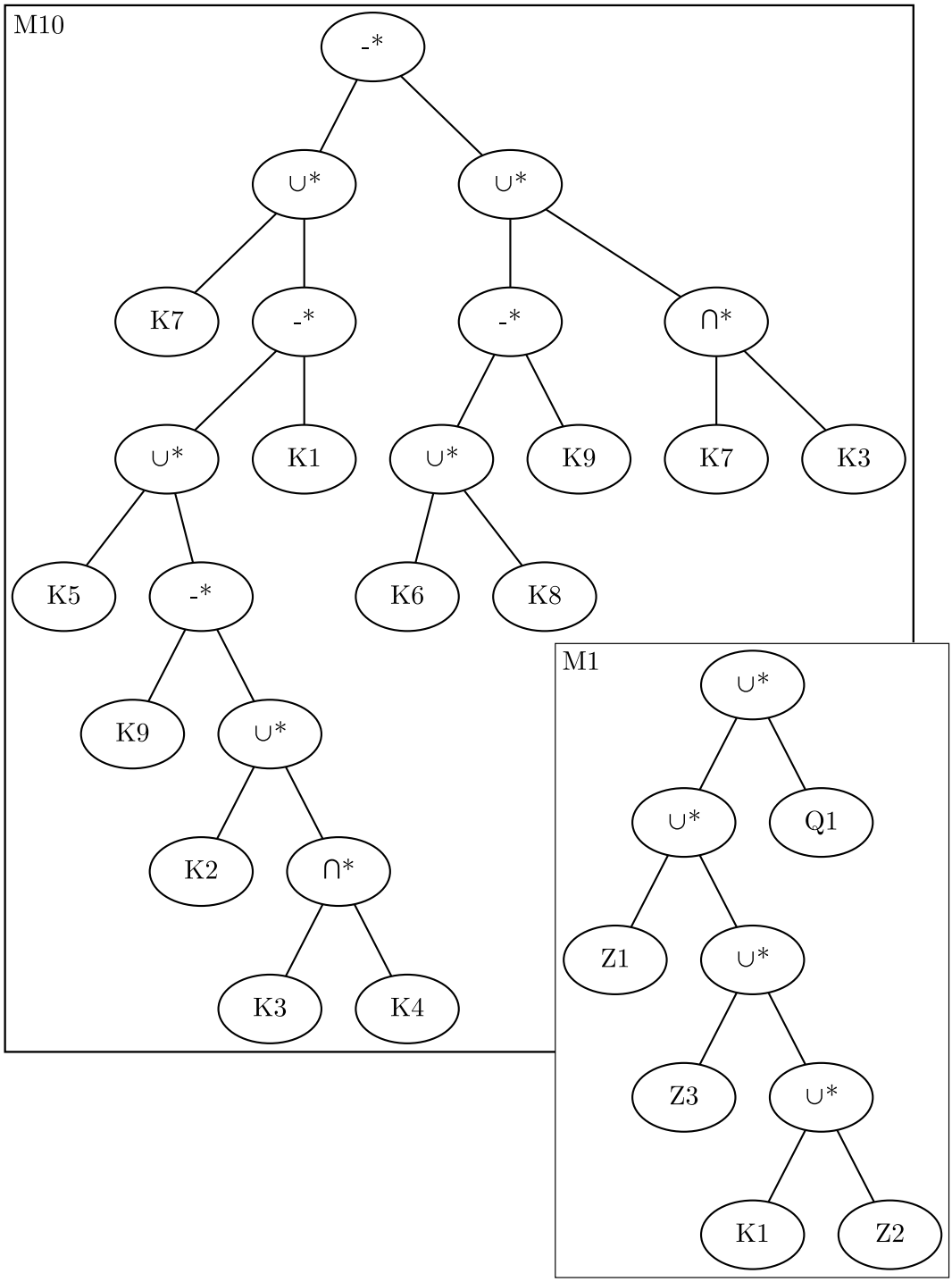


Abbildung 5.13.: Aus dem vorgestellten *System* resultierende KBs beispielhaft dargestellt anhand der Modelle M1 und M10.

5.6.1. Ergebnisqualität

Die Qualität erzielter Syntheseergebnisse lässt sich analog zur Zielfunktion (siehe Gleichung 5.5) anhand zweier Merkmale quantifizieren, dem geometrischen Einpassfehler $E_{O_f^0,N^0}(\Phi) = -G_{O_f^0,N^0}(\Phi)$ und der Größe $\#(\Phi)$ des resultierenden Baums Φ.

Geometrischer Einpassfehler. Abbildung 5.14 stellt den geometrischen Einpassfehler $E_{O_f^0,N^0}(\Phi)$ für alle Modelle und Methoden (*System*, *nur EA*) dar (kleiner ist besser). Bei den Modellen M3, M7, M8 und M11 ergeben sich für das *System* gegenüber der Variante *nur EA* Vorteile. Zu sehen sind diese auch in den Dreiecksnetzdarstellungen der Ergebnisse des *Systems* (siehe Abbildungen 5.11c und 5.12c) sowie in den sichtbaren Einpassfehlern bei der *nur EA*-Variante (siehe Abbildung 5.15). Zu erklären sind die besseren Ergebnisse für das *System* v.a. mit der Komplexität der entsprechenden Modelle (siehe Tabelle 5.4) bei gleichzeitiger Effektivität der angewandten Vereinfachungs- und Partitionierungsstrategien (siehe Tabelle 5.5).
Abweichungen in den Werten in Abbildung 5.14 bei den übrigen Modellen (z.B. bei Modell M2) lassen sich durch numerische Fehler sowie dadurch erklären, dass die VDF-Formulierungen für konvexe Polytope und reguläre Mengenoperatoren nur Approximationen darstellen (siehe Gleichung 2.28 und Kapitel 2.3.1.1). Damit können zwei KBs, die eigentlich denselben Festkörper beschreiben, leicht abweichende Werte für $E_{O_f^0,N^0}$ ergeben. Die zugehörigen Dreiecksnetzrepräsentationen sind jedoch, ebenso wie der beschriebene Festkörper, in diesen Fällen identisch.
Dass leicht unterschiedliche Werte für $E_{O_f^0,N^0}$ in der gleichen Dreiecksrepräsentation resultieren, kann auch an der dazu durchgeführten Dreiecksnetzkonversion von Φ liegen. Diese nutzt das *Marching Cubes*-Verfahren [113], welches immer approximativen Charakter hat, ebenso wie die Dreiecksnetzrepräsentation des KBs, die es erzeugt.

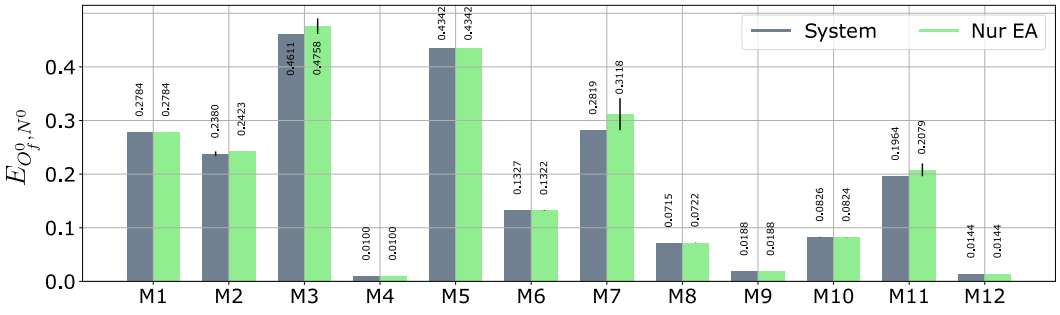
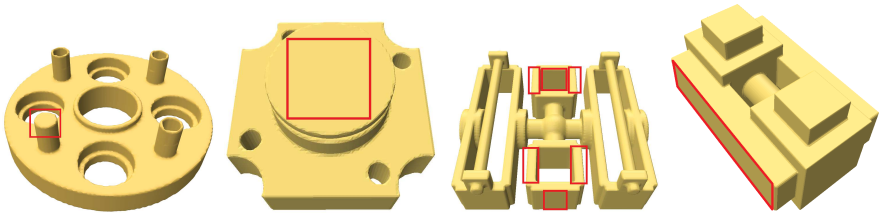


Abbildung 5.14.: Geometrischer Einpassfehler der Ergebnis-KBs.



(a) M3 (b) M7 (c) M8 (d) M11

Abbildung 5.15.: Auftretende Einpassfehler (rot) bei der Variante *nur EA*.

Größe. Die Größe #(·) der Ergebnis-KBs wird in Abbildung 5.16 dargestellt. Für die Modelle M1, M4, M6, M9 und M12 ist das *System* mit der Ground Truth identisch. Bei der *nur EA*-Variante ist das nur für Modell M1 der Fall. Dies ist wiederum auf die vom *System* genutzten Vereinfachungs- und Partitionierungsstrategien zurückzuführen (siehe Tabelle 5.5), da für Primitive, die auf diese Weise dem Problem entnommen werden, ein größenoptimaler Teilbaum synthetisiert wird. Diese Strategien wirken sich auch positiv auf die Größe der Modelle M3, M7 und M8 aus. Greifen genannte Methoden nicht oder nicht ausschlaggebend, erzielt aufgrund der Stochastizität des EAs, entweder das *System* (Modell M5) oder die *nur EA*-Variante (Modelle M2, M10 und M11) bessere Ergebnisse bzgl. der Baumgröße.

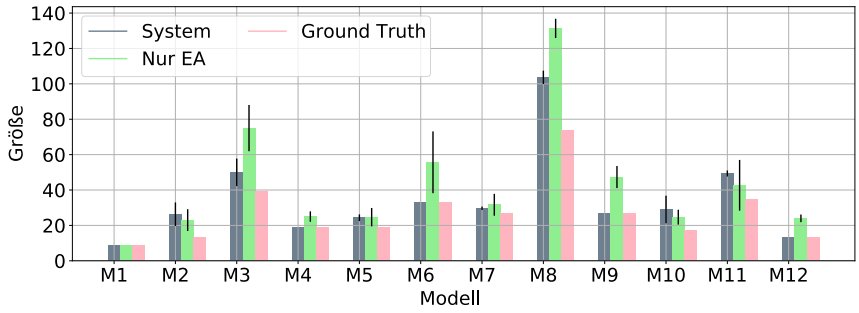


Abbildung 5.16.: Größe der Ergebnis-KBs.

Merkmal	**M1**	**M2**	**M3**	**M4**	**M5**	**M6**	**M7**	**M8**	**M9**	**M10**	**M11**	**M12**
# Ge-pruned	1	0	8	9	0	0	9	12	0	0	2	5
# Art.-Pkte.	0	0	0	0	0	0	0	9	0	0	0	0
# Verb. Komp.	1	1	1	1	1	17	1	1	14	1	1	2

Tabelle 5.5.: Werte der Merkmale *Anzahl durch Pruning entfernter Primitive* (Zeile 1), *Anzahl entfernter Artikulationspunkte* (Zeile 2) und *Anzahl initial vorhandener verbundener Komponenten* (Zeile 3).

5.6.2. Laufzeitbetrachtungen

In Abbildung 5.17 sind die Laufzeiten für die beiden Prozessschritte *Erzeugung des VDF-Gitternetzes* und *multikriterielle Optimierung* jeweils für das *System* (linker Balken pro Modell) und für die Ausführung der *nur EA*-Variante (rechter Balken pro Modell) dargestellt. Auf die Aufführung der Laufzeiten der übrigen Prozessschritte wurde verzichtet, da diese zusammengenommen immer kleiner als 1s und damit vernachlässigbar sind. Generell ist erkennbar, dass die *multikriterielle Optimierung* den größten Anteil an der Gesamtlaufzeit hat.
Für die Modelle M3, M4, M5, M7, M8 und M12 ist das *System* klar laufzeiteffizienter, was auf die im Schritt *Pruning* durchgeführte Vereinfachung zurückzuführen ist (siehe Tabelle 5.5, *Anzahl durch Pruning entfernter Primitive*). Eine Ausnahme bildet hier M5, für das das *Pruning* als Erklärung nicht greift. Für M8 kommt hinzu, dass die *Artikulationspunktanalyse* zur Verbesserung der Effizienz beitragen kann (siehe Tabelle 5.5, *Anzahl entfernter Artikulationspunkte*). Ebenso ergeben sich Laufzeitvorteile für das *System* für die Modelle M6 und M9. Grund hierfür ist die initiale Problempartitionierung in vergleichsweise viele Komponenten (siehe Tabelle 5.5, *Anzahl initial vorhandener verbundener Komponenten*). Für die Modelle M1, M2, M10 und M11 ergibt sich kein Vorteil für das *System*. Dies ist der Tatsache geschuldet, dass bei diesen Modellen keine signifikante Vereinfachung oder Partitionierung möglich ist (siehe Tabelle 5.5).

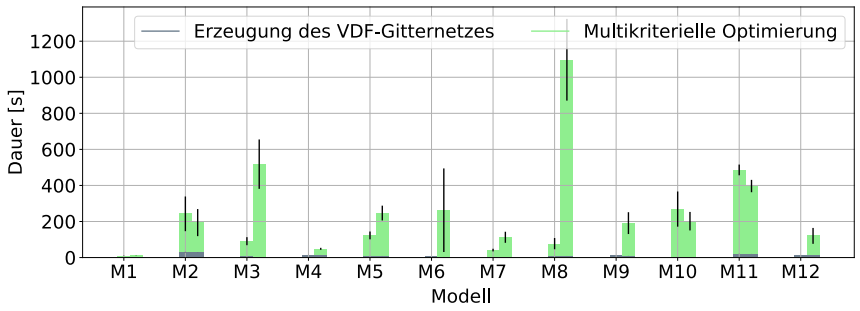


Abbildung 5.17.: Prozessdauer für das *System* (jeweils linker Balken) und die *nur EA*-Variante (jeweils rechter Balken).

5.6.3. Schlüsselergebnisse

Aus der Evaluation geht hervor, dass die entworfenen Strategien zur Problemvereinfachung und -partitionierung in vielen Fällen zu einer verbesserten Ergebnisqualität und verringerten Gesamtlaufzeit führen. Dies ist dann der Fall, wenn der zugrundeliegende Intersektionsgraph geeignet strukturiert ist, also z.B. viele baumartige Teilgraphen enthält, die das *Pruning* begünstigen (z.B. bei M3, M4, M7 und M8). Bei der Ergebnisqualität fällt die unzureichende Größenoptimalität auf, wenn die durch *multikriterielle Optimierung* zu lösenden Teilprobleme verhältnismäßig groß sind (z.B. bei M3 und M8). Dies

ist auf die Beschaffenheit der Zielfunktion zurückzuführen, die eine optimale Wahl der Gewichtungsfaktoren α und β voraussetzt (siehe Gleichung 5.5). Ist diese nicht optimal, führt dies entweder zu Ergebnis-KBs, die schlecht in die Eingabepunktwolke eingepasst sind, oder zu solchen, die eine erhöhte Anzahl redundanter Strukturen enthalten. Da die Wahl der Gewichtungsfaktoren eingabespezifisch ist, muss eine aufwändige Feinjustierung erfolgen.

5.7. Zusammenfassung und Ausblick

In diesem Kapitel wurde das Problem der KB-Synthese aus einer Menge von Eingabeprimitiven und einer Punktwolke im Detail durchleuchtet und bzgl. seiner Komplexität charakterisiert. Darauf aufbauend wurde ein System zur Lösung des Problems eingeführt, das das Konzept des Intersektionsgraphen nutzt, um eine Problempartitionierung und -vereinfachung vorzunehmen und dabei berücksichtigt, dass Eingabedaten messfehlerbehaftet sein können. In dessen Evaluation konnte gezeigt werden, dass sich damit für geeignete Probleminstanzen Vorteile bzgl. Laufzeiteffizienz und Ergebnisqualität ergeben.
Aufbauend auf den Ergebnissen dieses Kapitels ergibt sich eine Vielzahl zukünftiger Forschungsrichtungen.
So wäre eine direkte Unterstützung unbeschränkter Primitive, wie z.B. die von Ebenen, interessant. Dadurch könnte der vorhergehende Schritt der Primitivenextraktion (siehe Kapitel 4) stark vereinfacht werden, da auf eine Zusammenführung von Ebenen zu konvexen Polytopen verzichtet werden könnte. Außerdem ermöglicht die Unterstützung von Ebenen eine Lösung des Problems der separierenden Primitive (siehe Kapitel 2.3.4). Mit separierenden Ebenen lassen sich z.B. mit Zylindern abgerundete Quader darstellen (siehe Abbildung 2.11). Die drei nötigen separierenden Ebenen für einen Zylinder würden dabei durch dessen Mittelpunkt und entlang je einer Koordinatenachse verlaufen. Gleiches gilt für die drei nötigen separierenden Ebenen für eine Kugel. Für die Einführung unbeschränkter Primitive müsste auch untersucht werden, welche Auswirkungen dies auf die Struktur des Intersektionsgraphen und damit auf die eingeführten Vereinfachungs- und Partitionierungsstrategien hat.
Weiterhin interessant wäre die Konzeption zusätzlicher Strategien zur Partitionierung und Vereinfachung anhand des Intersektionsgraphen. Bisher wurden z.B. Zyklen hinsichtlich der Möglichkeit einer vereinfachten Detektion von dominanten Primitiven nicht ausgiebig untersucht. Ebenso könnten andere Muster innerhalb des Intersektionsgraphen ermittelt werden, die eine Vereinfachung oder Partitionierung desselben ermöglichen.
Ein Skalierbarkeitsproblem bzgl. wachsender Modelldimensionen ergibt sich aus der Erzeugung des *VDF-Gitternetzes* (siehe Kapitel 5.5.2). Der Speicherbedarf dieser Datenstruktur steigt kubisch mit der Größe und Auflösung des Gitters an. Größe und Auflösung sind wiederum abhängig von den räumlichen Dimensionen der Eingabepunktwolke bzw. von dem gewünschten geometrischen Detailgrad. Benötigt wird das *VDF-Gitternetz* bei der *Erzeugung der*

Fundamentalproduktpunktwolke (siehe Kapitel 5.5.3) sowie beim *Pruning* zur Ermittlung des Typs eines dominanten Primitivs (siehe Kapitel 5.5.5). Dabei wäre zu untersuchen, ob eine Repräsentation der Oberfläche und nicht des gesamten Volumens für diese Zwecke bereits ausreicht. Damit wäre ein quadratischer statt kubischer Speicherplatzbedarf möglich. Alternativ wäre die Verwendung einer räumlichen Hashtabelle anstatt des hier genutzten regulären Gitternetzes zu untersuchen, was die Speichereffizienz möglicherweise verbessert [127, 187]. Darüber hinaus könnte die Eingabepunktwolke geeignet segmentiert werden, was eine initiale Partitionierung des Problems ermöglichen würde (siehe Kapitel 3.2.3).
Wie aus der Diskussion der Evaluationsergebnisse hervorgeht, sind manche Ergebnis-KBs bezüglich ihrer Größe nicht optimal. Die Größe des resultierenden Baums ist dabei stark abhängig von der Gewichtung derselben innerhalb der *multikriteriellen Optimierung*. Um eine Verringerung der Größe sowie die Verbesserung anderer KB-spezifischer Metriken zu erreichen, wird ein weiterer Verarbeitungsschritt eingeführt. Dieser dient im Allgemeinen der nachträglichen Optimierung von KBs, wie sie Kapitel 6 zum Inhalt hat.

6. Optimierung von Konstruktionsbäumen

Das Einfache ist das höchste Ziel. Nachdem man eine Unmenge Noten und noch mehr Noten gespielt hat, tritt die Einfachheit als krönender Lohn der Kunst hervor.

—Frédéric Chopin

Die von Frédéric Chopin im Eingangszitat erwähnte Einfachheit ist auch im Kontext von KBs ein interessanter Aspekt. Da potentiell unendlich viele KBs zur Beschreibung desselben Festkörpers existieren, stellt sich grundsätzlich die Frage, nach welchen quantifizierbaren Merkmalen sich diese kategorisieren und bewerten lassen. Dabei ist die Einfachheit zentrales Merkmal, da diese großen Einfluss auf die schnelle kognitive Erfassung und die damit verknüpfte intuitive Editierbarkeit hat. Daher ist sie maßgeblich für die Eignung eines KBs in der weiterführenden Verarbeitung. Dies ist insbesondere von hoher Relevanz, da ein großer Vorteil der CSG-Repräsentation ihre leichte Erlernbarkeit und intuitive Anpassbarkeit ist (siehe Kapitel 2.3.1.1). Die folgenden Kapitel beschäftigen sich mit Aspekten der Einfachheit von KBs und wie sich diese sinnvoll quantifizieren und damit optimieren lassen.
Dazu wurde folgende Kapitelstruktur gewählt: Eine Auflistung von mit dem Kapitel assoziierten Vorveröffentlichungen findet sich in Kapitel 6.1. Dieser folgt eine genaue Beschreibung der Zielsetzung sowie der Motivation für die Auseinandersetzung mit dem Problem der KB-Optimierung (siehe Kapitel 6.2). Mit dem Thema verwandte Arbeiten werden in Kapitel 6.3 diskutiert. Darauf aufbauend erfolgt die Beschreibung einer Prozess-Pipeline zur Optimierung von KBs in Kapitel 6.4 und deren Evaluation in Kapitel 6.5. Das Kapitel schließt mit einer Zusammenfassung aller relevanter Inhalte und gibt einen Ausblick auf mögliche anknüpfende Forschungsrichtungen (siehe Kapitel 6.6).
Betrachtet man dieses Kapitel als Teil der Lösung des in dieser Arbeit behandelten Gesamtproblems (siehe Abbildung 3.1), ist das hier vorgestellte System als Lösung für *Problem 3* zu verstehen.

6.1. Vorveröffentlichungen

Die Inhalte dieses Kapitels wurden vom Autor bereits in mehreren Teilen veröffentlicht. So findet sich die Idee zur Formulierung von Problemen aus dem

Bereich der KB-Synthese und -Optimierung als QUBO-Probleme bereits in [48] wieder. Weiterhin wurden die hier beschriebene Prozess-Pipeline sowie die vorgestellten Optimierungsmetriken bereits in [58] publiziert. Die Evaluation wurde für diese Arbeit hingegen vollständig neu auf einer fehlerbereinigten Implementierung durchgeführt, wobei genutzte Datensätze ebenfalls aus [58] stammen. Das im Ausblick (siehe Kapitel 6.6) vorgestellte VR-System zur manuellen Bearbeitung von KBs ist ebenfalls bereits Teil einer Veröffentlichung [57]. Die Abbildungen 6.2, 6.3, 6.4, 6.5 sowie Tabelle 6.3 stammen aus [58], wobei Abbildung 6.3 marginal modifiziert wurde. Aus [57] wurden die beiden Abbildungen 6.15 und 6.16 übernommen.

6.2. Motivation und Zielsetzung

Bei der automatischen Synthese von KBs können Baumstrukturen entstehen, die redundante Teile enthalten und damit nicht optimal bezüglich der zur Repräsentation des Festkörpers tatsächlich benötigten Knoten sind (siehe Kapitel 5.6). Komplexe KBs stellen auch den menschlichen Bearbeiter vor Probleme bei der kognitiven Erfassung der Gesamtstruktur. Aus diesem Grund kann auch die manuelle Erzeugung nicht optimale Baumstrukturen zum Ergebnis haben.
Ungeachtet der Quelle dieser Probleme stellt sich die Frage, welche Konsequenzen sich aus ihnen ergeben. Dabei ist wieder die manuelle Bearbeitbarkeit zentral. So lassen sich redundante Strukturen nicht immer direkt erkennen und werden, gerade in komplexen Modellen, gerne übersehen. Dies führt zu unnötigen Schwierigkeiten, wenn z.B. Teilbäume selektiert und manipuliert werden sollen und nicht klar ersichtlich ist, welche Teile tatsächliche Auswirkungen auf die Form des Festkörpers haben. Ein verwandtes Problem ist die mangelnde Korrespondenz zwischen der Position von Teilbäumen im KB und der räumlichen Lage der durch sie repräsentierten Geometrie. Dies führt, ähnlich wie auftretende Redundanzen, zu erhöhtem kognitivem Aufwand bei der manuellen Manipulation der Baumstruktur und damit zu einem Effizienzverlust innerhalb der ausgeführten Modellierungsaufgabe.
Es stellt sich die Frage, wie das Problem der suboptimalen Baumstrukturen algorithmisch gelöst werden kann. Wichtig dabei ist, dass es sich bei der CSG-Repräsentation um eine nicht einzigartige Repräsentation handelt (siehe Kapitel 2.3.1.1). Es existiert also eine unendlich große Menge an möglichen KBs für ein und denselben Festkörper. Somit lassen sich KBs auswählen, die bezüglich bestimmter Kriterien zu bevorzugen sind. Formal ausgedrückt ergibt sich folgendes zu lösendes Problem: Ausgehend von einem Festkörper mit Punktmenge S sowie einem KB Φ mit Primitiven H für den $|\Phi(H)| = S$ gilt, lautet die Zielsetzung, einen neuen KB Φ_{opt} zu finden, für den neben $|\Phi_{opt}(H)| = S$ auch Optimalität bzgl. folgender Metriken gegeben ist:

- **Größe**: Die Anzahl der Primitiven- und Operationsliterale in Φ_{opt} soll

minimiert werden. Damit werden Redundanzen eliminiert, deren Existenz das manuelle Verändern von KBs erschwert, da potentiell mehr Änderungen für das gewünschte Ergebnis vorgenommen werden müssen.

