



Institut für Software & Systems Engineering
Universitätsstraße 6a D-86159 Augsburg



Masterarbeit im Studiengang
„Informatik“

Individualisierbarer Konstruktionsplaner mit automatischer Bauplandeduktion

Sebastian Rossi



Institut für Software & Systems Engineering
Universitätsstraße 6a D-86135 Augsburg



Individualisierbarer Konstruktionsplaner mit automatischer Bauplandeduktion

Sebastian Rossi

Erstgutachter: Prof. Dr. Wolfgang Reif
Zweitgutachter: Prof. Dr. Bernhard Bauer
Matrikelnummer: 1475010
Abgabe der Arbeit: 8. Januar 2023
Betreuer: Constantin Wanninger

Erklärung

Hiermit versichere ich, dass ich diese Masterarbeit selbstständig verfasst habe. Ich habe dazu keine anderen als die angegebenen Quellen und Hilfsmittel verwendet.

Augsburg, den 8. Januar 2023

Sebastian Rossi

Zusammenfassung

The lack of housing is a growing problem in modern states. High cost of skilled labor in building construction intensifies this problem. Use of full automation in this domain is quite uncommon to this day. This paper presents a solution by using a collection of different robots tasked with individual jobs. With this approach the introduced system can tackle a multitude of problems, which arise from the difficult conditions on construction site. Principles of the Semantic Plug&Play concept are used to ensure a simple application and adaptability. As proof of concept different scenarios with increasing levels of difficulty are used.

Inhaltsverzeichnis

1 Motivation	1
2 Problemstellung	3
3 Fallstudien	5
3.1 Planung und Bauplandeduktion eines LEGO Gebäude mit Einsteinmauerwerk	5
3.1.1 Problemstellung	6
3.2 Planung und Bauplandeduktion eines Lego Gebäude mit Verbandsmauerwerk	8
3.3 Szenario mit veränderbaren Bausteintypen	8
3.4 Sternchenaufgabe: Szenario mit "runden" Wänden und arbiträren (nicht rechteckige) Bausteinformen / schräge schnitte	8
4 Related Work	9
4.1 3D Druck und Additive Fertigung von Gebäuden	9
4.2 Legeroboter	9
4.2.1 Digital Plan of Brickwork Layout for Robotic Bricklaying Technology	9
4.3 Materialien	10
4.4 Bausteine	10
4.5 Wall detailing und das (3D) Bin Packing Problem	10
4.6 Ludwigs Dissertation	11
5 Grundlagen	13
5.1 Industry Foundation Classes	13
5.1.1 IFC 4.3.1.0 Aufbau	13
5.1.2 IfcPropertySets und IfcQuantitySets	14
5.1.3 Positionierung von IFCProducts	15
5.1.4 IfcOpeningElement	15
5.2 IFC for Blender	15
5.2.1 Blender	15
5.2.2 blenderbim	15
5.2.3 IfcOpenShell	16
5.3 Building Information Modeling	16
5.4 opensourcebim	17
5.5 Mauerwerksbau	18
5.5.1 Maßsysteme	18
5.5.2 Mauerwerksverband	19
5.6 LEGO	22

Inhaltsverzeichnis

5.7	BRep	23
5.8	Ontologie	23
5.8.1	Protégé	25
5.8.2	Owlready2	25
6	Konzept	27
6.1	Modellierung	27
6.1.1	Raster	27
6.1.2	Wandstück	28
6.1.3	Mauerwerksverband	28
6.1.4	Beziehungen zwischen Wandstücken	30
6.1.5	Lösen von Beziehungen	34
6.1.6	Öffnungen	37
6.1.7	Wandenden	37
6.2	Wall Detailing	38
6.3	Regelbasierte Bauplandeduktion	39
7	Realisierung	41
7.1	Modellierung	41
7.2	Wall Detailing	41
7.2.1	Konvertieren des IFC zu BREP	42
7.2.2	Anwenden des Moduls	42
7.2.3	Kombinieren passender Wandstücke	42
7.2.4	Finden und Lösen von Beziehungen	45
7.2.5	Anwenden der Öffnungen	46
7.2.6	Anwenden der Mauerwerksverbände	46
7.2.7	Nachbarschaftsbeziehungen zwischen Bausteinen berechnen	46
7.2.8	Export	46
8	Proof of Concept	49
9	Fazit und Ausblick	51
Literatur		53

1 Motivation

Der stetig steigende Fachkräftemangel trifft auch das Baugewerbe. Laut einer Umfrage des Hauptverbandes der Deutschen Bauindustrie e.V. stufen über 75 Prozent aller befragten Unternehmen sowohl den vorherrschenden Fachkräftemangel, als auch die steigenden Energie- und Rohstoffpreise als Risiko für das eigene wirtschaftliche Wachstum ein [30]. Abbildung 1.1 illustriert die Entwicklung dieser Sorge über einen Zeitraum von etwas mehr als zwanzig Jahren. Damit ist es wenig überraschend, dass eine Bewegung weg von menschlichen Arbeitskräften hin zur Automatisierung existiert. Neben dem Fachkräftemangel stellt aber auch die geringe Effizienz von Bauvorhaben ein Problem dar, welche sich über den gesamten Planungs- und Bauprozess erstreckt. Diese Ineffizienz entsteht aufgrund der Vielzahl der an Bauprojekten beteiligten Experten und Unternehmen und ist, als *Fragmentierungsproblem der Bauindustrie* bezeichnet, ein bekanntes Problem [46]. Deshalb etablieren sich derzeit Standards, um Bauprojekte digital zu begleiten. Mit diesen



Abbildung 1.1: Während wirtschaftliches Risiko durch eine potentiell sinkende Nachfrage nach Bauanträgen und eventuell steigender Arbeitskosten unverändert blieben oder sogar als weniger relevant bewertet wurden, ist ein deutlicher Anstieg aufgrund des vorherrschenden Fachkräftemangels und der Energie- und Rohstoffpreise zu erkennen.

soll gleichzeitig die Effizienz gesteigert, die Kommunikation zwischen den einzelnen Expertenteams vereinfacht, der Arbeitsplatz "Baustelle sicherer gestaltet und ein resourcensparender Bau ermöglicht werden [25] [50]. Gleichzeitig steigen mit der Zunahme an digitalen Informationen zu Bauprojekten, auch die Möglichkeiten diese besser zu analysieren, zu optimieren und an neue Technologien zu knüpfen. Erst dadurch wurde das seit einigen Jahren erforschte Gebiet der Additiven Fertigung von Gebäuden, etwa mit Beton druckenden Roboterarmen oder mobilen Robotern, realisierbar [18]. [TODO

