Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Leonard Opitz

Generierung von dreidimensionalen Darstellungen von Gebäuden aus Grundrissen

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Leonard Opitz

Generierung von dreidimensionalen Darstellungen von Gebäuden aus Grundrissen

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

im Studiengang Bachelor of Science Angewandte Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Philipp Jenke
Zweitgutachter: Prof. Dr.-Ing. Andreas Meisel

Eingereicht am: 8. September 2016

Leonard Opitz

Thema der Arbeit
Generierung von dreidimensionalen Darstellungen von Gebäuden aus Grundrissen

Stichworte
Grundriss Analyse, Generierung dreidimensionaler Darstellungen

Kurzzusammenfassung
Dieses Dokument betrachtet Möglichkeiten zur Generierung von dreidimensionalen Darstellungen aus zweidimensionalen Gebäudegrundrissen. Dabei wird ein Verfahren zur Strukturanalyse von auf Vektordaten basierten Grundrissen erläutert und die Einteilung der architektonischen Elemente (Wände, Türen und Fenster) in standardisiert darzustellende viereckige Abschnitte algorithmisch umgesetzt.

Das entwickelte System nimmt die Vektordaten im *drawing interchange file* Format entgegen und stellt die Analyseergebnisse mithilfe des *computergraphics* Frameworks der HAW unter Verwendung von Polygonnetzen dar.
Leonard Opitz

Title of the paper
Generating three dimensional representations of buildings from groundplans

Keywords
groundplan analysis, generating three dimensional representations

Abstract
This document examines methods for generating three dimensional representations from two dimensional groundplans. It defines a technique to analyse the structure of vector based groundplans and classifies their architectural elements (walls, windows and doors) into squares. These squares are then displayed consistently.

The developed system uses vector data of groundplans in the *drawing interchange file* format as input. It visually renders results based on polygon meshes through the *computergraphics* framework of the HAW.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

1.1 Motivation

Unter dem Sammelbegriff *Smart Home* wird im Allgemeinen ein umfassend mit Sensorik und Regeltechnik ausgestattes Wohnhaus verstanden. *Smart Home* Anwendungen dienen der Überwachung und Steuerung dieser Sensorik und Regeltechnik und der Verbesserung der allgemeinen Lebensqualität (Kazmierzak, 2015, vgl. Abstract).

Benutzeroberflächen von *Smart Home* Anwendungen können durch eine dreidimensionale Darstellung der Umgebung in ihrer intuitiven Bedienbarkeit unterstützt werden. Die Möglichkeit, digitale Grundrisse aus der Planungsphase eines Gebäudes direkt für *Smart Home* Anwendungen nutzbar zu machen, ist die Motivation dieser Bachelorarbeit.

1.2 Zielsetzung

Die Hautpaufgabenstellung dieser Ausarbeitung besteht darin, die vom Architekten erstellten Grundrisse in verwendbare, dreidimensionale Darstellungen für Java™ basierte *Smart Home* Anwendungen umzuwandeln.

Gängige CAD-Anwendungen, wie sie in Architekturbüros eingesetzt werden, können die darin angefertigten Pläne in das allgemeine und offene *Drawing Interchange File* Format(DXF) exportieren.

Ziel dieser Bachelorarbeit ist die Weiterverwendung dieser exportierten Grundrissdaten zur Generierung von dreidimensionalen Darstellungen von Gebäuden. Diese werden zur Veranschaulichung der Ergebnisse durch des Computergraphics Framework der HAW dargestellt.

Im Rahmen dieser Bachelorarbeit soll eine Anwendung[1] entwickelt werden, die auf Basis einer DXF Datei und Konfigurationsangaben in definierter XML-Form eine dreidimensionale Darstellung des zugrunde liegenden Grundrisses erzeugt. Die dreidimensionale Darstellung basiert dabei auf Polygongitterdaten.

[1] nachfolgend der Einfachheit halber als "das System" bezeichnet

1.3 Abgrenzung

In dieser Arbeit wird kein vollständiges Gebäude generiert. Es wird keine exakte Darstellung komplexer architektonischer Elemente umgesetzt. Vertikale Planmaße, wenn vorhanden, werden nicht ausgewertet. Die Höhe der Darstellung ergibt sich aus konfigurierbaren Werten. Die resultierenden Polygonnetze beschränken sich in der Höhe auf einzelne Stockwerke. Bei der Darstellung werden ausschließlich Wände, Türen und Fenster berücksichtigt. Möbel und Treppen werden nicht dargestellt.

1.4 Überblick über den Aufbau der Arbeit

Das Kapitel *Grundlagen* führt in die verwendeten Bibliotheken sowie in die Begriffswelt des Systems ein. Ausdrücke, die im Verlauf der Arbeit immer wieder vorkommen, werden darin vorab erklärt.

Im Kapitel *Problembetrachtung* werden die einzelnen Bereiche der Aufgabenstellung analysiert und mögliche Lösungsansätze diskutiert. Nacheinander werden die einzelnen Bereiche *Datenimport*, *Datenanalyse*, *Datenbewertung* und *Ergebnisdarstellung* abgehandelt und die Schwierigkeiten in diesen Bereichen vorgestellt.

Im Kapitel *Lösungskonzept* folgen dann die tatsächlich verwendeten Lösungen für die zuvor erläuterten Problemstellungen. In einzelnen Abschnitten analog zur Einteilung des vorangegangenen Kapitels werden die verwendeten mathematischen Methoden und Algorithmen im einzelnen vorgestellt und erläutert.

Das Kapitel *Implementierung* stellt die Entwicklungsarbeit vor und erläutert, welche Klassen in der Projektstruktur *smarthomevis.groundplan*[2] die in den vorangegangenen Kapiteln erläuterten Konzepte implementieren. Ausserdem wird der Inhalt der XML Konfigurationsdatei beschrieben.

Im Kapitel *Evaluation* werden die für die Entwicklung verwendeten Testfälle vorgestellt sowie Sonderfälle erläutert, die vom System nicht korrekt umgesetzt und dargestellt werden können. Wenn für diese Grenzbereiche theoretische Lösungen vorhanden sind, werden diese im Anschluss ebenfalls kurz umrissen.

In der *Schlussbetrachtung* folgen dann noch die Zusammenfassung der Arbeit sowie ein Fazit, und es werden die Möglichkeiten der Weiterentwicklung aufgezeigt, die für das entwickelte System denkbar sind.

[2] vollständiger Pfad im Projekt: [smart_home_visualization\src\main\java\smarthomevis\groundplan]

2 Grundlagen

2.1 Drawing Interchange File Format

Das *Drawing Interchange File* Format (DXF Format) ist die zugrunde liegende Datenquelle dieser Ausarbeitung. Das DXF Format ist von *AutoDesk* als Zeichnungsaustauschformat entwickelt worden, mit dem Daten zwischen verschiedenen Anwendungen übertragen werden können. DXF Dateien werden häufig verwendet, um Zeichnungsdaten zwischen CAD-Programmen auszutauschen (Autodesk, 2016, vgl.). Das DXF Format bietet sich für diese Arbeit an, da es von vielen Architekturprogrammen als vektorbasiertes Exportformat unterstützt wird.

Darstellungen im DXF basieren auf Definitionen von grafischen Primitiven (Linien, Kreise und Kurven) in einem dreidimensionalen Koordinatensystem. Es wird als Ausgangsformat verwendet, da es von vielen gängigen CAD- und Architekturprogrammen als Exportformat unterstützt wird.

2.1.1 Die Struktur von DXF Dateien

Die Grundstruktur einer DXF Datei ist in Sektionen eingeteilt. Diese sind *Header*, *Classes*, *Tables*, *Blocks*, *Entities* und *Objects*.

Das DXF Format unterstützt eine Reihe von Entitätentypen, die für die Darstellung verschiedener Elemente einer Grafik verwendet werden können. Diese Ausarbeitung stützt sich auf die Verwendung von Liniendaten und Ebenendefinitionen.

2.2 computergraphics Framework der HAW

Für die Darstellung der generierten Grundrisse wird das *computergraphics Framework* der HAW verwendet. Das *computergraphics Framework* (*cg Framework*) unterstützt verschiedene Formen der Darstellung. Das im Rahmen dieser Arbeit entwickelte System produziert Polygongitter-Darstellungen und lässt diese vom *cg-Framework* rendern.

Die Architektur des entwickelten Systems soll jedoch auch einen einfachen Austausch der zur Darstellung verwendeten Bibliothek ermöglichen.

2.2.1 Ergebnisdarstellung im computergraphics Framework der HAW

Das entwickelte System erstellt als Ergebnis dreidimensionale Polygonnetze der importierten Grundrissdaten. Um die Ergebnisse der hier entwickelten Algorithmen betrachten und beurteilen zu können, werden diese Polygonnetze an das *cg Framework* der HAW zur Renderung und Darstellung übergeben.

2.2.2 Darstellungen aus Polygonen

Als Polygon werden im Rahmen dieser Arbeit die dreieckigen Flächen bezeichnet, aus denen die Darstellungselemente konstruiert werden. Die einzelnen Polygone werden zu Polygonnetzen zusammengesetzt und ergeben gemeinsam die Gesamtdarstellung.

Das *computergraphics* Framework bietet für die Konstruktion von Polygonnetzen die Klassen *TriangleMesh*[1] und *Vertex*[2] an. Vektorkoordinaten werden zu *Vertex* Objekten umgewandelt und einem *TriangleMesh* Objekt hinzugefügt. Dem *TriangleMesh* Objekt können dann dreieckige Konstellationen aus den hinzugefügten *Vertex* Objekten mitgeteilt werden. Auf diese Weise wird die Gesamtdarstellung nach und nach aus einzelnen dreieckigen Flächen zusammengesetzt.

2.2.3 JOGL Projekt

Das *JOGL* Projekt ist eine Java™ Anbindung an die *OpenGL* API und bietet Hardware-Unterstützung für 3D-Anwendungen. *JOGL* unterstützt dabei die Einbindung in *AWT*, *Swing* und *SWT* Widgets sowie spezifische Oberflächen Toolkits die die *NativeWindow* API einbinden (JogAmp.org, 2016, vgl.).

Das *computergraphics* Framework nutzt das *JOGL* Projekt zur Darstellung von dreidimensionalen Grafiken in einer *Swing* basierten Benutzeroberfläche. Diese wird als Anwendungsrahmen für die Darstellung der Ergebnisse des entwickelten Systems verwendet.

2.3 Begriffsdefinitionen

2.3.1 Linie

Als eine Linie wird hier die grafische Einheit betrachtet, die durch einen Anfangs- und Endpunkt in Form von Ortsvektoren definiert ist. Eine Linie wird im System durch den Datentyp *GPLine* repräsentiert.

[1] cgresearch.graphics.datastructures.trianglemesh.TriangleMesh
[2] cgresearch.graphics.datastructures.trianglemesh.Vertex

2.3.2 Wand

Eine Wand ist die Menge von Linien, die innerhalb eines Bauplans ein festes Mauerwerk ergeben, welches seine Richtung nicht ändert und nicht von einem Fenster oder einem Druchgang beziehungsweise einer Tür unterbrochen wird. Ein Teil der Aufgabe dieses Systems ist die Erkennung von Wänden auf Basis von Linien mit gemeinsamen Punktvektoren oder parallelen Richtungsvektoren.

2.3.3 Paarung

Als Paarung werden im Rahmen dieser Ausarbeitung zwei Linien bezeichnet, wenn die beiden Linien aufgrund bestimmter Kriterien in eine gegenseitige Beziehung gesetzt werden. Linien in einer Paarung liegen stets parallel zueinander, haben einen definierten Abstand und liegen auf gleicher Höhe, das heißt der Betrag der Projektion der einen Linie auf die andere ist nicht Null.

2.3.4 Solid

Als Solid wird im Rahmen dieser Arbeit ein abgeschlossenes Segment bezeichnet, das in der Regel ein einfaches Wandelement darstellt. Es zeichnet sich insbesondere dadurch aus, dass alle Seitenflächen sowie die Ober- und Unterseite bei der dreidimensionalen Darstellung mit Polygonflächen versehen werden. Eine Herausforderung dieser Arbeit besteht darin, Wände im Grundplan zu erkennen und diese in für die Darstellung geeignete Solids einzuteilen. Die Darstellung einer Wand wird immer durch eine oder mehrere Solids umgesetzt. Ein Solid wird im System durch den Datentyp *GPSolid* repräsentiert.

2.3.5 Basislinien

Als die *Basislinien* eines Solids werden diejenigen beiden Linien betrachten, die zusammen in einer Paarung auftreten und zur Bildung des Solids geführt haben. Sie werden für die Umsetzung der Darstellung als gesonderte Referenz, zusätzlich zur normalen Liste aller am Solid beteiligten Linien, in jeweiligen Solid gespeichert.

2.3.6 Multiline

Eine Multiline ist eine Linie, die zu mehreren anderen Linien benachbart ist. Sie kommt in mehr als einer Paarung vor.

3 Problembetrachtung

3.1 Datenimport

Für die Darstellung eines im DXF Format vorliegenden Gebäudegrundrisses müssen zunächst Daten aus einer Quelldatei in das System importiert werden.

Die in einer DXF Datei enthaltenen Daten sind in einer ausschließlich für die maschinelle Verarbeitung ausgelegten Struktur organisiert, die für den menschlichen Betrachter nur mit sehr großem Aufwand zu interpretieren wäre. Für den Import von Daten sind mehrere Ansätze möglich.

Informationen im DXF Format liegen als ASCII Zeichen vor. Sie sind mit einem selbst entwickelten Parser auswertbar. Dieser müsste jedoch eine hohe strukturelle Komplexität handhaben und würde daher in der Entwicklung des Systems einen großen Aufwand bedeuten.

Verschiedene Bibliotheken stellen diese Funktionalität zum Teil kostenlos zur Verfügung. Für die Einbindung in eine Java™ Software Umgebung bietet sich die frei verfügbare *Kabeja* Bibliothek an. *Kabeja* unterstützt das Parsen, Verarbeiten und Konvertieren von DXF Dokumenten.

3.1.1 DXF Import mit Kabeja

Kabeja besitzt eine gut dokumentierte Schnittstelle für den Zugriff auf die Inhalte einer DXF-Datei. Der Zugriff auf die Inhalte der DXF-Quelldateien erfolgt dabei mit einem dem *document object model* ähnlichen Syntax. Die einzelnen Ebenen, die in den verwendeten Grundrissen verschiedene Elementtypen enthalten, können iterativ durchlaufen, aber auch individuell über einen definierten Namen abgerufen werden. Über den Zugriff auf die Ebenen des Dokuments erhält man dann Entitäten. Die Entitäten einer DXF-Ebene sind stets vom selben Typ. Das Auslesen der einzelnen Elemente geschieht sequentiell.

3.1.2 Variierende Qualität der Ausgangsdateien

Die innere Struktur von Dateien im Drawing Exchange Format kann sehr unterschiedlich ausfallen.

Werden in der Ausgangsanwendung keine Ebenen oder Blöcke zur Trennung von Elementen verwendet, sind diese auch in der resultierenden DXF-Datei nicht gegeben.

Daher ist für eine korrekte Generierung von dreidimensionalen Gebäudedarstellungen in der Regel eine Bearbeitung der DXF-Datei mit einem (Vektor-) Grafikprogramm notwendig. Für die einfache Unterscheidung von Fenstern, Türen und Wänden sind diese Elemente auf getrennte DXF-Ebenen zu unterteilen. Diese Ebenen und die Eigenschaft ihres Inhalts müssen außerdem der Anwendung mitgeteilt werden.

Anforderungen an das DXF Dokument

Die Ausgangsdatei enthält einen zweidimensionalen Grundriss eines Gebäudes. Das in dieser Bachelorarbeit entwickelte System verwendet derzeit lediglich grafische Primitive vom Typ *Line*.

Um vom System auswertbar zu sein, müssen die Grundelemente eines Grundrisses durch parallele Linien dargestellt sein. Jede Linie ist durch die Angabe zweier Ortsvektoren als Endpunkte definiert.

Für die korrekte Darstellung von Fenstern, Türen und Wänden müssen diese auf einzelnen Ebenen liegen. Überlappungen dieser Elemente führen zu Fehlern in der dreidimensionalen Darstellung.

Keine Gruppierungen

Das DXF Format unterstützt die Gruppierung von grafischen Primitiven in Blöcken. Sämtliche Testgrundrisse, die zur Entwicklung des Systems verwendet wurden, machen jedoch von dieser Methode keinen Gebrauch. Daher werden für die Analyse der Grundrissdaten Blöcke vom entwickelten System nicht berücksichtigt.

Grafische Primitive liegen als ungruppierte Entitäten vor. Die Findung von logischen Einheiten muss durch das System erfolgen.

Entitäten ohne eindeutige Benennung

Entitäten im DXF Format haben keine eindeutige Bezeichnung. Dieser Umstand verhindert die Identifizierung grafischer Primitiver auf der Basis von menschenlesbaren Zeichenketten und eine einfache Bewertung der Systemresultate.

3.1.3 Zusatzinformationen im XML Format

Aufgrund der Individualität eines jeden Grundrisses müssen eine Reihe von Werten in einer Konfigurationsdatei individuell an den jeweiligen Plan anpassbar sein. Grundrisse können sich sowohl in der Skalierung als auch in den Ausmaßen stark unterscheiden. Durch eine höhere Anzahl von Räumen entsteht auch eine höhere Anzahl von Linien, sodass gewisse Schwellwerte entsprechend höher eingestellt werden müssen.

Die Beschränkung auf zweidimensionale Daten macht zusätzliche Informationsangaben für verschiedene Bereiche des Systems notwendig. Diese zusätzlichen Daten müssen dem System zusammen mit der DXF Datei in wohlstrukturierter Form übergeben werden.

Für diesen Zweck können verschiedene Formate Verwendung finden. Aufgrund der hohen Flexibilität bietet sich eine Strukturierung im XML Format an. Das XML Format ist wohldefiniert, lässt sich einfach erweitern und mithilfe von Methoden der *javax* Klassenbibliothek strukturiert auslesen.

Die Konfigurationsdatei enthält eine nachfolgend aufgeführte Reihe von erforderlichen Informationen, die für jeden Grundriss individuell einstellbar sein müssen. Darunter sind Angaben zu den im DXF Dokument enthaltenen relevanten Ebenen und den darauf enthaltenen Entitäten (siehe 3.1.2), außerdem nachfolgend erläuterte Werte für die dreidimensionale Darstellung zweidimensionaler Daten, für die Analyse notwendige Toleranzwerte (siehe 3.2.3) sowie Schwellwerte zur Strukturanalyse (siehe 4.3.1).