- **Überschneidungsgrad:** Grundsätzlich können KBs mit Operationen konstruiert werden, deren Operanden Primitive sind, die sich räumlich in komplett anderen Bereichen des Festkörpers befinden (z.B. durch Nutzung des Vereinigungsoperators). Soll nun eine Anpassung der Repräsentation durchgeführt werden, indem z.B. eine Transformation auf einen bestimmten Teilbaum angewandt wird, sorgt dies für eine Änderung des zu repräsentierenden Festkörpers an den unterschiedlichsten, zum Teil räumlich weit auseinanderliegenden Stellen. Um dies zu verhindern, wird in dieser Arbeit der *Überschneidungsgrad* als rekursiv definiertes Maß für die Korrespondenz zwischen Teilbäumen und von ihnen repräsentierter Geometrie eingeführt.

Zusammenfassend sollen Möglichkeiten gefunden werden, KBs durch Verbesserung der obengenannten Metriken zu vereinfachen. Dazu werden verschiedene Verfahren aus dem Bereich der *Konstruktionsbaumoptimierung* (siehe Abbildung 3.1) in eine flexible Prozess-Pipeline eingebettet und vielversprechende Kombinationen miteinander auf aussagekräftigen Testdatensätzen verglichen. Die Lösungsidee wird anhand einer Prozess-Pipeline skizziert. Zentral dabei sind drei Bausteine zur Optimierung von KBs:

- Eine regelbasierte *Redundanzentfernung* für KBs, die in einem iterativen Prozess redundante Substrukturen erkennt und vereinfacht.

- Ein Verfahren zur rekursiven Faktorisierung von Primitiven, die vollständig innerhalb oder vollständig außerhalb des zu repräsentierenden Festkörpers liegen. Die Methode basiert dabei auf der *Dekomposition* (siehe Kapitel 3.2.6.2).

- Eine Auswahl von Verfahren zur Optimierung beider Metriken. Dazu zählt eine Methode, welche EAs) einsetzt und Strategien, die ausnutzen, dass KB-Ausdrücke Boolesche Algebren sind.

Nach der Beleuchtung verwandter Arbeiten im Folgekapitel, wird die in dieser Arbeit eingeführte Prozess-Pipeline beschrieben und sich ergebende Konfigurationsmöglichkeiten evaluiert.

6.3. Verwandte Arbeiten

Im Folgenden werden verwandte Arbeit zum Problem der KB-Optimierung diskutiert.

6.3.1. Größenoptimierung

Bei der Optimierung von KBs konzentrieren sich die meisten verwandten Arbeiten auf die Minimierung der Größe. So wird von Shapiro et al. eine Technik zur Extraktion von KBs aus einer Menge von Primitiven repräsentiert durch Begrenzungsflächenmodelle vorgeschlagen, die auf der rekursiven Faktorisierung von dominanten Primitiven beruht [172]. Diese, sog. *Dekomposition*, erzeugt größenoptimale KBs. Dies gilt jedoch nur für Eingangsmodelle, die sich vollständig faktorisieren lassen (siehe Kapitel 3.2.5.2). Ein ähnliches Schema zur Faktorisierung dominanter Primitive bei der Extraktion eines KBs wird von Buchele et al. vorgeschlagen [26]. Grundsätzlich lassen sich die genannten Dekompositionsverfahren auch für die Größenoptimierung existierender KBs verwenden (siehe Kapitel 3.2.6.2). Aus diesem Grund ist die *Dekomposition* wichtiger Bestandteil der hier vorgestellten Prozess-Pipeline.
Ebenfalls aus [172] stammt die Idee, bewährte Verfahren zur Minimierung von Logikschaltkreisen [22, 119, 130, 143] für das Problem der Minimierung der Größe einzusetzen. Dies ist möglich, da sich auf Festkörpern zusammen mit den regularisierten Mengenoperatoren $\cup^*$, $\cap^*$ und $\setminus^*$ eine Boolesche Algebra definieren lässt (siehe Kapitel 3.2.6.1).

6.3.2. Optimierung weiterer Eigenschaften

Eine Idee zur Optimierung von KBs hinsichtlich Größe und Editierbarkeit wird in [6] beschrieben. Primitive, die räumlich weit voneinander entfernt liegen, werden aus den entsprechenden Fundamentalprodukten entfernt. Dies verringert nicht nur die Größe, sondern auch die manuelle Bearbeitbarkeit des KBs, da lokale Änderungen nur lokale Auswirkungen haben. Das ist besonders von Nutzen, da der dort beschriebene Ansatz auch Ebenen als Primitive unendlicher Ausdehnung unterstützt, was zusätzliche Anforderungen an eine intuitive Bearbeitungstechnik stellt. Der hier beschriebene Ansatz führt diese Idee fort, indem eine Metrik definiert wird, die Teilbäume bzgl. ihrer räumlichen Überschneidung gruppiert.
In [74] wird ein Festkörper, repräsentiert durch einen KB, bezüglich des Gewichts und der Steifigkeit optimiert, indem ein multikriterielles Optimierungsproblem mittels eines EAs gelöst wird. Dabei steht v.a. die physikalische Herstellung des Festkörpermodells im Vordergrund. In [195] wird ebenfalls auf EAs zur Formoptimierung zurückgegriffen, jedoch werden nicht KBs optimiert, sondern konzeptionell verwandte CAD-Spezifikationsbäume. Ziel dieser Verfahren ist dabei also nicht die Verbesserung der Baumstruktur hinsichtlich gewählter Metriken, sondern die Optimierung der durch sie repräsentierten Form bezogen auf das definierte Einsatzgebiet des späteren Echtweltobjekts.
Eine in Gänze andere Zielsetzung verfolgt hingegen [153]. Dort wird beschrieben, wie die direkte Visualisierung von KBs, also ohne den Zwischenschritt der Umwandlung in ein Begrenzungsflächenmodell, für parallele Hardwarearchitekturen beschleunigt werden kann. Dies geschieht, indem der Eingabe-KB

in einen erweiterten Booleschen Ausdruck konvertiert wird. Für diesen existieren dann effiziente, parallel auswertbare Algorithmen, die für einen Punkt zurückgeben, ob dieser im beschriebenen Festkörper liegt oder nicht. Damit wird in [153] ein komplett anderes Optimierungsziel angestrebt, dessen Lösungsstrategie nur für diesen Zweck sinnvoll funktioniert.

6.3.3. Zusammenfassung

Bis auf [6] sind alle bisher diskutierten Methoden zur KB-Optimierung auf die Verringerung der Größe der Baumstruktur ausgelegt. In diesem Kapitel wird hingegen ein Verfahren vorgestellt, das neben der Größe auch andere relevante Metriken optimieren kann. Dabei wird, ähnlich wie in [47, 74] und [52], ein multikriterielles Optimierungsproblem formuliert und mithilfe eines EAs gelöst.

6.4. Konzept eines Systems zur Konstruktionsbaumoptimierung

Die Prozess-Pipeline besteht aus mehreren Schritten (siehe Abbildung 6.1): Zuerst werden redundante Teilbäume aus der Eingabe entfernt (*Redundanzentfernung*, siehe Kapitel 6.4.1). Im Anschluss findet eine Dekomposition des KBs in eine Menge von Fundamentalprodukten sowie eine Restpunktmenge statt (*Dekomposition*, siehe Kapitel 6.4.2). Ist die Restpunktmenge leer, ist der Prozess beendet und der Eingabe-KB größenoptimal reduziert. Ist das nicht der Fall, wird die Reststruktur mit einer Auswahl an Optimierungsmethoden optimiert (*Optimierung des Restfestkörpers*, siehe Kapitel 6.4.3).

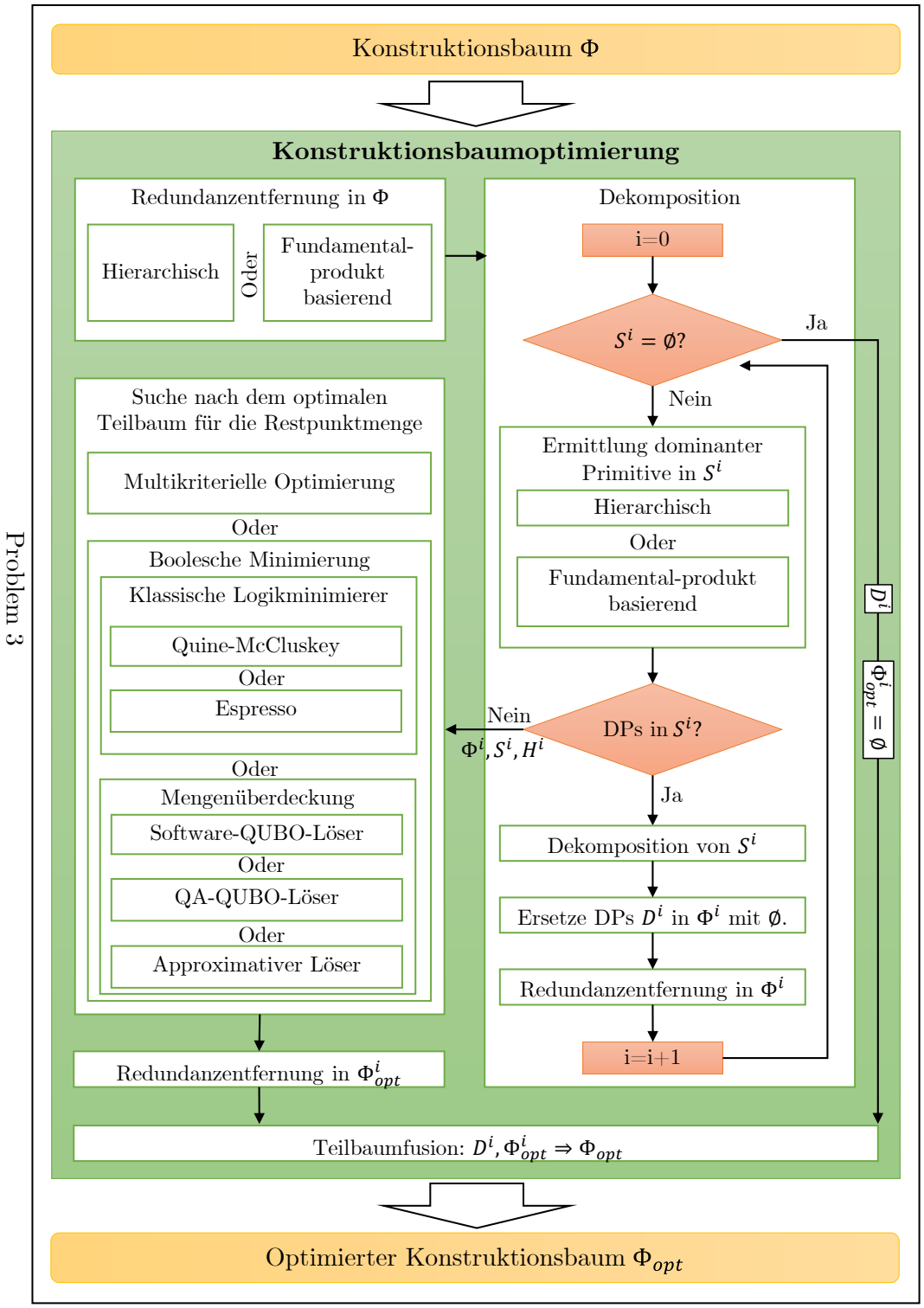


Abbildung 6.1.: Schematische Darstellung der Prozess-Pipeline zur Optimierung von KBs.

6.4.1. Redundanzentfernung

Die *Redundanzentfernung* ist ein iterativer, von [189] inspirierter Optimierungsprozess, der in jedem Durchlauf redundante Teilbäume markiert und entfernt. Dabei macht sich das Verfahren die gegebene geometrische Information über die Primitive in H zu Nutze, um einfache, regelbasierte Muster zu erkennen und zu vereinfachen. Im Folgenden werden diese Regeln aufgeführt.

- Beschreiben die zwei Operanden einer Schnittoperation disjunkte Punktmengen, wird der entsprechende Teilbaum mit der leeren Menge $\emptyset$ ersetzt. Das ist z.B. der Fall, wenn die Operanden zwei sich nicht überdeckende Primitive enthalten. Beschreiben die Operanden dieselbe Punktmenge, wird der entsprechende Teilbaum mit einem der Operanden ersetzt.

- Beschreibt der Subtrahend einer Differenzoperation eine Punktmenge, die disjunkt zur Punktmenge des Minuenden ist, wird der entsprechende Teilbaum durch den Minuenden ersetzt. Beschreiben beide Operanden dieselbe Punktmenge, wird der entsprechende Teilbaum durch die leere Menge $\emptyset$ ersetzt.

- Beschreiben zwei Operanden einer Vereinigungsoperation die exakt gleiche Punktmenge, wird der entsprechende Teilbaum durch einen der Operanden ersetzt.

- Findet sich ein Teilbaum, der das Komplement des Komplements eines Operanden darstellt, wird dieser mit dem Operand des inneren Komplements ersetzt.

Auf diese Weise entstehen bei jedem Durchlauf Teilbäume, die entweder das Literal der leeren oder das der Universalmenge, also $\emptyset$ bzw. W enthalten. Um diese zu entfernen, werden die folgenden Transformationsregeln angewandt:

- Beinhaltet eine Schnittoperation $\emptyset$ als Operanden, wird der entsprechende Teilbaum mit $\emptyset$ ersetzt. Beinhaltet sie W als Operanden, wird sie durch den jeweils anderen Operanden ersetzt.

- Beinhaltet eine Vereinigungsoperation $\emptyset$ als Operanden, wird der entsprechende Teilbaum mit dem jeweils anderen Operanden ersetzt. Beinhaltet sie W als Operanden, wird der entsprechende Teilbaum mit W ersetzt.

- Hat eine Differenzoperation $\emptyset$ als Subtrahenden, wird sie durch den Minuenden ersetzt. Enthält sie $\emptyset$ als Minuenden, wird sie durch $\emptyset$ ersetzt. Ist der Minuend W, wird der entsprechende Teilbaum durch das Komplement des anderen Operanden ersetzt. Ist der Subtrahend W, wird der entsprechende Teilbaum durch $\emptyset$ ersetzt.

Die Prozedur durchläuft den KB iterativ und wird solange wiederholt, bis der resultierende KB weder $\emptyset$- noch W-Literale enthält.

Zentral für das Verfahren ist die robuste und laufzeiteffiziente Ermittlung, ob zwei KBs die gleiche Punktmenge beschreiben sowie der Test, ob ein Teilbaum die leere Menge $\emptyset$ beschreibt. Da der Zusammenhang

$$|\Phi| = |\Psi| \Longleftrightarrow |\Phi| \cap^* (\setminus^* |\Psi|) = \emptyset \wedge |\Psi| \cap^* (\setminus^* |\Phi|) = \emptyset \tag{6.1}$$

gilt, genügt es hierbei, ein Verfahren zur Erkennung leerer Mengen zu entwickeln, um beide Probleme zu lösen. Dabei gilt für die Ermittlung der Gleichheit, dass zwei Tests auf die leere Menge ($|\Phi| \cap^* (\setminus^* |\Psi|) = \emptyset$ und $|\Psi| \cap^* (\setminus^* |\Phi|) = \emptyset$) ausgeführt werden müssen.
Für die Erkennung leerer Mengen wird in dieser Arbeit auf samplingbasierte Methoden zurückgegriffen. Die Idee ist dabei, dass für Punkte der Domäne des durch Φ beschriebenen Festkörpers (also für den achsenausgerichteten Quader, der den Festkörper vollständig umschließt) die VDF des zu überprüfenden KBs ausgewertet wird. Ist diese an jedem abgetasteten Punkt positiv, beschreibt der KB ausschließlich Punkte, die außerhalb liegen und damit die leere Punktmenge $\emptyset$. Für das Sampling werden dazu zwei Methoden vorgeschlagen, die in den beiden folgenden Unterkapiteln beschrieben werden.

6.4.1.1. Hierarchisches Sampling

Bei dieser Methode wird die Domäne des durch Φ beschriebenen Festkörpers festgelegt durch einen ihn umfassenden Quader der Größe (w_0, h_0, r_0) rekursiv in immer kleinere Quader unterteilt. Dieses Vorgehen ähnelt dabei der Einordnung von Punkten in der Octree-Repräsentation (siehe Kapitel 2.3.2.2). Für jede Rekursionsebene wird untersucht, ob die VDF des zu untersuchenden KBs für die Mittelpunkte der Quader positiv oder negativ ist. Ist sie negativ, wird abgebrochen, da dann sicher ist, dass ein Punkt innerhalb des durch den KB beschriebenen Festkörpers existiert und dieser somit nicht die leere Menge beschreibt. Es findet also ein frühes Stoppen statt, welches die Laufzeiteffizienz im Durchschnitt verbessert. Ist die VDF für alle Punkte positiv, werden die Quader der aktuellen Rekursionstiefe in acht Unterquader zerteilt und wieder ein entsprechender VDF-Test durchgeführt. Um eine endlose Rekursion bei leeren Mengen zu vermeiden, wird eine minimale Quaderbreite, -länge und -höhe $(w_{min}, h_{min}, r_{min})$ als benutzerdefinierte untere Schranke für die Quadergröße vorgegeben. Wird sie unterschritten, terminiert das Verfahren und gibt zurück, dass es sich bei dem untersuchten KB um eine Beschreibung der leeren Menge handelt (siehe Abbildung 6.2). Um die Laufzeiteffizienz zusätzlich zu steigern, wird ausgenutzt, dass bei der späteren Verwendung der Methode oft identische KBs geprüft werden. Damit ist eine Beschleunigung mithilfe einer Nachschlagetabelle für bereits geprüfte KBs lohnenswert.

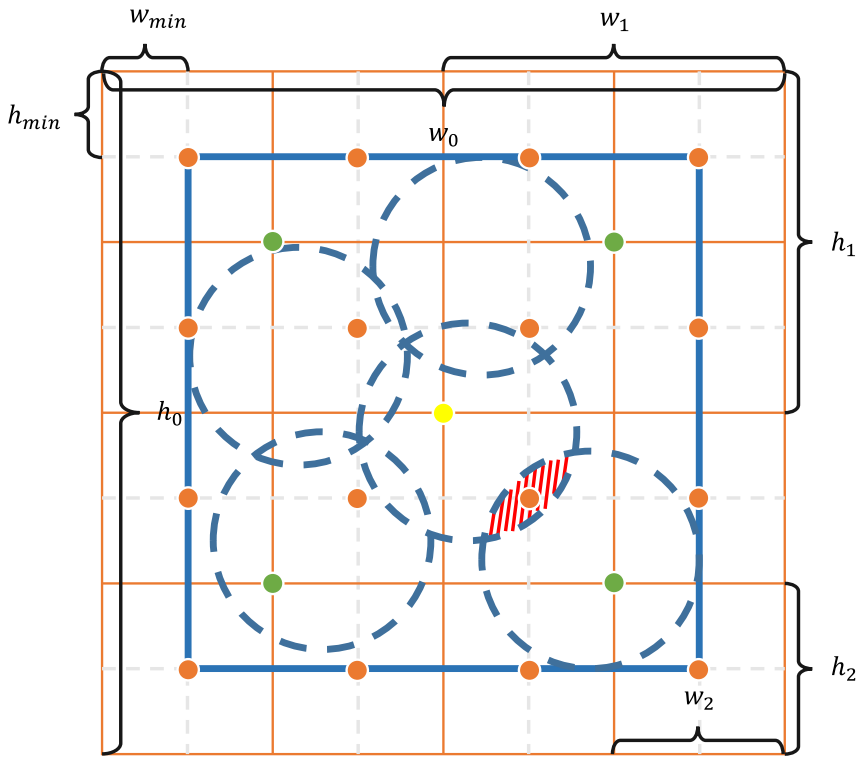


Abbildung 6.2.: Beispiel für *hierarchisches Sampling* mit rekursiv geteilten Quaderdimensionen (w_0, h_0), (w_1, h_1) und (w_2, h_2) (hier zur Vereinfachung nur in zwei Dimensionen), Primitiven mit blau gestrichelten Rändern und Punktmenge des zu prüfenden KBs in Rot schraffiert. Im ersten Schritt wird der Mittelpunkt (gelb) des allumfassenden Quaders (blauer Rand) der Größe (w_0, h_0) geprüft. Es folgen die beiden anderen Rekursionstiefen 1 (grün) und 2 (rot). Die Quaderauflösung bei Rekursionstiefe 2 ist ausreichend, um festzustellen, dass es sich bei dem zu prüfenden KB nicht um eine Beschreibung der leeren Menge handelt. Im Gesamten wäre noch eine weitere Rekursionstiefe mit Quadergröße (w_{min}, h_{min}) möglich, in diesem Fall aber nicht nötig.

6.4.1.2. Fundamentalproduktbasierendes Sampling

Bei dieser Methode werden die Fundamentalprodukte (siehe Kapitel 3.2.5.1) des Eingabe-KBs extrahiert und für jedes geprüft, ob die VDF für einen darin liegenden Punkt positiv ist. Ist das für alle der Fall, handelt es sich bei dem zu prüfenden KB um eine Beschreibung der leeren Menge. Da dabei potentiell weniger Punkte als bei der Anwendung des *hierarchischen Samplings* geprüft werden müssen, ist zu evaluieren, ob sich dieser Umstand in einem besseren Laufzeitverhalten widerspiegelt (siehe Kapitel 6.5).
Bei diesem Verfahren zentral ist die Bestimmung der Fundamentalprodukte, was wieder über ein einmaliges Sampling des den Festkörper umschließenden Quaders passiert. Für jeden Punkt wird dazu die VDF aller Primitive in H ausgewertet. Ist sie für ein Primitiv $h \in H$ positiv, ist der Punkt außerhalb des Primitivs. Somit geht das Komplement des Primitivs in das Fundamentalprodukt ein. Ist sie negativ, liegt der Punkt also innerhalb von h, geht das Primitiv direkt in das Fundamentalprodukt ein (siehe Abbildung 6.3). Als Resultat ergibt sich eine Menge von Fundamentalprodukten, deren Vereinigung exakt die Punktmenge S beschreiben, also die Punktmenge, die auch durch Φ beschrieben wird (siehe Kapitel 3.2.5.1).

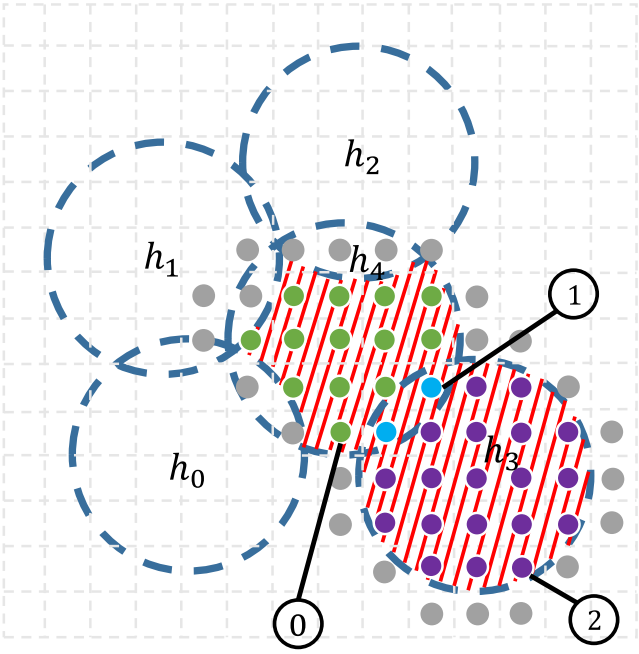


Abbildung 6.3.: Beispiel für die Gewinnung von Fundamentalprodukten durch Sampling. Mit rot schraffierter Punktmenge S und Primitivenmenge $H = \{h_0, h_1, h_2, h_3, h_4\}$. Resultierende Fundamentalprodukte: $\overline{h_0} \cdot \overline{h_1 \cdot h_2 \cdot h_3} \cdot h_4$ (0, Punkte in Grün), $\overline{h_0} \cdot \overline{h_1 \cdot h_2} \cdot h_3 \cdot h_4$ (1, Punkte in Hellblau) und $\overline{h_0} \cdot \overline{h_1 \cdot h_2} \cdot h_3 \cdot \overline{h_4}$ (2, Punkte in Lila). Graue Punkte liegen außerhalb von S.