1 Motivation

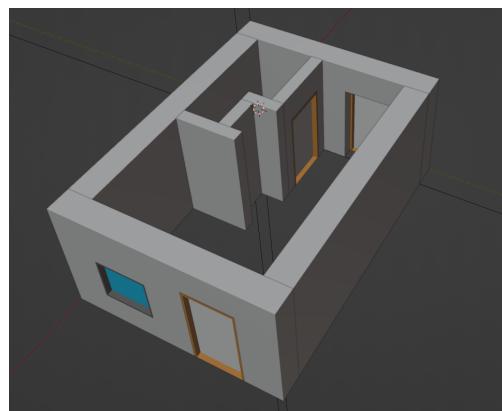
nochmal reinschauen ob das einigermaßen als Quelle passt]. Obwohl es mittlerweile viele Projekte zur Additiven Fertigung von Gebäuden gibt, haben diese oft den Nachteil der Nicht-Parallelisierbarkeit der druckenden Roboter, die durch die Höhe der temporären Stützstrukturen (wie Kräne, Gerüste, Aufhängungen) eingeschränkte Bauhöhe und die vergleichsweise lange Bauzeit [TODO Quelle oder lange bauzeit weg]. [Frage wegen Kommentar: Star Wars roboterschwarm/Marsbesiedelung; soll ich hier sagen dass ein derzeit in der SciFi Szene aufgekommener Trend von organisierten Roboterschwärmen, die was bauen gar nicht so blöd ist und wir das ähnlich angehen?] Diesen Einschränkungen soll nun mithilfe eines Schwarmes bodengebundener automoner Roboter, welche gleichzeitig an dem Bauprojekt arbeiten, entgegengewirkt werden. Dabei sollen sich die Roboter auf den Mauern des Gebäudes selbst bewegen können, während sie dieses errichten. In dieser Arbeit liegt der Schwerpunkt allerdings nicht auf dem Entwickeln der Roboter selbst, sondern das Erstellen eines Bauplans für den genannten Roboterschwarm basierend auf einem 3D Modells des Gebäudes. Das Erstellen des Modells innerhalb eines nutzerfreundlichen 3D Editors stellt nicht nur die geometrischen und physikalischen Eigenschaften des Gebäudes digital bereit, sondern verschlankt auch die Kommunikation zwischen Endnutzer und Architekt oder ersetzt letzteren komplett. Gleichzeitig bildet diese Arbeit damit auch den Trend hin zur sogenannten Massenpersonalisierung ab, welcher als Nachfolgetrend zur Massenproduktion und als "heiliger Gral" der Fertigung angesehen wird . Dieser Trend ist auch für die Bauindustrie interessant, denn auch hier schafft die Möglichkeit sämtliche Kundenwünsche an ein Produkt (oder Gebäude) umzusetzen, ohne dafür spezielles Werkzeug herstellen zu müssen, neue Gewinnmöglichkeiten. Mit der Option der Modellierung des Gebäudes durch den Kunden selbst, ist die Kommunikation mit den Architekten über dessen Wünsche effizienter, da beide Parteien zusammen an dem Modell arbeiten können. Im Anschluss an den Designprozess des 3D Models des Gebäudes, soll dieses in einen Bauplan übersetzt werden, welcher alle notwendigen Informationen für den oben genannten Roboterschwarm enthält, sodass dieser das Gebäude selbstständig errichten kann [TODO ugs]. Dieses Konzept wird anhand nachfolgender Fallstudie(n) getestet.

2 Problemstellung

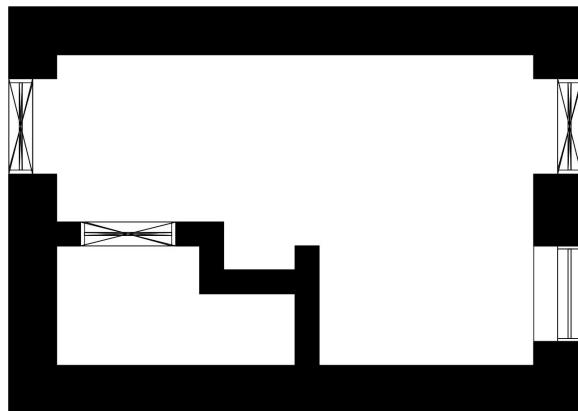
3 Fallstudien

3.1 Planung und Bauplandeduktion eines LEGO Gebäude mit Einsteinmauerwerk

Ziel dieses Szenarios ist es, aus dem in Abbildung 3.1a dargestellten 3D Modell ein Bauplan zu generieren. Zu sehen ist der Plan eines einfachen Hauses mit einem Stockwerk.



(a) 3D Modell innerhalb von Blender.



(b) Gebäudeplan des 3D Modells.

Abbildung 3.1: Modell eines Studentenzimmers (verschiedene Darstellungsformen).

Dieses besitzt eine Eingangstür, eine Terrassentür neben einem Fenster und eine Tür, die das Badezimmer vom Hauptraum trennt. Türen und Fenster stellen eine Herausforderung für den Planungsalgorithmus dar, da der Verlauf einer ansonsten durchgängigen Wand dadurch unterbrochen wird und Lücken aufweist. Details wie Duschen, Betten, Toiletten und ähnliche Komponenten, sind für dieses Szenario irrelevant, da diese keinen Effekt auf die Struktur der Wände haben und wurden aus diesem Grund bewusst weggelassen. Das Modell wurde mithilfe der in Kapitel 5 näher behandelten Technologien erstellt und entspricht in seiner Struktur einem verbreiteten Industriestandard. Dafür wurden zwei Wandtypen definiert, die jeweils unterschiedliche Wanddicken vorgeben. So gibt es breite Außen- und dünne Innenwände. Diese entsprechen in ihren Maßen dem Raster, welches das *LEGO System* (siehe Kapitel 5) vorgibt. Für breite Wände gilt, dass diese immer zwei Noppen breit, mindestens eine Noppe lang und einen Stein hoch sein muss. Für dünne Wände hingegen gilt eine feste Breite von einer Noppe, ebenfalls eine Mindestlänge von einer Noppe und eine Mindesthöhe von einem Stein. Beide Wandtypen können nur Höhen beziehungsweise Längen aufweisen, die jeweils einem Vielfachen der Höhe oder Länge des kleinsten Legosteins aufweisen, der zum Bau der Wände verwendet werden soll (hier der 1x1 Stein). Daraus resultiert ein Raster, welches ebenfalls für die

Ausmaße und Positionen der Fenster und Türen einzuhalten ist. Dieses Raster gilt es, abhängig des ausgewählten Wandtyps, in den Editor zu integrieren, um das Modellieren solcher Gebäude nutzerfreundlich zu gestalten und nicht durch ständiges Messen und Eintragen genauer Positionen oder Maße zu unterbrechen. Nicht nur die Abmessungen

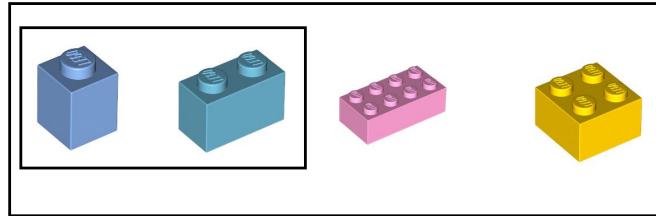


Abbildung 3.2: LEGO Steintypen für die Innen- und Außenwände. (TODO Bild ist sehr hässlich)

der Wände müssen in ein Raster fallen, auch deren Rotation wird in diesem Szenario auf 90° Schritte limitiert. Das stellt in diesem Fall eine vertretbare Enschränkung dar, da es ohnehin dem intuitiven Umgang mit LEGO Steinen und gleichzeitig dem Baustil der meisten einfachen Gebäuden entspricht. Folglich muss ein Format für die Bausteintypen entwickelt werden, aus welchen all diese Informationen abgeleitet werden können. Dieses Format muss sowohl von dem Editor selbst verwendet, als auch zur Berechnung innerhalb des Planungsalgorithmus herangezogen werden. Außerdem werden weitere Regeln benötigt, um den resultierenden Plan näher an das Vorgehen eines realen Baus zu bringen. So werden beim Errichten von Häusern zuerst die Ecken (der Schnittpunkt zweier Wände) um eine Stufe erhöht, um im Anschluss die geraden Wandabschnitte aufzufüllen. Damit wird vermieden, dass in den ohnehin schon komplexeren Eckbereichen auch noch zugeschnittene Ziegel notwenig werden. Stattdessen schneidet man diese erst zurecht, wenn sich dann eher mittig im Wandabschnitt Lücken ergeben, die kleiner sind als die vorhandenen Ziegel. Zwar wird dies im Fall von Legosteinen nie auftreten, aber es ist übersichtlicher das Problem in dem vorliegenden eingeschränkten Szenario zu beschreiben. Zusätzlich müssen Regeln in den Planungsalgorithmus eingeführt werden, die diverse Mauerwerksverbände (vorgestellt in Kapitel 5) erzwingen zu können, um damit die Stabilität und gleichmäßige Kraftverteilung innerhalb einer Wand zu gewährleisten und sich ebenfalls möglichst nah an der Realität des Mauerbaus zu bewegen.