Höhenangaben

Die in das System importierten zweidimensionalen Grafikdaten enthalten keine Angaben zur Höhe der enthalten Elemente. Dabei ist zu berücksichtigen, dass für die Darstellung von Fenstern und Türen zusätzlich zur Wandhöhe weitere Angaben notwendig sind. Für Fenster muss die Höhe von Ober- und Unterkante der Fensteröffnung und für Türen die Öffnungshöhe definiert werden.

Skalierungsfaktor

Das DXF Format ist ein allgemeines Übertragungsformat, weshalb DXF Dateien von verschiedenen Anwendungen erstellt sein können. Die Handhabung von Maßen und Dimension der enthaltenen Daten kann daher stark variieren. Oft sind auch in den Quelldaten keinerlei Dimensionsangaben enthalten. Für die Erzeugung einer verwendbaren Darstellung ist jedoch erforderlich, dass Höhenangaben in zweckdienlichen Maßeinheiten erfolgen können. Diese Höhenangaben stehen in direkter Verbindung zur Skalierung von Zeichnungselementen. Pas-

sen Höhenangaben und Skalierungsfaktor nicht zueinander, ist eine fehlerhafte Darstellung der Grundrisse die Folge.

Da die Ausmaße der zugrunde liegenden Vektorgrafiken sehr stark variieren können, ist eine zuverlässige Methode zur Bestimmung der Skalierungsgröße erforderlich.

Denkbar wäre hier eine Schätzung der Gesamtdimension des Gebäudes oder der Ausmaße eines geeigneten Raumes. Diese Methode erscheint jedoch wenig zuverlässig.

Ein geeigneter Ansatzpunkt ist hierfür die Breite von Türlaibungen beziehungsweise die in Bauplänen eingezeichneten Maueröffnungsbreiten, da diese durch die Deutsche Industrie Norm festgelegt sind. Für die Maueröffnungsbreite von einflügeligen Türen sind vier verschiedene Breiten gängig. Durch die Vermessung einer durchschnittlich breiten Tür im verwendeten Grundriss lässt sich über diese Standardwerte auf eine sinnvolle Skalierungsgröße schliessen.

3.2 Datenanalyse

3.2.1 Identifizierung von Elementen der Quelldatei

Von Architekten erstellte Grundrisse enthalten eine Vielzahl verschiedener Informationen und können aufgrund des Fehlens von Standards stark in der Darstellung variieren (de las Heras u. a., 2015, vgl. Kap. 1, Absatz 2). In einem Plan können Maße definiert sein, müssen jedoch nicht zwingend vorliegen. In vielen Darstellungen sind zur Verdeutlichung der Raummaße beispielhaft Möbel eingezeichnet.

Wand-, Tür- und Fensterdarstellungen können stark variieren. Abhängig von der gewählten Software oder auch den Vorlieben des Architekten sind Wände mal durch einfache parallele Linien oder auch durch ausgefüllte schwarze, graue oder schraffierte Quadrate dargestellt. Fenster können durch Doppellinien, einfache Linien oder auch einfach Lücken dargestellt werden. Türen können als Lücken mit Bögen oder auch Doppellinien repräsentiert sein. Einige Beispiele für verschiedene Darstellungsarten sind in der Abbildung 3.1 enthalten.

Unterscheidung von Räumen und Wänden

Eine Kernanforderung an das System ist die fehlerfreie Unterscheidung von Räumen und Wänden. Die Darstellung eines Raumes in einem zweidimensionalen Grundriss unterscheidet sich in einigen Fällen nur im Abstand der Linien von der Darstellung einer Wand. Für das menschliche Auge ist die Unterscheidung meist eine leichte Übung. Eine Schraffur innerhalb von Festkörpern kann diese Aufgabe zusätzlich erleichtern.

Abbildung 3.1: Beispiele verschiedener Darstellungsformen von Grundrissen

Für einen Algorithmus ist diese Aufgabe jedoch weniger eindeutig zu lösen. Ein Raum definiert sich, ebenso wie ein Wandabschnitt, durch umschliessende Linien. Es ist ein allgemeingültiges Regelwerk erforderlich, das die eindeutige und fehlerfreie Zuteilung von einzelnen parallelen Linien zu Wänden ermöglicht.

Ein solches Regelwerk wird in de las Heras u.a. im ersten Absatz des zweiten Kapitels beschrieben (de las Heras u. a., 2013b, vgl.). Die Autoren stellen sechs Regeln für die Wanderkennung auf.

1. Wände sind durch parallele Linien dargestellt.
2. Sie verlaufen in orthogonalen Richtungen.
3. Wände sind von rechteckiger Form, gewöhnlich länger als dick.

4. Sie definieren die Struktur des Gebäudes; treten im Plan gleichmäßig verteilt auf.
5. Verschiedene Wandstärken werden zur Modellierung von außen oder innen liegenden Wänden verwendet.
6. Die Wände innerhalb eines Dokuments sind mit dem gleichen Muster gefüllt (schraffiert, gekachelt, gefüllt usw.).

Diese Annahmen können nicht als vollständige Liste von unumstößlichen Regeln für alle existierenden realen Grundrisse angesehen werden. So gibt es zum Beispiel Grundrisse mit diagonalen oder in Kurven verlaufenden Wänden, Gebäude mit gleicher Wandstärke für innen oder außen liegende Wände und so weiter (de las Heras u. a., 2013b, vgl. Anmerkungen zu den Regeln).

Wände können durch eine beliebige Anzahl von Linien zusammengesetzt sein

Eine zweidimensionale Wanddarstellung in Draufsicht kann bereits durch vier Linien vollständig erfolgen. Sind einer oder mehrere weitere Wandabschnitte an eine Wand angesetzt, reichen drei beziehungsweise zwei Linien für die Darstellung der Wand.

Grundrissdarstellungen im DXF Format können Wandabschnitte jedoch durch eine beliebige Anzahl von Entitäten umsetzen. So kann eine Linie eines Wandabschnitts, deren Richtung sich nicht ändert, mit einer beliebigen Anzahl von Entitäten in der DXF Datei repräsentiert werden. Eine Eingrenzung eines Abschnitts anhand einer Anzahl von Linien ist nicht zulässig.

Im folgenden werden zwei Linien, die parallel sind und die im korrekten Abstand der Wandstärke gegenüber liegen als *benachbart* bezeichet.

Fenster, Türen und Wände unterscheiden

Die Darstellung von Fenstern, Türen und Wänden in einem Grundriss, nachfolgend auch als Elemente bezeichnet, kann mit verschiedenen Methoden erfolgen. Das Fehlen einer Standardnotation in Grundrissen führt zu unterschiedlichen grafischen Gestaltungsweisen je nach Architekturbüro oder Land. Daher können Wände von Grundriss zu Grundriss sehr unterschiedlich gestaltet sein (de las Heras u. a., 2013a, vgl. Kap. 1 Absatz 3). Fensterlaibungen können in Bauplänen nur durch eine andere Farbgebung von Wänden unterscheidbar dargestellt werden. Ein Durchbruch für einen Türrahmen unterscheidet sich in einer zweidimensionalen Draufsicht nicht zwingend von einem Durchbruch für ein Fenster.

Durch diesen Umstand ist eine eindeutige Differenzierung verschiedener Elemente eines Bauplans nicht vollständig oder nicht allgemein für jede Gestaltungsform von Bauplänen auto-

matisierbar. Um diese verschiedenen Elemente sicher voneinander differenzieren zu können, muss daher eine deutliche Trennung vorgenommen werden. Die Sortierung der einzelnen Elemente auf jeweils eigene DXF-Ebenen bietet sich hier an. Durch diese Trennung können die Vektordaten der einzelnen Elemente bereits beim Import in das System entsprechend sortiert und gekennzeichnet werden.

3.2.2 Strukturanalyse

Anhand der in Abschnitt 3.2.1 genannten Regeln kann ein systematisches Vorgehen abgeleitet werden. Zunächst ist eine Gruppierung aller Linien mit gleicher Richtung notwendig. Mithilfe dieser Gruppen können dann Distanzen zwischen parallelen Linien ausgewertet werden.

Abstände zwischen Linien werden dafür in einem Histogramm gesammelt, sodass häufige Ansammlungen von Abständen für regelmäßige Distanzen zwischen parallelen Linien stehen (de las Heras u. a., 2013b, vgl. Kap. 2, Absatz 6).

Berücksichtigt werden sollen für die Ansammlung ausschließlich Linien, die eine gemeinsame Projektion besitzen, also diejenigen Linien, bei denen die Länge eines berechneten Projektionsvektors der einen Linie auf die andere einen Betrag größer Null hat.

Abstände werden dabei in individuell konfigurierbaren Intervallen zusammengefasst. Durch die Verwendung von Intervallen werden Abweichungen kompensiert, die durch die manuelle Erstellung von Wandabschnitten entstehen können (siehe Abschnitt 3.2.3, Seite 12).

Aus dem Histogramm werden dann diejenigen Abstandsgruppen ausgewählt, die einerseits innerhalb eines sinnvollen Bereichs einer Wandstärke von zehn bis fünfzig Zentimetern liegen und außerdem in ihrer Häufigkeit einen in der Konfigurationsdatei individuell anpassbaren Schwellwert überschreiten.

Diejenigen Paarungen von Linien, die diese Abstandskriterien erfüllen, können dann für die nachfolgende strukturelle Bewertung verwendet werden.

3.2.3 Toleranzen

Sind Linien eines Grundrisses nicht ausschließlich durch Hilfsmittel erstellt oder von Hand nachberarbeitet, können Abweichungen in den Koordinaten von benachbarten Elementen vorkommen. Bei der vergleichenden Betrachtung von Elementen eines Grundrisses sind diese Abweichungen daher in einem definierten Rahmen zu tolerieren.

Punkttoleranz

Die Punkte zweier Linien, die dieselbe Ecke beschreiben, können durch unterschiedliche, nur geringfügig in einzelnen Ordinaten im Dezimalbereich abweichende Ortsvektoren definiert

sein. Bei einem Vergleich der Ordinaten können diese geringfügigen Abweichungen zu einem Fehlschlag führen, obwohl die Vektoren den gleiche Punkt beschreiben sollen. Daher sollten Vergleiche von Vektoren im System einen individuell für jeden Grundriss konfigurierbaren Toleranzwert berücksichtigen.

Richtungstoleranz

Linien einer Wand, die eine optische parallele Ausrichtung haben, können um wenige Kommastellen eines Grades in ihrer Richtung abweichen. Dies kann eine Folge der zuvor beschriebenen Ordinatenabweichung eines Punktes sein. Vergleiche der Richtung von Linien und Vektoren sollten ebenfalls einen individuell für jeden Grundriss konfigurierbaren Toleranzwert berücksichtigen.

Abstandsintervall

Die zuvor beschriebenen Abweichungen können des Weiteren einen uneinheitlichen Abstand zwischen den zwei Linien einer Wand zur Folge haben. Bei der Zählung der Häufigkeit des Vorkommens eines Abstands muss daher mit einem konfigurierbaren Intervallbereich gearbeitet werden, um eine nicht repräsentative Streuung der Häufigkeiten durch minimale Abweichungen zu unterbinden.

3.3 Datenbewertung

3.3.1 Gruppierung von Linien zu logischen Elementen (Solids)

Schliessung vertikaler Flächen

Hauptziel dieser Arbeit ist eine dreidimensionale Darstellung von zweidimensionalen Grundrissdaten. Ein naheliegender, einfacher Ansatz ist die direkte Projektion der gegebenen zweidimensionalen Daten in die dritte Dimension. Alle gegebenen Linien werden nach einem Festen Regelwerk für die verschiedenen Elementebenen des Grundrisses entlang der Höhenachse kopiert und verschoben.

Die vertikalen Flächen zwischen den ursprünglichen Linien und den Projektionen können dann ohne komplexen Aufwand mit Polygonen (siehe Abschnitt 3.4.1) versehen werden. Das Ergebnis dieser einfachen Projektion ist in der Abbildung 3.2 dargestellt.

Vertikale Wandflächen sind an dieser Stelle bereits vollständig gerendert. Da jedoch auch die Ober- und Unterseiten jedes Elements durch Flächen geschlossen dargestellt werden sollen, müssen die Linienpaare, die gemeinsam ein Element bilden, in der Datenstruktur miteinander

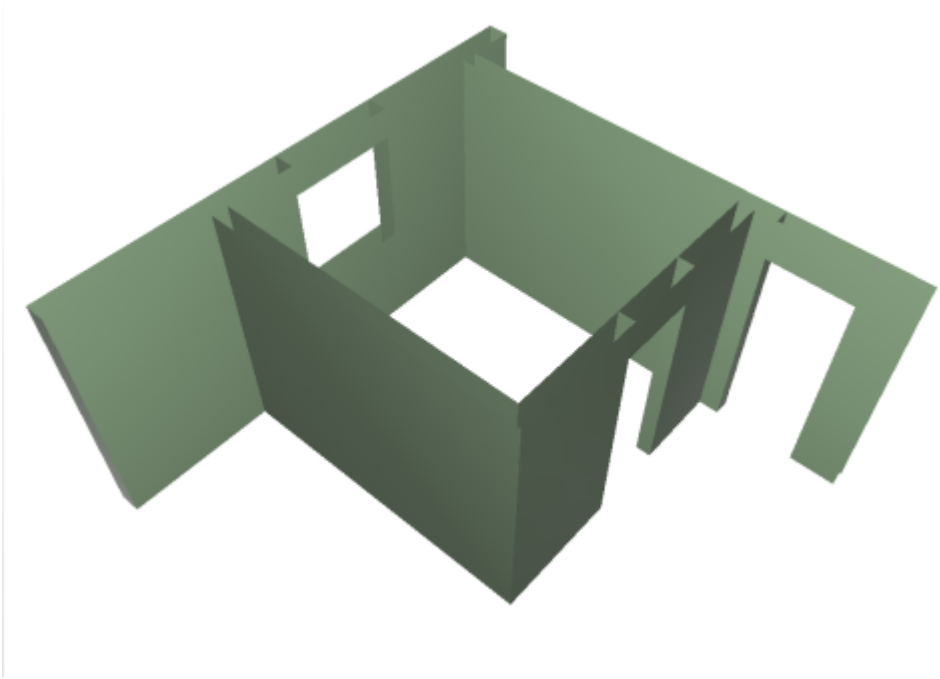

Abbildung 3.2: Geschlossene vertikale Flächen

assoziiert sein. Auf diese Weise können die horizontalen Flächen der Wandelemente mithilfe der vier Anfangs- und Endvektoren beider Linien geschlossen dargestellt werden.

Bildung von Solids

Die Kernanforderung an das System ist die verlässliche Bestimmung und Ergänzung von einzelnen Wandelementen, den Solids. Solids stellen dabei die kleinste notwendige Zerlegung eines Wandabschnitts dar. Diese Zerlegung findet statt, um die zwei parallelen Linien eines Elements für die spätere Darstellung miteinander zu assoziieren.

In der Abbildung 3.3 ist der häufigste und einfachste Fall eines Linienpaares abgebildet. Die Linien *Line A* und *Line B* sind benachbart. Die vier Vektoren A_1, A_2, B_1 und B_2 bilden die Eckpunkte des Solids und können für die Bildung zweier horizontaler Polygone zur Schließung der Decken und Bodenflächen herangezogen werden. *Line A* ist in dem gezeigten Fall zwar größer als ihre benachbarte *Line B*, da *Line A* jedoch keine weiteren benachbarten Linien besitzt, ist eine Assoziation beider vollständiger Linien an dieser Stelle kein Problem.

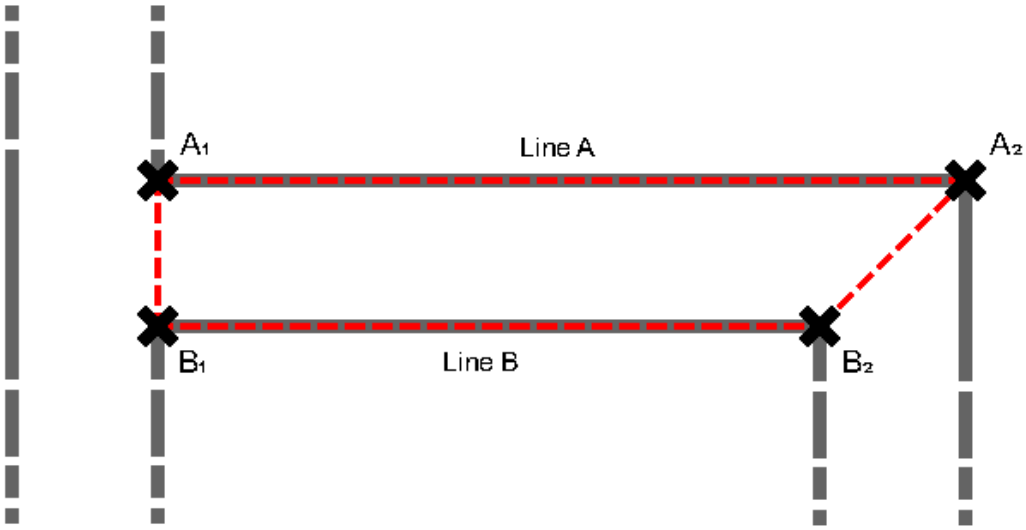

Abbildung 3.3: Einfache Solidbildung aus zwei benachbarten parallelen Linien

Asymmetrisches Verhältnis von Linien

Die Assoziation paralleler Linien ist dabei nicht immer symmetrisch. Verschiedene Gründe können in einigen Fällen dazu führen, dass jede Linie eines Elements beliebig viele parallele Nachbarlinien besitzen kann. Gründe hierfür können sowohl orthogonal angesetzte Wände als auch die in 3.2.1 angesprochene Zerlegung von Linien in eine beliebige Anzahl von Entitäten sein. Für die Bildung der vom System benötigten Solids ist eine Zerlegung von Linien erforderlich, um bei der Darstellung die horizontalen Flächen der verschiedenen Elemente korrekt und effizient durch Polygone schliessen zu können.

Linien mit mehreren parallelen Nachbarlinien werden nachfolgend als Multiline bezeichnet. Multilines müssen derart in einzelne Teillinien zerlegt werden, sodass jede parallele Linie X einen Nachbarabschnitt besitzt, der mindestens die Projektion der Linie X enthält.

3.3.2 Lücken in Solids durch angesetzte Wände

An Stellen, wo durch eine orthogonal angesetzte Wand keine Nachbarlinien vorhanden sind, entstehen bei der Solidbildung Lücken, die gesondert gehandhabt werden müssen. Für ein sauberes Rendering als Ergebnis ist es erforderlich, die einzelnen Bereiche dieser Wandabschnitte ohne Überschneidungen in Vierecke einzuteilen und dabei keine Lücken übrig zu lassen. Dabei

ist eine einzelne Lücke durch eine einzige angesetzte Wand weniger aufwändig zu schließen als eine Reihe von Lücken.