6.4.2. Dekomposition

Die in Kapitel 3.2.5.2 eingeführte *Dekomposition* ist ein rekursives Verfahren, um die Punktmenge S eines Festkörpers mit gegebenen Primitiven H in eine Kombination aus dominanten Primitiven und einer potentiell leeren Restpunktmenge bestehend aus nicht faktorisierbaren Primitiven zu zerlegen. Sie eignet sich dabei auch zur Optimierung der Größe eines KBs, wie es hier Ziel ist (siehe Kapitel 3.2.6.2). Folgende Erläuterungen stützen sich v.a. auf Gleichung 3.5 aus Kapitel 3.2.5.2 und verlaufen entlang der Prozessschritte, wie sie in Abbildung 6.1 dargestellt sind.
Der Prozess startet mit Iteration $i = 0$. Damit gilt für die aktuelle Restpunktmenge $S^i = S$, für die bisher nicht faktorisierten Primitive $H^i = H$ sowie für den aktuellen Eingabe-KB $\Phi^i = \Phi$. Die Sequenz dominanter Primitive D^i ist dabei initial leer.
Ist die aktuelle Restpunktmenge S^i leer, ist die KB-Optimierung abgeschlossen und D^i wird zusammen mit dem leeren optimalen Rest-KB Φ^i_{opt} zurückgegeben. Ist dies nicht der Fall, wird geprüft, ob Primitive aus H^i dominante Primitive in S^i sind (siehe Abbildung 6.1, *Ermittlung dominanter Primitive in* S^i). Wurden dominante Primitive gefunden, werden diese aus H^i entfernt, um H^{i+1} zu erhalten, und zusammen mit D^i zur Sequenz D^{i+1} verknüpft. Dabei werden an den Anfang von D^i die neu gefundenen dominanten Primitive angehängt: Zuerst die außenliegenden, dann die innenliegenden. Zudem werden die Literale der gefundenen dominanten Primitive im Eingabe-KB Φ^i mit dem $\emptyset$-Literal ersetzt und per *Redundanzentfernung* entfernt, was im neuen Eingabe-KB Φ^{i+1} resultiert. Damit ist die neue Restpunktmenge $S^{i+1} = |\Phi^{i+1}(H^{i+1})|$ ebenfalls festgelegt. Im letzten Schritt dieser Iteration wird die Iterationsvariable i inkrementiert und damit eine neue Iteration mit der Überprüfung der aktuellen Restpunktmenge S^i auf Leerheit begonnen (siehe Abbildung 6.1, $S^i = \emptyset ?$). Finden sich hingegen keine dominanten Primitive in S^i, wird ein optimaler KB für S^i gesucht (Φ^i_{opt}), wobei $S^i = |\Phi^i(H^i)| = |\Phi^i_{opt}(H^i)|$ gelten muss (siehe Abbildung 6.1, *Suche nach dem optimalen Teilbaum für die Restpunktmenge*). Verfahren, die dies bewerkstelligen, werden in Kapitel 6.4.3 erläutert.
Ein einfaches Beispiel für die *Dekomposition* findet sich in Abbildung 6.3. Dort sind h_0, h_1 h_2 außenliegende dominante Primitive in S^0, wohingegen h_3 ein innenliegendes dominantes Primitiv ist. Damit ist $D^0 = (h_3, h_2, h_1, h_0)$. In der nächsten Iteration ist h_4 in S^1 ein innenliegendes dominantes Primitiv, was in der finalen Sequenz dominanter Primitive $D^1 = (h_4, h_3, h_2, h_1, h_0)$ resultiert.
Zentral für die *Dekomposition* ist die Identifikation dominanter Primitive. Ist ein Primitiv dominant, liegen alle Punkte innerhalb des Primitivs auch innerhalb des von Φ^i beschriebenen Festkörpers mit der Punktmenge S^i oder komplett außerhalb desselben. Um dies zu prüfen, wird eine auf Sampling beruhende Strategie verfolgt. Dazu wird für jedes Primitiv aus H^i die VDF des Primitivs innerhalb des Quaders abgetastet, welcher das Primitiv vollständig umfasst. Für die Menge der so abgetasteten Punkte mit negativem VDF-Wert muss gelten, dass der ermittelte VDF-Wert des zu optimierenden KBs Φ^i für

alle Punkte der Menge entweder negativ oder positiv ist. Ist das der Fall, handelt es sich bei dem Primitiv um ein dominantes. Als Abtastschema können dabei wieder *hierarchisches Sampling* (siehe Kapitel 6.4.1.1) oder *fundamentalproduktbasierendes Sampling* (siehe Kapitel 6.4.1.2) eingesetzt werden. Bei Ersterem ist das Auftreten von innerhalb des zu prüfenden Primitivs liegenden Punkten, deren VDF-Werte von Φ^i unterschiedliche Vorzeichen haben das Abbruchkriterium für den frühen Stopp der Rekursion.

6.4.3. Suche nach dem optimalen Teilbaum für die Restpunktmenge

Kann bei der *Dekomposition* der Eingabe-KB Φ nicht vollständig in dominante Primitive faktorisiert werden, muss für die Punktmenge S^i, den aktuellen Eingabe-KB Φ^i sowie die Primitivenmenge H^i der aktuellen und letzten Dekompositionsiteration i ein optimaler Teilbaum Φ^i_{opt} gefunden werden. Zu diesem Zweck werden im Folgenden zwei Methodenfamilien vorgestellt, die als Teil der Prozess-Pipeline frei auswählbar sind.

6.4.3.1. Boolesche Minimierung

Bei Methoden diesen Typs werden zuerst alle Fundamentalprodukte in S^i ermittelt und mit $\cup^*$ verknüpft, um die KDNF von Φ^i, Φ^i_{kdnf}, zu erhalten (siehe Gleichung 5.1, Kapitel 5.4.1). Dies wird mit einem Sampling erreicht, wie es Kapitel 6.4.1.2 beschreibt und für das Beispiel aus Abbildung 6.3 zum KB $\Phi^0_{kdnf} = \overline{h_0} \cdot \overline{h_1 \cdot h_2 \cdot h_3} \cdot h_4 \cup^* \overline{h_0} \cdot \overline{h_1 \cdot h_2} \cdot h_3 \cdot h_4 \cup^* \overline{h_0} \cdot \overline{h_1 \cdot h_2} \cdot h_3 \cdot \overline{h_4}$ führt.

Klassische Logikminimierer. Auf Festkörpern lässt sich zusammen mit den regularisierten Mengenoperatoren $\cup^*$, $\cap^*$ und $\setminus^*$ eine Boolesche Algebra definieren (siehe Kapitel 3.2.6.1). Damit stehen die Werkzeuge zur Minimierung von DNF-Formen Boolescher Ausdrücke zur Verfügung. In dieser Arbeit wurden dazu zwei Verfahren zur Minimierung von Φ^i_{kdnf} betrachtet: Der *Espresso*-Logikminimierer [22] und die *Quine-McCluskey*-Methode [119, 143].

Mengenüberdeckung. Wie beim Algorithmus von *Quine* und *McCluskey* [119, 143], wird bei diesem Verfahren ausgenutzt, dass sich jeder Boolesche Ausdruck auch als Vereinigung einer Teilmenge seiner Primimplikanten beschreiben lässt und dies dann auch die minimale DNF-Form des Ausdrucks ist [172]. Die hier beschriebene Strategie folgt dabei dem gleichen Muster.
Zur besseren Übersicht wird bei den folgend neu eingeführten Bezeichnern auf die Sichtbarmachung des Indexes i verzichtet. Basierend auf den n Fundamentalprodukten $F = \{f_1, ..., f_n\}$ von Φ^i_{kdnf} wird eine Suche nach Primimplikanten durchgeführt. Dabei ist ein Fundamentalprodukt $f \in F$ prim, wenn $|f| \subset S^i$ gilt und gleichzeitig die Entfernung eines einzelnen

Primitivenliterals von f zu $|f| \not\subset S^i$ führt. Für das Beispiel in Abbildung 6.3 (ohne Berücksichtigung der *Dekomposition*) würde die Menge der Primimplikanten F_{prim} die Primimplikanten h_3 und $\overline{h_0 \cdot h_1 \cdot h_2} \cdot h_4$ enthalten. Ein einfacher Algorithmus, der alle Primimplikanten zurückgibt, prüft der obigen Definition folgend für jedes Fundamentalprodukt $f \in F$, welche Primitivenliterale entfernt werden können, ohne dass $|f| \not\subset S^i$ gilt. Das so ermittelte, reduzierte Fundamentalprodukt ist ein Primimplikant und wird F_{prim} hinzugefügt, sofern es nicht schon enthalten ist. Die Prüfung $|f| \notin S^i$ erfolgt durch eine samplingbasierte Leermengenprüfung $|f| \cap^* \overline{|\Phi^i|} = \emptyset$.
Um nun die relevanten Primimplikanten aus F_{prim} auszuwählen, wird eine Formulierung des Problems als Mengenüberdeckungsproblem gewählt: Die zu überdeckende Menge U enthält die Indizes aller Fundamentalprodukte in F (für das Beispiel in Abbildung 6.3: $U = \{0, 1, 2\}$).
Jeder Primimplikant aus F_{prim} deckt eine Teilmenge der Menge U ab, was in einer Menge von Indexteilmengen V mit $|V| = |F_{prim}|$ und Elementen $V_k \subset U, 1 \leq k \leq |F_{prim}|$ resultiert (für das Beispiel in Abbildung 6.3: $V = \{h_3 : \{1, 2\}, \overline{h_0 \cdot h_1 \cdot h_2} \cdot h_4 : \{0, 1\}\}$). Nun muss die kleinste Teilmenge aus V gefunden werden, sodass U vollständig abgedeckt wird (im Beispiel: $\{\{1, 2\}, \{0, 1\}\}$). Dieses Mengenüberdeckungsproblem ist $\mathcal{NP}$-*vollständig* [97] und wird hier als QUBO-Problem

$$H_A = A \sum_{\alpha=1}^{|U|} \left(1 - \sum_{m=1}^{|V|} \theta_{\alpha,m}\right)^2 + A \sum_{\alpha=1}^{|U|} \left(\sum_{m=1}^{|V|} m\theta_{\alpha,m} - \sum_{k:\alpha \in V_k} \theta_k\right)^2$$

und

$$H_B = B \sum_{k=1}^{|V|} \theta_k, \tag{6.2}$$

formuliert [115]. Dabei sind A und B zu wählende, positive Konstanten (mit $0 < B < A$). $\theta_k \in \{0, 1\}$ ist eine Binärvariable, welche 1 ist, wenn die Teilmenge V_k Teil der ausgewählten Teilmengen ist. Bei $\theta_{\alpha,m}$ handelt es sich ebenfalls um eine Binärvariable, die 1 ist, wenn die Anzahl ausgewählter Teilmengen V_k, die das Fundamentalprodukt mit Index α enthalten, gleich m ist. Betrachtet man den ersten Term in H_A, ist es sein Zweck dafür zu sorgen, dass für jedes Fundamentalprodukt mit Index α genau ein $\theta_{\alpha,m}$ 1 ist. Der zweite Term in H_A stellt sicher, dass die Anzahl der Vorkommen von Fundamentalprodukt α auch mit der Anzahl der aus V ausgewählten Teilmengen, die Fundamentalprodukt α als Element enthalten, übereinstimmt. Werden die in H_A formulierten Nebenbedingungen nicht erfüllt, ist das Ergebnis von Null verschieden. Schließlich sorgt H_B dafür, dass die Anzahl selektierter Teilmengen möglichst minimal ist. Aus Gleichung 6.2 geht hervor, dass die Anzahl benötigter Binärvariablen sich auf $N = |U| \cdot |V| + |V|$ beläuft. Damit ergibt sich das zu lösende Optimierungs-

problem aus

$$\underset{\Theta \in \{0,1\}^N}{\operatorname{argmin}} H_A + H_B. \tag{6.3}$$

Für die Lösung kann nun QA-Hardware (siehe Kapitel 2.4.3.3 und Abbildung 6.1, *QA-QUBO-Löser*) verwendet werden oder eine Software-Emulation derselben [18] (siehe Abbildung 6.1, *Software-QUBO-Löser*). In dieser Arbeit wurde für den Vergleich mit anderen KB-Optimierungsmethoden auf eine Software-Emulation zurückgegriffen, die auf der Tabusuche, einer probabilistischen Metaheuristik, basiert [18, 66]. Da nur diese für jeden Testdatensatz Ergebnisse zurückliefert, ist dies sinnvoll. Zusätzlich wurde das Mengenüberdeckungsproblem mit QA-Hardware separat evaluiert, um die Eignung des Verfahrens auch für diese zu demonstrieren. Zum Vergleich mit einer klassischen Methode, wird dazu auch ein approximatives Verfahren [31] herangezogen (*Approximativer Löser*, Abbildung 6.1).

6.4.3.2. Multikriterielle Optimierung

Für die *Multikriterielle Optimierung* als Strategie zur Verbesserung von KBs wird ein kombinatorisches Optimierungsproblem mit der Menge an möglichen Baumkombinationen, die mit den unterstützten regulären Mengenoperationen und den Primitiven H^i möglich sind, als Suchraum formuliert (siehe Kapitel 3.2.6.3). Dabei sollen drei Optimierungsziele erreicht werden:

- **Geometrische Passgenauigkeit**: Die durch die Lösung repräsentierte Punktmenge muss identisch mit $S^i = |\Phi^i(H^i)|$ sein.
- **Größe**: Die Lösung muss die kleinstmögliche Menge an Operationen und Primitiven enthalten, die ausreicht, um die *geometrische Passgenauigkeit* zu garantieren.
- **Überschneidungsgrad**: Die rekursiv definierte Metrik, die die räumliche Überschneidung zwischen Operanden regulärer Mengenoperatoren quantifiziert, soll für die Lösung einen möglichst großen Wert annehmen.

Gelöst wird das Optimierungsproblem mit einem EA für alle drei Optimierungsziele gleichzeitig. Im Vergleich zu DNF-basierten Optimierern hat diese Problemformulierung zwei große Vorteile: Zum einen ist es möglich, größenoptimale KBs zu finden, was im Allgemeinen mit DNF-Optimierern nicht möglich ist [172]. Zum anderen ist die Flexibilität der Formulierung, die es erlaubt, zusätzliche Optimierungsziele neben der Größenoptimalität miteinzubeziehen, ein großer Vorteil. Als größter Nachteil hingegen ist die Laufzeit von Metaheuristiken zur Lösung kombinatorischer Optimierungsprobleme zu nennen, die generell signifikant höher ist als die von spezialisierten DNF-Optimierern. Im Folgenden werden die Schritte des EAs beschrieben, wobei das Prozessdiagramm aus Abbildung 2.13 als Vorlage dient.

Start. Zu Beginn wird die Startpopulation (siehe Abbildung 2.13, *Zufällige Population* 0) mit 1 bis n_{pop} KBs befüllt, die entweder Kopien von Φ^i darstellen oder aber zufällig erzeugt wurden und dabei die gleiche Größe wie Φ^i aufweisen. Für die Ermittlung der geometrischen Passgenauigkeit im Schritt *Bewertung* müssen die Punktmengen O_{in} und O_{au} erzeugt werden. Erstere enthält für jedes Fundamentalprodukt, das innerhalb von $|\Phi^i|$ liegt einen Punkt, der auch innerhalb des Fundamentalprodukts liegt. Letztere enthält je einen Punkt für jedes Fundamentalprodukt, das außerhalb von $|\Phi^i|$ liegt. Ermittelt werden beide Punktmengen über eine Abtaststrategie (siehe Kapitel 6.4.3.1).

Bewertung. Die zu maximierende Zielfunktion, welche ein einzelnes Individuum Φ_c der aktuellen Population bewertet, ergibt sich aus

$$F_{\Phi^i,O_{in},O_{au}}(\Phi_c) = \alpha \cdot G_{O_{in},O_{au}}(\Phi_c) + \beta \cdot U(\Phi_c) + \gamma \cdot \#_{\Phi^i}(\Phi_c), \tag{6.4}$$

wobei $G_{O_{in},O_{au}}(\cdot)$ die geometrische Passgenauigkeit, $U(\cdot)$ den Überschneidungsgrad und $\#_{\Phi^i}(\cdot)$ die relative Größe des Ausdrucks quantifiziert. α, β und γ sind dabei benutzerdefinierte Gewichte. Die geometrische Passgenauigkeit zählt, wie viele Punkte aus O_{in} in $|\Phi_c|$ bzw. wie viele Punkte aus O_{au} nicht in $|\Phi_c|$ liegen und besteht folglich aus zwei Termen $G_{O_{in},O_{au}}(\Phi_c) = G_{O_{in}}(\Phi_c) + G_{O_{au}}(\Phi_c)$ mit

$$G_{O_{in}}(\Phi_c) = \frac{1}{|O_{in}|} \sum_{o \in O_{in}} \begin{cases} 1, & \text{falls } |F_{\Phi_c}(o)| \leq \epsilon \\ 0, & \text{sonst} \end{cases} \tag{6.5}$$

und

$$G_{O_{au}}(\Phi_c) = \frac{1}{|O_{au}|} \sum_{o \in O_{au}} \begin{cases} 1, & \text{falls } |F_{\Phi_c}(o)| > \epsilon \\ 0, & \text{sonst} \end{cases}, \tag{6.6}$$

wobei $F_{\Phi_c}(\cdot)$ die VDF des Lösungskandidaten und ϵ ein benutzerdefinierter Parameter ist. Der Term zur Quantifizierung des Überschneidungsgrads ergibt sich aus

$$U(\Phi_c) = \frac{P_{rek}(\Phi_c)}{\#(\Phi_c)}, \tag{6.7}$$

wobei $\#(\cdot)$ die Knoten eines KBs zählt und

$$P_{rek}(\Phi) = \begin{cases} 1, \text{ falls } \Phi \text{ ein Blattknoten ist} \\ P_{rek}(\Phi_1) + P_{rek}(\Phi_2) + \Delta(\Phi_1, \Phi_2) \text{ sonst} \end{cases} \tag{6.8}$$

mit den Kindknoten von Φ, Φ_1 und Φ_2, den Überschneidungsgrad rekursiv misst und

$$\Delta(\Phi_1, \Phi_2) = \begin{cases} 1 \text{ falls } |\Phi_1| \cap^* |\Phi_2| \neq \emptyset \\ 0 \text{ sonst} \end{cases} \tag{6.9}$$

die Überschneidung zweier KBs Φ_1 und Φ_2 detektiert. Zur Quantifizierung der Größe des Lösungskandidats Φ_c wird

$$\#_{\Phi^i}(\Phi_c) = \frac{\#(\Phi^i) - \#(\Phi_c) - \#^{min}_{\Phi^i}}{\#^{max}_{\Phi^i} - \#^{min}_{\Phi^i}} \tag{6.10}$$

verwendet, wobei $\#^{min}_{\Phi^i}$ und $\#^{max}_{\Phi^i}$ die minimale und maximale Baumgröße aller Bäume in der aktuellen Population darstellen.

Die Schritte *Elternauswahl*, *Rekombination*, *Mutation*, *Gesamtauswahl* und *Abbruch* sind identisch mit den in Kapitel 5.5.7 beschriebenen.

Selektion des besten Individuums. Der beste KB der Population der letzten Iteration bezogen auf die Zielfunktion lässt sich ermitteln und als Ergebnis selektieren. Jedoch ist es in diesem Fall hilfreicher, eine Menge von Lösungen auszuwählen, die sog. Pareto-Menge oder Pareto-Front [43]. In dem hier betrachteten Fall filtert man dazu zuerst Lösungskandidaten aus, deren Wert für die geometrische Passgenauigkeit $G_{O_{in},O_{au}}$ nicht exakt 1 ist, da genau die gleiche Punktmenge beschrieben werden soll, die auch Φ^i beschreibt. Dann wird für jeden verbliebenen Lösungskandidaten Φ_c das Tupel $(U(\Phi_c), \#_{\Phi^i}(\Phi_c))$ betrachtet. Ein Kandidat wird nach zwei Kriterien für die Ergebnismenge ausgewählt: 1) Es gibt keinen anderen Kandidaten, dessen beiden Tupelwerte mindestens so groß sind wie die des Kandidaten und 2) kein einzelner Wert eines anderen Kandidaten übersteigt den des ausgewählten.

Der hier beschriebene EA zur *Suche nach dem optimalen Teilbaum für die Restpunktmenge* wird durch die in Tabelle 6.1 dargestellten benutzerdefinierten Parameter konfiguriert.

Parameter	**Beschreibung**
n_{pop}	Größe der Population
n_{var}	Anzahl der Wiederholungen der Variationsoperatoren
n_{iter}	Max. Anzahl an Optimierungsiterationen
m_{iter}	Max. Anzahl an Iterationen ohne Verbesserung der Zielfunktion
α,β, γ	Gewichtungsfaktoren der Zielfunktion (Gleichung 6.4)
ϵ	Max. tolerierbarer Abstand in den Gleichungen 6.5 und 6.6
p_{mut}	Mutationswahrscheinlichkeit
p_{rek}	Rekombinationswahrscheinlichkeit
p_{neu}	Wahrscheinlichkeiten der Mutationsvariante *Neu*

Tabelle 6.1.: Benutzerdefinierte Parameter für die *multikriterielle Optimierung* zur *Suche nach dem optimalen Teilbaum für die Restpunktmenge.*

6.4.4. Teilbaumfusion

Aus den Schritten *Dekomposition* und *Suche nach dem optimalen Teilbaum für die Restpunktmenge* (siehe Abbildung 6.1) stammt für einen KB Φ und einer Menge von Primitiven H mit n dominanten Primitiven aus der Sequenz $D^i = \{d_1, ..., d_n\}$ der optimierte KB

$$\Phi_{opt} = ((...(\Phi^i_{opt} \oplus d_1) \oplus ...) \oplus d_{n-1}) \oplus d_n, \tag{6.11}$$

wobei $\oplus$ entweder $\cup^*$ oder $-^*$ ist, je nach Typ des folgenden dominanten Primitivs. Sind alle Primitive dominant ist Φ^i_{opt} leer. Formal korrekt betrachtet sind die Elemente in D^i sowie die Operatoren, für die $\oplus$ als Platzhalter fungiert, Elemente der Grammatik für KBs (siehe Abbildung 2.3).
Die *Dekomposition* optimiert ausschließlich die Größe eines KBs. Um auch den Überschneidungsgrad optimieren zu können, wird ein Sortiermechanismus genutzt. Dieser sorgt dafür, dass in der Operationskette aufeinander folgende Primitive auch möglichst räumlich zusammenhängen.

6.5. Evaluation

Im Folgenden wird die eingeführte Prozess-Pipeline zur Optimierung von KBs experimentell untersucht. Dabei steht neben den Evaluationsergebnissen (siehe Kapitel 6.5.1 und 6.5.2) auch die Erstellung des Evaluationsdatensatzes im Fokus. Alle Experimente wurden auf einem Laptop mit Vierkernprozessor (2,6GHz) und 16GB RAM durchgeführt. Auch die Berechnung der *Mengenüberdeckung* (siehe Kapitel 6.4.3.1) wurde auf diesem Gerät und nicht auf dedizierter QA-Hardware durchgeführt, da nur so ein vergleichbares Ergebnis für jedes Testmodell erzielt werden konnte. Eine Ausnahme bildet das Experiment in Kapitel 6.5.1, für das der Quanten-Annealer *D-Wave Advantage* der Firma *D-Wave Systems* mit 5000 physikalischen Qubits zum Einsatz kam. Die genutzten Parameter für den EA finden sich in Tabelle 6.2. Zudem nutzten alle Sampling-Methoden eine Schrittweite von 0,1.

Param.	n_{pop}	n_{var}	n_{iter}	m_{iter}	α,β, γ	ϵ	p_{mut}	p_{rek}	p_{neu}
Wert	150	1	1000	500	50, 1, 10	0,05	0,3	0,4	0,5

Tabelle 6.2.: Gewählte Parameter für die *multikriterielle Optimierung*.