Einschränkungen:

- * Alle Wände verwenden ein Ziegelset, in welchem alle Ziegel die gleiche Höhe haben
- * Alle Wände verwenden den selben Mauerwerksverband
- * Alle anderen sich berührenden Wände stehen in einem 90 Grad Winkel zueinander

3.1.1 Problemstellung

Bevor für das Modell ein passender Bauplan entwickelt werden kann, muss das sogenannte *Wall Detailing* (vgl. Kapitel 4.5) stattfinden. Darunter versteht man das Berechnen einer Bausteinkombination, die eine Wand möglichst gut abbildet. Dabei bedeutet "möglichst gut", dass die Proportionen der Bausteinkombination möglichst den der Ursprungswand entsprechen und keine nicht gewollten Lücken entstehen. In diesem Szenario wird durch

Auflegen des Rasters des LEGO Systems schon während des Modelliervorgangs sicher gestellt, dass das resultierende Modell mit Legosteinen baubar ist. Darum ist das Ziel des *Wall Detailing* Schrittes in diesem Szenario, jede Wand lückenlos mit Bausteinen zu füllen. Zusätzlich existieren folgende Einschränkungen und Eigenschaften, die für das Ergebnis dieses Szenarios gelten sollen.

Überbindemaß Obwohl in dem vorliegenden Modell ausschließlich Einsteinmauerwerke vorgesehen sind, gilt es die in Kapitel 5.5.2 erläuterte Regel zum Überbindemaß von Bausteinen zu beachten. Da es sich hierbei aber um LEGO Steine handelt, die wesentlich kleiner sind als die genormten Formate für Ziegelsteine, wird der für das Überbindemaß vorgesehene Mindestwert von 45mm ignoriert. Außerdem ist ein Versatz von unter 50% der Dicke eines Legosteines in den meisten Fällen nicht umsetzbar, da diese nicht frei übereinander gesteckt werden können. Darum wird für dieses Szenario ein Überbindemaß von exakt 50% der Steindicke verwendet, was den Einschränkungen des Überbindemaßes entspricht, welches einen Mindestversatz von 40% der Steindicke voraussetzt. Für die in Abbildung 3.2 dargestellten Lego Steine kann mit einem solchen Überbindemaß gleichzeitig auch das vorgesehene Raster des Lego Systems eingehalten werden, da diese alle eine gerade Anzahl an Noppen besitzen. TODO Bild von dumm gestapelter LEGO Wand zu LEGO Wand mit Überbindemaß.

Anstoßende Wandstücke Besonders herausfordernd ist die Einhaltung des Überbindemaßes an anstoßenden Wandstücken. Solche Wandstücke entstehen zum Beispiel an Ecken, also 90 Grad zueinanderstehende Wände oder aber an den Übergängen von Außenwänden zu Innenwänden. Wie in Abbildung 3.1a zu erkennen treten in diesem Szenario beide Fälle auf. Zunächst müssen daher alle anstoßenden Wandstücke aus dem Modell gefiltert werden. In Kapitel 5.5.2 werden gängige Praktiken zur Lösung von anstoßenden Wandstücken vorgestellt. Mit diesen Informationen muss der Planungsalgorithmus zunächst eine Lösung für solche Wandstücke finden und anschließend die restlichen Flächen mit dem Vorgehen von oben auffüllen.

Öffnungen Das Fenster und die drei Türen sind hier die einzigen Besonderheiten in dem Modell. Für den Mauerbau bedeutet dies an den entsprechenden Stellen Öffnungen zu lassen. Die Informationen über diese Öffnungen liegen innerhalb des Formates vor, in welchem das Modell erstellt wurde (siehe ??). So lassen sich Größe und Position leicht herausfinden und das Ergebnis der beiden obigen Schritte dementsprechend abändern.

Nun liegt eine Beschreibung des Modells mit dem zuvor definierten Set an Legosteinen vor. Als letzten Schritt gilt es daraus eine Abfolge an Bauinstruktionen zu finden, die unter Beachtung von diversen Einschränkungen zum gewünschten Ergebnis führen. Mit dieser Problemstellung hat sich Ludwig (TODO) in seiner Dissertation auseinandergesetzt (siehe Kapitel 4.6). Einschränkungen, die beim Bau von Legokonstruktionen gelten sind: Deadlock, Überhang etc. (TODO) Das Vorgehen in seiner Arbeit soll für dieses Szenario evaluiert und gegebenenfalls angepasst werden.

3.2 Planung und Bauplandeduktion eines Lego Gebäude mit Verbandsmauerwerk

Kein Einsteinmauerwerk -> dicke Wände, die mit Läufern und Bindern erstellt werden müssen. Eckfälle sehr komplex (siehe Basics)

3.3 Szenario mit veränderbaren Bausteintypen

Wie können wir die Bausteine veränderbar machen, sprich die Bearbeitungsmöglichkeiten während des Baus (schneiden eines Ziegels) miteinbeziehen in unser Bausteinformat. Erstmal das schneiden nur parallel zu den ebenen eines rechteckigen ziegelsteins betrachten, denn bei β chrägen β schnitten entstehen neue komplexe Formen.. Wie generiert man daraus dann Baupläne -> riesiges aber cooles Optimierungsproblem! Vlt geht das richtung constraint programming? Sprich den Bauplan als Lösung für ein beschränktes Problem ansehen. Total viele Fragen, keine Ahnung wo zu beginnen aber klingt cool

3.4 Sternchenaufgabe: Szenario mit "runden" Wänden und arbiträren (nicht rechteckige) Bausteinformen / schräge schnitte

Was machen wir wenn die Wand nicht perfekt mit den vorgegebenen Bausteintypen gebaut werden kann (Beispiel ein runder Turm) Bausteinverbindungen (wie Mörtel) betrachten und als *Verbindungselement* in das Bausteinformat mit aufnehmen? Wie kann man diesen so einschränken, dass nur physisch machbares ausgerechnet wird. Was wenn die Bausteine arbiträre Formen haben und nur in sehr komplexen Mustern eine "dichte" Wand ergeben -> tiling Probleme.

4 Related Work

4.1 3D Druck und Additive Fertigung von Gebäuden

4.2 Legeroboter

4.2.1 Digital Plan of Brickwork Layout for Robotic Bricklaying Technology

In diesem Paper stellen Usmanov et al. ein generelles Vorgehen für das Erstellen eines Ziegel-Legeplans für ein als digitales Modell vorliegendes Gebäude vor [16]. Dieses Vorgehen gliedern sie in sechs Schritte:

1. Das vorliegende IFC Modell (siehe 5.1) nach Wandelementen durchsuchen und diese in das sogennante BREP-Format (siehe 5.7) konvertieren.
2. Das Aufteilen des gesamten Modells in Schichten, die der Modulhöhe des verwendeten Ziegelsteinformats entspricht.
3. Verbindungen von getrennt modellierten Wandelementen heraussuchen. Dies ist zum Beispiel an Eckstücken der Fall, da dafür oft zwei einzelne Wandelemente modelliert werden, welche in einem Winkel zueinander stehen und sich berühren. Für die nachfolgenden Schritte sind diese Verbindungen relevante Informationen.
4. Mit den Informationen der vorhergegangenen Schritte können nun für jede Schicht kritische Bereiche identifiziert werden, an welchen später ein komplexer Legevorgang von Ziegeln von Nöten ist.
5. Nun werden zunächst die kritischen Bereiche anhand einer vorher definierten Legeanleitung mit teilweise angepassten Ziegelsteinen bestückt und im Anschluss die restlichen Bereiche aller Wände mit dem ausgewählten Standardziegel aufgefüllt. In diesem Schritt werden zusätzlich Fenster- und Türstürze über deren Öffnungen in den Wänden gelegt.
6. Dieser Schritt fasst das Anzeigen als 3D Modell und Konvertieren des Resultats in eine nicht konkreter definierte Listenform zusammen. Das Ergebnis für das von den Autoren ausgewählte Beispielgebäude ist in Abbildung 4.1 zu sehen.