Einfache Lücke in einem Wandabschnitt

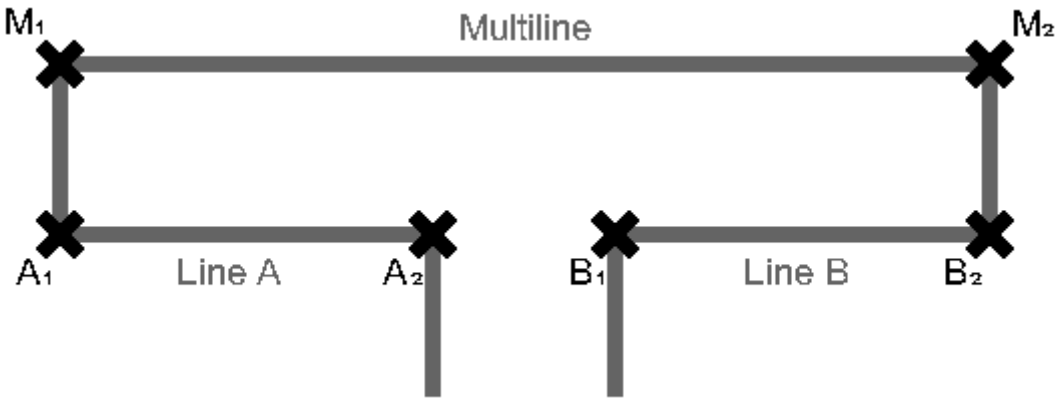

Abbildung 3.4: Multiline mit zwei benachbarten Linien und einer Lücke

In der Abbildung 3.4 ist eine einzelne Lücke dargestellt. Der *Multiline* sind zwei benachbarte Linien *Line A* und *Line B* gegenübergestellt. Den viereckigen Bereich zwischen den drei benachbarten Linien und der angesetzten Wand anhand der sechs Vektoren M_1, M_2, A_1, A_2, B_1 und B_2 so mit dreieckigen Flächenstücken (Polygone; siehe Abschnitt 3.4.1) zu belegen, dass durch die Schließung aller Lücken keinerlei Überschneidungen entstehen würden, ist nicht ohne weiteren Aufwand möglich. Zusätzlich muss eine Variante dieses Problems betrachtet werden, bei der sich die Linien *Line A* oder *Line B* noch in weitere einzelne Linienentitäten unterteilen, ohne dabei jedoch eine weitere Lücke in der Darstellung zu erzeugen. Um die Aufgabe zu lösen, wird der relevante Bereich in viereckige Teilflächen zerlegt. Dadurch können die Oberflächen bei der Darstellung auf einfache Weise mit Polygonen umgesetzt werden.

Schließung einfacher Lücken

Als mögliche Lösung kann die *Multiline* in mehrere Teilstücke zerlegt werden. Die Vektoren A_1 und A_2 von *Line A* und B_1 und B_2 von *Line B* werden auf die Multiline projeziert. Die dadurch erhalten Vektoren pA_1, pA_2, pB_1 und pB_2 können dann für die Bildung der Linienprojektionen *Line pA* und *Line pB* verwendet werden. Aus den Paarungen der Linien und ihren zugehörigen

Projektionen können dann Solids entstehen, wodurch die größere Fläche in einzelne Teilflächen aufgeteilt wurde.

Die Abbildung 3.5 veranschaulicht diesen Vorgang. Die roten Linien *Line pA* und *Line pB* sind die Projektionen der Linien *Line A* und *Line B* auf die Multiline. Der blau karierte Bereich sind die durch die Vektoren der Linie und ihrer jeweiligen Projektion gebildeten Solids. Der gelbe Bereich ist die durch die angesetzte Wand entstehende Lücke.

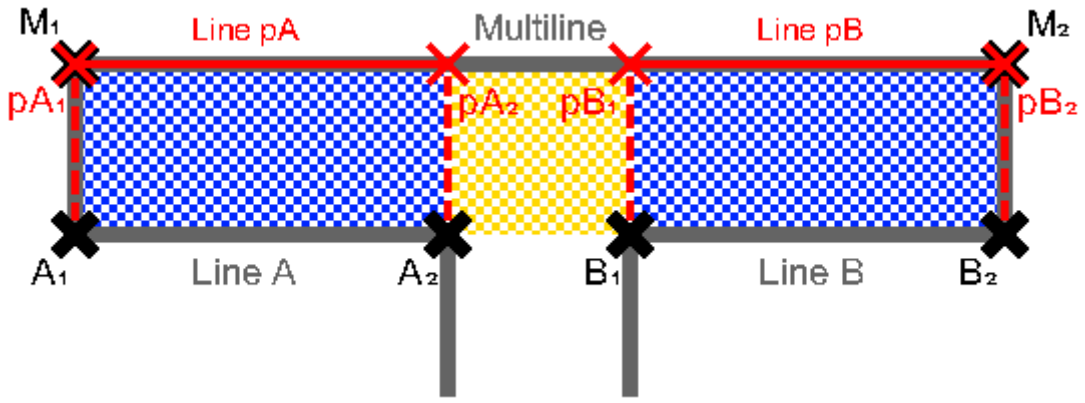

Abbildung 3.5: Einteilung des Wandabschnitts aus Abbildung 3.4 in Solids

Dieser Bereich soll ebenfalls durch ein Solid geschlossen werden. Ein Ansatz zur Lösung dieser Aufgabe sieht vor, die Nachbarlinien auf die Multiline zu projezieren und diese Projektionen von der Multiline zu subtrahieren. Im nächsten Schritt sollten dann die Ortsvektoren der Projektionspunkte als Stützvektoren für die verbliebene Differenzlinie getestet werden bis ein Ergebnis ohne Überschneidungen gefunden ist.

Jedoch erscheint dieser Ansatz weder zuverlässig noch effizient. Um die korrekte Position zu finden, müssten durch Testen alle Positionsvarianten geprüft werden. Des Weiteren ist diese Methode lediglich für den einfachen Fall einer einzelnen Lücke auf dem betrachteten Wandabschnitt hinreichend. Im Falle von mehrere Lücken auf einem Abschnitt (siehe 3.3.2) ist das Verfahren nicht anwendbar. Die Differenzlinie würde die Länge aller vorhandenen Lücken zusammen betragen. Daher wäre dann eine komplizierte, weitere Zerlegung der Differenzlinie in Teilabschnitte notwendig.

Der nachfolgend beschriebene Ansatz bietet den Vorteil, dass einzelne Schritte für die Lösung der verschiedenen Fälle identisch vollzogen werden können.

Dafür sind zunächst die projezierten Vektoren pA_2 und pB_1 erforderlich. Diese können als Ecken der Lücke identifiziert werden, indem aus der Menge der projezierten Vektoren diejenigen aussortiert werden, die zu den Vektoren M_1 und M_2 identisch sind.

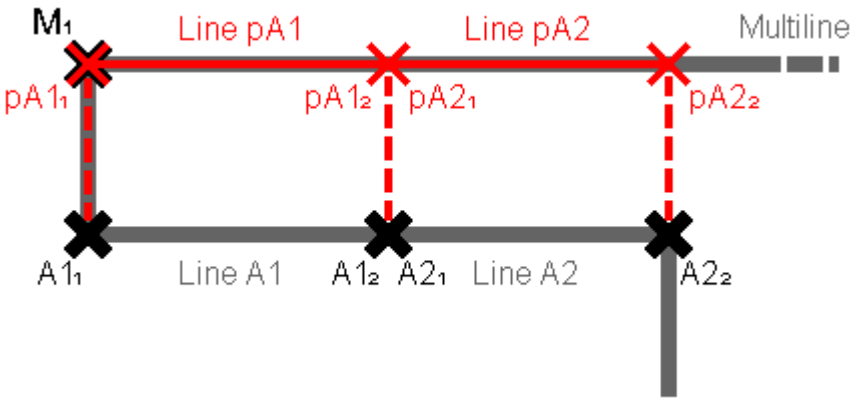

Abbildung 3.6: Sonderfall Multiline mit geteilter Nachbarlinie

Im Falle der oben erwähnten Variante, verdeutlicht in Abbildung 3.6, muss an dieser Stelle ein weiterer Schritt erfolgen. Beispielhaft wird angenommen, dass sich *Line A* aus zwei Linien *Line* $A1$ und *Line* $A2$ zusammensetzt. Dadurch ergeben sich statt der zwei Punkte A_1 und A_2 und ihrer Projektionen pA_1 und pA_2 jeweils die vier Vektoren $A1_1$, $A1_2$, $A2_1$ und $A2_2$ und ihre Projektionen $pA1_1$, $pA1_2$, $pA2_1$ und $pA2_1$. In dem gezeigten Fall sind die Koordinaten der projezierten Vektoren $pA1_2$ und $pA2_1$ identisch. Diese Identität kann dann als Auswahlkriterium dienen, um sie aus der Menge der auf die *Multiline* projezierten Vektoren ebenfalls auszusortieren.

Die dann verbliebenen zwei Vektoren sind die zwei Eckpunkte des gesuchten Solids. Eine einfache Methode die zugehörigen anderen zwei Eckpunkte, also die Vektoren A_2 und B_1, zu identifizieren, ist die Addition der gefundenen Vektoren mit dem invertierten Richtungsvektor zwischen einem der Vektoren von *Line A* oder *Line B* und seinem korrespondierenden Projektionsvektor. Addiert man diesen Vektor auf die Vektoren pA_2 und pB_1, erhält man die Ortsvektoren der Punkte A_2 und B_1. Mithilfe der vier gefundenen Vektoren kann dann ein Solid aus zwei Linien konstruiert werden, um die Lücke in der Wanddarstellung zu schließen.

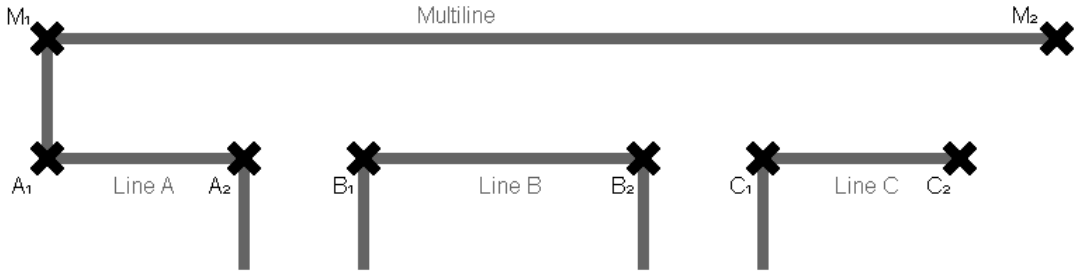

Abbildung 3.7: Multiline mit drei benachbarten Linien und zwei Lücken

Mehrfache Lücken in einem Wandabschnitt

Zunächst unterscheidet sich dieser Sonderfall wenig von der im vorangegangenen Abschnitt 3.3.2 erläuterten Aufgabe. Das Vorgehen für die in der Abbildung 3.7 dargestellte Wandkonstellation kann zunächst auf identische Weise gehandhabt werden. Das Schließen der mehrfachen Lücken in der Wanddarstellung erfordert jedoch ein anderes Vorgehen als im vorangegangenen Fall.

Schließung mehrfacher Lücken

Die Vektoren der zur *Multiline* benachbarten Linien *Line A*, *Line B* und *Line C* in der Abbildung 3.8 werden zunächst auf die Multiline projeziert. Mit diesen Projektionspunkten kann dann eine Projektionslinie konstruiert werden, die jeweils mit ihrer Ursprungslinie einen Solid bildet.

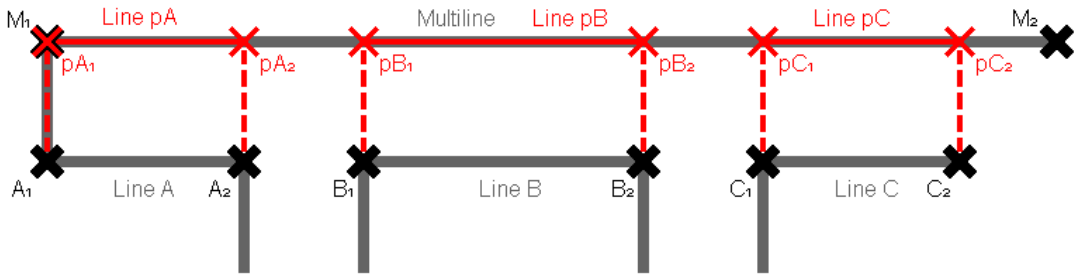

Abbildung 3.8: Projektionen von Vektoren der benachbarten Linien auf die Multiline

Für die Schließung der vorhandenen Lücken kann zunächst noch wie im Abschnitt 3.3.2 beschrieben vorgegangen werden. Aus der Menge der auf die *Multiline* projezierten Punkte werden gegebenenfalls die Endpunkte der *Multiline* entfernt. Zusätzlich werden doppelt vor-

kommende Vektoren aus der Liste aussortiert. Nach diesen Schritten bleiben dann diejenigen Vektoren übrig, die die gesuchten Lücken in der Wanddarstellung begrenzen.

Da es sich in diesem Fall um eine größere Menge Vektoren handelt, besteht die Aufgabe nun darin, diese Vektoren derart in Paare einzuteilen, sodass die mit ihnen gebildeten Solids zuverlässig die gesuchten Lücken schließen.

Zunächst müssen die verbliebenen Vektoren in ihrer Abfolge sortiert und bewertet werden. Dafür wird die Parameterform der Multiline gebildet und der skalare Faktor für die Position jedes Punktes auf der Multiline bestimmt. Anhand dieser Größe können dann die Punkte in Beziehung gesetzt werden. Auf diese Weise lässt sich zum Beispiel ausschließen, dass die Vektoren pA_2 und pC_1 der Abbildung 3.8 als Punktepaar für einen Solid in Frage kommen, da sie entlang der Multiline nicht direkt benachbart sind.

Die auf diese Weise sortierten benachbarten Punkte werden dann mit den Endpunkten der Projektionslinien der zuvor gebildeten Solids verglichen. Sind die zwei betrachteten Vektoren identisch zu den zwei Endpunkten eines Solids, wird die Kombination verworfen. Ist keine Projektionslinie eines Solids mit identischen Vektoren vorhanden, wird ein neuer Solid erstellt.

Für den neuen Solid wird zunächst die Linie aus den zuvor gefundenen, benachbarten Projektionspunkten gebildet. Die zweite, benachbarte Linie wird konstruiert, indem ein Distanzvektor auf die projezierten Vektoren addiert wird. Der Distanzvektor ist im Vorhinein aus einem Ursprungsvektor einer Nachbarlinie (zum Beispiel A_1 der Linie *Line A* aus Abbildung 3.8) und der dazugehörigen Projektion (pA_1) eines Solids gebildet worden. Durch die Addition des Distanzvektors erhält man also zwei weitere Vektoren, die zu einer Linie konstruiert werden können. Durch die Kombination der beiden Linien zu einem Solid ist die gefundene Lücke dann geschlossen.

3.3.3 Nicht zugeordnete Linien

In jedem Bauplan treten einzelne Linien auf, die sich bei der Distanzanalyse keinem Nachbarn zuordnen lassen. Diese Linien bilden entweder auf einem Wandende den Abschluss oder sie stellen die Trennung zwischen einem Fenster- oder Türelement und dem angrenzenden Wandelement dar.

Die in keiner Paarung enthaltenen Linien sind bereits im Schritt der Strukturanalyse, beschrieben in Abschnitt 3.2.2, gesondert zusammenzufassen.

Um diese Linien ebenfalls korrekt ihren Solids zuzuordnen, werden sämtliche gebildeten Solids durchiteriert und die Vektoren dieser Linien mit den Vektoren der Solidlinien verglichen. Dabei kann vorausgesetzt werden, dass die Vektoren dieser freien Linien zu jeweils nur einem der Vektoren beider benachbarter Linien eines Solids passen.

3.3.4 Lücken durch überstehende Multilines an Wandecken

Bei der in Abbildung 3.8 dargestellten Linienkonstellation wird der Punkt pA_1 aus der Liste der relevanten Projektionspunkte aussortiert, da er identisch zu M_1 ist. Für M_2 wird jedoch kein Projektionspunkt entfernt. Diese Eigenschaft kann für die Schließung einer noch bestehenden Darstellungslücke zwischen den Punkten C_2, pC_2 und M_2 verwendet werden.

Für die saubere Darstellung dieses Eckbereichs könnte die Handhabung dreieckiger Solids im System ermöglicht werden. Durch solch einen Solid könnte dann der Bereich zwischen den Punkten C_2, pC_2 und M_2 abgedeckt werden. Jedoch würde dieser Sonderfall eine gesonderte Behandlung dreieckiger Solids im Darstellungsaufbau erfordern.

Ein anderer Ansatz tauscht den zum Multiline Ende M_2 am nächsten gelegenen Vektor pC_2 in dem mit ihm gebildeten Solid mit M_2 aus. Auf diese Weise ist eine Handhabung des Solids bei der späteren Darstellung wie bei allen anderen Solids möglich, und der fragliche Bereich wird vollständig dargestellt.

3.4 Ergebnisdarstellung

Zur dreidimensionalen Visualisierung der zuvor gefundenen Solids sollen alle Linien eines jeden Solids als vertikale Seitenflächen sowie die Ober- und Unterseite geschlossen dargestellt werden.

3.4.1 Aufbau des Polygonnetzes

Die Darstellung erfolgt mithilfe von Polygonen. Dabei wird jedes dreidimensionale Objekt beziehungsweise jeder dreidimensionale geometrische Körper durch Dreiecke zusammengesetzt. Diese Methode ist für die Darstellung von Gebäuden gut geeignet, da sich die geometrischen Formen in der Regel auf rechteckige Strukturen beschränken, deren Flächen durch die Kombination von zwei Dreiecken zusammengesetzt werden können. Die Einteilung des Gebäudes in viereckige Elemente, die Solids, begünstigt dabei die dreidimensionale Darstellung durch Polygone. Mit einem einfachen Regelwerk können die verschiedenen Elemente eines Grundrisses individuell aufgebaut werden.

3.4.2 Darstellung von Wänden, Türen und Fenstern

Das System muss dabei berücksichtigen, um welche Art von architektonischem Element es sich bei den einzelnen Solids handelt. Die Darstellungen von Wänden, Fenstern und Türen variiert in vertikaler Richtung.

Aus den Vektordaten der Quelldatei sind für die Bildung der Dreiecke jeweils zwei Punkte einer Linie in vektorieller Form bekannt. Da sich die Ausgangsdaten auf den zweidimensionalen Raum beschränken ist die dritte Ordinate entlang der vertikalen Achse stets 0 gesetzt.