In einem ersten Schritt wurden insgesamt 11 CAD-Modelle in CSG-Repräsentation manuell erstellt (siehe Abbildungen 6.4 und 6.5). Um verschiedene Aspekte suboptimaler Modellierung zu simulieren, fügt ein Generator den Eingabemodellen zufallsbasiert Redundanzen hinzu:

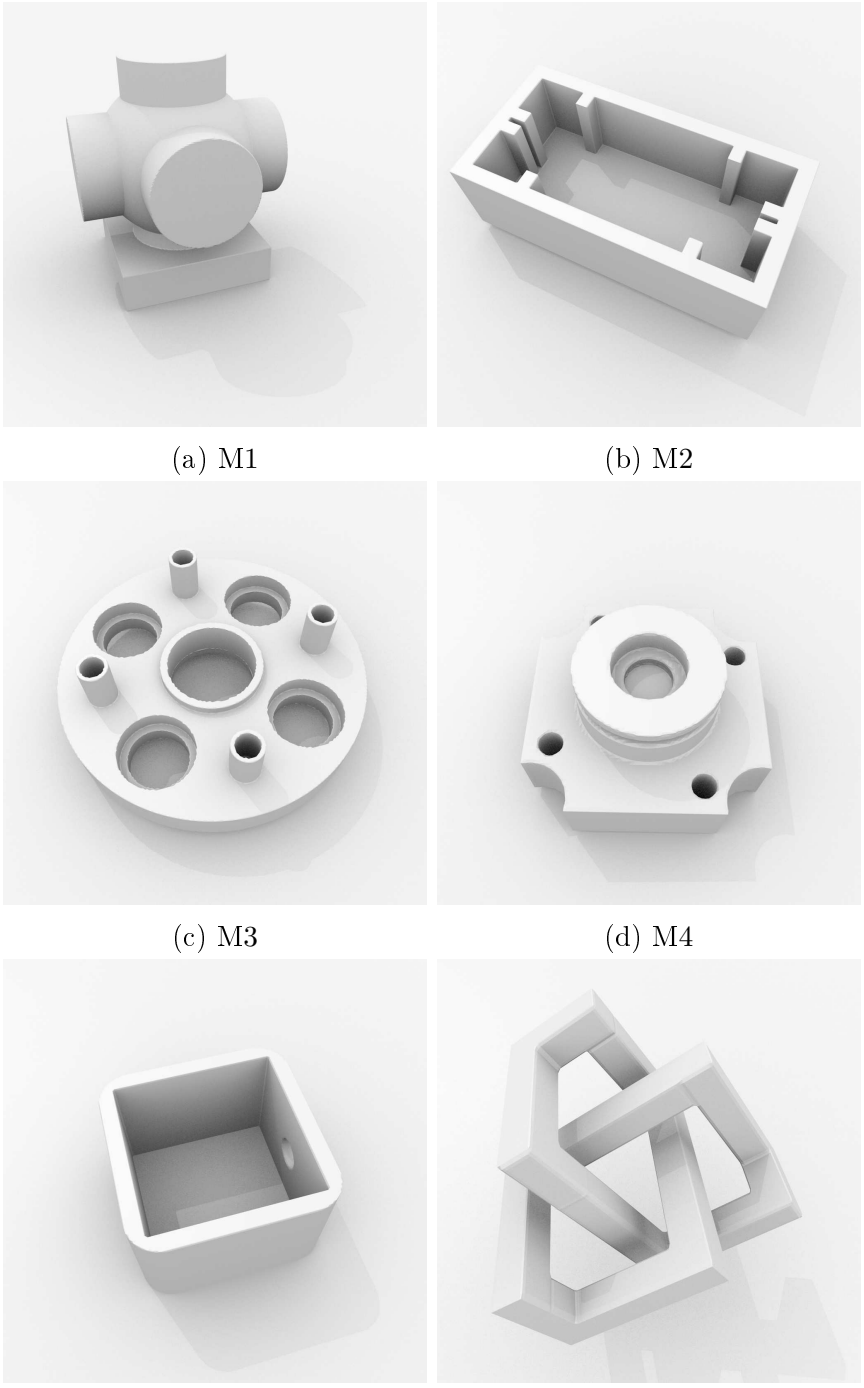
- **Einfügen eines kopierten Teilbaums ($S1$):** An zufälliger Stelle wird eine Vereinigungs- oder Schnittoperation eingefügt, wobei an dieser Stelle Kopien des Teilbaums als Operanden fungieren.

- **Einfügen eines doppelten Komplements ($S2$):** An zufälliger Stelle wird ein doppeltes Komplement eingefügt.
- **Anwendung des Distributivgesetzes ($S3$):** Auf einen zufällig gewählten Teilbaum der Struktur $A \cap^* (B \cup^* C)$ oder $A \cup^* (B \cap^* C)$ werden die Distributivgesetze $A \cap^* (B \cup^* C) = (A \cap^* B) \cup^* (A \cap^* C)$ bzw. $A \cup^* (B \cap^* C) = (A \cup^* B) \cap^* (A \cup^* C)$ angewandt.
- **Anwendung des Absorptionsgesetzes ($S4$):** Auf einen zufällig gewählten Teilbaum werden die Absorptionsgesetze $A = A \cup^* (A \cap^* B)$ oder $A = A \cap^* (A \cup^* B)$ angewandt.
- **EA-basiertes Einfügen von Redundanzen ($S5$):** Der EA (siehe Kapitel 6.4.3.2) wird mit negativen Gewichten für die Terme zur Quantifizierung der Größe und des Überschneidungsgrads ausgeführt, um Redundanzen und suboptimale Teilbaumüberdeckungen zu erzeugen.

Parametrisiert werden kann der Generator mit dem 6-Tupel $(N_{iter}, P_{S1}, P_{S2}, P_{S3}, P_{S4}, P_{S5})$, wobei N_{iter} die Anzahl an Generatoriterationen darstellt und P_{Si} die Wahrscheinlichkeiten für den Einsatz der Strategien zum Einfügen von Redundanzen sind. Für die Evaluation wurden zwei Datensätze mit den Parametern $(10, 1, 1, 1, 1, 1)$ (Datensatz D1) und $(20, 0, 0, 0, 0, 1)$ (Datensatz D2) erzeugt, wie Tabelle 6.3 zeigt.

Modelle (# Knoten, Überschneidungsgrad, (Dimensionen))	**D**1	**D**2
M1 (9, 0.75, (20.0×22.0×20.0))	(97, 0.442)	(78, 0.35)
M2 (13, 0.833, (15.8×31.7×11.3))	(41, 0.952)	(23, 0.727)
M3 (39, 0.474, (21.0×6.0×21.0))	(79, 0.675)	(57, 0.536)
M4 (27, 1, (13.4×13.4×12.0))	(65, 0.97)	(97, 0.708)
M5 (19, 0.667, (24.0×18.0×27.0))	(59, 0.645)	(48, 0.76)
M6 (19, 0.556, (23.1×10.0×10.0))	(99, 0.667)	(43, 0.857)
M7 (91, 0.706, (21.6×7.4×21.8))	(144, 0.725)	(151, 0.728)
M8 (73, 0.722, (31.5×12.7×4.5))	(145, 0.74)	(129, 0.813)
M9 (171, 0.471, (29.7×3.84×30.1))	(191, 0.51)	(216, 0.519)
M10 (17, 0.875, (13.0×12.5×13.0))	(41, 0.864)	(32, 0.563)
M11 (37, 0.789, (26.0×13.0×22.0))	(72, 0.897)	(53, 0.704)

Tabelle 6.3.: Metriken aller Modelle für die Datensätze D1 und D2.

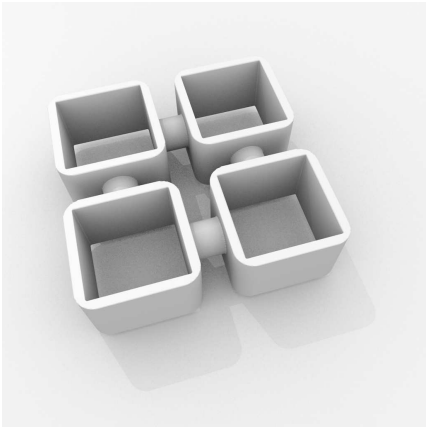


(a) M1 (b) M2

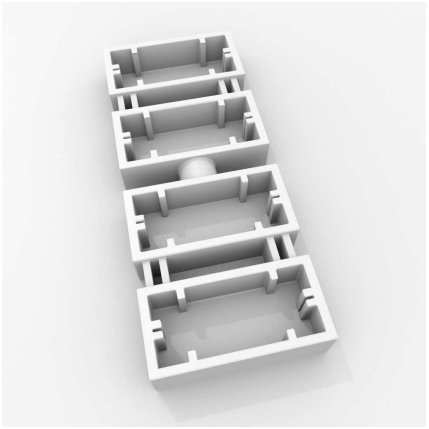
(c) M3 (d) M4

(e) M5 (f) M6

Abbildung 6.4.: Modelle M1-M6.

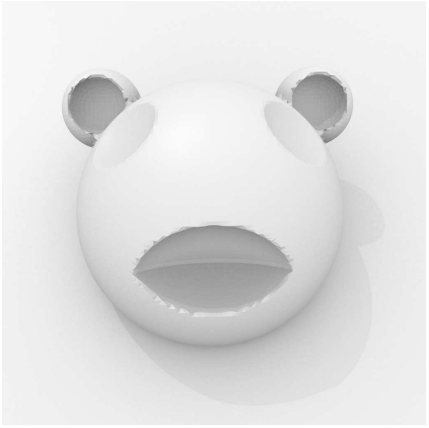


(a) M7

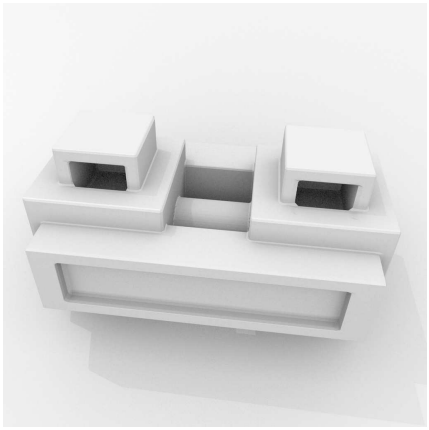


(b) M8

(c) M9



(d) M10

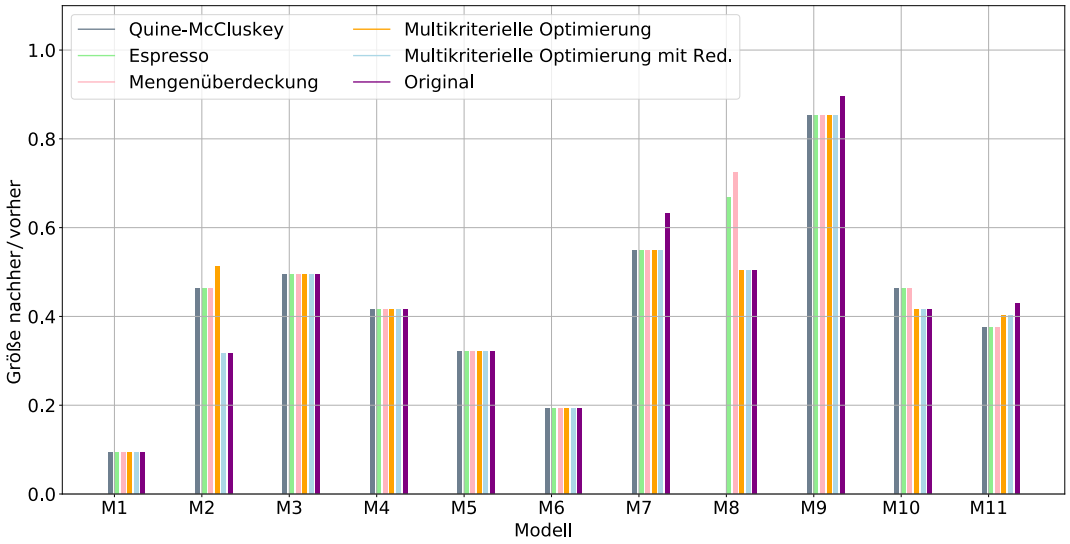


(e) M11

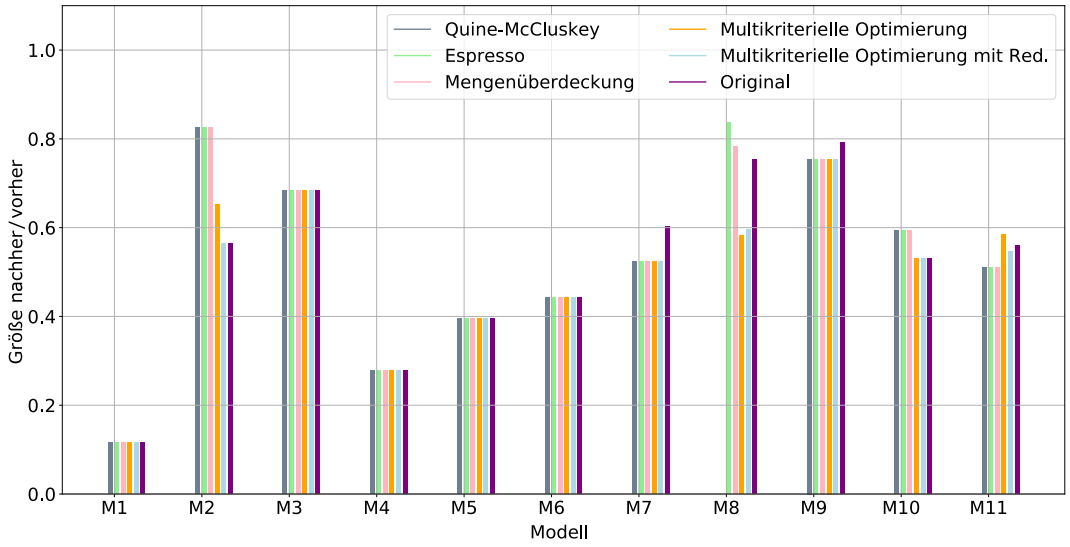
Abbildung 6.5.: Modelle M7-M11.

6.5.1. Ergebnisqualität

Die beiden zu optimierenden Metriken Größe und Überschneidungsgrad werden im Folgenden nun hinsichtlich der vorgestellten Optimierungsmethoden genauer beleuchtet. Dies geschieht auf Basis der Darstellungen in den Abbildungen 6.6 und 6.7. Dabei wird jeweils immer das Verhältnis zwischen dem Wert einer Metrik für den resultierenden KB Φ_{opt} und dem Wert derselben Metrik für den KB des Eingabemodells Φ dargestellt.

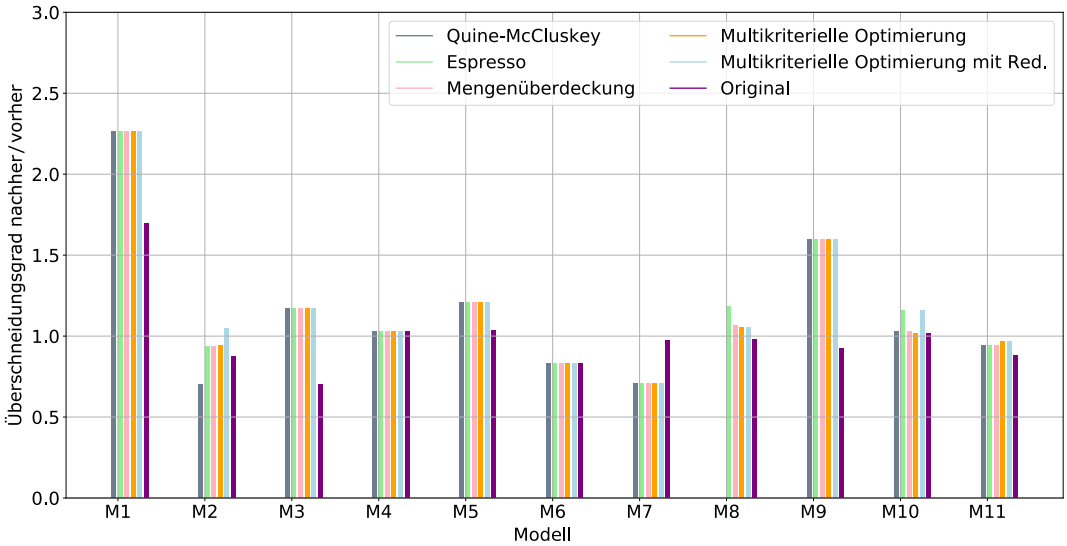


(a) D1

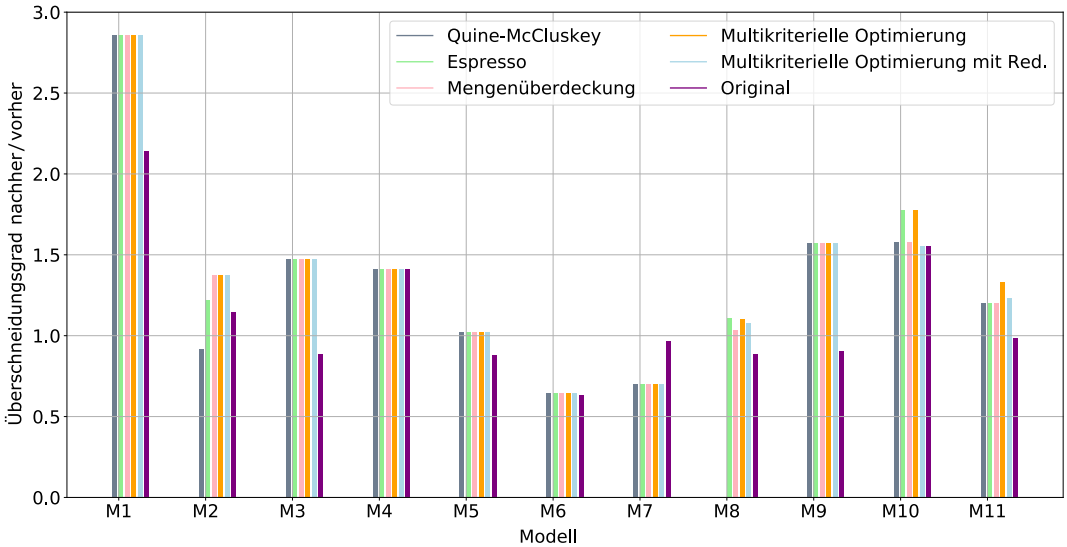


(b) D2

Abbildung 6.6.: Verhältnis zwischen den Größen der Eingabe- und optimierten Ausgabebäumen Φ_{opt} für beide Datensätze. In Lila ist die Größe des manuell erstellten Originalmodells abgebildet.



(a) D1

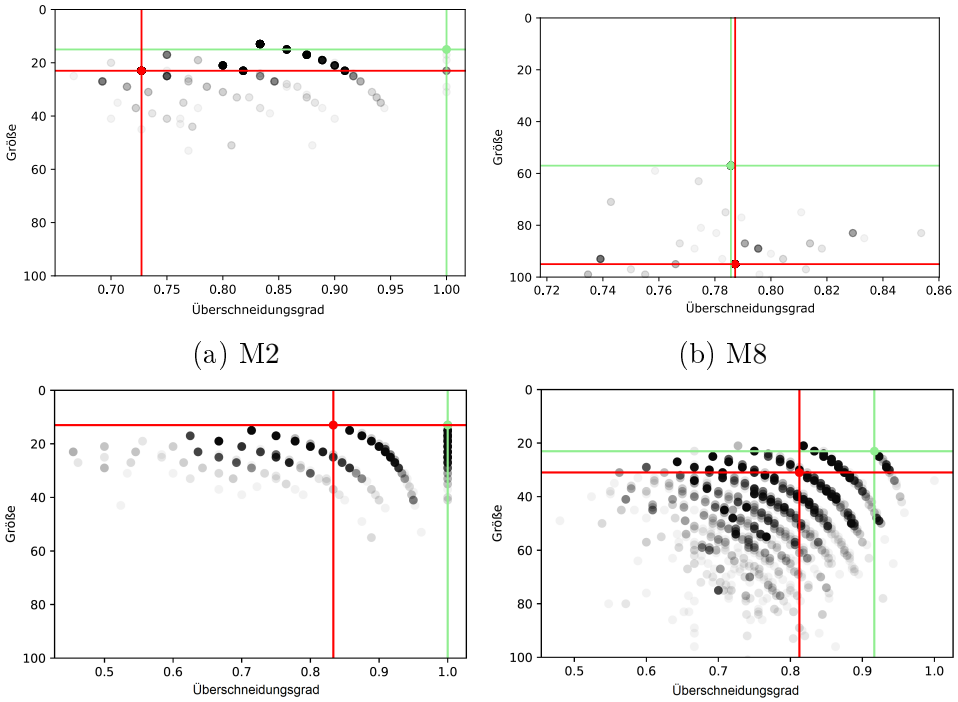


(b) D2

Abbildung 6.7.: Verhältnis zwischen den Werten für den Überschneidungsgrad der Eingabe- und optimierten Ausgabebäumen Φ_{opt} für beide Datensätze. In Lila ist der Überschneidungsgrad des manuell erstellten Originalmodells abgebildet.

Für die Modelle M1, M3, M4, M5, M6, M7 und M9 ergeben sich dabei für alle Optimierungsmethoden und jeweils beide Datensätze identische Werte für Größe und Überschneidungsgrad. Dies ist insofern zu erwarten, als dass bei der vorangehenden *Dekomposition* für diese Modelle keine Restpunktmenge übrig bleibt und damit keine *Suche nach dem optimalen Teilbaum für die Restpunktmenge* stattfindet. Positiv fällt die erreichte Verringerung der Größe bei den Modellen M7, M9 und M11 für beide Datensätze und M8 für Datensatz D2 verglichen mit dem manuell erstellten Eingabemodell auf (siehe Abbildung 6.6, kleiner ist besser).
Die benutzte Implementierung für die Methode *Quine-McCluskey*, welche immer das beste, mit DNF-basierten Minimierern mögliche Ergebnis liefert, scheitert im Fall von Modell M8 an der Problemgröße. Dies liegt daran, dass bei diesem Verfahren sowohl Speicherverbrauch als auch Laufzeit exponentiell mit der Anzahl der Primitive wächst [119].
Für alle Modelle gilt, dass es mindestens ein Verfahren gibt, das eine Reduzierung der Größe auf das Niveau des manuell erstellten Eingabedatensatzes erreicht. Interessant sind auch die erzielten Verbesserungen des Überschneidungsgrads für die Modelle M1, M2, M3, M5, M8, M9 und M11 für beide Datensätze (siehe Abbildung 6.7, größer ist besser). Für die Modelle M4, M6 und M10 wurden Ergebnisse für den Überschneidungsgrad erreicht, die zumindest nicht schlechter sind, als die für das manuell erstellte Eingabemodell. Dies trifft nicht für Modell M7 zu, was mit der erreichten Reduzierung der Größe und dem dadurch möglichen reduzierten Überdeckungsgrad zwischen Teilbäumen zu erklären ist.
Für die Modelle M2, M8 und M10 sowie beide Datensätze resultiert die *multikriterielle Optimierung* in den besten Resultaten, was die Größe betrifft. Eine Ausnahme macht dabei Modell M11, ebenfalls für beide Datensätze. Interessant ist dabei auch der Vergleich zwischen *multikriterieller Optimierung* mit *Redundanzentfernung in* Φ und ohne. Generell lässt sich festhalten, dass sich deren Aktivierung positiv auf das Ergebnis der Größenoptimierung auswirkt (v.a. bei M2, Ausnahme: M8 Datensatz $D1$).
Jedoch erreicht die *multikriterielle Optimierung* nicht in jedem Fall den besten Wert für den Überschneidungsgrad, wie z.B. am Resultat für Modell M8 in Verbindung mit Datensatz D1 zu sehen ist. Dafür lassen sich drei Gründe anführen: Erstens handelt es sich bei der verwendeten Lösungsmethode per EA um ein nicht-deterministisches Verfahren, was nicht immer optimale Lösungen garantiert. Zweitens findet sich eine Erklärung in der Wahl der Gewichte für die Terme zur Quantifizierung von Größe und Überschneidungsgrad in der Zielfunktion (siehe Gleichung 6.4). Dort ist das Gewicht für die Größe um den Faktor 10 größer als das für den Überschneidungsgrad, was auch aus Tabelle 6.2 hervorgeht. Kleinere Bäume werden also gegenüber Bäumen mit besserem Überschneidungsgrad bevorzugt. Drittens ist intuitiv klar, dass das durch eine optimale Größe bedingte Fehlen von Redundanzen eine geringere Teilbaumüberdeckung und damit einen geringeren Wert für den Überschneidungsgrad

zur Folge hat. Auffällig ist dies auch in den vergleichsweise guten Ergebnissen der Methode der *Mengenüberdeckung* bzgl. des Überschneidungsgrads bei gleichzeitig schlechterem Abschneiden bei der Optimierung der Größe.
Ein wichtiger positiver Aspekt der *multikriteriellen Optimierung* ist die Möglichkeit, die Pareto-Front der erzeugten Lösungen für den Restausdruck zu visualisieren und damit die besten Kandidaten manuell auszuwählen. Eine derartige Darstellung ist dabei beispielhaft in Abbildung 6.8 für die Modelle M2, M8, M10 und M11 und Datensatz D2 aufgeführt. Dabei deutlich zu sehen ist z.B., dass für die Modelle M2 und M11 eine Verbesserung von beiden Metriken erreicht werden konnte, wobei die Optimierung für Modell M10 in einem höheren Überdeckungsgrad bei gleichbleibender Größe resultiert. Bei Modell M8 wurde eine signifikant geringere Größe mit einem leicht schlechteren Überdeckungsgrad erkauft.