Besonders detailliert ist ihr Ansatz für das Finden von besagten kritischen Teilbereichen einer Wand mithilfe mathematischer Gleichungen, die alle Wände zueinander in Beziehung stellen. Diese Teilbereiche sind Wanddecken, T-Kreuzungen (wie etwa der Übergang einer Innenwand an eine Außenwand) und Öffnungen innerhalb einer Wand und sind ebenfalls in Kapitel 3.1.1 dieser Arbeit als Problemstellung aufgeführt. Anhand der von ihnen zusammengetragenen Informationen konnten die Autoren erfolgreich die

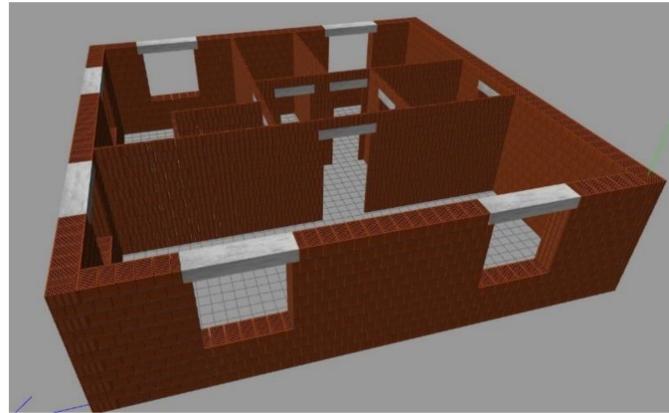


Abbildung 4.1: Ergebnis des Verfahrens zur Erstellung eines Ziegel-Legeplans nach Usmanov et al. [16].

bestmögliche Platzierung von vier Roboterarmen errechnen, die den schnellstmöglichen Bau des Gebäudes ermöglichen. Dennoch werden zum Schluss noch einige Einschränkungen ihres Verfahrens angesprochen. Vor allem die Beschränkung auf 90 Grad Ecken und das Gebunden sein an einen einzigen Standardziegel werden als besonders restriktiv wahrgenommen.

4.3 Materialien

4.4 Bausteine

4.5 Wall detailing und das (3D) Bin Packing Problem

TODO hinführen über Bin Packing hin zu Spezialfall "Wall detailing mit arbiträren Bausteinen und Eigenschaften (wie versetzen der ziegel)"

TODO über bin packing schreiben, erklären paper suchen, lösungsansätze zu np hartem problem

Xu Chengran et al. haben in ihrem Paper "Optimal brick layout of masonry walls based on intelligent evolutionary algorithm and building information modeling" verschiedene Optimierungsansätze aus dem Bereich des 2D Packaging Problems getestet [17]. Konkret wurden drei Algorithmen verwendet: Differential Evolution, Particle Swarm Optimization und Neighbourhood Field Optimization. Außerdem wird ein drei-phäsiges Vorgehen vorgeschlagen: Data collection, Brick layout und Data Output. Dieses Vorgehen eignet sich auch für das Finden von Bausteinkonfigurationen in dieser Arbeit, da zuerst alle relevanten geometrischen Daten (in diesem Fall Wände, Fenster, Türen usw.) aus dem 3D Modell gesammelt werden müssen, bevor das Detailing stattfinden kann. Nach dem Optimieren der Bausteinkonfiguration muss das Ergebnis ebenfalls in ein Format gebracht werden, das für die folgenden Schritte verwendet und eventuell auch dem Nutzer angezeigt werden kann.

Soft items: <https://arxiv.org/abs/2206.15116>

Irregular Shaped items: <https://link.springer.com/content/pdf/10.1631/FITEE.1400421.pdf>
"Parametric Blockwall-Assembly Algorithms for the Automated Generation of Virtual Wall Mockups Using BIM"

4.6 Ludwigs Dissertation

5 Grundlagen

Zunächst wird für diese Arbeit ein geeignetes Speicherformat für 3D Modelle von Gebäuden benötigt. Um den Bau von Gebäuden zu automatisieren, ist es notwendig die Domäne *Gebäude* möglichst vollständig digital abbilden zu können. Hilfreich ist dabei, wichtige Daten über bestimmte Bestandteile des Gebäudes direkt in das Modell zu integrieren, auf Basis derer etwa Kostenberechnungen durchgeführt oder Materialmengen herausgefunden werden können. Da diese Informationen oft von Experten verschiedener Fachbereiche (etwa aus den Bereichen der Architektur, des Bauwesens oder der Statik) stammen, muss das Format sehr flexibel und im besten Fall auch zeitgleich bearbeitbar sein. Dafür werden seit dem Jahr 2000 die *Industry Foundation Classes* (IFC) von buildingsmart entwickelt, deren Anwendung im internationalen Bauwesen mittlerweile weit verbreitet ist [36].

5.1 Industry Foundation Classes

In der Spezifikation des Standards selbst, wird dieser wie folgt beschrieben: „Die Industry Foundation Classes (IFC) sind ein offener internationaler Standard für Daten des Building Information Model (BIM), welche zwischen Softwareanwendungen ausgetauscht, gemeinsam genutzt und von den verschiedenen Akteuren der Bauindustrie und des Gebäudemanagements verwendet werden. Der Standard enthält Definitionen für Daten, die für die Lebenszyklen von Gebäude- und Infrastrukturarbeiten erforderlich sind. Die bis jetzt in die IFC aufgenommenen Infrastrukturtypen umfassen Brücken, Straßen, Eisenbahnen, Wasserstraßen und Hafenanlagen“ (aus dem Englischen) [40]. Eine frühere Version des IFC Standards ist unter der Bezeichnung ISO 16739 registriert (siehe [41]). Da die IFC aber nach wie vor kontinuierlich weiterentwickelt werden, wird in dieser Arbeit die derzeit neueste Version verwendet. Diese ist die IFC Spezifikation 4.3.1.0 [39]. Das verbreitetste Austauschformat für IFC ist das Step Physical File Format, welches im ISO 10303 Teil 21 registriert ist [34]. Zudem gibt es speicherreduzierte Formate wie ifcZip oder für Menschen lesbarere Formate wie ifcXML [35][32][29].