Über die Manipulation der dritten Ordinate lassen sich die Punkte entlang der Höhenachse verschieben, wordurch auf unkomplizierte Weise verschiedene Elemente des Grundrisses dargestellt werden können.

Die Höhenwerte zur Darstellung der einzelnen Elemente sind in der Konfigurationsdatei enthalten und können individuell justiert werden. Für die Darstellung eines Wandelements reicht eine einzelne Höhenangabe: die Gesamthöhe des darzustellenden Stockwerks. Für eine Tür muss zusätzlich die Höhe der Türöffnung definiert sein. Fensterlemente benötigen für die dreidimensionale Darstellung zusätzlich zur Gesamthöhe des Stockwerks zwei weitere Werte: die Höhe der Unterkante der Fensteröffnung sowie die Höhe der Oberkante. Die verschiedenen Höhenwerte sind in der Abbildung 3.9 als rote Pfeile von der zweidimensionalen Grundrissdarstellung in vertikaler Richtung dargestellt.

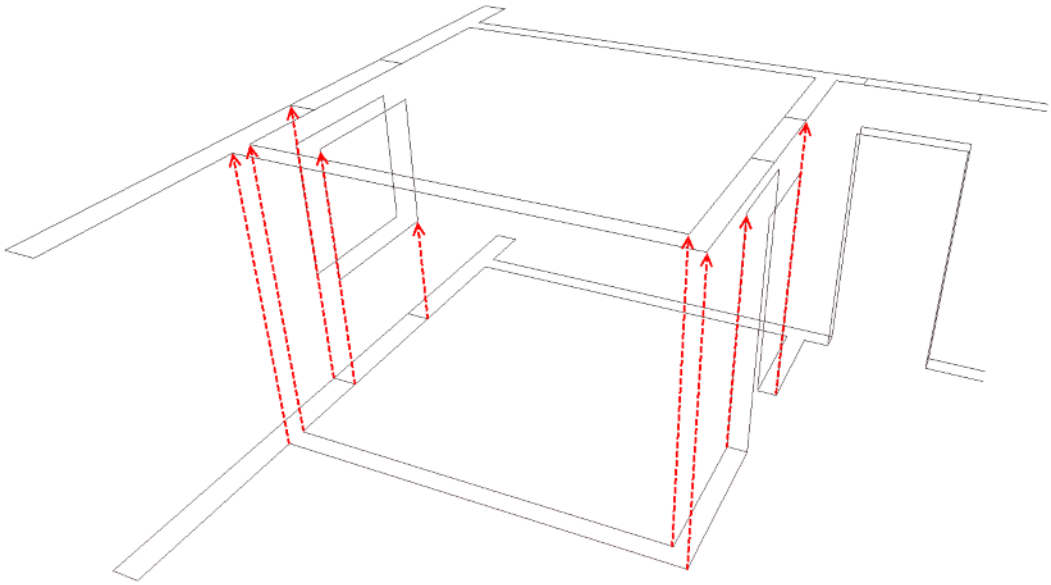

Abbildung 3.9: Projektion der 2D-Daten entlang der Höhenachse

Die vertikalen Flächen von Wand- und Türlementen können aus zwei Polygonen zusammengesetzt werden, wobei die Tür sich lediglich darin von der Wand unterscheidet, dass die

Unterkante des aus zwei Polygonen konstruierten Vierecks nicht auf Bodenhöhe sondern auf Höhe der Türöffnung liegt. Eine vertikale Fläche für ein Fenster wird aus vier Polygonen konstruiert. Zwei davon ergeben ein Viereck vom Boden bis zur Unterkante der Fensteröffnung, die anderen zwei bilden ähnlich dem Türelement das Viereck oberhalb der Fensteröffnung bis zur Gesamthöhe.

Horizontale Flächen werden aus den vier Vektoren gebildet, die die zur Solidbildung verwendeten benachbarten Linien definieren. Zwei Polygone zwischen diesen vier Eckpunkten ergeben eine geschlossene Fläche. Während Wand- und Türelemente zwei dieser Flächen zur Schließung der Ober- und Unterseite erfordern, benötigen Fensterlemente vier horizontale Flächen. Zwei davon sind für die Ober- und Unterseite, die anderen zwei sind für die Fensteröffnungsflächen erforderlich.

Ausserdem müssen im Falle eines Fensterelements noch zusätzlich die vertikalen Seitenflächen in der Fensteröffnung mit jeweils zwei Polygonen versehen werden.

4 Lösungskonzept

4.1 Datenimport

4.1.1 Anforderungen an das DXF Dokument

Für den reibungslosen Import der DXF Daten müssen einige Bedingungen erfüllt sein.

Vorbereitung der Datei

Damit das System die Vektordaten des Grundrisses korrekt interpretieren und darstellen kann, muss das System die Daten den drei verschiedenen Grundtypen der Wanddarstellung zuordnen können. Mithilfe eines entsprechenden Grafikprogramms können die relevanten Linien auf eigene Ebenen sortiert werden. Diese Ebenen müssen eindeutige Namen besitzen, die dann anhand der XML-Konfigurationsdatei mit den entsprechenden Typen assoziiert werden. Für die entsprechenden Anpassungen der verwendeten Grundrisse wurde das frei verfügbare Programm Inkscape verwendet. Damit lassen sich dem Dokument neue Ebenen hinzufügen und mithilfe der Befehle *Markieren*, *Ausschneiden* und *Einfügen* die entsprechenden Entitäten auf die Ebenen sortieren.

4.1.2 Eigene Datentypen

Für die Handhabung der Exportdaten innerhalb des Systems dienen drei individuelle Klassen.

Grundriss Linien

Die aus dem DXF Dokument exportierten Linien werden in ein systemeigenes Objekt überführt. Auf diese Weise wird die weitere Handhabung der Daten von den Datentypen der *Kabeja* Bibliothek unabhängig. Die Vektoren werden dabei in ein Vektorformat der computergraphics Bibiliothek überführt. Desweiteren erhalten Linien in diesem Schritt einen individuellen Bezeichner, der der besseren Nachvollziehbarkeit in den Ausgaben des Programmablaufs dienen soll.

Solid

Für die Organisation der Daten im Schritt der Datenanalyse ist ein Datentyp Solid vorhanden. Ein Solid kann beliebig viele Linien enthalten. In der Regel sind es jedoch zwei bis vier Linien zur Formung eines Vierecks. Ausserdem enthält jeder Solid eine Referenz auf zwei dieser Linien zur Identifizierung desjenigen benachbarten Linienpaares, das die Erstellung des Solids bedingt hat. Dies ist notwendig, um bei der späteren Darstellung von horizontalen Flächen zusätzlichen Aufwand zur Identifizierung der korrekten Vektorkonstellation und Polygonerstellung einzusparen.

Grundriss Datentyp

Linien und Solid Objekte werden dabei von einer weiteren Klasse organisiert. Diese Klasse wird innerhalb des Systems zur Speicherung von Referenzen aller importierten oder konstruierten Objekte in verschiedenen Listenstrukturen verwendet.

4.1.3 XML Konfigurationsdatei

Die im Abschnitt 3.1.3 beschriebenen Problemstellungen führen zu einer Reihe von Konfigurationswerten, die im Nachfolgenden genauer erläutert werden sollen. Die Konfigurationsdatei ist zwingend erforderlich und muss zur einfachen Erkennung den selben Dateinamen wie die zugehörige DXF-Datei, abweichend ausschließlich in der Dateiendung, besitzen.

Skalierung

In der Konfiguration ist ein Skalierungsfaktor enthalten, der zu den importierten DXF-Daten passen muss. Dabei spielen sowohl die Dimension der enthaltenen Vektordaten als auch die zugrunde liegende Bezugsgröße eine Rolle.

Bezugsgröße

Bei der Speicherung einer Vektorgrafik im DXF Format muss eine Bezugsgröße gewählt werden. Da Vektorgrafiken sich dadurch auszeichnen, dass sie nicht pixelbasiert, sondern in der Darstellung frei skalierbar sind ist hierfür die Wahl einer metrischen Einheit wie zum Beispiel Millimeter oder Zentimeter zu bevorzugen.

Bestimmung des Skalierungsfaktors

Für die Bestimmung des Skalierungsfaktors können im Idealfall die bekannten Maße des Grundrisses verwendet werden. Sobald ein Aussenmaß oder die Maße eines anderen Elements

des Grundrisses genau bekannt sind, kann mit Hilfe der in der Datei gemessenen Distanz der genaue Skalierungsfaktor, wie nachfolgend beschrieben, berechnet werden.

Andernfalls kann, wie bereits im Abschnitt 3.1.3 vorgeschlagen, der Skalierungsfaktor über die genormten Maße von Türlaibungen bestimmt werden.

Zunächst muss die Einheit der Bezugsgröße des DXF Dokuments festgelegt sein. In dieser Einheit kann dann in einem Grafikprogramm entweder das bekannte Element oder eine durchschnittlich breit erscheinende Türöffnung gewählt und vermessen werden.

Mithilfe des gemessenen Wertes lässt sich dann die Berechnung ausführen. Die Formel $Ax = D$ mit A als Meßwert und D als angenommene Normbreite ergibt nach x umgestellt den Skalierungsfaktor. Dabei muss berücksichtigt werden in welcher Einheit die restlichen in der Konfigurationsdatei enthaltenen Maßangaben angegeben sind. Sind die Höhenangaben in Metern definiert, sollte der Wert für D ebenfalls in Metern angegeben sein. Als Standardwert hat sich eine Maueröffnungsbreite von 80 Zentimetern bewährt. Für D müsste folglich der Wert 0.8 eingesetzt werden.

Beispielrechnung Werte: A = 12,2 mm; D = 0,8m

$$Ax = D \tag{4.1}$$

$$x = \frac{D}{A} \tag{4.2}$$

$$x = \frac{0,8}{12,2} \tag{4.3}$$

$$x = 0,066 \tag{4.4}$$

Der Skalierungsfaktor in diesem Beispiel beträgt 0,066. Wird dieser Wert in die Konfigurationsdatei eingetragen, sind alle weiteren Angaben in Metern vorzunehmen.

Datenskalierung bei Import

Die Vektordaten aus dem DXF werden im System bereits beim Import skaliert in die internen Datentypen umgewandelt. Auf diese Weise erfolgt die Analyse der importierten Daten bereits im gewählten Maßstab. Für die Analyse notwendige Konfigurationswerte können dadurch ebenfalls in der gewählten Einheit in reellen Wertebereichen erfolgen.

Toleranzen

Das Konfigurationsdokument enthält drei Werte für Toleranzen, die vom System bei der Analyse berücksichtigt werden.

Punkttoleranz

Für Vektoren kann in der Konfigurationsdatei ein Toleranzwert in der entsprechenden Bezugsgröße definiert werden. Dieser Wert wird bei Vergleichen von Vektoren als Toleranzbereich für die einzelnen Ordinaten verwendet.

Richtungstoleranz

Für den Vergleich von Richtungsvektoren kann ein Toleranzwert in Grad definiert werden. Bei Winkelvergleichen von Richtungsvektoren wird dieser Wert als Toleranzbereich für die Abweichung der Richtung zwischen zwei Vektoren verwendet.

Abstandsintervall

Das Abstandsintervall gibt an, in welchen Intervallgrößen Abstandshäufigkeiten in der Analyse gruppiert werden sollen.

Höheninformationen für die Elementdarstellung

Für jeden Grundriss können individuell die Höheninformationen für die Darstellung der drei verschiedenen Elementarten definiert werden. Die Angabe zur Wandhöhe betrifft dabei alle drei Elementarten. Sie definiert die Oberkante aller Elemente. Die Angabe für die Türhöhe definiert die Unterkante oberhalb der Maueröffnung und ergibt zusammen mit der Wandhöhe den Darstellungsbereich für Türelemente. Fensterelemente werden durch weitere zwei Angaben zur Unter- und Oberkante von Fensteröffnungen definiert, die zusammen mit der Wandhöhe die dreidimensionale Darstellung von Fenstern ermöglicht.

Werte für die Wahl relevanter Abstände im Histogramm

Für die Bildung von Solids müssen zunächst Paarungen von Linien mit den relevanten Abständen erfolgen. Diese Abstände werden vom System anhand der Analyseergebnisse selektiert. Die Selektion erfolgt dabei anhand von drei frei definierbaren Konfigurationswerten. Zunächst gibt es zwei Werte für die Ober- und Untergrenze von Wandstärken. Mit diesen Grenzwerten lässt sich der Bereich der in Frage kommenden Abstandswerte eingrenzen. Zusätzlich definiert ein Schwellwert für die Häufigkeit von Abständen welche Abstände selektiert werden.

4.2 Datenanalyse

4.2.1 Gruppierung paralleler Linien

Nach dem Import der Vektordaten analysiert das System diese zunächst. Dabei gilt es in einem ersten Schritt, alle parallelen Linien zu finden. In einem weiteren Schritt sind dann diejenigen parallelen Linien zu identifizieren, die benachbart zueinander sind, also eine Projektion aufeinander größer Null besitzen.

Sortierung von Linien anhand ihrer normierten Richtungsvektoren

Zunächst wird für jede Grundrisslinie l definiert über die Ortsvektoren l_1 und l_2 der normierte Richtungsvektor $\vec{n}$ berechnet um die Vektoren aufgrund der Normierung untereinander vergleichen zu können.

$$\vec{r} = l_2 - l_1$$

$$\vec{n} = \frac{\vec{r}}{|\,\vec{r}\,|}$$

Die Linien werden dann in Listen einer Map mit dem normierten Richtungsvektor $\vec{n}$ als Schlüssel einsortiert. Dabei wird geprüft, ob ein berechneter Richtungsvektor zu einem der bereits in der Map als Schlüssel vorhandenen Vektoren passt, wobei die in Abschnitt 4.1.3 angesprochene Winkeltoleranz t berücksichtigt wird, um Fehlsortierungen aufgrund minimaler Abweichungen zu vermeiden. Die Berechnung des Winkels φ zwischen der betrachteten Linie mit dem normierten Richtungsvektor $\vec{n}$ und einem bereits existierenden Schlüssel $\vec{k}$ der Map erfolgt durch die Formel

$$\cos\varphi = \frac{\vec{n}\cdot\vec{k}}{|\,\vec{n}\,|*|\,\vec{k}\,|} \quad \Longrightarrow$$

$$\varphi = \arccos\left(\frac{\vec{n}\cdot\vec{k}}{|\,\vec{n}\,|*|\,\vec{k}\,|}\right).$$

Dabei muss φ im Bereich $[0° - t \leq \varphi \leq 0° + t]$ beziehungsweise $[180° - t \leq \varphi \leq 180° + t]$ liegen, damit die Linie zum bestehenden Schlüsselvektor $\vec{k}$ hinzusortiert werden kann. Ist kein zu $\vec{n}$ passender Vektor $\vec{k}$ vorhanden, wird $\vec{n}$ als neuer Schlüssel zur Map hinzugefügt und die Linie seiner Liste hinzugefügt. Das Ergebnis dieses Schrittes ist eine Map, die für jeden gefundenen, normierten Richtungsvektor $\vec{n}$ eine Liste aller zugehörigen Linien enthält.

4.2.2 Projektionsprüfung und Abstandsbestimmung

Nun werden im nächsten Schritt die jeweils parallelen Linien untereinander verglichen und in Paare eingeteilt. Für den Vergleich wird zunächst jede Linie auf alle anderen Linien mit derselben Richtung projeziert. Das Vorgehen für die Berechnung einer Überlappung zweier Geraden wird dabei mithilfe des folgenden Verfahrens bestimmt.

Die beiden Linien A und B werden dafür zunächst in ihre Parameterdarstellung überführt:

$$A : \vec{a_0} + \lambda(\vec{a_1} - \vec{a_0})$$

$$B : \vec{b_0} + \lambda(\vec{b_1} - \vec{B_0})$$

Die Vektoren $\vec{b_0}$ und $\vec{b_1}$ der Linie B werden dann auf die Linie A projeziert, sodass sich die skalaren Werte λ_{b_0} und λ_{b_1} ergeben, die den projezierten Bereich der Linie B auf die Linie A darstellen:

$$\lambda_{b_0} = (\vec{b_0} - \vec{a_0}) \cdot (\vec{a_1} - \vec{a_0})$$

$$\lambda_{b_1} = (\vec{b_1} - \vec{a_0}) \cdot (\vec{a_1} - \vec{a_0})$$

Die Strecke A erstreckt sich in der Parameterdarstellung für alle Parameterwerte λ aus dem Intervall $[0; \beta]$ mit $\beta = (\vec{a_1} - \vec{a_0}) \cdot (\vec{a_1} - \vec{a_0})$.

Das Intervall I der Überlappung ergibt sich dann zu

$$I = [0; \beta] \cap [\lambda_{b_0}; \lambda_{b_1}].$$

Ist $I = \emptyset$, überlappen sich die Strecken nicht. Andernfalls besitzen die Strecken eine Überlappung und werden vom System als potentiell relevante Paarung vorgemerkt. Mit dem berechneten Wert λ_{b_0} und den Vektoren $\vec{a_0}$ und $\vec{a_1}$ der Linie A sowie den Vektoren $\vec{b_0}$ und $\vec{b_1}$ der Linie B ergibt sich dann der Abstand d beider Linien zu

$$d = ||\vec{b_0} - (\vec{a_0} + \lambda_{b_0}(\vec{a_1} - \vec{a_0}))||.$$

Die Linienpaarung wird unter der Referenz des Abstands d in einer Map gespeichert. Außerdem wird ein Zähler für das Abstandsintervall, in den der Abstand d fällt, um eins erhöht.

Erfassung und Ausgabe der Abstandshäufigkeit

Jedes Auftreten eines Abstands wird in einer speziellen Map erfasst. Die Map wird mithilfe des in der Konfigurationsdatei definierten Abstandsintervalls erweitert, wenn ein berechneter

Abstand den bisherigen Wertebereich der Liste überschreitet. Dabei werden alle Werte zwischen dem zuletzt höchsten Abstand und dem neuen Höchstwert in Schritten von der Größe des konfigurierten Abstandintervalls zur Map hinzugefügt. Auf diese Weise ergibt sich eine linearisierte Abstandsachse frei von Sprüngen.

Die Map wird vom System nach der Berechnung aller Abstände innerhalb des Grundrisses in Form einer csv-Datei ausgegeben. Anhand dieser ausgegebenen Daten kann dann ein Histogramm erstellt werden. Damit können die Werte für den Bereich der relevanten Wandstärken sowie der korrekte Schwellwert für die Häufigkeit so eingestellt werden, dass das System im nächsten Schritt die Bildung von Solids aus den entsprechenden Linienpaarungen durchführen kann.