(a) M2 (b) M8

(c) M10 (d) M11

Abbildung 6.8.: Darstellung der Größe und des Überschneidungsgrads aller durch *multikriterielle Optimierung* ermittelten Lösungskandidaten für den optimalen Restausdruck Φ^i_{opt} mit einer geometrischen Passgenauigkeit $G_{O_{in},O_{au}}$ von exakt 1. Der rote Punkt repräsentiert den Eingangsbaum Φ^i, der grüne Punkt die selektierte Lösung. Die Helligkeit eines Punkts gibt Aufschluss über die Auftrittshäufigkeit des entsprechenden Lösungskandidaten (je dunkler, desto häufiger). Die Darstellung enthält Ergebnisse, die für den Datensatz D2 gewonnen wurden.

Betrachtet man die Methode der *Mengenüberdeckung* genauer, fällt auf, dass die erzielten Ergebnisse für die Optimierung der Größe meist auf dem Niveau der Verfahren *Espresso* und *Quine-McCluskey* liegen. Das macht insofern Sinn, als dass alle drei Methoden zur Kategorie der DNF-basierten Minimierer gehören und somit die gleiche untere Schranke für die Optimierung der Größe haben. Eine Ausnahme bildet dabei Modell M8, für das einmal *Espresso* (D1) und einmal die *Mengenüberdeckung* (D2) besser abschneidet. Das lässt sich damit erklären, dass sowohl *Espresso* als auch die *Mengenüberdeckung* heuristische Ansätze sind, welche abhängig von der Eingabe potentiell unterschiedliche Ergebnisse erzielen können. Einen Ansatz zur Erklärung des schlechteren Abschneidens der *Mengenüberdeckung* bei M8 und Datensatz D1 liefert das Ergebnis der Auswahl der Primimplikanten (siehe Abbildung 6.9). Im Vergleich zum *approximativen Löser*, wählt der *Software-QUBO-Löser* zwei Primimplikanten mehr aus (22 vs. 20), was in einem größeren Gesamtausdruck resultiert.

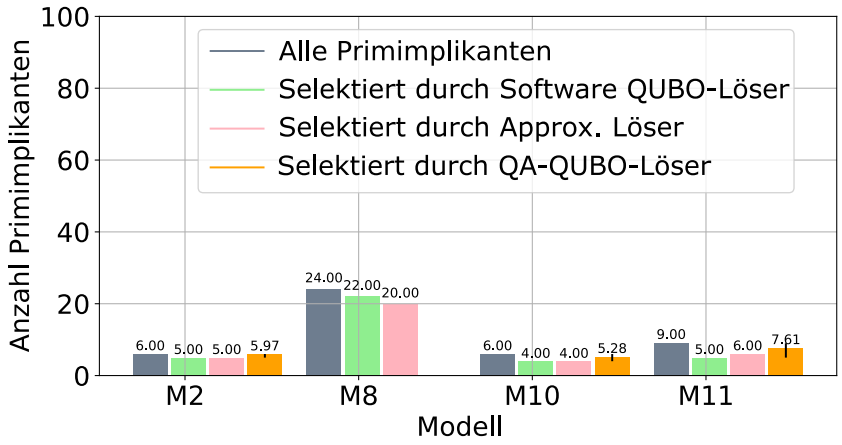


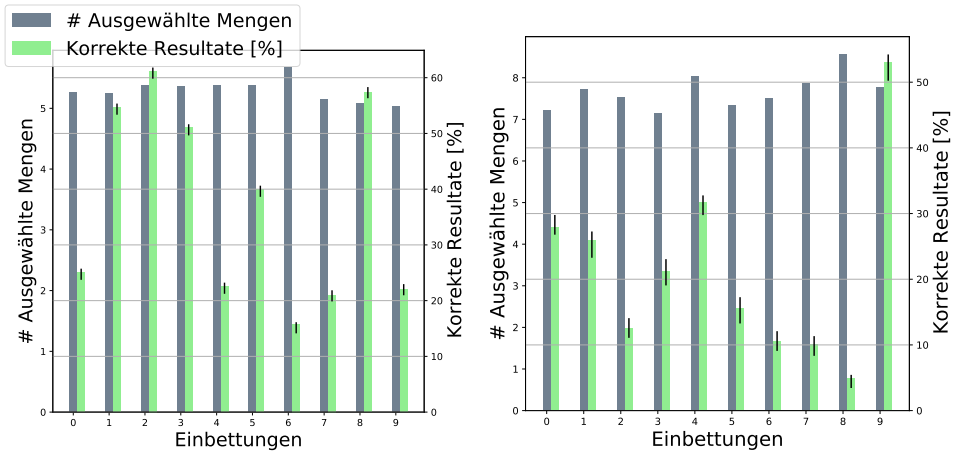
Abbildung 6.9.: Vergleich der Anzahl ausgewählter Primimplikanten für die Software-Emulation des QUBO-Lösers (*Software-QUBO-Löser*), den approximativen Löser und die genutzte QA-Hardware (*QA-QUBO-Löser*) auf Datensatz D1.

Um zu zeigen, dass die Methode der *Mengenüberdeckung* auch für QA-Hardware geeignet ist, wurde ein gesondertes Experiment auf Datensatz D1 durchgeführt. Dazu wurden für die Modelle M2, M10 und M11 je zehn Einbettungen in die *Pegasus*-Topologie erzeugt (siehe Kapitel 2.4.3.3) und pro Einbettung 500 Optimierungsprozesse durchgeführt. Die Erzeugung einer Einbettung für das Modell M8 schlug hingegen aufgrund von dessen Komplexität fehl (1825 logische Qubits, siehe Tabelle 6.4). Eine Übersicht über die Eigenschaften der QUBO-Matrix sowie der erzeugten Einbettungen für die verschiedenen Modelle findet sich in Tabelle 6.4.

Eigensch./Modell	**M**2	**M**8	**M**10	**M**11
# log. Qubits	114	1825	72	126
# phys. Qubits (min max μ σ)	1863 3080 2361, 3 410, 2	-	716 833 770, 6 42, 4	2791 4195 3223, 1 436, 9
Füllgrad QUBO-Matrix (%)	8, 6	1, 6	14, 6	10, 3

Tabelle 6.4.: Eigenschaften der QUBO-Problemformulierungen für die Modelle M2, M8, M10 und M11.

Aus dieser geht u.a. hervor, dass Einbettungen bzgl. der Anzahl genutzter physikalischer Qubits für dieselbe Probleminstanz variieren. Das Verhältnis der Anzahl von logischen Qubits zur Anzahl benötigter physikalischer Qubits zeigt zudem, dass hier um eine Größenordnung mehr physikalische Qubits zur Problemlösung auf der QA-Hardware benötigt werden.
Für die Durchführung wurden voreingestellte Standardparameter verwendet. Minimum, Maximum und Mittelwerte der Ergebnisse für alle Einbettungen zusammengenommen sind in Abbildung 6.9 dargestellt. Zu sehen ist, dass für keines der Modelle optimale Ergebnisse erzielt werden können. In Abbildung 6.10 wird beispielhaft für die Modelle M10 und M11 gezeigt, wie sehr sich einzelne Einbettungen hinsichtlich resultierender Ergebnisqualität und -korrektheit unterscheiden.



(a) M10 (b) M11

Abbildung 6.10.: Eigenschaften der Einbettungen für die Modelle M10 und M11.

Generell ist es schwer auszumachen, von welchen Parametern die Ergebnisqualität abhängt. Faktoren sind, neben der benutzerdefinierten Parametrisierung des Optimierungsprozesses und dessen inhärentem Ergebnisrauschen, auch Größe und Grad der Befüllung der Eingabe-QUBO-Matrix (siehe Tabelle

6.4). Zudem variiert die Qualität der erzeugten Einbettung, u.a. durch das Verhältnis zwischen Anzahl logischer und Anzahl physikalischer Qubits bestimmt, aufgrund der dazu verwendeten Heuristik.

6.5.2. Laufzeitbetrachtungen

In diesem Kapitel werden die gemessenen Laufzeiten für die einzelnen Prozessschritte aufgeführt und diskutiert.

Konfiguration der Prozess-Pipeline. Im Folgenden werden die Laufzeiten für insgesamt sechs Konfigurationen der Prozess-Pipeline untersucht. Eine Konfiguration wird dabei durch ein 3-Tupel $(a, b, c), a, b, c \in \{0, 1\}$ identifiziert. a definiert die verwendete Abtastmethode für die Dekomposition (0: *hierarchisch*, 1: *fundamentalproduktbasierend*). b beschreibt die gleiche Auswahl für die verschiedenen Schritte, die eine *Redundanzentfernung* beinhalten. c legt fest, ob die initiale *Redundanzentfernung in* Φ durchgeführt werden soll (1) oder nicht (0). Ein Beispieltupel wäre $(1, 0, 1)$ für eine Dekomposition mit *fundamentalproduktbasierendem Sampling* und einer *Redundanzentfernung* mit *hierarchischem Sampling*. Zudem wird eine initiale *Redundanzentfernung in* Φ durchgeführt.
Um die Vergleichbarkeit zu gewährleisten, wurde für den Schritt *Suche nach dem optimalen Teilbaum für die Restpunktmenge* für alle Konfigurationen auf den *Espresso*-Logikminimierer (siehe Kapitel 6.4.3.1) zurückgegriffen. Die gemessenen Laufzeiten für beide Datensätze und alle Konfigurationsmöglichkeiten finden sich in den Abbildungen 6.11 und 6.12. Aus diesen geht hervor, dass *fundamentalproduktbasierendes Sampling* in keinem Fall Effizienzvorteile bezüglich der Laufzeit bringt. Zudem führt eine initiale *Redundanzentfernung in* Φ für keines der getesteten Modelle zu einer Steigerung der Laufzeiteffizienz. Die besonders erhöhten Laufzeiten für die *Dekomposition* bei den Modellen M7 und M9 erklären sich aus den Dimensionen der Objekte und der Größe der zu optimierenden KBs (siehe Tabelle 6.3).

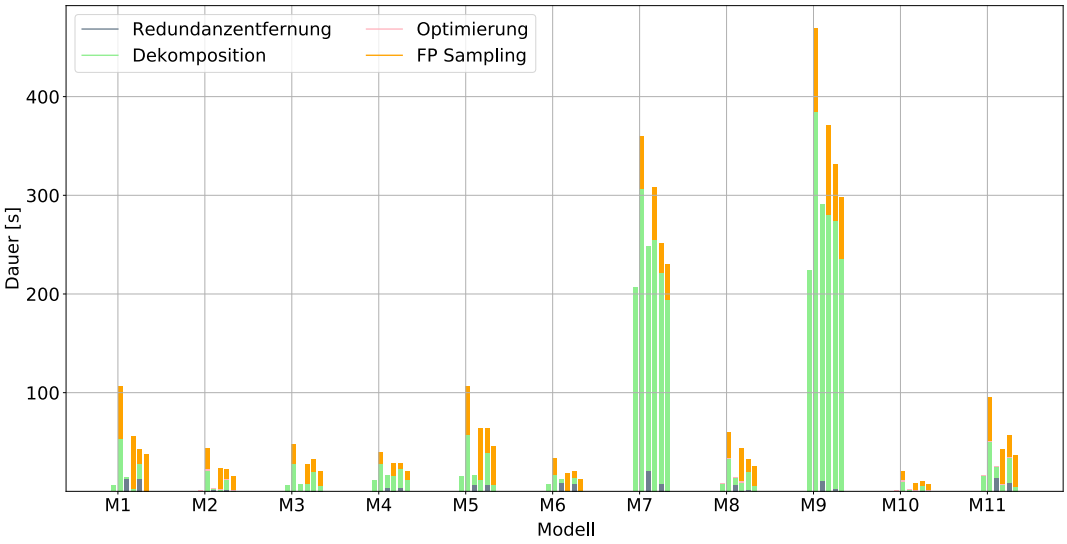


Abbildung 6.11.: Laufzeiten für verschiedene Konfigurationen der Prozess-Pipeline und Datensatz D1. Konfigurationen für jedes Modell von links nach rechts: $(0, 0, 0)$, $(1, 0, 0)$, $(0, 0, 1)$, $(0, 1, 1)$, $(1, 0, 1)$ und $(1, 1, 1)$.

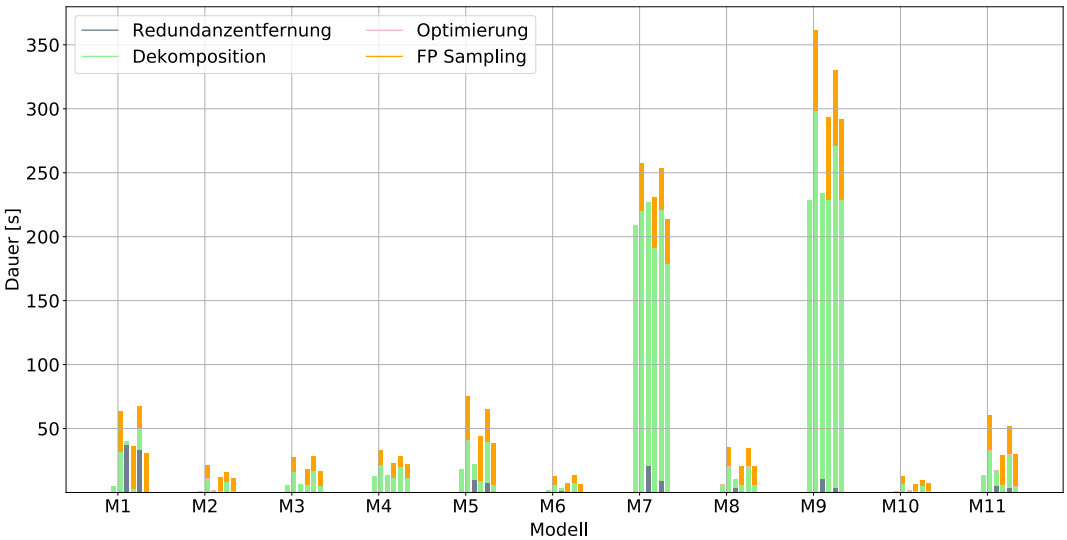
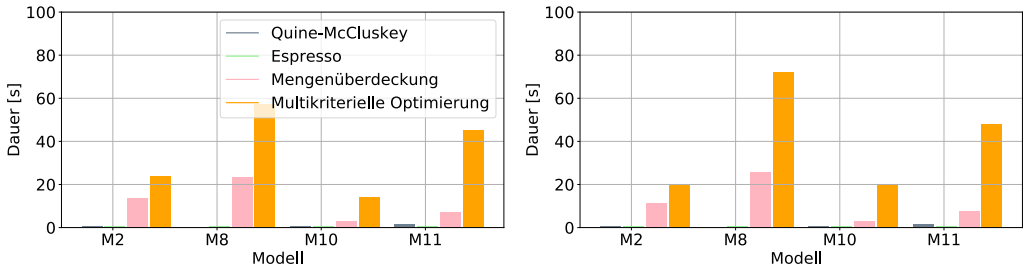


Abbildung 6.12.: Laufzeiten für verschiedene Konfigurationen der Prozess-Pipeline und Datensatz D2. Konfigurationen für jedes Modell von links nach rechts: $(0, 0, 0)$, $(1, 0, 0)$, $(0, 0, 1)$, $(0, 1, 1)$, $(1, 0, 1)$ und $(1, 1, 1)$.

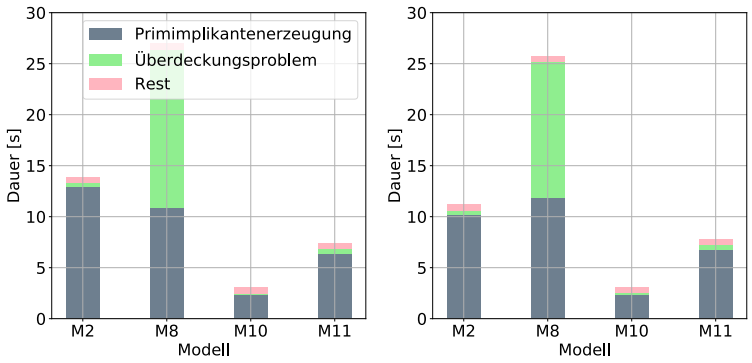
Optimierung. In diesem Abschnitt werden die gemessenen Laufzeiten für den Prozessschritt *Suche nach dem optimalen Teilbaum für die Restpunktmenge* diskutiert. Dazu wurde die Pipeline-Konfiguration $(0, 0, 0)$ für alle Durchläufe gewählt. Relevant sind diese nur für die Modelle M2, M8, M10 und M11, da nur diese im Schritt *Dekomposition* nicht vollständig faktorisiert werden können. Die Ergebnisse finden sich in den Abbildungen 6.13a (Datensatz D1) und 6.13b (Datensatz D2).



(a) D1 (b) D2

Abbildung 6.13.: Laufzeiten des Schritts *Suche nach dem optimalen Teilbaum für die Restpunktmenge* für beide Datensätze und Pipeline-Konfiguration $(0, 0, 0)$.

Espresso ist in jedem Fall das schnellste Verfahren, wohingegen die *multikriterielle Optimierung* per EA die längsten Laufzeiten aufweist. Wie erwähnt, liefert die *Quine-McCluskey*-Methode keine Lösung für Modell M8, was der Komplexität des Modells geschuldet ist. Die Laufzeiten der *Mengenüberdeckung* sind die zweithöchsten. Wie sich die Laufzeit auf einzelne Teilschritte dieser Methode verteilt, findet sich in den Abbildungen 6.14a und 6.14b.



(a) D1 (b) D2

Abbildung 6.14.: Laufzeiten der Methode *Mengenüberdeckung* aufgeschlüsselt nach Ermittlung der *Primimplikanten*, Lösung des Mengenüberdeckungsproblems (*Überdeckungsproblem*) und den vernachlässigbaren Anteilen (*Rest*) für beide Datensätze und Pipeline-Konfiguration $(0, 0, 0)$.

Dabei ist die Ermittlung der Primimplikanten der dominante Faktor, wobei für Modell M8 aufgrund der vergleichsweise hohen Anzahl von Primimplikanten (24) auch das Lösen des Mengenüberdeckungsproblems ins Gewicht fällt. Wie bereits erwähnt wird dafür der *Software-QUBO-Löser* verwendet. Eine Laufzeitmessung mit dem *approximativen Löser* ergibt dabei für keines der hier zu lösenden Probleme eine messbare Laufzeit > 1ms. Die genaue Messung der Laufzeit für den *QA-QUBO-Löser* gestaltet sich als schwierig, da zur eigentlichen Lösungsdauer noch die Dauer der Suche nach einer geeigneten Einbettung sowie die Verzögerungen des genutzten Cloud-Dienstes miteinbezogen werden müssen. Hierbei lässt sich jedoch festhalten, dass die Suche nach einer geeigneten Einbettung um Größenordnungen mehr Zeit in Anspruch nimmt als die eigentliche Optimierung (Millisekunden vs. Sekunden bzw. Minuten).

6.5.3. Schlüsselergebnisse

Möchte man die Erkenntnisse aus der Evaluation zu einigen Kernaussagen verdichten, wären das die folgenden:
Auf vielen in der Evaluation verwandten Datensätzen ist die *Dekomposition* ausreichend für ein größenoptimales Ergebnis, da der Eingabe-KB mit den gegebenen Primitiven vollständig faktorisiert werden kann. Dies entspricht auch der Realität, in welcher komplexe Kombinationen mit mehr als einmal vorkommenden Primitiven eher die Ausnahme bilden. Im Besonderen gilt das für Primitive, die aus einem automatischen Rekonstruktionsverfahren gewonnen wurden (siehe Kapitel 4).
Für nicht vollständig faktorisierbare KBs hingegen sind die auf *Boolescher Minimierung* basierenden Methoden bei der *Suche nach dem optimalen Teilbaum für die Restpunktmenge* bezüglich Laufzeit überlegen, nicht aber was die Ergebnisqualität angeht. Grund dafür ist, dass diese einen DNF-Ausdruck als Ergebnis haben und zudem den Überschneidungsgrad nicht mitberücksichtigen.
Nimmt man hingegen eine erhöhte Laufzeit in Kauf, ist die *multikriterielle Optimierung* den anderen Verfahren überlegen, da sie in der überwiegenden Mehrzahl der Fälle bezüglich Größenoptimalität und Überschneidungsgrad die besten Ergebnisse erzielt und den Vorteil einer übersichtlichen Auswahl von Lösungskandidaten bietet.

6.6. Zusammenfassung und Ausblick

In diesem Kapitel wurde eine Prozess-Pipeline zur Optimierung von KBs motiviert, konzipiert und evaluiert. Dazu wurden zwei relevante Metriken eingeführt: 1) Die Größe eines KBs sowie 2) der rekursiv definierte räumliche Überschneidungsgrad, der den Grad der räumlichen Überdeckung von Operanden der im KB enthaltenen Mengenoperatoren quantifiziert. Letztere Metrik wurde dabei neu entwickelt und stellt somit ein Alleinstellungsmerkmal

des Systems dar. Zur Optimierung der beiden Metriken wurden mehrere Strategien vorgestellt und miteinander verglichen. Als besonders interessant hat sich die Formulierung des Problems als ein multikriterielles Optimierungsproblem herausgestellt, welches mithilfe eines EAs gelöst wurde. Dabei lassen sich Populationen von Lösungskandidaten visualisieren und übersichtlich manuell auswählen, je nach präferierter Metrik.
Ebenfalls hervorzuheben ist die Vielfalt der zur Problemlösung herangezogenen Methoden: Zum einen wurde die Verbindung zwischen KB-Ausdrücken und Booleschen Algebren genutzt, um bestehende Verfahren zur Optimierung von Schaltkreisen adaptieren zu können. Zum anderen wurde eine Formulierung des Größenoptimierungsproblems als Mengenüberdeckungsproblem eingeführt, das aufgrund seiner Darstellung als QUBO-Problem auf QA-Hardware lösbar ist. Dies konnte auch anhand einer funktionstüchtigen Implementierung auf Basis eines QA-Systems der Firma *D-Wave Systems* gezeigt werden.
Abschließend stellt sich die Frage, welche Forschungsrichtungen aufbauend auf dem hier vorgestellten System als besonders vielversprechend angesehen werden können. Im Folgenden wird dazu eine Reihe passender Schwerpunkte erläutert.
Eine interessante Richtung wäre die Einführung neuer Metriken, deren Optimierung den Bearbeitungsprozess von KBs weiter verbessert. Dazu könnten z.B. Primitive oder ganze Teilbäume mit semantischer Information zu den Zusammenhängen der durch sie repräsentierten Bauteile versehen werden. Die Optimierung einer entsprechenden Metrik könnte dann ermöglichen, dass Teilbäume zu repräsentierten Bauteilmodulen im Konstruktionsplan korrespondieren, was die Effizienz bei Exploration und Bearbeitung des Modells signifikant erhöhen würde.
Ein weiterer Aspekt zukünftiger Forschung wäre die Verknüpfung des Systems mit einer Umgebung zur manuellen Modellbearbeitung (*Manuelle Editierung*, Abbildung 3.1). Zu diesem Zweck existieren bereits veröffentlichte Vorarbeiten zu einem VR-basierten Interaktionskonzept, das die Bearbeitung von CSG-basierten Modellen mithilfe einfacher Handgesten und Sprachkommandos ermöglicht. Abbildung 6.15 zeigt dessen Grobarchitektur, wohingegen Abbildung 6.16 einzelne Elemente der Benutzerschnittstelle sowie Impressionen aus deren Verwendung zeigt. Wie in [57] beschrieben, spricht die kurze Einarbeitungszeit klar für das Bedienkonzept, wohingegen dessen prototypischer Status, die Komplexität und Erkennungsrate der Sprachkommandos sowie die Unterstützung komplexer Modelle verbesserungswürdig sind. Dennoch könnte der Prototyp mit dem hier vorgestellten System zur automatischen Optimierung kombiniert werden, indem KBs vor der manuellen Bearbeitung automatisch optimiert werden. Zudem würde sich der Prototyp eignen, die in dieser Arbeit formulierte Hypothese, dass ein höherer Wert für den Überschneidungsgrad mit einer besseren manuellen Editierbarkeit einhergeht, experimentell zu untersuchen.