5.1.1 IFC 4.3.1.0 Aufbau

Im Grunde definieren die *Industry Foundation Classes* eine Vielzahl an Klassen, die in einer komplexen Hierarchie angeordnet den Grundstock des Datenmodells bilden. Diese sind anfangs abstrakte Konzepte, die sich mit zunehmender Tiefe in der Hierarchie konkretisieren. Da sich diese Arbeit zum größten Teil mit aus Wänden bestehenden Gebäuden befasst, wird nachfolgend die Klasse *IfcWall* wiederholt als Beispiel herangezogen. Der für diese Klasse relevante Ausschnitt aus der Klassenhierarchie ist in Abbildung 5.1 dargestellt. Objekte werden von dem Standard in Relation zueinander gestellt, um komplexe

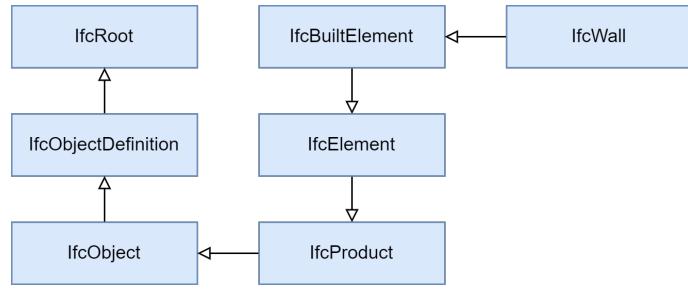


Abbildung 5.1: Klassenhierarchie am Beispiel der Klasse *IfcWall*

Zusammenhänge darzustellen. In Abbildung 5.2 erkennt man den Zusammenhang zwischen einem Objekt des Types *IfcWall*, des Stockerwerks, welches diese Wand referenziert und wiederum selbst Teil eines *IfcBuildings* ist, bis hin zur obersten Komponente eines Ifc Projektes, dem gleichnamigen *IfcProject*.

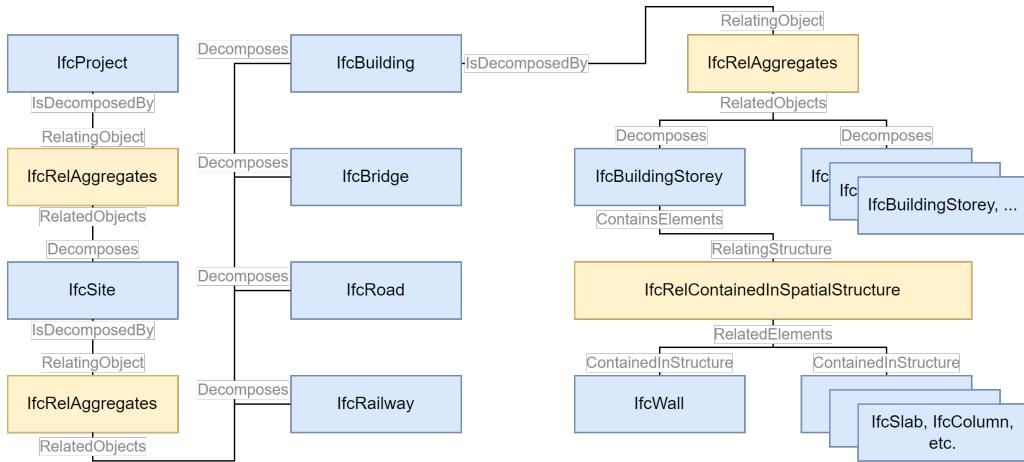


Abbildung 5.2: Relation der *IfcWall* und einem *IfcProject*

5.1.2 IfcPropertySets und IfcQuantitySets

Mit dem Erweitern des Gebäudemodells um möglichst viele Informationen, schafft man einen detaillierten digitalen Zwilling, der neben der bloßen Darstellung des Gebäudes als 3D Modell zusätzlich die Untersuchung auf andere Eigenschaften ermöglicht. Dazu zählen zum Beispiel eine präzisere Kosten- und Zeitschätzung, Sicherheitsaspekte und Emissionen eines Bauvorhabens [35] [9]. Dies ist mithilfe von *IfcPropertySets* und *IfcQuantitySets* umsetzbar, welche Objekten oder Objekttypen angehängt werden können. Dabei werden *IfcQuantitySets* vorwiegend dazu verwendet numerische Werte zu geometrischen oder physikalischen Eigenschaften auszudrücken. *IfcPropertySets* dienen hingegen dazu bestimmten Objekten oder Objekttypen mit sämtlichen nicht numerischen Werten zu annotieren. Beispiele dafür sind etwa Material oder Farbe eines bestimmten Objekts. Die Verwendung der Bezeichnung *Set* röhrt daher, dass diese Informationen in einer Baumstruktur vorliegen und demnach verschachtelt sein können. Es gibt zu

vielen Klassenbeschreibungen des IFC Standards vordefinierte PropertySets. Das hilft dabei wichtige Informationen zu bestimmten Objekttypen einheitlich angeben zu können. Im Falle des *IfcWallType* existiert beispielsweise das PropertySet mit dem Namen *Pset_WallCommon* [38]. Darin können die Ersteller von neuen WallTypes unter anderem Informationen über Brennbarkeit, thermisches Verhalten oder Akustik hinterlegen. Bei Erstellen einer IfcWall-Instanz aus einem bestimmten WallType werden die damit verbundenen Property- und QuantitySets automatisch daran angehängt, sodass jedes Objekt relevante Informationen über sich selbst bereithält.

5.1.3 Positionierung von IFCProducts

TODO: Positionierung ist anscheinend bisschen verschachtelt Blablabla so können aus einem IFC Projekt alle relevanten Klassen mit den für diese Arbeit erforderlichen Eigenschaften extrahiert werden. This is a test

5.1.4 IfcOpeningElement

Neben Wänden stellen Fenster und Türen wichtige Elemente eines Gebäudes dar. Diese sind ebenfalls Teil der Klassen des IFC Standards. Objekte des Typs *IfcOpeningElement* dienen dazu die notwendigen Lücken in einer Wand zu definieren in die später eine Tür oder ein Fenster eingebaut werden soll. In der Dokumentation wird dies wie folgt formuliert (aus dem Englischen): „[Das IfcOpeningElement] stellt eine Lücke in jedem Element dar, das eine physische Manifestation hat“ [37]. Es gibt zwei Arten an IfcOpeningElement, je nachdem ob die Öffnung durch die gesamte Breite eines Objektes reicht oder nicht. Folglich entsteht dadurch eine Unterscheidung zwischen Nischen und tatsächlichen Öffnungen, wobei nur letztere im Zusammenhang mit Fenstern oder Türen Sinn ergibt. TODO: Diagramm zwischen IFCWall und IfcOpeningElement.

5.2 IFC for Blender

5.2.1 Blender

Blender ist eines der beliebtesten Open Source Programme zur Modellierung von 3D Modellen und Animationen [26]. Eine umfangreiche Python API erlaubt es Blender durch sogenannte Addons an die eigenen Bedürfnisse anzupassen [21] [23]. Aufgrund dessen existiert auch eine Vielzahl an freien Erweiterungen – unter anderem auch eine Integration von IFC Projekten.

5.2.2 blenderbim

Neben kommerziellen Produkten wie etwa revit von autodesk zur Modellierung von IFC Modellen, gibt es auch für Blender ein freies Plugin, um IFC Modelle zu erstellen [47] [27]. Dieses Plugin ermöglicht es neben dem bloßen Designen des Gebäudes in kurzer Zeit z.B. detaillierte Zeichnungen verschiedener Perspektiven herauszuarbeiten, die z.B. von Bauingenieuren verwendet werden können, um einzelne Stockwerke oder Verkabelungen

zu planen. Blenderbim selbst kapselt unter anderem die Open Source Python Bibliothek *IfcOpenShell*, sodass diese in der Blender Laufzeitumgebung zur Verfügung steht [33].