4.3 Datenbewertung

Die vorliegenden Analyseergebnisse werden nun im nächsten Schritt ausgewertet und die relevanten Vektordaten und Beziehungen selektiert und weiter analysiert.

4.3.1 Strukturelle Ermittlung

Wahl relevanter Abstände

Für die Auswahl der relevanten Distanzhäufigkeiten werden die im vorangegangenen Abschnitt eingestellten Schwellwerte aus der Konfigurationsdatei herangezogen. Betrachtet werden dabei nur Distanzen, die innerhalb der definierten Ober- und Untergrenzen für die Wandstärke liegen. Treten Abstände im relevanten Wertebereich häufig genug auf, werden sie für die weitere Berwertung herangezogen.

Relevante Paarungen speichern und Vorkommen jeder Linie zählen

Im nächsten Schritt werden nun die Paarungen, die die oben erwähnten Kriterien erfüllen, selektiert und gespeichert. Dabei wird in einer weiteren Map gezählt, in wie vielen Paarungen eine Linie vertreten ist. Anhand dieser Daten kann dann im weiteren Verlauf bewertet werden, wie mit einer Paarung verfahren wird. Wie in Abschnitt 3.3.1 beschrieben wurde, ist für das weitere Verfahren entscheidend, ob die Linien einer Paarung lediglich in einer Paarung auftreten oder ob sie in mehreren Paarungen enthalten sind.

Ungepaarte Linien filtern

Um die Menge der nicht in einer Paarung enthalten Linien zu erhalten, wird zunächst eine Liste mit allen Lineobjekten gesammelt. Dann wird ein Set erstellt, in dem jedes Linienobjekt aus jeder Paarung vorkommt. Nun kann das Set mit den in Paarungen enthaltenen Linien von der Liste aller Linienobjekte abgezogen werden, um die Menge aller ungepaarten Linien zu erhalten.

4.3.2 Paarungen mit nur einfach vorkommenden Linien

Anhand der Auswertung zur Häufigkeit der Linien in Paarungen können nun sämtliche Paarungen von Linien selektiert werden, deren Linien ausschließlich in der jeweiligen Paarung vorkommen. Diese Selektion von Linienpaarungen kann im nächsten Schritt direkt in Solid Objekte umgewandelt werden.

Ausserdem wird an dieser Stelle für jedes Solid geprüft, ob eine der in Abschnitt 3.3.3 beschriebenen ungepaarten Linien zugeordnet werden kann. Dabei wird für die beiden Vektoren jeder ungepaarten Linie ein Vergleich mit den Vektoren der benachbarten Linien des Solids durchgeführt. Gibt es eine Übereinstimmung eines Vektor mit einem der Vektoren einer gepaarten Linie, wird sichergestellt, dass auch der zweite Vektor der ungepaarten Linie zu einem Vektor der anderen gepaarten Linie passt. Nur wenn jeweils ein Vektor der ungepaarten Linie jeweils einem Vektor der gepaarten Linien zugewiesen werden kann, gehört die Linie eindeutig zum Solid. Sie wird dem Solid Objekt in diesem Fall hinzugefügt und aus der Liste der ungepaarten Linien entfernt.

4.3.3 Paarungen mit mehrfach vorkommenden Linien

Zerlegung von Geraden in Teilstücke

Für den nächsten Schritt werden nun alle Paarungen eingesammelt, die Multilines enthalten. Es werden alle Paarungen jeder Ebene durchiteriert. Bei jeder Paarung wird die Häufigkeit des Vorkommens geprüft. Ist eine Multiline gefunden wird festgestellt, ob diese Multiline bereits in einem vorangegangenen Iterationsschritt bearbeitet worden ist. Ist das nicht der Fall, werden weitere mit dieser Multiline gepaarte Linien gesucht und die Gruppe alle gefundenen Linien in einer neuen Datenstruktur zusammengefasst.

Im Anschluss kann nun über die neu erstellten Strukturen iteriert werden, um die Liniengruppen nacheinander abzuarbeiten. Dabei wird zunächst die Multiline identifiziert. Dann werden die Vektoren jeder Linie, die zu dieser Multiline benachbart ist, auf die Multiline projiziert.

Für die Bestimmung eines Projektionspunktes auf eine Gerade in Parameterform nutzen wir eine Eigenschaft des Skalarprodukts. Sind zwei Vektoren $\vec{x}$ und $\vec{y}$ orthogonal zueinander, so ist das Skalarprodukt $\vec{x} \cdot \vec{y} = 0$. Der Vektor, der von einem Punkt $\vec{x}$ orthogonal auf die Gerade g zeigt, ergibt sich durch Bildung des Differenzvektors zu $\vec{d} = P_g(\vec{x}) - \vec{x}$. Mit dem Richtungsvektor $\vec{u}$ der Geraden ergibt sich daraus in das Skalarprodukt eingesetzt

$$(P_g(\vec{x}) - \vec{x}) \cdot \vec{u} = 0$$

und mit der Parameterform $P_g(\vec{x}) = \vec{r} + \lambda\vec{u}$ eingesetzt wiederum

$$(\vec{r} + \lambda\vec{u} - \vec{x}) \cdot \vec{u} = 0.$$

Nach λ aufgelöst erhalten wir dann

$$\lambda = \frac{(\vec{x} - \vec{r}) \cdot \vec{u}}{\vec{u} \cdot \vec{u}},$$

setzen dies nun in die Parameterform $P_g(\vec{x}) = \vec{r} + \lambda\vec{u}$ ein und erhalten die Formel zur Bestimmung der orthogonalen Projektion $P_g(\vec{x})$ eines Vektors $\vec{x}$ auf die Gerade g:

$$P_g(\vec{x}) = \vec{r} + \frac{((\vec{x} - \vec{r}) \cdot \vec{u})}{\vec{u} \cdot \vec{u}}\vec{u}$$

Der Differenzvektor $\vec{d}$ wird im nächsten Arbeitsschritt noch einmal relevant und daher vorläufig behalten.

Die berechneten Projektionen werden für die Lückenanalyse in einer Listenstruktur gesammelt. Mit den jeweils gefundenen Projektionspunkten kann nun eine Projektionslinie gebildet werden, die zur Ursprungslinie benachbart und in der Multiline enthalten ist. Mit der Ursprungslinie und der Projektionslinie wird dann ein neuer Solid erstellt. Dieser Schritt wird für alle zur Mutliline benachbarten Linien wiederholt.

Schließung vorhandener Lücken

Die im Schritt zuvor eingesammelten Projektionspunkte werden nun anhand ihres Skalarwerts sortiert. Für die Bestimmung des Skalarwerts wird zunächst erneut eine Geradengleichung in Parameterform der Multiline

$$G_{ML}(\vec{x}) = \vec{r} + \lambda(\vec{x_1} - \vec{x_0})$$

mit ihren Ortsvektoren $\vec{x_0}$ und $\vec{x_1}$ gebildet. Diese wird dann mit dem Vektor des jeweiligen Projektionspunkts $\vec{p}$ gleichgesetzt und nach λ aufgelöst:

$$\vec{p} = \vec{r} + \lambda(\vec{x_1} - \vec{x_0})$$

$$\lambda = \frac{\vec{p} - \vec{r}}{(\vec{x_1} - \vec{x_0})}$$

Dieser Arbeitsschritt macht sich zunutze, dass die Projektionspunkte bereits auf der Multiline liegen und daher eine Gleichsetzung der Projektionspunkte mit der Geradengleichung möglich ist. Die Ordinatenachsen werden dabei einzeln betrachtet. Die Ergebnisse der beiden Achsen werden verglichen und die Identität der berechneten λ-Werte überprüft. Abweichende λ-Werte würden bedeuten, dass der betrachtete Projektionspunkt nicht auf der Geraden liegt.

Für die Auffindung von Lücken werden nun von den sortierten Projektionspunkten gegebenenfalls jene Punkte entfernt, die zu den Orstvektoren der Multiline identisch sind. Wenn diese Ortsvektoren nicht zu einem Projektionspunkt passen, wird das vom System vermerkt, um die überstehenden Punkte der Multiline später einem Solid zuzuordnen.

Außerdem werden Punkte aussortiert, die mehrfach vorhanden sind. Diese kommen nur wie in Abbildung 3.6 dargestellt vor, also bei zwei direkt aufeinander abfolgenden Linien ohne Lücke. Des weiteren werden die sortierten Projektionspunkte noch mit einem Index versehen. Dieser ist für die nachfolgende Untersuchung der Projektionspunkte notwendig. Anhand der Anzahl der verbliebenen Projektionspunkte kann nun die Art des Lückenproblems unterschieden werden.

Einfache Lücke

Sind lediglich zwei Projektionspunkte verblieben, handelt es sich um eine einfache Lücke. Dieser Fall ist in Abbildung 4.1 dargestellt. Die Punkte pA_1 und pB_2 wurden aufgrund der Identität zu M_1 und M_2 aussortiert. Übrig sind dann die Punkte pA_2 mit dem Index 1 und pB_1 mit dem Index 2. Anhand Indizes kann beurteilt werden, ob diese beiden Punkte entlang der Multiline direkt benachbart sind. Benachbarte Punkte besitzen Indizes mit einer Differenz von 1. Ist dies nun der Fall, wird geprüft, ob der Solid bereits existiert. Wenn mit diesen beiden Punkten noch kein Solid gebildet worden ist, kann ein neuer konstruiert werden.

Dazu wird eine Linie zwischen den beiden Projektionspunkten pA_2 und pB_1 erstellt. Durch die Addition des im vorangegangenen Abschnitt berechneten Distanzvektors $\vec{d}$ zu den Projektionspunkten ergeben sich die Vektoren für die gegenüberliegende Linie. Durch die Verwendung

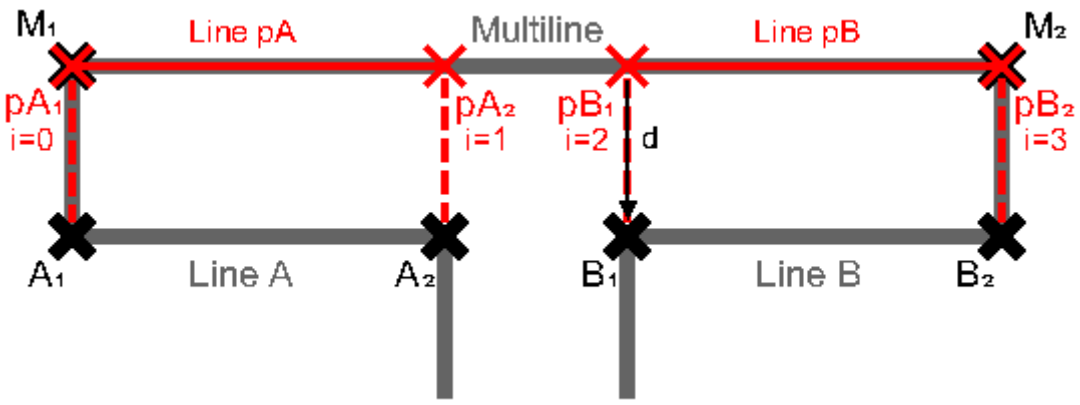

Abbildung 4.1: Multiline mit Projektionsindizes und Distanzvektor

des Distanzvektors wird der für die aktuelle Wandstärke korrekte Abstand zur ersten Linie eingehalten.

Mehrfache Lücken

Sind mehr als zwei Projektionspunkte verblieben, liegen mehrere Lücken auf der Multiline. Die relevanten Vektoren werden dann sortiert nach ihrem Index betrachtet, angefangen beim niedrigsten Index. Dabei wird zunächst das erste Element aus der Liste relevanter Projektionspunkte sowie das zu diesem Element direkt folgende betrachtet. Haben diese beiden Elemente eine Indexdifferenz von 1, sind also benachbart, werden sie mit den Linienvektoren der bisher auf der Multiline gebildeten Solids verglichen. Wenn keiner der Solids zu dieser Paarung passt, ist eine Lücke erfolgreich identifiziert worden. In diesem Fall kann aus den beiden Vektoren analog zum Verfahren im vorangegangenen Abschnitt mithilfe des Distanzvektors ein neuer Solid konstruiert werden.

Schließung von Lücken durch überstehende Multilines an Wandecken

Überstehende Linien an Wandecken fordern, wenn es sich bei beteiligten Linien um Multilines handelt, eine gesonderte Handhabung (Beschreibung in Abschnitt 3.3.4). Die zuvor nicht einem Projektionspunkt zugeordneten Ortsvektoren der Multiline können nun noch zu den passenden Solids zugeordnet werden.

Für diesen Schritt werden die zuvor berechneten Skalarwerte herangezogen, um festzustellen welcher der projezierten Punkte auf der Multiline dem nicht zugeordneten Punkt am nächsten

liegt. Da für die Berechnung der Skalarwerte die Parameterdarstellung der Multiline verwendet wurde, können die zwei Punkte als *Start-* beziehungsweise *Endpunkt* entsprechend des gebildeten Richtungsvektors betrachtet werden. Die Projektionspunkte sind in einer *TreeMap* anhand der natürlichen Ordnung ihrer Skalarwerte sortiert. Je nachdem, ob es sich um den Start- oder Endpunkt der Multiline handelt, kann also das erste beziehungsweise letzte Element dieser TreeMap als Kandidat für eine Ersetzung im zugehörigen Solid gewählt werden.

Die für die jeweils betrachtete Multiline gebildeten Solids werden nach dem zu ersetzenden Vektor durchsucht. Ist der Vektor in der Linie eines Solids gefunden, wird eine neue Linie mit dem zweiten Vektor der Linie und dem Multiline Vektor gebildet und die entsprechende Linie des Solids ersetzt.

4.4 Ergebnisdarstellung

Die Darstellung der vom System erstellten Solids erfolgt nach einem einfachen Schema. Die Liste aller Solids wird durchiteriert und für jedes Solid nacheinander ein Polygonnetz aufgebaut.

4.4.1 Aufbau vertikaler Flächen

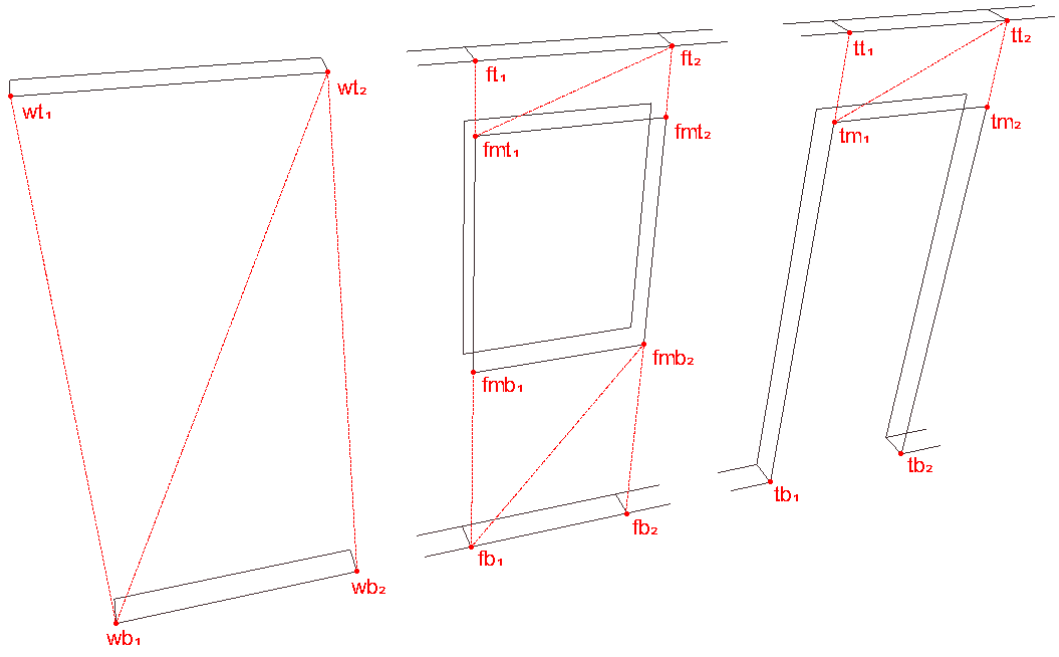

Abbildung 4.2: Vertikaler Polygonaufbau verschiedener Elementtypen

Im ersten Schritt wird die Darstellung der vertikalen Flächen umgesetzt. Für jede Linie eines Solids werden die beiden Vektoren entsprechend des Elemtenttyps auf verschiedene Höhen dupliziert. Die verschiedenen Höhenebenen der Elemente sowie die Belegung der vertikalen Flächen mit Polygonen ist in Abbildung 4.2 dargestellt.

Für ein Wandelement müssen die beiden Vektoren wb_1 und wb_2 der Bodenlinie kopiert und ihre Höhenordinate auf die maximale Wandhöhe geändert werden, sodass sich die Vektoren wt_1 und wt_2 ergeben. Nun kann mit den drei Vektoren wb_1, wb_2 und wt_2 ein Polygon konstruiert werden. Das zweite Polygon ergibt sich aus den Vektoren wb_1, wt_1 und wt_2. Dieser Schritt erfolgt für jede Linie eines Solids, sodass alle im Grundriss verzeichneten Seiten des Solids mit geschlossenen Flächen versehen sind.

Für die Linien eines Fensterelements müssen die Vektoren auf der Bodenlinie auf jeweils drei verschiedene Höhenwerte dupliziert werden. Die Konfiguration enthält für diesen Zweck Werte für die Höhe der Unter- und Oberkante der Fensteröffnung. Zusammen mit der Höhe der Wandoberkante kann die Darstellung dann durch vier Polygone zusammengesetzt werden. Dabei wird die Fläche unterhalb der Fensteröffnung durch ein Polygon aus den Punkten fb_1, fb_2 und fmb_2 sowie ein Polygon aus den Punkten fb_1, fmb_1 und fmb_2 zusammengesetzt. Die Fläche oberhalb der Fensteröffnung bildet sich dann aus dem Polygon mit den Vektoren fmt_1, fmt_2 und ft_2 sowie einem Polygon mit den Vektoren fmt_1, ft_1 und ft_2.

Für die vertikalen Flächen einer Türöffnung werden pro Linie wiederum lediglich 2 Polygone benötigt, da nur eine viereckige Fläche gebildet wird. Die Vektoren jeder Bodenlinie werden dafür auf zwei verschiedene Höhen dupliziert, und zwar sowohl auf den in der Konfiguration ebenfalls definierten Höhenwert für die Oberkante der Türöffnung als auch den Höhenwert der Wandoberkante, der bereits bei den zuvor beschriebenen Elementtypen verwendet worden ist. Zwischen der Türöffnungoberkante und der Wandoberkante werden dann die benötigten zwei Polygone aus den Vektoren tm_1, tm_2 und tt_2 sowie den Vektoren tm_1, tt_1 und tt_2 gebildet.