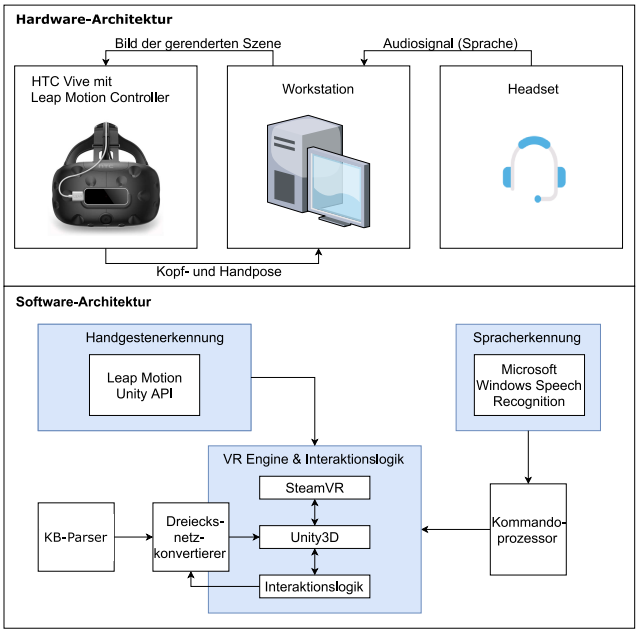
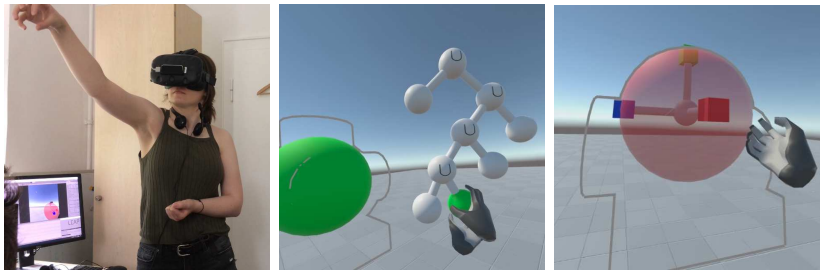


Abbildung 6.15.: Architektur des VR-Prototyps zur manuellen Bearbeitung von KBs. Als VR-Brille wurde die *HTC Vive* der Firma *HTC* verwendet. Als VR-Softwareplattform diente *SteamVR* entwickelt von der Firma *Valve Corporation*. Für die *Handgestenerkennung* kam der *Leap Motion Controller* der Firma *Ultraleap* zum Einsatz. Ein KB wird in die Anwendung geladen (*KB-Parser*) und zur Darstellung mit der 3D-Engine *Unity3D* in ein Dreiecksnetz konvertiert (*Dreiecksnetzkonvertierer*). Informationen aus der *Gestenerkennung* und der *Spracherkennung* bzw. dem *Kommandoprozessor* zur Erkennung einzelner Kommandos werden im Modul *Interaktionslogik* verarbeitet und resultieren in einem neuen Anwendungszustand, der dann dargestellt wird.



(a) (b) (c)

Abbildung 6.16.: Impressionen der Benutzerschnittstelle und deren Nutzung: a) Benutzung des Systems, b) Primitivenselektion mittels eingeblendetem KB und Handgesten, c) Primitivenmanipulation mittels Handgesten.

Die momentane Dynamik im Bereich des Quanten-Computings aufgreifend wäre auch eine noch umfassendere Beleuchtung des Themas der Größenminimierung von KBs mittels QUBO-Formulierungen interessant. Die hier vorgestellte Idee hat den Nachteil, dass mögliche Geschwindigkeitsvorteile durch die klassisch durchgeführte Suche nach Primimplikanten meist zunichte gemacht werden. Um dies zu verhindern, könnten Verfahren entwickelt werden, die auch die Ermittlung der Primimplikanten als QUBO-Problem beschreiben oder ganz auf exakte Primimplikanten verzichten und an deren Stelle andere, effizienter zu ermittelnde Teilmengen der Fundamentalproduktmenge bestimmen (z.B. die Teilmengen, die dominante Primitive beschreiben). In die gleiche Richtung würde die Ersetzung der momentan implementierten Strategie zur Primimplikantensuche durch eine effizientere Methode gehen, welche klassisch berechnet werden könnte.
Die Beispiele zukünftig möglicher Forschungsrichtungen zeigen, dass der in diesem Kapitel behandelte Problemkomplex äußerst vielschichtige Lösungsstrategien erfordert, was ihn für weitere Forschung prädestiniert.

7. Zusammenfassung und Ausblick

In dieser Arbeit wurde das Problem der KB-Synthese aus unstrukturierten räumlichen Daten umfassend beleuchtet und entsprechend neu entwickelte Lösungsmethoden im Detail beschrieben. Dabei wurde für die Darstellung der Daten auf die Punktwolkenrepräsentation zurückgegriffen, die auch entsprechende fehlerhafte Datenpunkte beinhalten kann. Für die übersichtliche Problemdarstellung wurde in Kapitel 3 eine Prozess-Pipeline entwickelt, die zur Lösung des Gesamtproblems notwendige Teilverarbeitungsschritte beinhaltet. Dieses Verarbeitungsmodell ist auch strukturgebend für die Aufteilung der Arbeit und spiegelt sich in den drei großen Inhaltskapiteln wieder (siehe Kapitel 4, 5 und 6), die je einen der zuvor identifizierten drei Problemkomplexe behandeln. Dies ermöglicht auch eine Kombination der in den Kapiteln beschriebenen Verfahren zu einem Gesamtsystem, das die ausgegebene Problemstellung vollständig löst und dabei im Vergleich zu anderen Methoden eine Reihe von Alleinstellungsmerkmalen aufweist.

Nach der Einführung wichtiger Grundlagen (siehe Kapitel 2) und der Diskussion der Problemstellung und des Pipeline-Modells (siehe Kapitel 3), wurde in Kapitel 4 mit der Beschreibung und Evaluation eines neuartigen Systems zur Extraktion von beschränkten Primitiven aus Punktwolken fortgefahren. Im Vergleich zu bestehenden Arbeiten unterstützt das System, neben Zylindern und Kugeln, auch arbiträre konvexe Polytope und nutzt zudem Verfahren, die gewährleisten, dass ermittelte Primitive beschränkt sind. Für die initiale Primitiveneinpassung wurde mit RANSAC auf ein bewährtes Verfahren zurückgegriffen, das mit Methoden des MLs hinsichtlich seiner Robustheit verbessert werden konnte (siehe Kapitel 4.4.4.1). Zur Unterstützung konvexer Polytope wurde ein kombinatorisches Optimierungsproblem formuliert, welches die optimale Kombination eingepasster Ebenen zu konvexen Polytopen zum Ziel hat (siehe Kapitel 4.4.4.2). Gelöst wurde dies mittels evolutionären Techniken, was jedoch nur für relativ kleine Probleminstanzen zufriedenstellende Ergebnisse erzielte. Aus diesem Grund wurde das Verfahren um ein Problempartitionierung erweitert, welche die Eingabepunktwolke in schwach-konvexe Segmente unterteilt (siehe Kapitel 4.5). In der Evaluation (siehe Kapitel 4.6) konnte gezeigt werden, dass das Gesamtverfahren sowohl auf synthetisch erzeugten Datensätzen als auch auf mit photogrammetrischen Methoden erstellten Echtweltmodellen funktioniert.

Zudem konnte der Nachweis erbracht werden, dass die schwach-konvexe Segmentierung eine zuverlässige Extraktion konvexer Polytope aus komplexen Eingabedaten ermöglicht.
Im folgenden Kapitel 5 wurde ein neuartiges System zur Lösung des KB-Syntheseproblems aus einer Menge extrahierter Primitive und einer Eingabepunktwolke im Detail beschrieben und experimentell untersucht. Dabei wurde auf die Möglichkeit der Nutzung von Primitiven Wert gelegt, die aus dem vorhergehend eingeführten Extraktionssystem stammen (siehe Kapitel 4). Als wichtiger Beitrag ist auch die ausführliche Charakterisierung des zu lösenden Syntheseproblems zu nennen, das klare Aussagen über dessen Komplexität zulässt und grundlegend für die Entwicklung des vorgestellten Lösungsverfahrens ist (siehe Kapitel 5.4). Das eingeführte System nutzt den Intersektionsgraphen der Eingabeprimitive, um eine robuste Ermittlung von dominanten Primitiven zu gewährleisten, die dann für eine rekursive Problemvereinfachung und -partitionierung eingesetzt werden (siehe Kapitel 5.5.4, 5.5.5 und 5.5.6). Für die Lösung des vereinfachten und aufgeteilten Problems wurde eine Formulierung als multikriterielles kombinatorisches Optimierungsproblem eingeführt, welches mithilfe eines EAs gelöst wird (siehe Kapitel 5.5.7). Es konnte experimentell bestätigt werden, dass das vorgestellte System im Vergleich zu verwandten Arbeiten als einziges alle gestellten Anforderungen erfüllt. Diese umfassen dabei die Möglichkeit der Verarbeitung von potentiell fehlerhaften Eingabepunktwolken, die Unterstützung von Modellen mit mehr als zehn Primitiven sowie die Berücksichtigung arbiträrer KBs, die z.B. einzelne Primitive auch mehrfach enthalten können (siehe Kapitel 5.3 und 5.6).

In Kapitel 6 stand die Optimierung von KBs im Fokus, was direkt aus den experimentellen Ergebnissen der vorhergehenden Kapitels folgt (siehe Kapitel 5). Dort wurde aufgezeigt, dass mit dem vorgestellten Verfahren synthetisierte KBs Redundanzen enthalten können und damit nicht optimal hinsichtlich ihrer Größe sind (siehe Kapitel 5.6).
In diesem Kapitel wurde eine weitere Optimierungsmetrik neben der Baumgröße eingeführt, der sog. Überschneidungsgrad. Diese quantifiziert den Zusammenhang zwischen der räumlichen Überschneidung von Primitiven und der Lage der zugehörigen Teilbäume innerhalb des KBs. Damit lässt sich forcieren, dass Lagebeziehungen im Raum auf die Struktur des KBs übertragen werden, was eine manuelle Editierung vereinfacht.
Des weiteren wurde eine flexible Prozess-Pipeline eingeführt, die den Einsatz unterschiedlichster Optimierungsalgorithmen erlaubt. Als zentrales Element dieser Pipeline wurde ein Größenreduktionsverfahren vorgestellt, dass auf dem Prinzip der rekursiven Faktorisierung von dominanten Primitiven basiert und mit einem zusätzlichen Mechanismus zur Optimierung der Überschneidungsgrad-Metrik ausgestattet ist (siehe Kapitel 6.4.2). Für Teilbäume, die auf diese Weise nicht vollständig optimiert werden können, wurden unterschiedliche Strategien implementiert und evaluiert. Eine Gruppe

von Algorithmen entstammt dabei der klassischen Minimierung von Logikschaltkreisen und konnte v.a. hinsichtlich Ihrer Laufzeiteffizienz überzeugen. Bezüglich Größenoptimalität jedoch ergaben sich für diese Methodenfamilie Nachteile, da sie grundsätzlich DNF-basierte Ausdrücke zum Ergebnis haben, die per Definition meist nicht optimal hinsichtlich der Baumgröße sind (siehe Kapitel 6.4.3.1 und 6.5). Zusätzlich wurde ein Verfahren vorgestellt, das, ähnlich zur klassischen *Quine-McCluskey*-Methode [119, 143], nach der optimalen Auswahl von Primimplikanten sucht. Dazu wurde das Problem auf ein Mengenüberdeckungsproblem abgebildet und als QUBO-Problem formuliert. Letzteres wurde auch mit QA-Hardware gelöst. Damit konnte gezeigt werden, dass eine Formulierung von Hybriden aus klassischen Methoden und Quanten-Algorithmen auch im vorliegenden Problembereich möglich ist (siehe Kapitel 6.4.3.1).
Darüber hinaus wurde eine weitere Methode zur Optimierung von KBs vorgestellt, die auf einer multikriteriellen Optimierung mithilfe eines EAs beruht und gleichzeitig Baumgröße und Überschneidungsgrad-Metrik optimiert. Die Evaluation dieses Verfahrens zeigte, dass es zwar die schlechteste Laufzeiteffizienz aller verglichenen Methoden aufweist, jedoch den Vorteil hat, dass beide Metriken gleichzeitig optimiert werden und zudem eine minimale Baumgröße erreichbar ist (siehe Kapitel 6.4.3.2).

Zusammenfassend lässt sich festhalten, dass mit den aufeinander abgestimmten Teilsystemen, beschrieben in den drei Inhaltskapiteln, das in dieser Arbeit behandelte KB-Syntheseproblem umfassend, robust und effizient gelöst werden kann. Der Nachweis konnte dabei durch eine gründliche Evaluation der Teilsysteme sowie der darin enthaltenen Komponenten erbracht werden.
Im Folgenden soll nun ein Ausblick auf mögliche zukünftige Forschungsrichtungen und Entwicklungen gegeben werden, denen das hier vorgestellte Gesamtsystem als Grundlage dienen könnte. Dabei werden v.a. Aspekte angesprochen, die nicht bereits in den entsprechenden Inhaltskapiteln Erwähnung finden (siehe Kapitel 4.7, 5.7 und 6.6).
Ein wichtiger Aspekt betrifft die tiefe Integration der einzelnen Teilkomponenten in eine leicht zu bedienende Plattform zur automatischen Synthese von KBs. Versehen mit Standarteinstellungen für die wichtigsten Systemparameter könnte diese zu einem effizienten Werkzeug für das geometrische RE ausgebaut werden. Zusätzlich wäre dabei eine Möglichkeit zur benutzergesteuerten Anpassung von Teilergebnissen wünschenswert, um z.B. Eingabepunktwolken vor Prozessstart manuell aufzubereiten oder die Parameter suboptimal eingepasster Primitive händisch anpassen zu können. Die Spannbreite der möglichen Erweiterungen erstreckt sich dabei über das gesamte Spektrum des entwickelten Pipeline-Modells.
Daran anknüpfend ist die Frage nach der optimalen Ausnutzung verfügbarer Rechenressourcen auch hinsichtlich der Nutzbarmachung heterogener Hardwarelandschaften interessant. So werden zur Ausführung der Prozessschritte

bis auf wenige Ausnahmen klassische CPUs verwendet. Es wäre nun zu eruieren, welche Programmteile von einer massiv-parallelen Ausführung auf GPUs profitieren könnten. Darüber hinaus wäre es vielversprechend, über den Tellerrand der klassisch-imperativen Programmierparadigmen hinauszublicken und zu evaluieren, ob einzelne Teilprobleme entweder mittels DP oder mit dem Einsatz von QA-Hardware beschleunigt werden können. In dieser Arbeit wurden dazu bereits erste Grundsteine hin zu hybriden Algorithmen gelegt.
Als zielgerichtete Forschungsfrage sind auch die Möglichkeiten zur effizienten Verarbeitung großer Eingabepunktwolken interessant. Bisher ist die Punktwolkengröße ein limitierender Faktor des Gesamtsystems, der Auswirkungen auf die maximal mögliche Modellgröße und den erreichbaren Modelldetailgrad hat. Eine Lösungsmöglichkeit umfasst dabei die Anwendung eines intelligenten Segmentierungsschritts, der die Punktwolke geeignet vorpartitioniert. Damit könnte das Gesamtproblem für jede Partition separat gelöst und Teilergebnisse entsprechend zusammengeführt werden. Die Segmentierung sollte dabei laufzeiteffizient sein, jedoch möglichst nur Schnittkanten entlang von Primitivenrändern setzen. Eine weitere Möglichkeit wäre die Konzeption eines Ausdünnungsalgorithmus, der nur Punkte in der Punktwolke belässt, die strukturgebend und detailerhaltend für das Gesamtmodell sind. Dabei liegt die Schwierigkeit bei der effizienten Ermittlung dafür aussagekräftiger Merkmale.
Eine weitere Forschungsrichtung umfasst die Erweiterung der Grammatik der genutzten CSG-Repräsentation. Damit würde der Tatsache Rechnung getragen, dass in professionellen RE-Anwendungsfällen Modelle rekonstruiert werden müssen, die durch komplexe Befehlsketten innerhalb von CAD-Software entstanden sind. So könnte die reguläre Baumstruktur eines KBs durch einen Graphen ersetzt werden, der auch Operationen, wie Schleifen und andere Programmiersprachenkonstrukte, unterstützt. Zudem würde die bereits erwähnte Erweiterung der Menge unterstützter Primitiventypen, z.B. um NURBS, zu einer breiteren Anwendbarkeit führen (siehe Kapitel 4.7). Die Herausforderung besteht nun darin, der erhöhten Problemkomplexität mit modifizierten Algorithmen zu begegnen, ohne dabei die Komplexität der Eingabemodelle einschränken zu müssen.
Diese Auflistung von Ideen zukünftiger Forschung erhebt keinen Anspruch auf Vollständigkeit, zeigt aber die Vielschichtigkeit des in dieser Arbeit behandelten Problems. Für die Zukunft bleibt zu entscheiden, welche Richtung dabei die vielversprechendste ist.

Abkürzungsverzeichnis

AQC *Adiabatischer Quantencomputer.* In seiner nicht-stochastischen Variante ein universelles Modell eines Quantencomputers, das auf dem Adiabatischen Theorem basiert.

BNF *Backus-Naur-Form.* Ein Formalismus zur Beschreibung kontextfreier Grammatiken.

CAD *Computer-Aided Design.* Der computerunterstützte Entwurf von Bauteilen und Produkten.

CAE *Computer-Aided Engineering.* Die computerunterstützte Durchführung von Simulations- und Optimierungsaufgaben im Entwicklungsprozess von Bauteilen und Produkten.

CAM *Computer-Aided Manufacturing.* Die computerunterstützte Herstellung von Werkstücken.

CAQ *Computer-Aided Quality Assurance.* Die computersunterstützte Qualitätssicherung.

CPU *Central Processing Unit.* Die zentrale Rechenheit eines Computers.

CSG *Constructive Solid Geometry.* Ein implizites Repräsentationsschema für Festkörper.

DBSCAN *Density-Based Spatial Clustering of Applications with Noise.* Ein dichtebasiertes Clustering-Verfahren.

DNF *Disjunktive Normalform.* Ein Boolescher Ausdruck, der aus Disjunktionen von Konjunktionen besteht.

DP *Differenzierbare Programmierung.* Ein Programmierparadigma, das auf der Kombination differenzierbarer Funktionen zur Modellierung von Algorithmen beruht.

EA *Evolutionäre Algorithmen.* Metaheuristiken zur Lösung von Optimierungsproblemen, die auf evolutionären Prozessen basieren.

EKEI *Eine-Klasse-Eine-Instanz.* Ein Primitivendetektionsproblem, bei dem genau ein Primitiventyp und genau eine Primitiveninstanz berücksichtigt werden muss.

EKMI *Eine-Klasse-Mehrere-Instanzen.* Ein Primitivendetektionsproblem, bei dem genau ein Primitiventyp, jedoch mehrere Primitiveninstanzen berücksichtigt werden müssen.

FDF *Formdurchmesserfunktion.* Eine Metrik zur approximativen Charakterisierung der zu einem Oberflächenpunkt assoziierten Teilmenge eines Festkörpers.

FNN *Faltende Neuronale Netze.* Eine neuronale Netzarchitektur, die auf der Anwendung von Filterkerneln mit lernbaren Gewichten auf multidimensionale Eingabedaten beruht.

FPS *Farthest Point Sampling.* Ein Sampling-Verfahren zur gleichmäßigen Ausdünnung von Punktwolken.

FSG *Faltungsschichtgruppe.* Eine Gruppe von hintereinandergeschalteten Faltungsschichten (siehe FNNs).

GPU *Graphics Processing Unit.* Die zentrale Rechenheit einer Grafikkarte.

ICP *Iterative Closest Point.* Ein Verfahren zur Registrierung von Punktwolken.

JÄK *Jaccard Ähnlichkeitskoeffizient.* Eine Metrik zur Ermittlung der Prädiktionsgüte bei Klassifikationsproblemen mit mehr als zwei Klassen.

KB *Konstruktionsbaum.* Die Beschreibung eines Festkörpers mittels hierarchisch angeordneten Mengenoperationen und geometrischen Primitiven.

KDNF *Kanonische Disjunktive Normalform.* Ein Boolescher Ausdruck, der aus Disjunktionen von Konjunktionen besteht, wobei jede Konjunktion jede Variable genau einmal enthält.

KI *Künstliche Intelligenz.* Eine Disziplin, die sich mit der berechenbaren Abbildung von Intelligenz beschäftigt.

KNN *Künstliches Neuronales Netz.* Netze bestehend aus abstrahierten Neuronen, die für maschinelles Lernen eingesetzt werden.

LSTM *Long Short-Term Memory.* Eine vielgenutzte Variante eines RNNs.

MKEI *Mehrere-Klassen-Eine-Instanz.* Ein Primitivendetektionsproblem, bei dem genau eine Primitiveninstanz, jedoch mehrere Primitiventypen berücksichtigt werden müssen.

MKMI *Mehrere-Klassen-Mehrere-Instanzen.* Ein Primitivendetektionsproblem, bei dem sowohl mehrere Primitiventypen als auch mehrere Primitiveninstanzen berücksichtigt werden müssen.

ML *Maschinelles Lernen.* Ein Teilbereich der *Künstlichen Intelligenz*, in dem Algorithmen zur Anwendnung kommen, die auf Basis von Trainingsdaten lernen, bestimmte Aufgaben zu erfüllen.

MSP *Mehrschichtiges Perzeptron.* Eine neuronale Netzarchitektur, bei der Neuronen in Schichten angeordnet sind und jedes Neuron einer Schicht mit jedem Neuron der darunterliegenden Schicht verbunden ist.

MVS *Multi-view Stereo.* Ein Verfahren zur 3D-Rekonstruktion auf Basis von zwei oder mehreren Sichten kalibrierter Kameras.

NURBS *Nicht-uniforme Rationale Basis-Spline.* Eine Repräsentation von Kurven, Oberlächen und Volumen mithilfe von Spline-Basisfunktionen.

OCB *Operationscharakteristik eines Beobachters.* Eine Metrik zur Ermittlung der Prädiktionsgüte bei Zwei-Klassen-Klassifikationsproblemen.

PNP *Perspektivischer n-Punkt Algorithmus.* Ein Verfahren zur Ermittlung der externen Kameraparameter basierend auf bekannten Korrespondenzen und zugehörigen dreidimensionalen Punkten.

QA *Quanten-Annealing.* Eine Metaheuristik zur Lösung von diskreten Optimierungsproblemen, welche das theoretische Konzept des Adiabatischen Quantencomputers umsetzt.

QUBO *Quadratic Unconstrained Binary Optimization.* Die Klasse der Optimierungsprobleme mit quadratischen Polynomen definiert auf den Binärzahlen als Zielfunktionen und ohne Nebenbedingungen.

RANSAC *Random Sample Consensus.* Eine iterative, stochastische Methode zur Schätzung von Modellparametern.

RE *Reverse Engineering.* Die Analyse des inneren Aufbaus und der Funktionsweise existierender Produkte ohne das Vorhandensein entsprechender Baupläne und Spezifikationen.

RNN *Rekurrente Neuronale Netze.* Eine neuronale Netzarchitektur, bei der einzelne Neuronen einer Schicht mit Neuronen der gleichen oder einer darunterliegenden Schicht verbunden sein können.

SfM *Structure-from-Motion.* Ein Verfahren zur 3D-Rekonstruktion auf Basis unkalibrierter Kameras.

SIFT *Scale-invariant Feature Transform.* Eine quantitative Beschreibung von Bildmerkmalen, die skalierungsinvariant ist.

SVM *Stützvektormaschinen.* Ein überwachtes Lernverfahren für Klassifizierungs- und Regressionsprobleme.

VDF *Vorzeichenbehaftete Distanzfunktion.* Eine analytische Funktion, die die vorzeichenbehaftete Distanz von einem Punkt zur Oberfläche eines Festkörpers angibt.

VR *Virtual Reality*. Eine berechnete Simulation der Realität.

Abbildungsverzeichnis

Literatur

[1] M. Agoston und M. Agoston. *Algebraic Topology: A first Course*. Earl J.Taft and Edwin Hewitt. M. Dekker, 1976. ISBN: 9780824763510.

[2] D. Aharonov, W. van Dam, J. Kempe, Z. Landau, S. Lloyd und O. Regev. "Adiabatic Quantum Computation is equivalent to standard Quantum Computation". In: *SIAM J. Comput.* 37.1 (Apr. 2007), 166–194. ISSN: 0097-5397. DOI: 10.1137/S0097539705447323.

[3] M. A. Aizerman, E. A. Braverman und L. Rozonoer. "Theoretical foundations of the potential function method in pattern recognition learning." In: *Automation and Remote Control,* Automation and Remote Control, 25. 1964, S. 821–837.