5.2.3 IfcOpenShell

```
1 import ifcopenshell
2 from ifcopenshell import geom
3 from stl import mesh, Mode
4 import numpy as np
5
6 settings = ifcopenshell.geom.settings()
7 settings.set(settings.USE_WORLD_COORDS, True)
8
9 ifc_file = ifcopenshell.open("model.ifc")
10 products = ifc_file.by_type("IfcProduct")
11 meshes = []
12
13 for product in products:
14     if product.Representation and product.is_a("IfcWall"):
15         shape = ifcopenshell.geom.create_shape(settings, product)
16         vertices = np.array(shape.geometry.verts).reshape((-1, 3))
17         edges = np.array(shape.geometry.edges)
18         faces = np.array(shape.geometry.faces).reshape((-1, 3))
19
20         m = mesh.Mesh(np.zeros(faces.shape[0], dtype=mesh.Mesh.dtype))
21         for i, f in enumerate(faces):
22             for j in range(3):
23                 m.vectors[i][j] = vertices[f[j], :]
24         meshes.append(m)
25
26 # Create the combined mesh
27 combined = mesh.Mesh(np.concatenate([m.data for m in meshes]))
28 combined.save('model.stl', mode=Mode.ASCII)
```

Listing 5.1: Beispielprogramm zur Extraktion bestimmter Daten einer IFC Datei und Generierung eines Meshes aus deren geometrischen Representationen.

IfcOpenShell ist eine frei verfügbare Bibliothek, die es erleichtert mit Daten im IFC Format zu arbeiten. Sie bietet unter anderem eine Python API an und ist wie bereits erwähnt ein Teil des Blender Addons blenderbim. Natürlich ist es dennoch möglich diese Bibliothek auch ohne Blender zu verwenden. In Listing 5.1 ist ein kurzes Beispiel eines Programmes gegeben, das in einer IFC Datei alle Objekte des Typs IfcWall ausfindig macht und dessen geometrische Representation in ein herkömmliches Mesh konvertiert. Damit existiert ein intuitiver Zugang zu den Daten von IFC Dateien.

5.3 Building Information Modeling

Ein weiterer Punkt, der für die Verwendung von IFC spricht ist das sogenannte *Building Information Modeling* (BIM) [28]. Ein Definitionsvorschlag lautet wie folgt: „BIM ist definiert als der Einsatz von Informations- und Kommunikationstechnologien zur Verschlankung der Prozesse im Lebenszyklus von Gebäuden, um eine sicherere und

produktivere Umgebung für die Bewohner zu schaffen, die Umwelt so wenig wie möglich zu belasten und die Effizienz der Betriebsabläufe für die Eigentümer während des gesamten Lebenszyklus des Gebäudes zu erhöhen“ (Übersetzt aus dem Englischen) [45]. Zum Lebenszyklus eines Gebäudes gehören etwa anfangs das Planen und Designen, später das Bauen, das Verwenden und Instandhalten und nach eventuellen Renovierungen das Abreißen. BIM kommt in all diesen Phasen zum Tragen und erleichtert diese Prozesse durch Anbieten einer einheitlichen Schnittstelle für alle am Infrastrukturbau und -management beteiligten Personen. Zusätzlich ermöglicht BIM eine exakte Dokumentation des Geschehens in sämtlichen Phasen des Bauwerks, was unter anderem zu einer genaueren Zeit- und Kostenplanung führt [9]. Auch Verantwortlichkeiten sind Teil von BIM, was zu einer erhöhten Produktivität beiträgt. Um nun das Zusammenarbeiten der unterschiedlichen Fachbereiche zu erleichtern, gibt es sogenannte BIM-Server auf welchen mehrere Arbeitende synchron an einem Projekt arbeiten können, während sie jeweils die für ihren Aufgabenbereich passende Ansicht vor sich haben. BIM-Server unterstützen zusätzlich eine Versionierung des Fortschritts an einem Projekt.

In einem Gespräch mit einem Ingenieur aus dem Bereich „Energysystemtechnik“ kam zur Sprache, dass viele Bereiche von BIM noch nicht ganz Einzug in Deutschland gefunden haben. Eben jene „Kollaboration über einen BIM-Server mit Änderungsmanagement etc. [sei] (noch) nicht üblich, da noch nicht alle Beteiligten dazu in der Lage sind. Vor allem Bauherren, Architekten und Baufirmen können es nicht“. Weiter sei „auch unklar, wer für falsche Angaben haftet und wer die Konsistenz aller Daten gewährleistet“. Auf der anderen Seite sei „das im BIM festegelegte Datenformat IFC das Maß der Dinge und auch bei uns so in Verwendung“. Auch das Einpflügen „ergänzende[r] Bauteilinformationen (z.B. zu Gewicht, Dämmwert, Recyclebarkeit, CO₂ Fußabdruck, etc.)“ finden Einsatz und sind Teil seines Alltags. Für ihn wichtig ist ebenfalls der Betrieb des Gebäudes. Hier unterstützt BIM, indem sämtliche Teile der Installationen in einem Gebäude, wie z.B Fensterdichtungen, Kabel, Rohre, Sicherungen oder eine Umwälzpumpe individuelle Teilenummern zugewiesen bekommen, hinter welchen alle Daten wie etwa Hersteller, Bestellnummern, Lebensdauer, Wartungshistorie oder Entsorgungsnachweise vermerkt sind. Dies wurde allerdings „angesichts der Realität der Handwerker und Gebäudenutzer für völlig unrealistisch und auch etwas over-engineered“ eingestuft. Trotzdem sei „BIM [...] das große Ding in der Bauwelt und der einzige echte Standard“.

5.4 ***opensourcebim***

Während es vorwiegend kommerzielle Produkte gibt, die Unternehmen das Arbeiten mit BIM ermöglichen, existiert auch hier eine aktive Open Source Bewegung. Unter dem Namen „The open source BIM collective“ oder kurz „opensourceBIM“ werden derzeit um die 70 Repositories betrieben. Darin enthalten sind unter anderem ein BIM-Server inklusive verschiedener Clients für Endanwender auf unterschiedlichen Systemen und Werkzeuge, die es erleichtern den IFC Files zu arbeiten [49]. Ein kurzer Test hat gezeigt, dass die in Blender modellierte IFC Files tatsächlich über einen „Anzeige-Client“, der mit einem lokal gehosteten BIM-Server verbunden ist, angezeigt werden können. Obwohl die Verwendung des BIM-Servers für diese Arbeit nicht notwendig ist, besteht die Option

diesen künftig mit in den Workflow zu integrieren, da damit auch das simultante Arbeiten an einem IFC File möglich ist, was in Blender nur teilweise und mit dem Einsatz von Plugins ermöglicht wird. Das stellt einen Praxisbezug zum aktuell verwendeten Stand dieser Technologien her, was in der oftmals konzeptionellen Natur der Forschung nicht immer der Fall ist. Wie auch Blender unterstützt BIM-Server das Einbinden von eigenen Plugins, sodass eine Erweiterung um neue Funktionalität möglich ist. Die Plugins werden in Java geschrieben. Der Server bietet aber auch eine REST Schnittstelle an, um Clients in anderen Sprachen anzubinden.

5.5 Mauerwerksbau

Der Mauerwerksbau ist eine Art des Massivbaus, bei welchem Natur- oder Formsteine aufgeschichtet werden, um Wände beziehungsweise Mauern zu errichten. Eine derart erbaute Wand besteht demnach aus (Bau)-Steinen und den dazwischen entstehenden Fugen. Mörtel ist dabei nicht zwangsläufig notwendig. Man spricht von trocken versetzten Steinen oder einer Trockenmauer, wenn darauf verzichtet wird. Heutzutage wird fast ausschließlich mit quaderförmigen Formsteinen gebaut. Zur Beschreibung solcher Formsteine existieren zwei relevante Größen. Die eine ist das sogenannte *Baunennmaß*, mit dem die tatsächliche Größe des Steins angegeben wird. Die andere das *Baurichtmaß*, das sich aus Baunennmaß und dem Fugenmaß zusammensetzt.