Die vertikalen Flächen innerhalb der Fenster- und Türlaibungen orthogonal zum Wandverlauf werden in diesem Verfahren automatisch durch die Renderung aller Wandelementlinien geschlossen dargestellt. Sollten in einem Grundriss an diesen Stellen keine schließenden Linien an den Elementgrenzen vorhanden sein, müssten diese für eine korrekte Darstellung von Fenster- und Türöffnungen ergänzt werden.

4.4.2 Aufbau horizontaler Flächen

Nachdem nun das Polygonnetz eines Solids bereits alle vertikalen Flächen enthält, muss es noch um die notwendigen horizontalen Flächen ergänzt werden. Auch in diesem Schritt ist wiederum

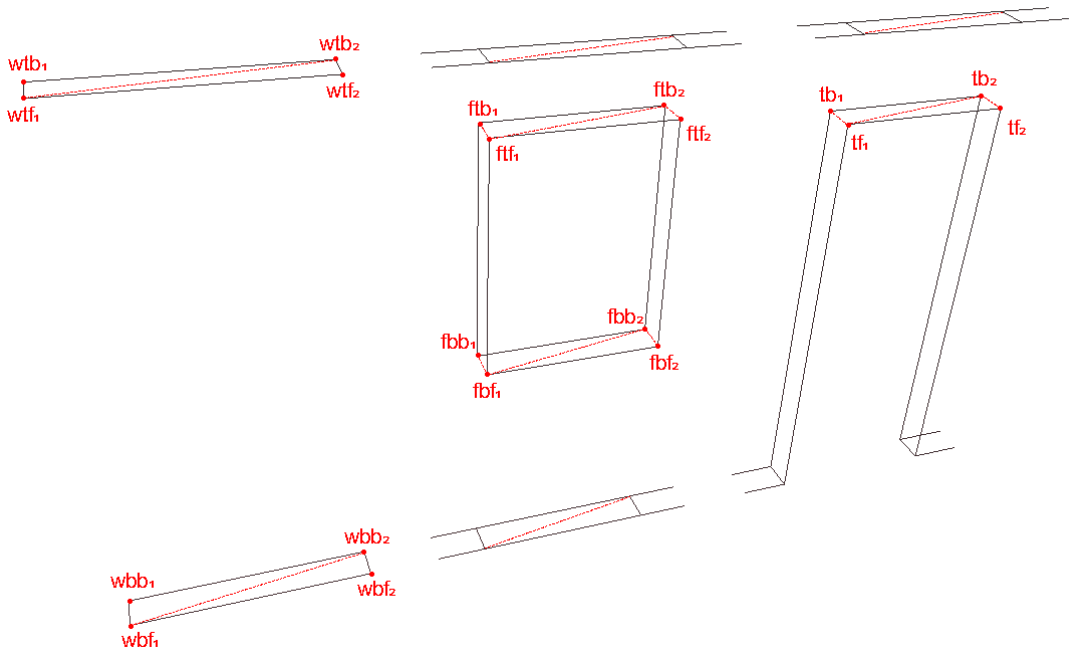

Abbildung 4.3: Horizontaler Polygonaufbau verschiedener Elementtypen

relevant, um welche Art von Element es sich bei dem Solid handelt, wie die Abbildung 4.3 zeigen soll.

Für die Schließung der Flächen werden ausschließlich die beiden Basislinien jedes Solids verwendet. Die Bildung zweier Polygone zu einem vollständigen Viereck erfordert dabei, dass die beiden Linien zunächst auf ihre Richtung hin untersucht werden. Die normalisierten Richtungsvektoren beider Linien werden gebildet und verglichen. Anhand dieses Vergleichs kann dann im weiteren Verlauf entschieden werden, wie die vier Vektoren und ihre jeweiligen Duplikate zu den zwei Gruppen von jeweils drei Vektoren zusammengesetzt werden müssen, um Lücken in der Darstellung der Flächen zu verhindern.

Nun wird geprüft, ob es sich bei dem aktuellen Solid um eine Türelement handelt. Ist dies nicht der Fall, so kann die Bodenfläche, in der Abbildung 4.3 im Falle des Wandelements mit den Vektoren wbf_1, wbf_2, wbb_1 und wbb_2 markiert, mit Polygonen versehen werden. Diese muss sowohl bei Wand- als auch bei Fensterelementen geschlossen dargestellt werden. Lediglich Türöffnungen erfordern ein Weglassen dieser Polygone.

Wenn es sich um ein Türelement handelt, werden die vier Vektoren der Basislinie mithilfe des Werts für die Türöffnungshöhe aus der Konfiguration entsprechend zu den Vektoren tf_1, tf_2, tb_1 und tb_2 dupliziert und zu einer horizontalen, viereckigen Fläche auf der erwähnten

Höhe aus zwei Polygonen zusammengesetzt. Dadurch ist die Unterseite der Wand oberhalb der Türöffnung nun geschlossen.

Handelt es sich bei dem aktuellen Element um ein Fenster, werden die beiden Flächen der Ober- und Unterseite der Fensteröffnung erstellt. Für diesen Zweck werden die vier Vektoren der Basislinien zweimal dupliziert. Die Höhenwerte für diese zwei Flächen sind ebenfalls in der Konfigurationsdatei enthalten. Mithilfe dieser Höhenwerte ergeben sich die vier Vektoren fbf_1, fbf_2, fbb_1 und fbb_2 auf Höhe der Fensteröffnungsunterkante und die vier Vektoren ftf_1, ftf_2, ftb_1 und ftb_2 auf Höhe der Fensteröffnungsoberkante. Diese beiden Gruppen aus jeweils vier Vektoren lassen sich dann zu jeweils zwei Polygonen kombinieren, sodass die horizontalen Flächen der Fensteröffnung nun ebenfalls geschlossen dargestellt werden.

Zum Schluss muss nun noch unabhängig vom Typ des Solids die horizontale Fläche an der Oberseite durch zwei Polygone geschlossen dargestellt werden. Diese ist in der Abbildung 4.3 am Wandelement beispielhaft mit den Vektoren wtf_1, wtf_2, wtb_1 und wtb_2 markiert.

Ein Beispiel für das Gesamtergebnis dieser einzelnen Schritte stellt die Abbildung 4.4 dar.

Abbildung 4.4: Vollständig gerenderte Elementtypen

5 Implementierung

Das System erfordert für die Ausführung einen Parameter beim Aufruf der Schnittstellenmethode beziehungsweise ein Argument beim Ausführen über die Systemkonsole. Dieser Parameter ist der Dateiname der DXF- und XML-Datei des darzustellenden Grundrisses ohne die Dateiendung.

5.1 XML Konfigurationsdatei

Die Konfiuration enthält unterhalb des Dokument-Elements, welches als Attribut den Dateinamen ohne Endung des zugehörigen Grundrisses hat, zwei verschiedene Knotentypen. Das attributlose *config* Element enthält alle Konfigurationswerte als Kindknoten mit dem konkreten Werten als Inhalt. Die *layer* Elemente definieren als Attribute die Namen der verschiedenen Elementebenen im DXF sowie den jeweils auf den Ebenen enthalten Elementtyp.

5.1.1 Konfigurationswerte des *config* Elements

Die nachfolgenden Werte sind für das System für die verschiedenen Arbeitsschritte erforderlich und beeinflussen sowohl die Analyse des Grundrisses als auch die dreidimensionale Darstellung. Um aus einer DXF-Datei eine verwendbare Darstellung zu erhalten, müssen diese Werte passend justiert werden. Alle Werte können als ganze Zahlen oder mit Dezimalstellen angegeben werden, sollten sich jedoch immer ausschließlich im positiven Bereich bewegen.

wall_top_height Höhe der Wandoberkante; beeinflußt die Darstellung; muss zur Einheit von D in Abschnitt 4.1.3 passen

window_bottom_height Höhe der Fensteröffnungsunterkante; beeinflußt die Darstellung; muss zur Einheit von D in Abschnitt 4.1.3 passen

window_top_height Höhe der Fensteröffnungsunterkante; beeinflußt die Darstellung; muss zur Einheit von D in Abschnitt 4.1.3 passen

door_top_height Höhe der Türöffnungskante; beeinflußt die Darstellung; muss zur Einheit von D in Abschnitt 4.1.3 passen

groundplan_scaling_scalar Skalierungsfaktor; beschrieben in Abschnitt 4.1.3

angle_tolerance Winkeltoleranz; Gradzahl zwischen 0° und 90°

point_tolerance Punkttoleranz; muss zur Einheit von D in Abschnitt 4.1.3 passen

distance_interval Abstandsintervallgröße i_d; Erfassung von Abstandshäufigkeiten wird in Gruppen der Größe i_d eingeteilt

lower_wallthickness_limit Untere Grenze für den Bereich relevanter Abstände

upper_wallthickness_limit Obere Grenze für den Bereich relevanter Abstände

wall_count_threshold Schwellwert für die Häufigkeit relevanter Abstände

5.1.2 Eigenschaften der *layer* Elemente

Ein *layer* Element hat die Attribute *name* und *type*. Das *name*-Attribut gibt dabei den exakten Namen eines Layers im DXF-Dokument an und das *type*-Attribut weist der genannten DXF-Ebene einen Darstellungstypen zu. Mögliche Darstellungstypen sind *WALL*, *WINDOW* und *DOOR*. Mit diesen Attributen lässt sich steuern, wie alle auf einem Layer befindlichen Solids bei der grafischen Darstellung gehandhabt werden müssen.

5.2 Packet- und Klassenstruktur

Nachfolgend werden die für das System implementierten Klassen der package-Struktur *smarthomevis.groundplan*[1] und ihre jeweilige Funktion erläutert.

5.2.1 Interface IGroundPlan.java und Haupteinstiegsklasse GroundPlan.java

Das System besitzt für die Einbindung anderer Anwendungen eine Interface-Klasse *IGroundPlan.java*. Diese dient dem gekapselten Zugriff auf die Funktionen des in dieser Ausarbeitung entwickelten Systems.

Für die Ausführung als eigenständige Anwendung beinhaltet die Klasse *GroundPlan.java* eine *main* Methode. Sie bindet außerdem das Interface *IGroundPlan.java* ein und dient als Implementationsklasse des Interfaces für den externen Aufruf der Funktionalitäten des Systems.

[1] vollständiger Pfad im Projekt: [smart_home_visualization\src\main\java\smarthomevis\groundplan]

IGroundPlan.java

Dieses Interface bietet drei Methoden für den externen Aufruf an:

CGNode convertDXFPlanToCgNode(String gpName)

Analysiert einen DXF Grundriss unter der Berücksichtigung der zugehörigen Konfigurationsparameter und generiert eine auf Solids basierende Darstellung zur Rueckgabe. Als Parameter ist der Dateiname des Grundrisses ohne Dateiendung erforderlich. Setzt das Vorhandensein einer Konfigurationsdatei im XML Format mit gleichem Dateinamen voraus.

void renderAndDisplayPlan(String planName)

Analysiert einen DXF Grundriss unter der Berücksichtigung der zugehörigen Konfigurationsparameter und generiert eine auf Solids basierende Darstellung. Diese wird dann mit dem *JOGL* Anwendungsrahmen des *computergraphics* Frameworks dargestellt.

CgNode construct3DMeshFromData(GPDataType data)

Nimmt als Parameter ein Objekt der Klasse *GPDataType.java* entgegen und stellt dieses nur anhand der darin enthaltenen *GPLine.java* Objekte dar. Dieser Aufruf ist veraltet und führt zu nicht vollständig aufgebauten Repräsentationen eines Grundrisses.

GroundPlan.java

Implementiert die Methoden des Interface *IGroundPlan.java* und enthält die *main* Methode als Einstiegspunkt zur eigenständigen Ausführung des Programms. Die *main* Methode erwartet als Parameter den Dateinamen des darzustellenden Grundrisses ohne Dateiendung und erfordert außerdem das Vorhandensein einer gleichnamigen XML-Konfigurationsdatei. Wird dem Aufruf kein Dateiname als Argument übergeben, wird ein allgemeiner Test-Grundriss ("*4H-HORA Projekt1.dxf*") beispielhaft analysiert und dargestellt.

Die Klasse enthält ausserdem eine Reihe von Testmethoden zur unterschiedlich gestalteten Renderung von Grundrissdaten, die im Laufe der Systementwicklung veraltet sind.

5.2.2 Java™ package *smarthomevis.groundplan.config*

GPDataImporter.java

Hauptkontrollklasse für den Systeminput. Liest mithilfe von GPConfigXMLReader.java die XML-Konfigurationswerte ein und steuert das Importieren aller relevanten DXF Daten. Wandelt alle DXF Entities in neue Vektordatenobjekte um, nimmt dabei die Skalierung der Vektordaten vor und speichert die Referenzen aller transformierten Daten in einer Instanz der Klasse *GPDataType.java*.

GPConfig.java

Implementierungsklasse für die Xml-Konfigurationsinhalte. Enthält Methoden für den Zugriff auf Konfigurationswerte, das Hinzufügen von Werten sowie die eindeutigen Bezeichner aller in der XML-Datei enthaltenen *config*-Elemente.

GPConfigXMLReader.java

Klasse für das Einlesen von XML-Daten. Liest die Konfigurationsdatei vollständig ein und legt alle Werte in einer Instanz der Klasse *GPConfig.java* ab.

5.2.3 Java™ package *smarthomevis.groundplan.data*

GPDataType.java

Transport- und Persistenzklasse für importierte Daten sowie alle vom System generierten Objekte und Relationen. Hält außerdem eine Referenz auf das *GPConfig* Objekt mit allen Konfigurationswerten.

GPLine.java

Klasse zur systeminternen Repräsentation von Linienobjekten. Enthält zwei Vektoren als Start- und Endpunkt der repräsentierten Linie in Form von *cgresearch.core.math.Vector* Objekten. Enthält außerdem einen *String* mit der eindeutigen Bezeichnung der Linie sowie einen Enumerator *LineType* zur Definition des Elementtyps einer *GPLine* Instanz. *LineType* kann die Werte *WALL*, *DOOR* und *WINDOW* annehmen.

GPSolid.java

Klasse zur systeminternen Repräsentation von Solidobjekten. Enthält eine Liste von *GPLine* Objekten der zu diesem Solid gehörenden Linien. Enthält außerdem eine gesonderte Referenz auf die zwei benachbarten Paarungslinien, die zur Generierung des *GPSoldid* Objekts geführt haben.

5.2.4 GPAnalyzer.java

Die Klasse *GPAnalyzer.java* nimmt in der Methode *Map<Double, List<String[]» analyzeData(GPDataType data)* die importierten und bereits skalierten Vektordaten des DXF-Dokuments in Form eines *GPDataType* Objekts entgegen. Diese Daten werden dann, wie im Abschnitt 4.2 erläutert, analysiert und zu Linienpaarungen sortiert. Die Rückgabe der Methode ist eine *Map* der Distanzen und dazugehörende Linienpaarungen. Die Tabelle der berechneten Abstände

und ihrer Häufigkeiten wird außerdem aus dieser Klasse heraus in einer csv-Datei zur externen Betrachtung im Dateisystem abgelegt.

5.2.5 GPEvaluator.java

Die Klasse *GPEvaluator.java* nimmt in der Methode *GPDataType processData(GPDataType data, Map<Double, List<String[]» pairsToDistanceMap)* die voranalysierten Daten der Klasse *GPAnalyzer.java* entgegen. Sie nimmt dann die Bewertung der Abstandsdaten und Paarbildungen vor und generiert, wie im Abschnitt 4.3 erläutert, die Solids in Form von Objekten der Klasse *GPSolid.java*. Die generierten Solids werden dann als Referenz an das *GPDataType* Objekt des Systems übergeben.

5.2.6 GPRenderer.java

Die Klasse *GPRenderer.java* ist für die Umsetzung der verschiedenen Darstellungsarten zuständig. Im Laufe der Systementwicklung sind dabei verschiedene Abstufungen des Darstellungsaufbaus entstanden. Objekte der Klasse *GPRenderer.java* erfordern bei der Instanzierung die Übergabe einer *GPDataType* Instanz mit den gesammelten und generierten Datenobjekten zur Darstellung. Die wichtigste Methode zur Generierung einer Darstellung ist dabei die Umsetzung auf Basis von *GPSolid* Objekten durch den Aufruf der Methode *CgNode render3DMeshViewFromSolids()*.

Methoden zur Darstellungsgenerierung sind:

CgNode render3DMeshViewFromSolids() dient der vollständigen Darstellung aller vom System generierten Solids auf Basis von Polygonen wie im Kapitel 4 erläutert.

CgNode renderScaled3DMeshViewOfLayer(String layerName, List<GPLine> lineList) dient der Darstellung aller importierten Linienobjekte auf Basis von Polygonen ohne Berücksichtigung von horizontalen Flächen.

CgNode render3DGridViewFromGPDataType() dient der Darstellung des Grundrisses in einer auf Linien basierten Form (z.B. verwendet in Darstellungen 4.2 und 4.3) ohne die Verwendung von Polygonen.

CgNode render2DViewFromGPDataType() dient der Darstellung der importierten Grundrissrohdaten in zweidimensionaler Form.

5.2.7 GPUtility.java

Die Klasse *GPUtility.java* enthält eine Reihe von statischen Hilfsmethoden für die Bearbeitung von *List* Objekten, die Ausgabe der Histogrammdaten in eine csv-Datei sowie sämtliche als Hilfsmittel notwendigen Implementierungen von Vektoroperationen. Diese Methoden bedienen sich dabei für Berechnungen der Klasse *java.math.BigDecimal* zur Vermeidung von Rundungsfehlern.

5.3 Verwendung systemfremder Klassen

Neben einer Reihe von Klassen der Standard API der Java™ Umgebung finden im System verschiedene Klassen externer Bibliotheken Verwendung:

5.3.1 Klassen der *Kabeja* Bibliothek

Die Verwendung von Klassen der *kabeja*-Bibliothek beschränkt sich im System ausschließlich auf die Klasse *smarthomevis.groundplan.config.GPDataImporter.java*. Die in Form von Kabejaklassen importierten Vektordaten werden innerhalb des Importers in systemeigene Klassen sowie Klassen des *cg Frameworks* umgewandelt.

5.3.2 Klassen der *computergraphics* Bibliothek

Die Handhabung von Vektordaten erfolgt im System ausschließlich mithilfe der *cg Framework* Klasse *cgresearch.core.math.Vector*. Diese Klasse dient in den systemeigenen Datenklassen *GPLine.java* und *GPSolid.java* der Repräsentation von dreidimensionalen Vektordaten und wird aus diesem Grund ebenfalls in allen anderen Klassen des Systems als Abhängigkeit importiert.