[4] T. Albash und D. A. Lidar. "Adiabatic Quantum Computation". In: *Reviews of Modern Physics* 90.1 (2018), S. 015002.

[5] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin und C. T. Silva. "Computing and Rendering Point Set Surfaces". In: *IEEE Transactions on visualization and computer graphics* 9.1 (2003), S. 3–15.

[6] J. L. Andrews. "User-guided inverse 3D Modeling". Diss. University of California, Berkeley, 2013.

[7] F. Angiulli und C. Pizzuti. "Fast Outlier Detection in high dimensional Spaces". In: *Principles of Data Mining and Knowledge Discovery*. Hrsg. von T. Elomaa, H. Mannila und H. Toivonen. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, S. 15–27.

[8] N. Anwer und L. Mathieu. "From Reverse Engineering to Shape Engineering in Mechanical Design". In: *CIRP Annals* 65.1 (2016), S. 165 –168. ISSN: 0007-8506. DOI: 10.1016/j.cirp.2016.04.052.

[9] K. S. Arun, T. S. Huang und S. D. Blostein. "Least-Squares Fitting of two 3-D Point Sets". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 9.5 (Mai 1987), 698–700. ISSN: 0162-8828. DOI: 10.1109/TPAMI.1987.4767965.

[10] S. Asafi, A. Goren und D. Cohen-Or. "Weak convex Decomposition by Lines-of-Sight". In: *Computer graphics forum* 32.5 (2013), S. 23–31.

[11] A. Avetisyan, A. Dai und M. Nießner. "End-to-end CAD Model Retrieval and 9dof Alignment in 3d Scans". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, S. 2551–2560.

[12] D. Barath und J. Matas. "Multi-class Model Fitting by Energy Minimization and Mode-Seeking". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, S. 221–236.

[13] J.-P. Bauchet und F. Lafarge. "Kinetic Shape Reconstruction". In: *ACM Transactions on Graphics* (2020). DOI: 10.1145/3376918.

[14] R. E. Bellman und S. E. Dreyfus. *Applied Dynamic Programming*. Princeton university press, 2015.

[15] Y. Ben-Shabat, M. Lindenbaum und A. Fischer. "Nesti-net: Normal Estimation for unstructured 3d Point Clouds using Convolutional Neural Networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, S. 10112–10120.

[16] M. Berger, A. Tagliasacchi, L. M. Seversky, P. Alliez, G. Guennebaud, J. A. Levine, A. Sharf und C. T. Silva. "A Survey of Surface Reconstruction from Point Clouds". In: *Computer Graphics Forum*. Bd. 36. Wiley Online Library. 2017, S. 301–329.

[17] P. J. Besl und R. C. Jain. "Segmentation through variable-order Surface Fitting". In: *IEEE Transactions on pattern analysis and machine intelligence* 10.2 (1988), S. 167–192.

[18] M. Booth und S. P. Reinhardt. *Partitioning Optimization Problems for hybrid Classical / Quantum Execution*. Techn. Ber. D-Wave Systems, 2017.

[19] K. Boothby, P. Bunyk, J. Raymond und A. Roy. *Next-Generation Topology of D-Wave Quantum Processors*. 2020. arXiv: 2003.00133 [quant-ph].

[20] B. E. Boser, I. M. Guyon und V. N. Vapnik. "A Training Algorithm for optimal Margin Classifiers". In: *Proceedings of the fifth annual workshop on Computational learning theory*. 1992, S. 144–152.

[21] V. Bouzas, H. Ledoux und L. Nan. "Structure-aware Building Mesh Polygonization". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 167 (2020), S. 432–442. ISSN: 0924-2716. DOI: 10.1016/j.isprsjprs.2020.07.010.

[22] R. K. Brayton, G. D. Hachtel, C. McMullen und A. Sangiovanni-Vincentelli. *Logic Minimization Algorithms for VLSI Synthesis*. Bd. 2. Springer Science & Business Media, 1984.

[23] L. Breiman, J. H. Friedman, R. A. Olshen und C. J. Stone. *Classification and Regression Trees*. Monterey, CA: Wadsworth und Brooks, 1984.

[24] C. Bron und J. Kerbosch. "Algorithm 457 : Finding all Cliques of an undirected Graph [H]". In: *Communications of the ACM* 16.9 (1973), S. 575–577. ISSN: 0001-0782. DOI: 10.1145/362342.362367.

[25] D. Brujic, I. Ainsworth und M. Ristic. "Fast and accurate NURBS fitting for reverse engineering". In: *The International Journal of Advanced Manufacturing Technology* 54.5-8 (2011), S. 691–700.

[26] S. F. Buchele und R. H. Crawford. "Three-dimensional Halfspace Constructive Solid Geometry Tree Construction from implicit Boundary Representations". In: *Computer-Aided Design* 36.11 (2004), S. 1063–1073.

[27] P. I. Bunyk, E. M. Hoskinson, M. W. Johnson, E. Tolkacheva, F. Altomare, A. J. Berkley, R. Harris, J. P. Hilton, T. Lanting, A. J. Przybysz und J. Whittaker. "Architectural Considerations in the Design of a Superconducting Quantum Annealing Processor". In: *IEEE Transactions on Applied Superconductivity* 24.4 (2014), S. 1–10. DOI: 10.1109/TASC.2014.2318294.

[28] E. Catmull und J. Clark. "Recursively generated B-Spline Surfaces on arbitrary topological Meshes". In: *Computer-aided design* 10.6 (1978), S. 350–355.

[29] F. Cazals und M. Pouget. "Estimating differential Quantities using polynomial Fitting of osculating Jets". In: *Computer Aided Geometric Design* 22.2 (2005), S. 121–146.

[30] B. Chazelle. "Convex Partitions of Polyhedra: A lower Bound and worst-case optimal Algorithm". In: *SIAM J. Comput.* 13.3 (Juli 1984), 488–507. ISSN: 0097-5397. DOI: 10.1137/0213031.

[31] V. Chvatal. "A greedy Heuristic for the Set-Covering Problem". In: *Mathematics of Operations Research* 4.3 (1979), S. 233–235. ISSN: 0364765X, 15265471.

[32] R. T. Collins. "A Space-Sweep Approach to true Multi-Image Matching". In: *Proceedings CVPR IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE. 1996, S. 358–363.

[33] T. H. Cormen, C. E. Leiserson, R. L. Rivest und C. Stein. *Introduction to Algorithms*. MIT press, 2009.

[34] T. M. Cover. "Geometrical and Statistical Properties of Systems of linear Inequalities with Applications in Pattern Recognition". In: *IEEE Transactions on Electronic Computers* EC-14.3 (1965), S. 326–334. DOI: 10.1109/PGEC.1965.264137.

[35] G. Cybenko. "Approximation by Superpositions of a sigmoidal Function". In: *Mathematics of control, signals and systems* 2.4 (1989), S. 303–314.

[36] B. Deng, K. Genova, S. Yazdani, S. Bouaziz, G. Hinton und A. Tagliasacchi. "CvxNet: Learnable Convex Decomposition". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, S. 31–44.

[37] O. Deussen. *Modeling in Computer Graphics.* Vorlesungsskript. 2013.

[38] L. Du, Y. Lai, C. Luo, Y. Zhang, J. Zheng, X. Ge und Y. Liu. "E-quality Control in Dental Metal Additive Manufacturing Inspection using 3D Scanning and 3D Measurement". In: *Frontiers in Bioengineering and Biotechnology* 8 (2020), S. 1038. ISSN: 2296-4185. DOI: 10.3389/fbioe.2020.01038.

[39] T. Du, J. P. Inala, Y. Pu, A. Spielberg, A. Schulz, D. Rus, A. Solar-Lezama und W. Matusik. "InverseCSG: Automatic Conversion of 3D Models to CSG Trees". In: *ACM Trans. Graph.* 37.6 (Dez. 2018). ISSN: 0730-0301. DOI: 10.1145/3272127.3275006.

[40] R. O. Duda und P. E. Hart. "Use of the Hough Transformation to detect Lines and Curves in Pictures". In: *Communications of the ACM* 15.1 (1972), S. 11–15.

[41] J. D. Eblen, C. A. Phillips, G. L. Rogers und M. A. Langston. "The Maximum Clique Enumeration Problem: Algorithms, Applications and Implementations". In: *Bioinformatics Research and Applications.* Hrsg. von J. Chen, J. Wang und A. Zelikovsky. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, S. 306–319. ISBN: 978-3-642-21260-4.

[42] H. Edelsbrunner und E. P. Mücke. "Three-dimensional Alpha Shapes". In: *Transactions on Graphics* 13.1 (1994), 43–72.

[43] M. Ehrgott. *Multicriteria Optimization.* Bd. 491. Springer Science & Business Media, 2005.

[44] M. Ester, H.-P. Kriegel, J. Sander und X. Xu. "A density-based Algorithm for Discovering Clusters in large spatial Databases with Noise". In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining.* KDD'96. Portland, Oregon: AAAI Press, 1996, 226–231.

[45] H. Fang und F. Lafarge. "Connect-and-Slice: An hybrid Approach for Reconstructing 3D Objects". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).* 2020.

[46] E. Farhi, J. Goldstone, S. Gutmann und M. Sipser. *Quantum Computation by Adiabatic Evolution.* 2000. arXiv: quant-ph/0001106 [quant-ph].

[47] P.-A. Fayolle und A. Pasko. "An evolutionary Approach to the Extraction of Object Construction Trees from 3D Point Clouds". In: *Computer-Aided Design* 74 (2016), S. 1–17.

[48] S. Feld, M. Friedrich und C. Linnhoff-Popien. "Optimizing Geometry Compression Using Quantum Annealing". In: *2018 IEEE Globecom Workshops (GC Wkshps).* 2018, S. 1–6. DOI: 10.1109/GLOCOMW.2018.8644358.

[49] M. A. Fischler und R. C. Bolles. "Random Sample Consensus: a Paradigm for Model Fitting with Applications to Image Analysis and automated Cartography". In: *Communications of the ACM* 24.6 (1981), S. 381–395.

[50] A. W. Fitzgibbon. "Robust Registration of 2D and 3D Point Sets". In: *Image and vision computing* 21.13-14 (2003), S. 1145–1153.

[51] M. Friedrich und P.-A. Fayolle. "Reconstruction of Convex Polytope Compositions from 3D Point-clouds". In: *Proceedings of the 16th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications, VISIGRAPP 2021, Volume 1: GRAPP, Online Streaming, February 8-10, 2021*. Hrsg. von A. A. de Sousa, V. Havran, J. Braz und K. Bouatouch. SCITEPRESS, 2021, S. 75–84. DOI: 10.5220/0010297100750084.

[52] M. Friedrich, P.-A. Fayolle, T. Gabor und C. Linnhoff-Popien. "Optimizing Evolutionary CSG Tree Extraction". In: *Proceedings of the Genetic and Evolutionary Computation Conference*. GECCO '19. Prague, Czech Republic: Association for Computing Machinery, 2019, 1183–1191. ISBN: 9781450361118. DOI: 10.1145/3321707.3321771.

[53] M. Friedrich, S. Feld, T. Phan und P.-A. Fayolle. "Accelerating Evolutionary Construction Tree Extraction via Graph Partitioning". In: *Proceedings of the 26th International Conference on Computer Graphics, Visualization and Computer Vision (WSCG)*. 2018.

[54] M. Friedrich, F. Guimera Cuevas, A. Sedlmeier und A. Ebert. "Evolutionary Generation of Primitive-Based Mesh Abstractions". In: *Proceedings of the 27th International Conference on Computer Graphics, Visualization and Computer Vision (WSCG)*. 2019.

[55] M. Friedrich., S. Illium., P.-A. Fayolle. und C. Linnhoff-Popien. "A Hybrid Approach for Segmenting and Fitting Solid Primitives to 3D Point Clouds". In: *Proceedings of the 15th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - GRAPP,* INSTICC. SciTePress, 2020, S. 38–48. ISBN: 978-989-758-402-2. DOI: 10.5220/0008870600380048.

[56] M. Friedrich, S. Illium, P.-A. Fayolle und C. Linnhoff-Popien. "CSG Tree Extraction from 3D Point Clouds and Meshes Using a Hybrid Approach". In: *Computer Vision, Imaging and Computer Graphics Theory and Applications*. Hrsg. von K. Bouatouch, A. A. de Sousa, M. Chessa, A. Paljic, A. Kerren, C. Hurter, G. M. Farinella, P. Radeva und J. Braz. Cham: Springer International Publishing, 2022, S. 53–79. ISBN: 978-3-030-94893-1.

[57] M. Friedrich, S. Langer und F. Frey. "Combining Gesture and Voice Control for Mid-air Manipulation of CAD Models in VR Environments". In: *Proceedings of the 16th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications, VISIGRAPP 2021, Volume 2: HUCAPP, Online Streaming, February 8-10, 2021*. Hrsg. von A. Paljic, T. C. Peck, J. Braz und K. Bouatouch. SCITEPRESS, 2021, S. 119–127. DOI: 10.5220/0010170501190127.

[58] M. Friedrich, C. Roch, S. Feld, C. Hahn und P.-A. Fayolle. "A Flexible Pipeline for the Optimization of Construction Trees". In: *28th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG 2020)*. 2020, S. 10. DOI: 10.24132/CSRN.2020.3001.10.

[59] K. Fukuda und A. Prodon. "Double Description Method Revisited". In: *Combinatorics and Computer Science*. 1995.

[60] K. Fukushima und S. Miyake. "Neocognitron: A self-organizing Neural Network Model for a Mechanism of Visual Pattern Recognition". In: *Competition and cooperation in neural nets*. Springer, 1982, S. 267–285.

[61] M. Gadelha, A. RoyChowdhury, G. Sharma, E. Kalogerakis, L. Cao, E. Learned-Miller, R. Wang und S. Maji. "Label-efficient Learning on Point Clouds using approximate convex Decompositions". In: *European Conference on Computer Vision (ECCV)*. 2020.

[62] R. Gal, A. Shamir und D. Cohen-Or. "Pose-oblivious Shape Signature". In: *IEEE transactions on visualization and computer graphics* 13.2 (2007), S. 261–271.

[63] M. Gammon. "A Review of common Geometry Issues affecting Mesh Generation". In: *2018 AIAA Aerospace Sciences Meeting*. 2018, S. 1402.

[64] M. Garey, D. Johnson und H. Witsenhausen. "The Complexity of the generalized Lloyd - Max Problem (Corresp.)" In: *IEEE Transactions on Information Theory* 28.2 (1982), S. 255–256. DOI: 10.1109/TIT.1982.1056488.

[65] A. Géron. *Hands-on Machine Learning with Scikit-Learn and TensorFlow : Concepts, Tools, and Techniques to build intelligent Systems*. Sebastopol, CA: O'Reilly Media, 2017. ISBN: 978-1491962299.

[66] F. Glover. "Future Paths for Integer Programming and Links to Artificial Intelligence". In: *Computers & operations research* 13.5 (1986), S. 533–549.

[67] T. F. Gonzalez. "Clustering to minimize the Maximum Intercluster Distance". In: *Theoretical Computer Science* 38 (1985), S. 293 –306. ISSN: 0304-3975. DOI: 10.1016/0304-3975(85)90224-5.

[68] T. D. Goodrich, T. S. Humble und B. D. Sullivan. *Optimizing Adiabatic Quantum Program Compilation using a Graph-Theoretic Framework*. 2017. arXiv: 1704.01996 [quant-ph].

[69] M. Grieves. "Digital Twin: Manufacturing Excellence through virtual Factory Replication". In: *White paper* 1 (2014), S. 1–7.

[70] E. Grilli, F. Menna und F. Remondino. "A Review of Point Clouds Segmentation and Classification Algorithms". In: *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 42W3 (Feb. 2017), S. 339–344. DOI: 10.5194/isprs-archives-XLII-2-W3-339-2017.

[71] C. Grimme und J. Bossek. *Einführung in die Optimierung: Konzepte, Methoden und Anwendungen*. Springer Fachmedien Wiesbaden, 2018. ISBN: 9783658211509.

[72] R. H. Hahnloser, R. Sarpeshkar, M. A. Mahowald, R. J. Douglas und H. S. Seung. "Digital Selection and analogue Amplification coexist in a cortex-inspired Silicon Circuit". In: *Nature* 405.6789 (2000), S. 947–951.

[73] A. Haleem und M. Javaid. "3D Scanning Applications in Medical Field: A literature-based Review". In: *Clinical Epidemiology and Global Health* 7.2 (2019), S. 199–210.

[74] K. Hamza und K. Saitou. "Optimization of Constructive Solid Geometry via a Tree-based Multi-objective Genetic Algorithm". In: *Genetic and Evolutionary Computation – GECCO 2004*. Hrsg. von K. Deb. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, S. 981–992. ISBN: 978-3-540-24855-2.

[75] H. Han, X. Han, F. Sun und C. Huang. "Point Cloud Simplification with preserved dge based on Normal Vector". In: *Optik-International Journal for Light and Electron Optics* 126.19 (2015), S. 2157–2162.

[76] X. Han, Z. Li, H. Huang, E. Kalogerakis und Y. Yu. "High-resolution Shape Completion using Deep Neural Networks for global Structure and local Geometry Inference". In: *Proceedings of the IEEE international conference on computer vision*. 2017, S. 85–93.

[77] R. Hartley und A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge university press, 2003.

[78] A. Hattab und G. Taubin. "3D Modeling by Scanning Physical Modifications". In: *2015 28th SIBGRAPI Conference on Graphics, Patterns and Images*. 2015, S. 25–32. DOI: 10.1109/SIBGRAPI.2015.8.

[79] F. R. Helmert. *Die Ausgleichungsrechnung nach der Methode der kleinsten Quadrate: mit Anwendungen auf die Geodäsie, die Physik und die Theorie der Meßinstrumente*. Leipzig: Teubner, 1872.

[80] P. Hermosilla, T. Ritschel und T. Ropinski. "Total Denoising: Unsupervised Learning of 3D Point Cloud Cleaning". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, S. 52–60.

[81] A. Hernández und J. M. Amigó. *Differentiable Programming and its Applications to dynamical Systems*. 2020. arXiv: 1912.08168 [math.DS].

[82] T. K. Ho. "Random Decision Forests". In: *Proceedings of 3rd international conference on document analysis and recognition*. Bd. 1. IEEE. 1995, S. 278–282.

[83] S. Hochreiter und J. Schmidhuber. "Long short-term Memory". In: *Neural computation* 9.8 (1997), S. 1735–1780.

[84] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald und W. Stuetzle. "Surface Reconstruction from unorganized Points". In: *Proceedings of the 19th annual conference on Computer graphics and interactive techniques*. 1992, S. 71–78.

[85] K. Hornik. "Approximation Capabilities of Multilayer Feedforward Networks". In: *Neural networks* 4.2 (1991), S. 251–257.

[86] K. Hornik, M. Stinchcombe und H. White. "Universal Approximation of an unknown Mapping and its Derivatives using Multilayer Feedforward Networks". In: *Neural Networks* 3.5 (1990), S. 551–560. ISSN: 0893-6080. DOI: 10.1016/0893-6080(90)90005-6.

[87] K. Hui und S. Tan. "Construction of a hybrid Sweep-CSG Modeler — The Sweep-CSG Representation". In: *Engineering with computers* 8.2 (1992), S. 101–119.

[88] L. Hyafil und R. L. Rivest. "Constructing optimal binary Decision Trees is NP-complete". In: *Information Processing Letters* 5.1 (1976), S. 15–17. ISSN: 0020-0190. DOI: 10.1016/0020-0190(76)90095-8.

[89] K. Ikeuchi, Hrsg. *Computer Vision, A Reference Guide*. Springer, 2014. ISBN: 978-0-387-30771-8. DOI: 10.1007/978-0-387-31439-6.

[90] E Ising. "Beitrag zur Theorie des Ferromagnetismus". In: *Z. Phys.* 31 (1925), S. 253–258.

[91] B. Jian und B. C. Vemuri. "Robust Point Set Registration using Gaussian Mixture Models". In: *IEEE transactions on pattern analysis and machine intelligence* 33.8 (2010), S. 1633–1645.

[92] T. Kadowaki und H. Nishimori. "Quantum Annealing in the transverse Ising Model". In: *Phys. Rev. E* 58 (5 1998), S. 5355–5363. DOI: 10.1103/PhysRevE.58.5355.

[93] O. V. Kaick, N. Fish, Y. Kleiman, S. Asafi und D. Cohen-Or. "Shape Segmentation by approximate Convexity Analysis". In: *ACM Transactions on Graphics (TOG)* 34.1 (2014), S. 1–11.

[94] A. Kaiser, J. A. Y. Zepeda und T. Boubekeur. "A Survey of simple geometric Primitives Detection Methods for captured 3D Data". In: *Computer Graphics Forum* 38.1 (2019), S. 167–196.

[95] G. Kamiske und J. Brauer. *Qualitätsmanagement von A bis Z: Wichtige Begriffe des Qualitätsmanagements und ihre Bedeutung*. Hanser, 2011. ISBN: 9783446425811.

[96] K. Kania, M. Zięba und T. Kajdanowicz. *UCSG-Net – Unsupervised Discovering of Constructive Solid Geometry Tree*. 2020. arXiv: 2006.09102 [cs.CV].

[97] R. M. Karp. "Reducibility among combinatorial problems". In: *Complexity of Computer Computations*. Springer, 1972, S. 85–103.

[98] M. Kazhdan, M. Bolitho und H. Hoppe. "Poisson Surface Reconstruction". In: *Proceedings of the fourth Eurographics symposium on Geometry processing*. Bd. 7. SGP '06. Cagliari, Sardinia, Italy: Eurographics Association, 2006, 61–70.

[99] J. Kempe, A. Kitaev und O. Regev. "The Complexity of the local Hamiltonian Problem". In: *SIAM Journal on Computing* 35.5 (2006), S. 1070–1097.

[100] V. G. Kim, W. Li, N. J. Mitra, S. Chaudhuri, S. DiVerdi und T. Funkhouser. "Learning part-based Templates from large Collections of 3D Shapes". In: *ACM Transactions on Graphics (TOG)* 32.4 (2013), S. 1–12.

[101] D. E. Knuth. *The Art of Computer Programming, Volume 4, Fascicle 4: Generating All Trees; History of Combinatorial Generation*. Addison-Wesley Professional, 2006.

[102] A. Kuş. "Implementation of 3D Optical Scanning Technology for Automotive Applications". In: *Sensors* 9.3 (2009), S. 1967–1979. ISSN: 1424-8220. DOI: 10.3390/s90301967.

[103] F. Lafarge und P. Alliez. "Surface Reconstruction through Point Set Structuring". In: *Computer Graphics Forum* 32.2 (Mai 2013), S. 225–234.

[104] J. E. Lenssen, C. Osendorfer und J. Masci. "Deep iterative Surface Normal Estimation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, S. 11247–11256.

[105] M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. Anderson, J. Davis, J. Ginsberg u. a. "The digital Michelangelo Project: 3D Scanning of large Statues". In: *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. 2000, S. 131–144.

[106] L. Li, M. Sung, A. Dubrovina, L. Yi und L. J. Guibas. "Supervised Fitting of geometric Primitives to 3d Point Clouds". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, S. 2652–2660.

[107] T.-M. Li, M. Gharbi, A. Adams, F. Durand und J. Ragan-Kelley. "Differentiable Programming for Image Processing and Deep Learning in Halide". In: *ACM Trans. Graph. (Proc. SIGGRAPH)* 37.4 (2018), 139:1–139:13.

[108] Y. Li, R. Bu, M. Sun, W. Wu, X. Di und B. Chen. "PointCNN: Convolution On X-Transformed Points". In: *Advances in Neural Information Processing Systems*. Hrsg. von S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi und R. Garnett. Bd. 31. Curran Associates, Inc., 2018.

[109] Y. Li, X. Wu, Y. Chrysanthou, A. Sharf, D. Cohen-Or und N. J. Mitra. "GlobFit: Consistently Fitting Primitives by Discovering global Relations". In: *ACM Transactions on Graphics* 30.4 (2011), 52:1–52:12.

[110] C. Little, D. Patterson, B. Moyle und A. Bec. "Every Footprint Tells a Story: 3D Scanning of Heritage Artifacts as an interactive Experience". In: *Proceedings of the Australasian Computer Science Week Multiconference*. 2018, S. 1–8.

[111] J. Liu, D. Xu, J. Hyyppa und Y. Liang. "A Survey of Applications with combined BIM and 3D Laser Scanning in the Life Cycle of Buildings". In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* (2021), S. 1–1. DOI: 10.1109/JSTARS.2021.3068796.

[112] S. Lloyd. "Least Squares Quantization in PCM". In: *IEEE Transactions on Information Theory* 28.2 (1982), S. 129–137. DOI: 10.1109/TIT.1982.1056489.