Des Weiteren bedienen sich die Klassen *GroundPlan.java* und *GPRenderer.java* sowie Klassen des packages *smarthomevis.groundplan.config* einer Reihe von Klassen aus den *computergraphics* packages *cgresearch*, *cgresearch.graphics* und *cgresearch.core* des *cg Frameworks*.

Generierung und Rückgabe der Darstellung

Für den Aufbau der Polygonnetze sowie die Rückgabe der fertigen Darstellung werden Instanzen der Klasse *cgresearch.graphics.scenegraph.CgNode* verwendet.

6 Evaluation

Für den Test der Implementierung wurden eine Reihe von Dateien verwendet. Dazu gehören sowohl aus dem Internet bezogenes Beispielmaterial als auch selbst erstellte Testgrafiken für die Prüfung von verschiedenen Linien- und Lückenkonstellationen.

Die verwendeten Grundrisse und Testdateien werden nachfolgend vorgestellt. Im Anschluss werden noch ungelöste Sonderfälle erläutert und mögliche Lösungen diskutiert.

6.1 Verwendete DXF Testdateien

Die Qualität der verwendeten Grundrisse ist entscheidend für den Erfolg der zuvor beschriebenen Prozesse. Die implementierten Algorithmen arbeiten unter den richtigen Bedingungen zuverlässig. Jedoch gibt es einige Konstellationen von Linien, die von den entwickelten Methoden nicht sauber übersetzt werden können und daher zu einer unsauberen Darstellung führen.

Die Gestaltung der Grundrisse ist stets abhängig vom verwendeten Programm und der Arbeitsweise des Erstellers. Die für die Entwicklung des Systems verwendeten fremden Beispieldateien sowie die selbst erstellten DXF-Testdateien werden zu einem hohen Grad korrekt dargestellt.

4H-HORA Projekt1.dxf

Der Beispielgrundriss *4H-HORA Projekt1* ist eine aus dem Internet heruntergeladene DXF Datei, die für die Erstellung eines Gebäudes in Holztafelbauweise gedient hat. Auf der Webseite sind die genauen Außenmaße des Gebäudes sowie die verwendeten Wandstärken für außen- und innenliegende Wände angegeben (pcae GmbH, 2016, vgl.). Mit diesen konkreten Maßen konnte die Bestimmung des Skalierungsfaktors sowie die Selektion von relevanten Abständen getestet werden.

Für die Verwendung dieser Datei mussten die in Abschnitt 4.1.1 beschriebenen Schritte zur Vorbereitung durchgeführt werden. Mit einem Grafikprogramm wurden dazu zwei weitere Ebenen hinzugefügt und die entsprechenden Elementlinien der Vektorgrafik auf die jeweiligen Ebenen verschoben.

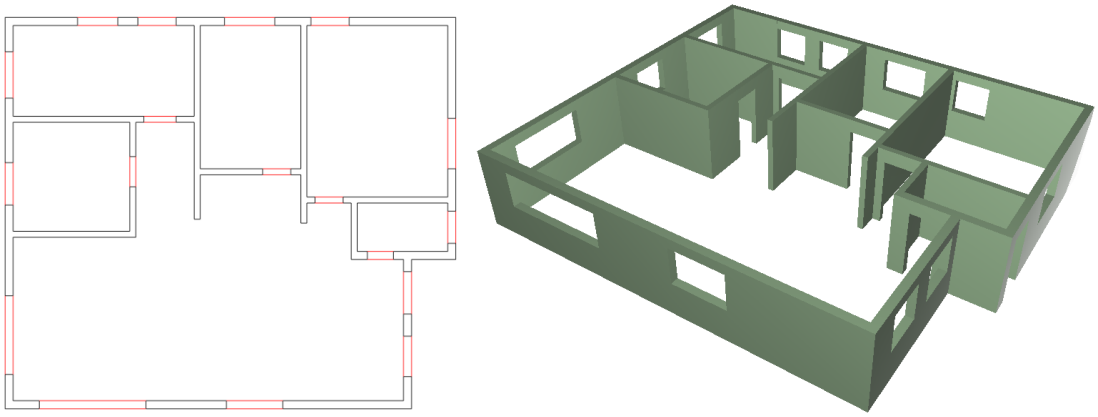

Abbildung 6.1: Beispielgrundriss *4H-HORA Projekt1*

Im weiteren Verlauf der Ausarbeitung haben Tests mit diesem Grundriss grundlegende Problemstellungen im Bereich der Solidbildung und der Schließung von Lücken aufgezeigt, welche nun analysiert und vom System entsprechend gehandhabt werden können (siehe Abschnitte 3.3.2 und 4.3.3).

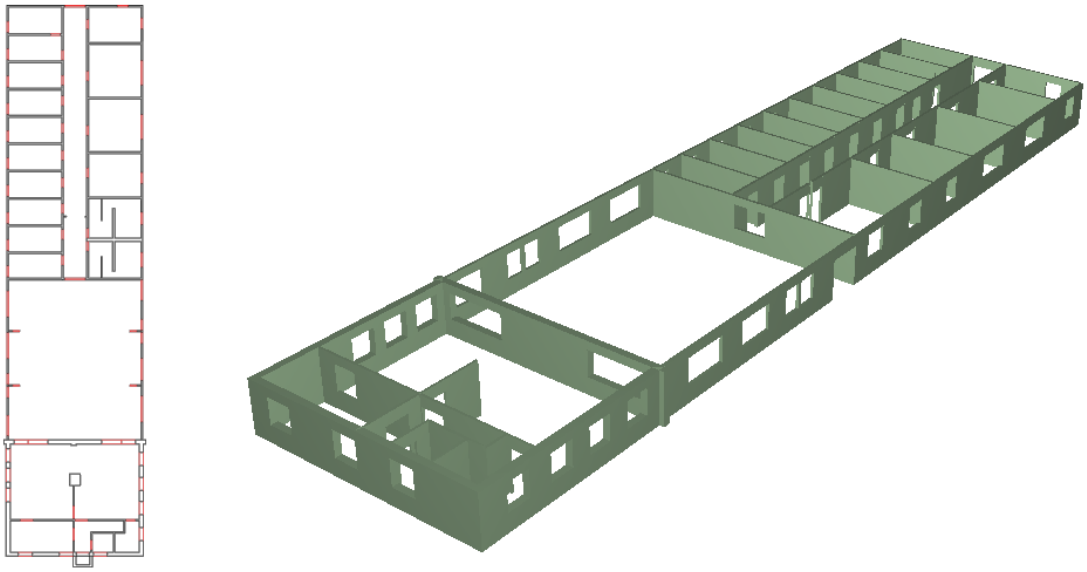

Abbildung 6.2: Beispielgrundriss *Grundriss_Haus_02*

Grundriss_Haus_02.dxf

Grundriss_Haus_02 ist ebenfalls ein aus dem Internet[1] geladener Testgrundriss. Dieser Grundriss zeigt Grenzen des Systems auf. Einige Bereiche dieses komplexen Gebäudes werden nicht vollständig dargestellt. Diese Grenzbereiche werden im weiteren Verlauf dieses Kapitels noch genauer behandelt.

In diesem Grundriss sind die verschiedenen architektonischen Elemente (Fenster, Türen und Wände) bereits von vornherein auf getrennten Ebenen gelagert. Der Skalierungsfaktor ist durch die standardisierten Maße für Türöffnungen bestimmt worden.

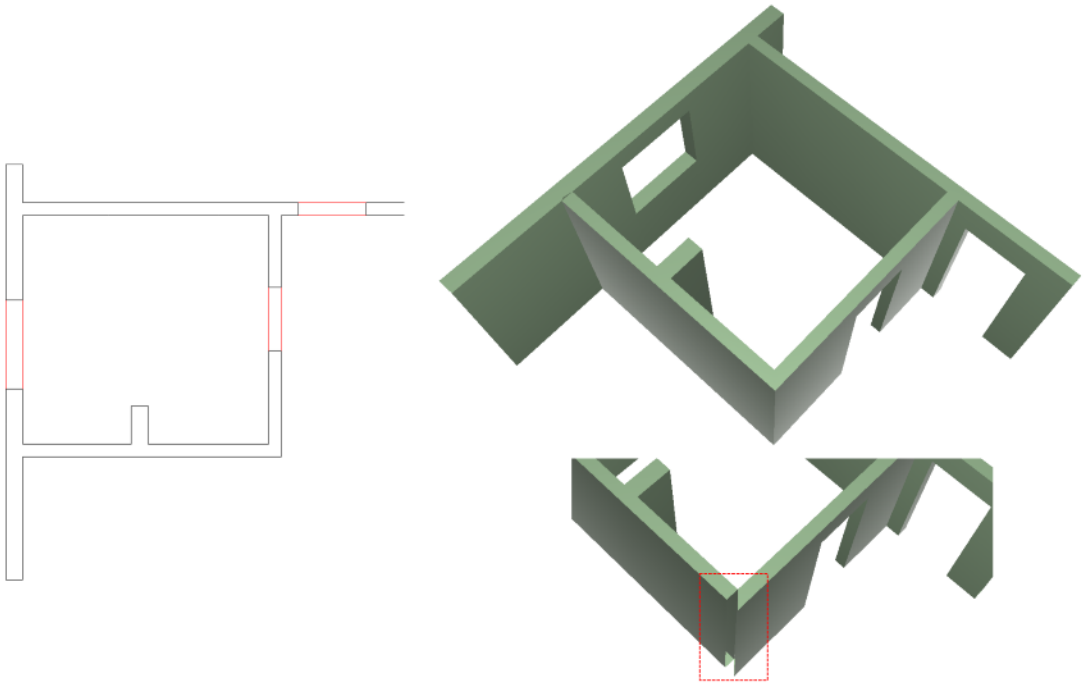

Abbildung 6.3: Testgrafik *TestRaum3*

TestRaum3.dxf

Die Grafik *TestRaum3.dxf* ist eine Untermenge der Grafik *4H-HORA Projekt1.dxf* und wurde für verschiedene Testszenarios mehrfach modifiziert (Versionen 1-3). Die hier genannte letzte Iteration zeigte eine ursprünglich nicht behandelte Konstellation aus Lücken und überstehenden Multilines auf, die in keinem der zuvor erwähnten Grundrisse vorgekommen ist. Anhand der

[1] https://projekt.beuth-hochschule.de/ping/graebendorf-2013/download/cad-zeichnungen/ (Stand 05.09.2016)

Testdatei konnte dieses Problem (rot gestricheltes Rechteck in Abbildung 6.3) entdeckt und behoben werden (siehe Abschnitte 3.3.4 und 4.3.3).

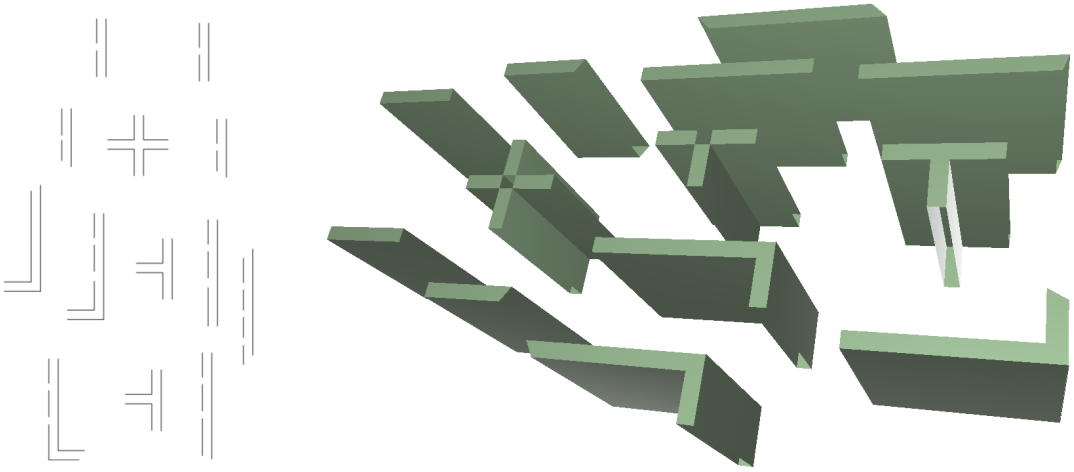

Abbildung 6.4: Testgrafik *lueckenTest*

lueckenTest.dxf

Die Datei *lueckenTest.dxf* wurde angefertigt, um die Darstellung von verschiedenen Kombinationen aus Linienunterbrechungen mit Lücken, Linienunterbrechungen ohne Lücken (in Abbildung 6.4 nicht verdeutlicht) sowie Wandecken und -kreuzungen zu prüfen. Anhand der Abbildung 6.4 wird sichtbar, dass zwei weitere Sonderfälle aus Linienkonstellationen vom System nicht korrekt umgesetzt werden können. Diese Sonderfälle werden im weiteren Verlauf dieses Kapitels ebenfalls noch genauer behandelt.

6.2 Nicht gelöste Sonderfälle

Die vollständige Darstellung vertikaler Flächen wurde schon zu Beginn der Ausarbeitung schnell erreicht. Dabei wurde jedoch früh deutlich, dass für eine saubere Darstellung von horizontalen Flächen eine generalisierende, komplexere Lösung notwendig sein würde. Die Komplexität des entwickelten Systems liegt hauptsächlich in der Einteilung einheitlicher Wandabschnitte: den Solids. Während der Standardfall der Solidbildung mit zwei parallelen Linien aus nur einer Paarung eine einfache Lösung besitzt, sind im Laufe der Ausarbeitung nach und nach komplexere Problemstellungen offenkundig geworden.

Die Lösungsfindung für die regelmäßig auftretenden Lücken in Wänden durch orthogonal angrenzende Wandabschnitte und die vielen Variationen mit beliebig vielen beteiligten Linien und mehreren Lücken erfordert eine allgemeine Problembetrachtung. Für diese Problemstellungen mussten weitere verlässliche Methoden mithilfe der linearen Algebra entwickelt werden, um Linienkonstellationen in jeder Variante lückenlos in einzelne Wandabschnitte einzuteilen. Es existieren jedoch noch Grenzfälle, die vom System nicht korrekt gehandhabt werden können.

Die nachfolgend beschriebenen Sonderfälle 6.2.1 und 6.2.2 sind in der Abbildung 6.4 bereits erkennbar. Theoretische Lösungsansätze für diese Probleme sind vorhanden, wurden jedoch aufgrund ihrer Komplexität und technischen Unvollständigkeit nicht mehr implementiert. Die möglichen Lösungen werden im Anschluss an die Beschreibung der Problemfälle erläutert.

6.2.1 Sonderfall: T-förmige Wandkreuzung mit unterbrochener Linie

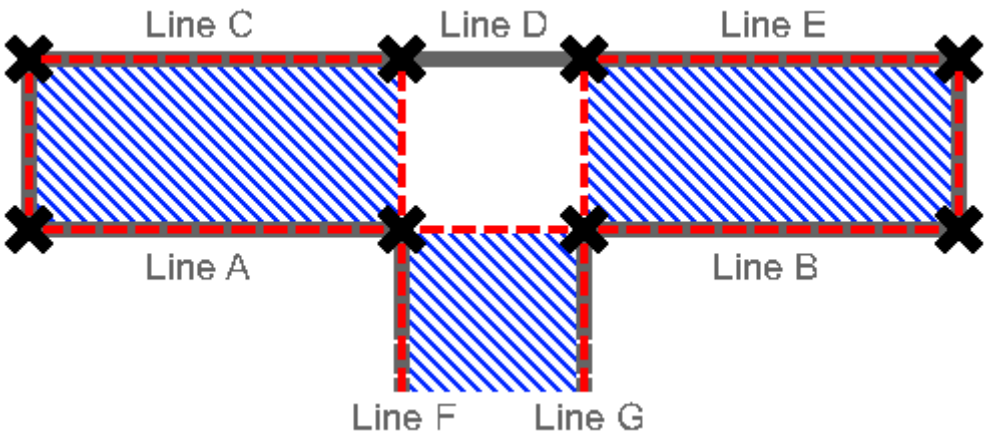

Abbildung 6.5: Ungelöster Sonderfall T-Kreuzung mit freier Linie *Line D*

Der in Abbildung 6.5 dargestellte Sonderfall unterscheidet sich von den zuvor behandelten Problemstellungen in dem Punkt, dass die durchgehende *Multiline* aus Abbildung 3.4 in diesem Fall in drei einzelne Linien *Line C*, *Line D* und *Line E* unterteilt ist. Durch diese spezielle Unterteilung betrachtet das System diesen Abschnitt nicht als ganzheitlichen Problemfall, sondern bildet die Solids der Paarungen [*Line A, Line C*], [*Line B, Line E*] und [*Line F, Line G*] unabhängig voneinander nach dem Verfahren für einfache Paarungen. Dabei bleibt der Wandabschnitt von *Line D* unbearbeitet, da dieser Linie keine Nachbarlinie gegenüber zu stellen ist.

Variante des Sonderfalls

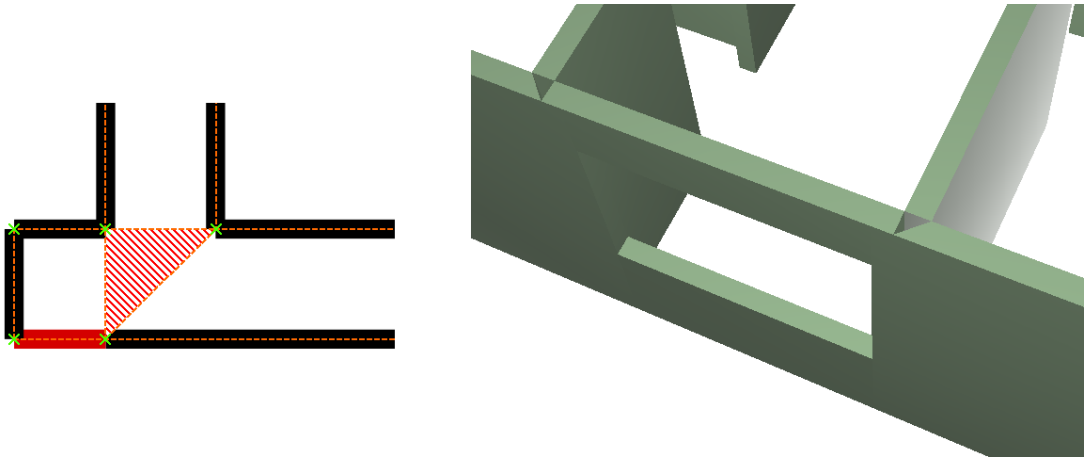

Abbildung 6.6: Variante des Sonderfalls 6.2.1

Die in Abbildung 6.6 dargestellte Variante des Problems 6.2.1 ist der Darstellung *Grundriss_Haus_02* direkt entnommen. Dabei entsteht durch die rot eingefärbte Linie eine Konstellation, die zu einer unvollständigen Darstellung führt. Auch in diesem Fall entsteht die dreieckige, rot markierte freie Fläche durch die voneinander unabhängige Betrachtung der drei angrenzenden Solids.

Lösungsansatz

Um diese speziellen Lücken zu identifizieren und zu schließen, kann ein Alleinstellungsmerkmal der beteiligten Vektoren in Relation zu ihren Solids genutzt werden. Die vom System gebildeten Solids haben stets vier Eckpunkte. In allen zuvor behandelten Fällen besitzen zwei benachbarte Solids stets zwei gemeinsame Punkte, beziehungsweise zwei annähernd identische Vektoren.

Diese Regel gilt für alle benachbarten Solids bis auf diejenigen Nachbarschaften, die in den genannten Sonderfall passen. Solids, die direkt an eine der beschriebenen Lücken grenzen, können mehr als einen Eckpunkt mit anderen Solids teilen. Die relevanten Eckpunkte sind jedoch diejenigen Verbindungspunkte, deren beteiligte Nachbarsolids ausschließlich diesen einen Punkt gemeinsam haben.

Um also die relevanten Vektoren zu erhalten, muss zunächst eine Betrachtung der Nachbarschaftsbeziehung von Solids angestellt werden. Dabei wird ein Hauptaugenmerk auf diejenigen Punkte gelegt, die von zwei Solids ohne weitere gemeinsame Punkte geteilt werden.

Die Isolation dieser gesuchten Vektoren ist im System testweise implementiert worden[2], ein hinreichend zuverlässiges Regelwerk für die weitere Verarbeitung der gefundenen Vektoren zu Solids wurde jedoch nicht implementiert.

Ein denkbarer Ansatz, erläutert anhand des Beispiels in Abbildung 6.5, ist die Identifizierung des Solids [*Line F, Line G*] in der gezeigten Konstellation anhand der zwei einzeln benachbarten Punkte. Die beiden Solids [*Line A, Line C*] und [*Line B, Line E*] weisen lediglich einen einzeln benachbarten Punkt auf. Anhand dieses Kriteriums könnten die Linien *Line F* und *Line G* dann bis auf die Punkte der freien Linie *Line D* extrapoliert und bei einer Übereinstimmung zur Linie im entsprechenden Solid korrigiert werden.

Ein anderer Ansatz, ebenfalls an der Zeichnung erläutert, wäre eine vergleichende Betrachtung der freien Linie *Line D*, die in den vorangegangenen Schritten keinem Solid zugewiesen werden konnte, mit den nicht in eine Nachbarschaft eingebundenen Vektoren der Linien *Line C* und *Line E*. Sind beide Vektoren der Linie *Line D* eindeutig den beiden freien Vektoren der Solids [*Line A, Line C*] und [*Line B, Line E*] zuzuweisen, wäre ein erster Anhaltspunkt bewiesen. Nun könnte die einzelne, symmetrische Relation der zwei erwähnten Solids zum Solid [*Line F, Line G*] dazu genutzt werden, diese einzeln benachbarten Vektoren zu identifizieren und mit diesen und den Vektoren der Linie *Line D* einen neuen Solid zu erstellen.

Beide Ansätze müssten jedoch für die zweite Variante des Sonderfalls entsprechend abgeändert werden.

Für eine direkte Behebung des Problems kann außerdem die entsprechende Wandkonstellation mit einer Grafiksoftware derart geändert werden, dass diese Sonderfälle nicht länger bestehen. Bei der in Abbildung 6.5 gezeigten Problematik müssten zwei der Linien *Line C*, *Line D* und *Line E* entfernt und die dritte auf die gesamte Länge erweitert werden. Im Fall der Abbildung 6.6 wäre das Entfernen der rot dargestellten Linie sowie die Verlängerung der gleichförmig anschließenden Linie bis zum Wandecke ausreichend.

6.2.2 Sonderfall: X-förmige Wandkreuzung ohne freie Linien

In dem in Abbildung 6.7 gezeigten Sonderfall ist keine unzugeordnete Linie beteiligt. Der Sonderfall ist dennoch ähnlich der in Abschnitt 6.2.1 beschriebenen Konstellation. Aufgrund der vier einzeln gebildeten Solids ist in diesem Fall eine Lücke in der Darstellung vorhanden.

[2] Methoden [*GPEvaluator.sortSolidsAfterAlignedCorners()*] und [*GPEvaluator.analyzeMatchedCornersAndBuildMissingSolids(...)*]

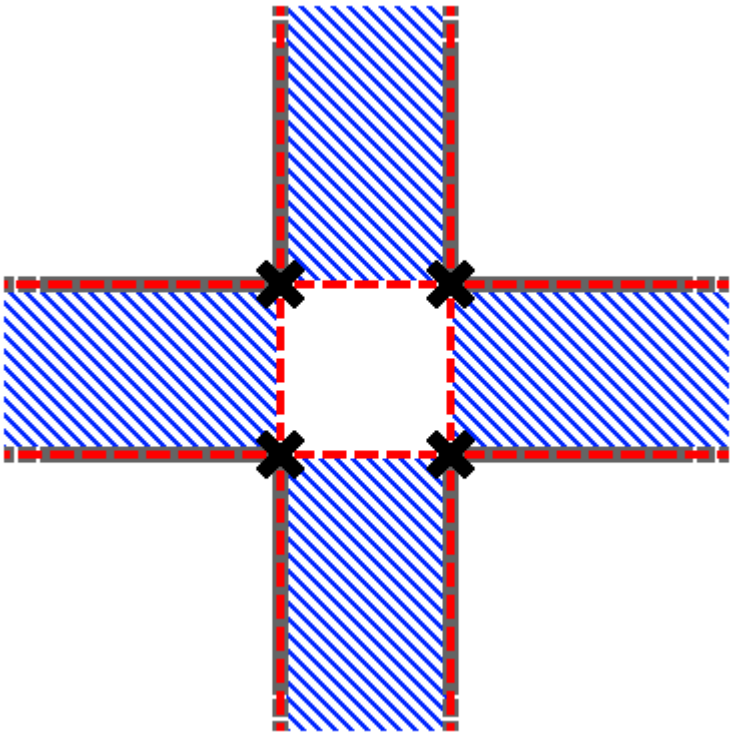

Abbildung 6.7: Ungelöster Sonderfall zentrierte X-Kreuzung

Lösungsansatz

Auch für diesen Sonderfall kann zunächst eine Identifizierung der beteiligten Punkte anhand ihrer einzelnen Beziehung zum angrenzenden Solid durchgeführt werden. Anhand der vier gefundenen Punkte beziehungsweise der acht zugehörigen Vektoren müsste die Kreisbeziehung der vier Solids eindeutig erkannt werden, um dann unter Verwendung der beteiligten Vektoren zwei parallele Linien zu bilden und mit diesen einen Solid zu erstellen.

6.3 Vereinfachung der Polygonbildung

Ursprünglich erforderte das *computergraphics* Framework beim Vorgehen der Polygonerzeugung eine von der Position des Betrachters abhängige Reihenfolge. Diese Anforderung wurde bereits zu Beginn der Umsetzung des Systems aus dem Framework entfernt[3], sodass eine akute Beachtung bei der Erzeugung der Polygone nicht weiter notwendig war.

[3] eventuell nur vorläufig

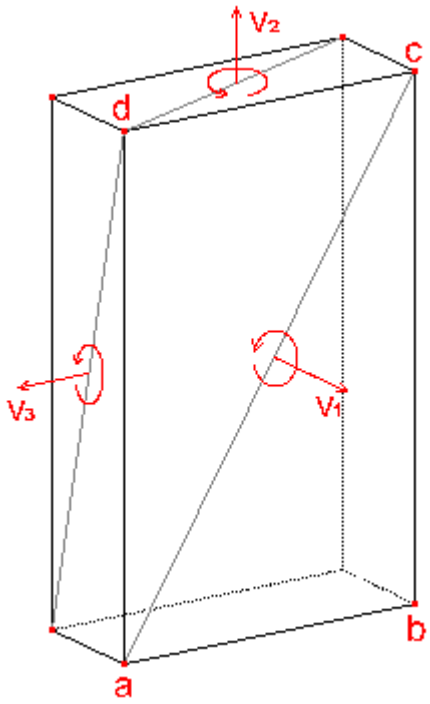

Abbildung 6.8: Reihenfolge der Polygonpunkte abhängig vom Betrachter

Das Framework sieht im ursprünglichen Fall vor, dass die drei an einem Polygon beteiligten Vektoren dem Polygon gegen den Uhrzeigersinn nacheinander hinzugefügt werden. Der Umstand, dass die Linien eines Solids nicht einer geregelten Richtung unterliegen, erschwert das Erstellen von Polygonen in dieser Reihenfolge. Die Linien einer Vektorgrafik werden über zwei Vektoren definiert. Diese Vektoren haben eine zufällige oder durch die Erstellungsrichtung der Linie bedingte Reihenfolge innerhalb der Liniendefinition. Die Linie zwischen den Vektoren a und b in Abbildung 6.8 könnte den Punkt a oder auch den Punkt b zuerst enthalten.

6.3.1 Lösungsansatz

Um die Vektoren folglich einem Polygon in der richtigen Reihenfolge hinzuzufügen, muss die Richtung einer Linie in Verbindung mit der Position im Solid betrachtet werden.

Im Falle eines Solids könnte dafür ein Vektor berechnet werden, der orthogonal auf der darzustellenen Ebene steht und aus dem Körper des Solids heraus zeigt. Die Vektoren v_1, v_2 und v_3 in Abbildung 6.8 sollen dies verdeutlichen. Diese Vektoren zeigen dann in Richtung des Betrachters und können für die Entscheidung der Reihenfolge verwendet werden. Die

beiden Polygone der großen Frontfläche des abgebildeten Solids müssen den Polygonen in der Reihenfolge $a \to b \to c$ und $a \to c \to d$ hinzugefügt werden.

Wenn die Fläche mit der gleichen Orientierung auf der gegenüberliegenden Seite des Solids liegen würde, ergäben sich die gegenläufigen Reihenfolgen $b \to a \to c$ und $a \to d \to c$.

Anders formuliert könnte die Reihenfolge abhängig von einem orthogonal auf der Ebene stehender Vektor v_1, v_2 oder v_3 entschieden werden, je nachdem ob er aus dem Solid heraus oder in den Solid hinein gerichtet ist.

Ein mathematisches Verfahren für die Bewertung dieser Vektorausrichtung ist jedoch im Rahmen der Ausarbeitung nicht gefunden worden.

7 Schlussbetrachtung

7.1 Zusammenfassung und Fazit

Die Generierung von dreidimensionalen Darstellungen von Gebäuden aus Grundrissen ist mithilfe einiger vorbereitenden Anpassungen der Quelldaten und einer Reihe von gezielt bestimmbaren Konfigurationsparametern, gespeichert in einer XML Datei, erfolgreich durchführbar.

Die Verwendung der *Kabeja* Bibliothek macht einen strukturierten und effizienten Import von Vektordaten aus dem DXF Format möglich. Das Sortieren von Linien der Wand-, Fenster- und Türelemente auf jeweils eigene Ebenen der Vektorgrafik sowie eine generell anwendbare Methode zur Bestimmung eines geeigneten Skalierungsfaktors sind dafür notwendige Vorbereitungen.

Für die Generierung einer verwendbaren Darstellung ist eine Strukturanalyse des Grundrisses erforderlich, die sich hauptsächlich auf die Distanzen zueinander parallel liegender Linien stützt. Mithilfe einer Häufigkeitsbestimmung der im Grundriss vorhandenen Distanzen und einigen eingrenzenden Werten lässt sich eine Einteilung der vorhandenen Vektordaten in Gruppen von Linien durchführen, die gemeinsam die Wandabschnitte des Grundrisses bilden.

Diese Gruppen von relationalen Linien sind die Basis für eine Zerlegung der komplexen Grundrissstrukturen in einheitliche, viereckige Elemente. Bei dieser Einteilung müssen einige Sonderfälle von Linienkonstellationen berücksichtigt werden, um einen möglichst hohen Darstellungsgrad zu erreichen. Abhängig von den Eigenheiten der Quelldaten sind dabei Darstellungsfehler möglich, die sich jedoch in der Theorie automatisiert oder durch eine geringfügige manuelle Optimierung der Quelldaten in vielen Fällen beheben lassen.

Die aufwändige Zerlegung der Strukturen in einzelne einheitliche Elemente ist für eine strukturierte und vereinheitlichte Generierung einer dreidimensionalen Darstellung erforderlich. Während die Darstellung vertikaler Flächen auf Basis der Rohdaten auf einfache Weise umsetzbar ist, erfordert die korrekte Ergänzung von horizontalen Flächen die Einteilung der architektonischen Strukturen in eine Menge ausschließlich viereckiger Elemente.

Auf Basis dieser vereinheitlichten Elemente ist dann die Generierung von typabhängigen dreidimensionalen Grundrisselementen auf strukturierte Art möglich. Die Darstellung auf Basis des *computergraphics* Frameworks erfolgt mit individuell für jeden Grundriss konfigurierbaren Höhenangaben für Wände, Fenster und Türen.

Je nach äußerer Form eines Grundrisses kann mit dem erarbeiteten System auf effizientem und schnellem Wege eine dreidimensionale Darstellung erreicht werden. Die Findung von funktionierenden Parametern ist in der Regel einfach und schnell erreicht. Die in den Quelldaten nicht vorhandenen Höheninformationen lassen sich dabei individuell auf das dargestellte Gebäude einstellen.

Zwar lassen sich die Ergebnisse der entwickelten Algorithmen bisher nicht speichern und ohne erneuten Systemdurchlauf wiederverwenden, allerdings ist das Ziel der Generierung von dreidimensionalen Darstellungen aus zweidimensionalen Gebäudegrundrissen im Allgemeinen erreicht worden. Das entwickelte System hat jedoch noch Potenzial für Verbesserungen und Erweiterungen, welche im nachfolgenden, abschließenden Ausblick auf Möglichkeiten der Weiterentwicklung erläutert wird.

7.2 Möglichkeiten der Weiterentwicklung

7.2.1 Sonderfälle

Zunächst werden die im Abschnitt 6.2 beschriebenen Sonderfälle vom System noch nicht automatisiert korrigiert. Die aufgezeigten, groben Lösungsskizzen können dabei einen durchführbaren Ansatz bieten, da die Identifizierung der relevanten Vektoren bereits implementiert ist.

7.2.2 Reihenfolge in der Polygongenerierung

Des Weiteren ist der in Abschnitt 6.3 beschriebene Aufbau von Polygonen in der ursprünglich vom *computergraphics* Framework vorausgesetzten Reihenfolge nicht implementiert. Eine verwendbare Lösung für diese Aufgabe ist jedoch nicht zufriedenstellend bekannt.

7.2.3 Definition und Darstellung verschiedener Höhen für Fenster und Türen

Während die Wände eines Stockwerks nahezu immer und Türöffnungen zumindest meistens eine einheitliche Höhe haben, variieren Fenster innerhalb eines Gebäudes meist stark in ihrer Größe beziehungsweise ihrer Höhe. Eine Umstrukturierung der Höhenmaßangaben in der

Konfigurationsdatei ist denkbar, um die Angabe beliebig vieler Fensterebenen mit jeweils individuellen Höhenmaßen zu ermöglichen.

Dies würde jedoch auch eine Ergänzung in der Typdefinition von Linien und Solids sowie eine Erweiterung bei der Generierung von Fensterdarstellungen erforderlich machen.

Eine solche Erweiterung würde die Darstellung von bodenhohen Fenstern, hüfthohen Fenstern sowie schmalen Oberlichtern in Wänden innerhalb desselben Grundrisses zulassen und damit eine originalgetreuere Darstellung ermöglichen.

7.2.4 Darstellung mehrerer Stockwerke

Eine Darstellung mehrerer Stockwerke ist bereits über die Positionierung der einzelnen Stockwerke nebeneinander in einer Grafik möglich. Um jedoch eine Art schichtweise Darstellung mehrerer Stockwerke und im Idealfall der verbindenden Treppen zu ermöglichen, müsste das System umfangreich erweitert werden.

7.2.5 Speicherung der Bewertungsergebnisse und daraus generierter Darstellungen

Der übergeordnete Gedanke dieser Ausarbeitung ist die Verwendung der Darstellungsergebnisse in *Smart Home* Anwendungen. Um für diese Anwendungen nicht bei jedem Start erneut eine vollständige Analyse und Generierung der Solids und der daraus resultierenden Darstellungen vornehmen zu müssen, ist die Persistenz der Ergebnisse erforderlich.

Zu diesem Zweck sind verschiedene Ansätze denkbar. Sowohl die Speicherung der generierten Solids für eine nachträglich durch Konfigurationswerte anpassbare Darstellung als auch die Speicherung der generierten Polygonnetze kann für verschiedene Anwendungsfälle sinnvoll sein.

Literaturverzeichnis

[Autodesk 2016] Autodesk, Inc.: Importieren und Exportieren von DXF-Dateien. (2016). – URL https://knowledge.autodesk.com/de/support/autocad/learn-explore/caas/CloudHelp/cloudhelp/2016/DEU/AutoCAD-Core/files/GUID-D4242737-58BB-47A5-9B0E-1E3DE7E7D647-htm.html. – Zugriffsdatum: 06.09.2016

[de las Heras u. a. 2013a] de las Heras, Lluís ; Fernández, David ; Valveny, Ernest ; Lladós, Josep ; Sanchez, Gemma: Unsupervised Wall Detector in Architectural Floor Plans. (2013)

[de las Heras u. a. 2015] de las Heras, Lluís ; Terrades, Oriol R. ; Lladós, Josep: Attributed Graph Grammar for Floor Plan Analysis. (2015)

[de las Heras u. a. 2013b] de las Heras, Lluís ; Valveny, Ernest ; Sanchez, Gemma: Combining structural and statistical strategies for unsupervised wall detection in floor plans. (2013)

[pcae GmbH 2016] GmbH pcae: Projekt 1 - Holztafelbau. (2016). – URL http://www.pcae.de/main/progs/alfa/hora/projekt1.htm. – Zugriffsdatum: 05.09.2016

[JogAmp.org 2016] JogAmp.org: Java binding for the OpenGL API Overview. (2016). – URL http://jogamp.org/jogl/www/. – Zugriffsdatum: 05.09.2016

[Kazmierzak 2015] Kazmierzak, Florian: Smart Home Environment - Concepts and Solutions. (2015)

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 8. September 2016 Leonard Opitz