[113] W. E. Lorensen und H. E. Cline. "Marching Cubes: A high Resolution 3D Surface Construction Algorithm." In: *SIGGRAPH*. Hrsg. von M. C. Stone. ACM, 1987, S. 163–169. ISBN: 0-89791-227-6. DOI: 10.1145/37401.37422.

[114] D. G. Lowe. "Distinctive Image Features from scale-invariant Keypoints". In: *International journal of computer vision* 60.2 (2004), S. 91–110.

[115] A. Lucas. "Ising Formulations of many NP Problems". In: *Frontiers in Physics* 2 (2014), S. 5.

[116] S. Luo und W. Hu. "Differentiable Manifold Reconstruction for Point Cloud Denoising". In: *Proceedings of the 28th ACM International Conference on Multimedia*. MM '20. Seattle, WA, USA: Association for Computing Machinery, 2020, 1330–1338. ISBN: 9781450379885. DOI: 10.1145/3394171.3413727.

[117] W. Ma und J.-P. Kruth. "NURBS Curve and Surface Fitting for Reverse Engineering". In: *The International Journal of Advanced Manufacturing Technology* 14.12 (1998), S. 918–927.

[118] D. W. Marquardt. "An Algorithm for Least-Squares Estimation of nonlinear Parameters". In: *Journal of the society for Industrial and Applied Mathematics* 11.2 (1963), S. 431–441.

[119] E. J. McCluskey Jr. "Minimization of Boolean Functions". In: *Bell system technical Journal* 35.6 (1956), S. 1417–1444.

[120] T. M. Mitchell. *Machine Learning*. New York: McGraw-Hill, 1997. ISBN: 978-0-07-042807-2.

[121] C. Moenning und N. A. Dodgson. "A new Point Cloud Simplification Algorithm". In: *Proc. Int. Conf. on Visualization, Imaging and Image Processing*. 2003, S. 1027–1033.

[122] C. Moenning und N. A. Dodgson. "Fast Marching farthest Point Sampling". In: *Eurographics 2003 - Posters*. Eurographics Association, 2003. DOI: 10.2312/egp.20031024.

[123] M. Muja und D. G. Lowe. "Fast approximate nearest Neighbors with automatic Algorithm Configuration". In: *VISAPP (1)* 2.331-340 (2009), S. 2.

[124] Y. Nakamura, T. Matsuura, K. Satoh und Y. Ohta. "Occlusion detectable Stereo-Occlusion Patterns in Camera Matrix". In: *Proceedings CVPR IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE. 1996, S. 371–378.

[125] L. Nan und P. Wonka. "PolyFit: Polygonal Surface Reconstruction from Point Clouds". In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, S. 2372–2380. DOI: 10.1109/ICCV.2017.258.

[126] A. Y. Ng und M. I. Jordan. "On discriminative vs. generative Classifiers: A Comparison of Logistic Regression and Naive Bayes". In: *Advances in neural information processing systems*. 2002, S. 841–848.

[127] M. Nießner, M. Zollhöfer, S. Izadi und M. Stamminger. "Real-time 3D Reconstruction at Scale using Voxel Hashing". In: *ACM Transactions on Graphics (ToG)* 32.6 (2013), S. 1–11.

[128] D. Nister und H. Stewenius. "Scalable Recognition with a Vocabulary Tree". In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*. Bd. 2. Ieee. 2006, S. 2161–2168.

[129] B. Nourse, D. Hakala, R. Hillyard und P. Malraison. "Natural Quadrics in Mechanical Design". In: *Proceedings of the AUTOFACT West Conference*. Bd. 1. Anaheim, CA, USA: Association for Computing Machinery, 1980, 363–378.

[130] J. O'Rourke. "Polygon Decomposition and Switching Function Minimization". In: *Computer Graphics and Image Processing* 18.4 (1982), S. 382 –391.

[131] S. Ochmann, R. Vock, R. Wessel und R. Klein. "Automatic Reconstruction of parametric Building Models from Indoor Point Clouds". In: *Computers Graphics* 54 (2016). Special Issue on CAD/Graphics 2015, S. 94–103. ISSN: 0097-8493. DOI: 10.1016/j.cag.2015.07.008.

[132] I. Olkin und F. Pukelsheim. "The Distance between two random Vectors with given Dispersion Matrices". In: *Linear Algebra and its Applications* 48 (1982), S. 257 –263. ISSN: 0024-3795. DOI: 10.1016/0024-3795(82)90112-4.

[133] S. Osher und R. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Applied Mathematical Sciences. Springer New York, 2006. ISBN: 9780387227467.

[134] B. M. Ozyildirim und M. Kiran. *Do Optimization Methods in Deep Learning Applications matter?* 2020. arXiv: 2002.12642 [cs.LG].

[135] J. Pan, S. Chitta und D. Manocha. "FCL: A general purpose Library for Collision and Proximity Queries". In: *2012 IEEE International Conference on Robotics and Automation*. 2012, S. 3859–3866. DOI: 10.1109/ICRA.2012.6225337.

[136] M. Pauly, M. Gross und L. P. Kobbelt. "Efficient Simplification of point-sampled Surfaces". In: *IEEE Visualization, 2002. VIS 2002.* IEEE. 2002, S. 163–170.

[137] K. Pearson. "LIII. On Lines and Planes of closest Fit to Systems of Points in Space". In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2.11 (1901), S. 559–572.

[138] L. Piegl und W. Tiller. *The NURBS Book*. second. New York, NY, USA: Springer-Verlag, 1996.

[139] M. Pilz und H. A. Kamel. "Creation and Boundary Evaluation of CSG-Models". In: *Engineering with computers* 5.2 (1989), S. 105–118.

[140] F. Pomerleau, F. Colas und R. Siegwart. "A Review of Point Cloud Registration Algorithms for mobile Robotics". In: *Foundations and Trends in Robotics* 4.1 (2015), S. 1–104.

[141] C. R. Qi, H. Su, K. Mo und L. J. Guibas. "PointNet: Deep Learning on Point Sets for 3d Classification and Segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, S. 652–660.

[142] C. R. Qi, L. Yi, H. Su und L. J. Guibas. "PointNet++: Deep hierarchical Feature Learning on Point Sets in a metric Space". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Long Beach, California, USA: Curran Associates Inc., 2017, 5105–5114. ISBN: 9781510860964.

[143] W. V. Quine. "The Problem of Simplifying Truth Functions". In: *American Math. Monthly* 59.8 (1952), S. 521–531.

[144] M.-J. Rakotosaona, V. La Barbera, P. Guerrero, N. J. Mitra und M. Ovsjanikov. "PointCleanNet: Learning to denoise and remove Outliers from dense Point Clouds". In: *Computer Graphics Forum* 39.1 (2020), S. 185–203. DOI: 10.1111/cgf.13753.

[145] S. Ramaswamy, R. Rastogi und K. Shim. "Efficient Algorithms for Mining Outliers from large Data Sets". In: *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*. 2000, S. 427–438.

[146] A. Requicha. *Mathematical Models of Rigid Solid Objects (TM-28)*. Techn. Ber. Production Automation Project, University of Rochester, 1977.

[147] A. Requicha. "Representations for Rigid Solids: Theory, Methods, and Systems". In: *ACM Comput. Surv.* 12.4 (Dez. 1980), 437–464. ISSN: 0360-0300. DOI: 10.1145/356827.356833.

[148] A. Requicha und R. Tilove. *Mathematical Foundations of Constructive Solid Geometry: General Topology of Closed Regular Sets*. Techn. Ber. Production Automation Project, University of Rochester, 1978.

[149] A. Requicha und H. B. Voelcker. "Solid Modeling: A historical Summary and contemporary Assessment". In: *IEEE Computer Graphics and Applications* 2.2 (1982), S. 9–24. DOI: 10.1109/MCG.1982.1674149.

[150] A. Ricci. "A Constructive Geometry for Computer Graphics". In: *The Computer Journal* 16.2 (1973), S. 157–160.

[151] F. Rosenblatt. *The Perceptron, a Perceiving and Recognizing Automaton Project Para*. Report: Cornell Aeronautical Laboratory. Cornell Aeronautical Laboratory, 1957.

[152] D. A. Ross, J. Lim, R.-S. Lin und M.-H. Yang. "Incremental Learning for robust Visual Tracking". In: *International journal of computer vision* 77.1-3 (2008), S. 125–141.

[153] J. Rossignac. "Ordered Boolean List (OBL): Reducing the Footprint for Evaluating Boolean Expressions". In: *IEEE Transactions on Visualization and Computer Graphics* 17.9 (2011), S. 1337–1351.

[154] S. T. Roweis und L. K. Saul. "Nonlinear Dimensionality Reduction by locally linear Embedding". In: *science* 290.5500 (2000), S. 2323–2326.

[155] D. E. Rumelhart, G. E. Hinton und R. J. Williams. "Learning Representations by Back-Propagating Errors". In: *nature* 323.6088 (1986), S. 533–536.

[156] R. B. Rusu, Z. C. Marton, N. Blodow, M. Dolha und M. Beetz. "Towards 3D Point Cloud based Object Maps for Household Environments". In: *Robotics and Autonomous Systems* 56.11 (2008), S. 927–941.

[157] R. Schnabel, P. Degener und R. Klein. "Completion and Reconstruction with Primitive Shapes". In: *Computer Graphics Forum (Proc. of Eurographics)* 28.2 (März 2009), S. 503–512.

[158] R. Schnabel, R. Wahl und R. Klein. "Efficient RANSAC for Point-Cloud Shape Detection". In: *Computer Graphics Forum* 26.2 (Juni 2007), S. 214–226.

[159] U. Schneider und D. Werner. *Taschenbuch der Informatik*. 6. Aufl. Fachbuchverlag Leipzig, 2007.

[160] B. Schölkopf, A. Smola und K.-R. Müller. "Kernel Principal Component Analysis". In: *International conference on artificial neural networks*. Springer. 1997, S. 583–588.

[161] J. L. Schönberger, E. Zheng, J.-M. Frahm und M. Pollefeys. "Pixelwise View Selection for unstructured Multi-View Stereo". In: *European Conference on Computer Vision*. Springer. 2016, S. 501–518.

[162] J. L. Schönberger und J.-M. Frahm. "Structure-from-Motion Revisited". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[163] E. Schubert, S. Hess und K. Morik. "The Relationship of DBSCAN to Matrix Factorization and Spectral Clustering." In: *LWDA*. 2018, S. 330–334.

[164] E. Schubert, J. Sander, M. Ester, H. P. Kriegel und X. Xu. "DBSCAN revisited, revisited: why and how you should (still) use DBSCAN". In: *ACM Transactions on Database Systems (TODS)* 42.3 (2017), S. 1–21.

[165] K. Schwab. *Die vierte industrielle Revolution*. Pantheon Verlag, 2016.

[166] T. W. Sederberg, J. Zheng, A. Bakenov und A. Nasri. "T-Splines and T-NURCCs". In: *ACM Trans. Graph.* 22.3 (Juli 2003), 477–484. ISSN: 0730-0301. DOI: 10.1145/882262.882295.

[167] L. Shapira, A. Shamir und D. Cohen-Or. "Consistent Mesh Partitioning and Skeletonisation using the Shape Diameter Function". In: *The Visual Computer* 24.4 (2008), 249–259.

[168] V. Shapiro. "Semi-analytic Geometry with R-Functions". In: *Acta Numerica* 16 (2007), S. 239 –303.

[169] V. Shapiro. "A convex Deficiency Tree Algorithm for curved Polygons". In: *International Journal of Computational Geometry & Applications* 11.02 (2001), S. 215–238.

[170] V. Shapiro. "Chapter 20 - Solid Modeling". In: *Handbook of Computer Aided Geometric Design*. Hrsg. von G. Farin, J. Hoschek und M.-S. Kim. Amsterdam: North-Holland, 2002, S. 473 –518. ISBN: 978-0-444-51104-1. DOI: 10.1016/B978-044451104-1/50021-6.

[171] V. Shapiro. *Theory of R-functions and Applications: A Primer*. Techn. Ber. Cornell University, 1991.

[172] V. Shapiro und D. L. Vossler. "Construction and Optimization of CSG Representations". In: *Computer-Aided Design* 23.1 (1991), S. 4–20. ISSN: 00104485. DOI: 10.1016/0010-4485(91)90077-A.

[173] V. Shapiro und D. L. Vossler. "Separation for Boundary to CSG Conversion". In: *ACM Trans. Graph.* 12.1 (Jan. 1993), 35–55. ISSN: 0730-0301. DOI: 10.1145/169728.169723.

[174] G. Sharma, R. Goyal, D. Liu, E. Kalogerakis und S. Maji. "CSGNet: Neural Shape Parser for Constructive Solid Geometry". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[175] G. Sharma, D. Liu, S. Maji, E. Kalogerakis, S. Chaudhuri und R. Měch. "ParSeNet: A parametric Surface Fitting Network for 3D Point Clouds". In: *Computer Vision – ECCV 2020*. Hrsg. von A. Vedaldi, H. Bischof, T. Brox und J.-M. Frahm. Cham: Springer International Publishing, 2020, S. 261–276. ISBN: 978-3-030-58571-6.

[176] C.-H. Shen, H. Fu, K. Chen und S.-M. Hu. "Structure Recovery by Part Assembly". In: *ACM Transactions on Graphics (TOG)* 31.6 (2012), S. 1–11.

[177] B.-Q. Shi, J. Liang und Q. Liu. "Adaptive Simplification of Point Cloud using k-Means Clustering". In: *Computer-Aided Design* 43.8 (2011), S. 910–922.

[178] S. Silva, P.-A. Fayolle, J. Vincent, G. Pauron, C. Rosenberger und C. Toinard. "Evolutionary Computation Approaches for Shape Modelling and Fitting". In: *Portuguese Conference on Artificial Intelligence*. Springer. 2005, S. 144–155.

[179] D. Smirnov, M. Fisher, V. G. Kim, R. Zhang und J. Solomon. "Deep parametric Shape Predictions using Distance Fields". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, S. 561–570.

[180] M. Sokovic und J. Kopac. "RE (Reverse Engineering) as necessary Phase by rapid Product Development". In: *Journal of Materials Processing Technology* 175.1 (2006), S. 398 –403. ISSN: 0924-0136. DOI: 10.1016/j.jmatprotec.2005.04.047.

[181] C.-Y. Sun, Q.-F. Zou, X. Tong und Y. Liu. "Learning adaptive hierarchical Cuboid Abstractions of 3D Shape Collections". In: *ACM Trans. Graph.* 38.6 (Nov. 2019). ISSN: 0730-0301. DOI: 10.1145/3355089.3356529.

[182] M. Sung, V. G. Kim, R. Angst und L. Guibas. "Data-Driven Structural Priors for Shape Completion". In: *ACM Trans. Graph.* 34.6 (Okt. 2015). ISSN: 0730-0301. DOI: 10.1145/2816795.2818094.

[183] H. Süße und E. Rodner. "Geometrie der Abbildungsprozesse". In: *Bildverarbeitung und Objekterkennung: Computer Vision in Industrie und Medizin*. Wiesbaden: Springer Fachmedien Wiesbaden, 2014, S. 293–337. ISBN: 978-3-8348-2606-0. DOI: 10.1007/978-3-8348-2606-0_14.

[184] R. S. Sutton und A. G. Barto. *Reinforcement Learning: An Introduction*. Second. The MIT Press, 2018.

[185] R. Tarjan. "Depth-first Search and linear Graph Algorithms". In: *SIAM Journal on Computing* 1.2 (1972), S. 146–160. DOI: 10.1137/0201010. eprint: 10.1137/0201010.

[186] A. Tarrida. *Affine Maps, Euclidean Motions and Quadrics*. Springer Undergraduate Mathematics Series. Springer London, 2011. ISBN: 9780857297105. DOI: 10.1007/978-0-85729-710-5.

[187] M. Teschner, B. Heidelberger, M. Müller, D. Pomerantes und M. H. Gross. "Optimized Spatial Hashing for Collision Detection of deformable Objects." In: *Vmv*. Bd. 3. 2003, S. 47–54.

[188] Y. Tian, A. Luo, X. Sun, K. Ellis, W. T. Freeman, J. B. Tenenbaum und J. Wu. "Learning to infer and execute 3D Shape Programs". In: *International Conference on Learning Representations*. 2019.

[189] R. B. Tilove. "A Null-Object Detection Algorithm for Constructive Solid Geometry". In: *Commun. ACM* 27.7 (Juli 1984), S. 684–694. ISSN: 0001-0782. DOI: 10.1145/358105.358195.

[190] O. Van Kaick, H. Zhang, G. Hamarneh und D. Cohen-Or. "A Survey on Shape Correspondence". In: *Computer Graphics Forum*. Bd. 30. Wiley Online Library. 2011, S. 1681–1707.

[191] W. Vinci und D. A. Lidar. "Non-stoquastic Hamiltonians in Quantum Annealing via geometric Phases". In: *npj Quantum Information* 3.1 (2017). ISSN: 2056-6387. DOI: 10.1038/s41534-017-0037-z.

[192] U. Von Luxburg. "A Tutorial on Spectral Clustering". In: *Statistics and computing* 17.4 (2007), S. 395–416.

[193] D. Wagner und F. Wagner. "Between Min Cut and Graph Bisection". In: *Mathematical Foundations of Computer Science 1993*. Hrsg. von A. M. Borzyszkowski und S. Sokołowski. Berlin, Heidelberg: Springer Berlin Heidelberg, 1993, S. 744–750. ISBN: 978-3-540-47927-7.

[194] F. Wang, D. Zheng, J. Decker, X. Wu, G. M. Essertel und T. Rompf. "Demystifying Differentiable Programming: Shift/reset the penultimate Backpropagator". In: *Proceedings of the ACM on Programming Languages* 3.ICFP (2019), S. 1–31.

[195] D. Weiss. "Geometry-based structural Optimization on CAD Specification Trees". Diss. ETH Zurich, 2009.

[196] H. Weyl. "Elementare Theorie der konvexen Polyeder". In: *Commentarii Mathematici Helvetici* 7.1 (1934), S. 290–306.

[197] D. H. Wolpert. "The Lack of A Priori Distinctions between Learning Algorithms". In: *Neural Computation* 8.7 (1996), S. 1341–1390. DOI: 10.1162/neco.1996.8.7.1341.

[198] Q. Wu, K. Xu und J. Wang. "Constructing 3D CSG Models from 3D Raw Point Clouds". In: *Computer Graphics Forum* 37.5 (2018), S. 221–232. DOI: 10.1111/cgf.13504.

[199] Z. Xiang und R. A. Plastock. *Computergrafik: Einführung in die theoretischen Grundlagen*. mitp, 2007.

[200] J. Xiao und Y. Furukawa. "Reconstructing the World's Museums". In: *Computer Vision – ECCV 2012*. Hrsg. von A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato und C. Schmid. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, S. 668–681. ISBN: 978-3-642-33718-5.

[201] A. Yao. "Applications of 3D Scanning and Reverse Engineering Techniques for Quality Control of Quick Response Products". In: *The International Journal of Advanced Manufacturing Technology* 26.11-12 (2004), S. 1284–1288. DOI: 10.1007/s00170-004-2116-5.

[202] S. Zbinden, A. Bärtschi, H. Djidjev und S. Eidenbenz. "Embedding Algorithms for Quantum Annealers with Chimera and Pegasus Connection Topologies". In: *High Performance Computing*. Hrsg. von P. Sadayappan, B. L. Chamberlain, G. Juckeland und H. Ltaief. Cham: Springer International Publishing, 2020, S. 187–206. ISBN: 978-3-030-50743-5.

[203] L. Zelnik-Manor und P. Perona. "Self-tuning Spectral Clustering". In: *Proceedings of the 17th International Conference on Neural Information Processing Systems*. NIPS'04. Vancouver, British Columbia, Canada: MIT Press, 2004, 1601–1608.

[204] L. Zhang, B. Curless und S. M. Seitz. "Rapid Shape Acquisition using Color structured Light and multi-pass Dynamic Programming". In: *Proceedings. First International Symposium on 3D Data Processing Visualization and Transmission*. IEEE. 2002, S. 24–36.

[205] Y. Zhao, Z. Nasrullah und Z. Li. "PyOD: A Python Toolbox for scalable Outlier Detection". In: *Journal of Machine Learning Research* 20.96 (2019), S. 1–7.

[206] X. Zhong, Y. Wu, D. Wang, Q. Wang, C. Xu und F. Gao. *Generating large convex Polytopes directly on Point Clouds*. 2020. arXiv: 2010.08744 [cs.RO].

[207] Y. Zhong und W. Ma. "An Approach for Solid Modelling in a Virtual Reality Environment". In: *Virtual and Augmented Reality Applications in Manufacturing*. Springer, 2004, S. 15–42.

[208] H. Zhu, B. Guo, K. Zou, Y. Li, K.-V. Yuen, L. Mihaylova und H. Leung. "A Review of Point Set Registration: From pairwise Registration to groupwise Registration". In: *Sensors* 19.5 (2019), S. 1191.

[209] C. Zou, E. Yumer, J. Yang, D. Ceylan und D. Hoiem. "3D-PRNN: Generating Shape Primitives with Recurrent Neural Networks". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, S. 900–909.

Anhang

A. Parameter für die Konstruktion Konvexer Polytope

In diesem Kapitel werden die gewählten Parameter des EAs für die *Konstruktion konvexer Polytope*, wie sie in Tabelle 4.1 aufgeführt sind, für jedes Modell dargestellt. Die Parameter für die Durchläufe ohne schwach-konvexe Segmentierung sind in den Tabellen A.1 und A.2 aufgeführt; die für die Durchläufe mit schwach-konvexer Segmentierung in Tabelle A.3. Modell und segmentierungsunabhängige Parameter sind in Tabelle A.4 dargestellt. Für die Parameter p_{ers}, p_{mod}, p_{hin} und p_{ent} wurde jeweils der Wert $0,15$ gewählt, unabhängig von Modell und schwach-konvexer Segmentierung.

Parameter	**M1**	**M2**	**M3**	**M4**	**M5**	**M6**
n_{pop}	150	150	150	150	150	150
$n_{e\,max}$	6	6	6	6	6	6
$n_{p\,max}$	10	32	10	10	100	100
n_{iter}	100	2000	100	100	500	750
m_{iter}	50	1000	50	50	250	375
p_{qu}	$0,5$	$1,0$	$0,5$	$0,5$	$1,0$	$1,0$
α	$1,0$	$2,0$	$1,0$	$1,0$	$1,0$	$1,0$
γ	$0,01$	$0,0$	$0,01$	$0,01$	$0,01$	$0,01$
$V_{G\,min}$	$0,9$	$0,8$	$0,9$	$0,9$	$0,7$	$0,9$

Tabelle A.1.: Parameter des EAs zur *Konstruktion konvexer Polytope*. Modelle M1-M6, ohne schwach-konvexe Segmentierung.

Parameter	**M7**	**M8**	**M9**	**M10**	**M11**	**M12**
n_{pop}	100	150	100	200	100	150
$n_{e\,max}$	6	6	8	10	8	6
$n_{p\,max}$	5	20	15	20	1	20
n_{iter}	100	200	4000	3000	1000	3000
m_{iter}	50	100	2000	2000	500	1000
p_{qu}	0,0	0,0	0,0	0,0	0,0	0,5
α	1,0	1,0	1,0	1,0	1,0	1,0
γ	0,01	0,01	0,1	0,0	0,1	0,0
$V_{G\,min}$	0,7	0,9	0,9	0,8	0,7	0,8

Tabelle A.2.: Parameter des EAs zur *Konstruktion konvexer Polytope*. Modelle M7-M12, ohne schwach-konvexe Segmentierung.

Parameter	**M5**	**M6**	**M7**	**M9**	**M10**	**M11**	**M12**
n_{pop}	100	100	100	100	200	100	100
$n_{e\,max}$	8	6	6	8	10	8	6
$n_{p\,max}$	1	2	1	2	20	3	1
n_{iter}	200	300	200	400	3000	500	10
m_{iter}	100	300	100	200	2000	250	10
p_{qu}	1,0	1,0	0,0	0,0	0,0	0,0	0,5
α	1,0	1,0	1,0	1,0	1,0	1,0	1,0
γ	0,1	0,0	0,0	0,1	0,0	0,1	0,0
$V_{G\,min}$	0,5	0,8	0,5	0,9	0,8	0,7	0,8

Tabelle A.3.: Parameter des EAs zur *Konstruktion konvexer Polytope*. Modelle M5-M12 (ohne M8), mit schwach-konvexer Segmentierung.

Parameter	n_{var}	V_{max}	β	ϵ	p_{mut}	p_{rek}	p_{neu}
Wert	1	1	1	0,02	0,4	0,4	0,4

Tabelle A.4.: Parameter des EAs zur *Konstruktion konvexer Polytope*, unabhängig von Modell und schwach-konvexer Segmentierung.