5.5.1 Maßsysteme

Als Maßsystem bezeichnet man das Vorgeben der Größen von Bausteinen anhand eines konkret definierten (Grund-)Moduls. Zwei in Deutschland populäre Grundmodule sind zum einen das *ISO-2848-Basismodul* mit Länge 100mm und zum anderen der Achtelmeter, verankert in der *DIN 4172 Maßordnung im Hochbau* [42][51]. Letzteres bezeichnet man daher auch als das oktometrische Maßsystem. Nachfolgend wird auf dieses Maßsystem näher eingegangen.

Oktometrisches Maßsystem

Baurichtmaße sind gemäß dem oktometrischen Maßsystem immer ein Vielfaches von $12,5\text{cm}$ (das entspricht $1/8\text{m}$) und nach der Norm aber mindestens 6.25cm . Dies gilt sowohl für Länge und Breite als auch für die Höhe der Steine. Das System ist in der *DIN 4172 Maßordnung im Hochbau* geregelt und ist ein fest definiertes Grundmaß für das Bauwesen in Europa [51]. Daraus gehen insbesondere folgende zwei Formate für Ziegelsteine hervor: Das Normalformat (NF) mit $240 \times 115 \times 71\text{mm}$ und das Dünnformat (DF) mit $240 \times 115 \times 52\text{mm}$ (Länge \times Breite \times Höhe) [15]. Alle anderen Formate werden mithilfe dieser beiden Grundsteine angegeben. So sind zum Beispiel die in Abbildung 5.3 gezeigten 2 DF und 6 DF Steine eine Kombination aus mehreren Steinen im Dünnformat. Dabei sieht die Norm ein Fugenmaß von 10mm für Stoßfugen (vertikal) und 12mm für Lagerfugen (horizontal) vor. Für Systeme, die eine schmalere oder keine Fuge benötigen, werden entsprechend größere Steine hergestellt, um der Maßordnung weiterhin zu entsprechen. Durch Einhalten eines Systems ist man zusätzlich in der Lage Türen

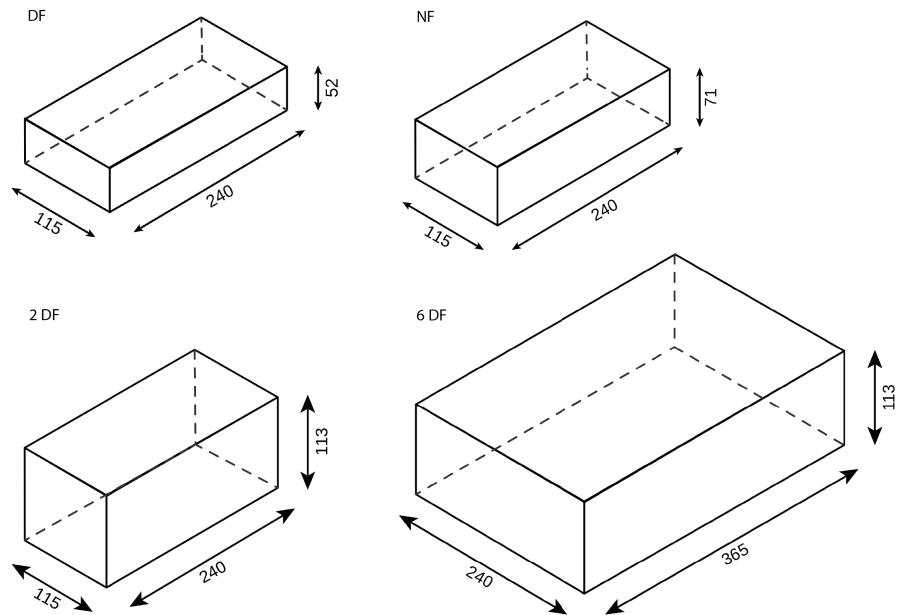


Abbildung 5.3: Darstellung verschiedener Steinformate nach DIN 4172 (Baunennmaß in Millimetern) [48]

und Fenster an die daraus entstehenden Öffnungsgrößen anzupassen und vermeidet dadurch zeitaufwendiges, nachträgliches Anpassen. Da Höhe, Breite und Länge der Steine zusammen mit den dazwischenliegenden Fugen aufgrund des Maßsystems jeweils Vielfache voneinander sind, ergeben sich viele Möglichkeiten zur Aufschichtung und Aneinanderreihung der Bausteine. Einige davon sind in Abbildung 5.4 zu sehen und sind gleichzeitig Beispiele für einen sogenannten *Mauerwerksverband*.

5.5.2 Mauerwerksverband

Als Mauerwerksverband bezeichnet man bestimmte, gleichmäßige Anordnungen von Mauersteinen, um einen homogenen Mauerwerkskörper zu erreichen [44]. Damit kann eine gleichmäßige Kraftverteilung innerhalb der Mauer gewährleistet werden. Eine wichtige Rolle nimmt dabei das Überbindemaß ein, welches die Mindestüberlappung von Mauersteinen aus zwei Schichten der Mauer vorgibt. Für das planmäßige Überbindemaß l_{ol} gilt für übliche Mauersteine mit Schichthöhen $h_u \leq 249\text{mm}$ nach DIN EN 1996-1-1: $l_{ol} \geq 0,4h_u \geq 45\text{mm}$ [24][52]. Zudem wird darin die Mindestwanddicke für tragendes Mauerwerk, „sofern aus Gründen der Standsicherheit, der Bauphysik oder des Brandschutzes nicht größere Dicken erforderlich sind“ [24], auf $t_{min} = 115\text{mm}$ festgelegt [52]. Dies ist, wie in Abbildung 5.3 zu sehen, exakt die Breite der kleinsten Ziegelformate NF und DF. Man unterscheidet zwei Arten von Mauerwerk: das Einsteinmauerwerk und das Verbandsmauerwerk. Wie schon dem Namen zu entnehmen, handelt es sich beim Einsteinmauerwerk um ein Mauerwerk, bei welchem die Wanddicke der Steindicke entspricht. Hier muss das Überbindemaß lediglich über die Wandlängsrichtung eingehalten werden. Bei Verbandsmauerwerk gilt dies zusätzlich für die Wandquerrichtung [22].

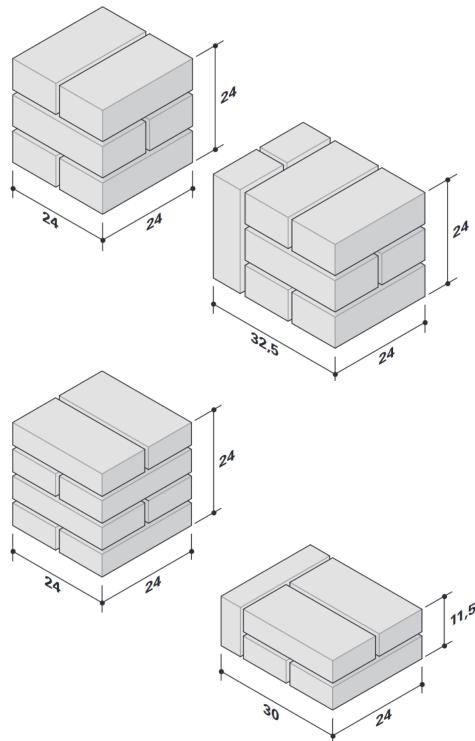


Abbildung 5.4: Besondere Eigenschaften der oktametrischen Maßordnung [15]

Einige Beispiele sind in Abbildung 5.5 zu sehen. Als Wandstück wird von nun an ein gerader Wandabschnitt bezeichnet. Dieser hat eine Länge, eine Höhe und durch einen vorgegebenen Bausteintyp und einem gewünschten Verband eine gewisse Breite. Treffen zwei oder mehrere Wandstücke aufeinander, so gilt es diese dem Überbindemaß entsprechend miteinander zu verzähnen. Dabei treten verschiedene Sonderfälle auf, die für unterschiedliche Verbände nach unterschiedlichen Lösungen verlangen:

1. **Ecken** werden durch zwei sich an Wandenden berührenden, meist rechtwinklig zueinanderstehenden Wandstücken gebildet.
2. **Kreuzungen** stellen zum Beispiel zwei sich kreuzende Wandstücke dar.
3. **T-Kreuzungen** entstehen, wenn ein Wandende eines Wandstücks auf einer anderen Wand steht. Sowohl bei Kreuzungen als auch T-Kreuzungen kann auf das aufwendige Verahnen verzichtet und stattdessen die sogenannte Stumpfstoßtechnik angewandt werden. Dabei werden Stahlanker zwischen der Wand und den darauf treffenden „stumpfen“ Wandenden verwendet, um die beiden Wände sicher miteinander zu verbinden.
4. **Wandenden** sind die „Enden“ eines Wandstücks, die kein anderes Wandstück berühren. Dafür muss der verwendete Mauerwerksverband zu einem geraden Abschluss gebracht werden. Oftmals lässt sich das Anpassen der Bausteine (etwa durch zerschneiden) nicht vermeiden.

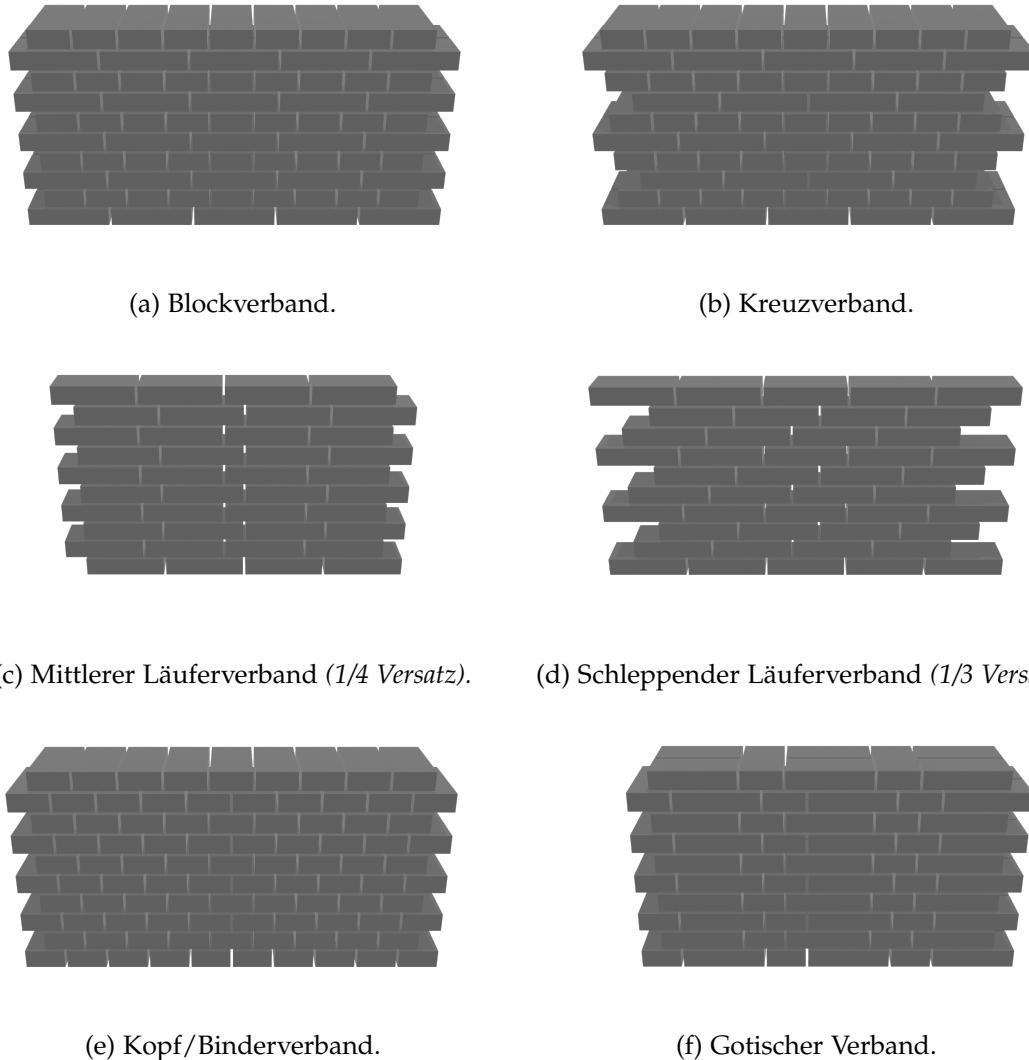


Abbildung 5.5: Typische Mauerwerksverbände.

5. **Öffnungen** innerhalb eines Wandstücks können in der selben Art behandelt werden, da der vorherrschende Verband in den betroffenen Schichten gerade unterbrochen werden muss. Über Öffnungen für Fenster und Türen wird ein sogenannter Sturz gelegt, welcher ebenfalls in den der Wand zugrunde liegenden Mauerwerksverband eingebunden werden muss.

Die Lösungen für die oben genannten Situationen variieren je nach angestrebten Mauerwerksverband und der verwendeten Modulgröße stark. Einige Beispiele sind in Abbildung 5.6 und Abbildung 5.7 zu sehen [15][31]. Während etwa beim Läuferverband darauf verzichtet werden kann Bausteine für Eckbereiche und Kreuzungen zu zerschneiden, ist dies für andere Verbände teilweise unumgänglich.

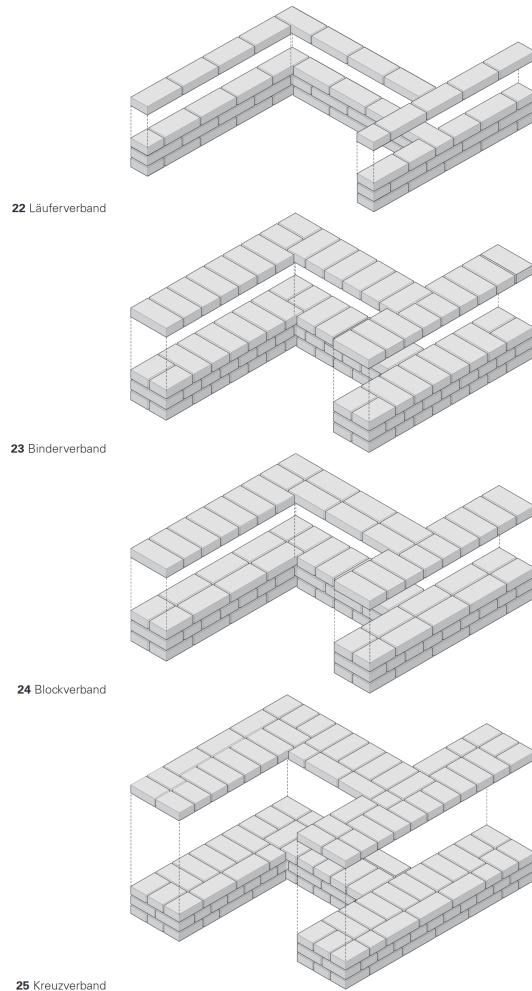


Abbildung 5.6: Lösungen für Ecken und T-Kreuzungen unterschiedlicher Verbände [15]

5.6 LEGO

Ein 1×1 LEGO Stein hat eine quadratische Grundfläche von $7.8mm \times 7.8mm$. Dies entspricht demnach dem Baunennmaß des 1×1 LEGO Steins. Zwischen zwei nebeneinander platzierten Steinen ist ein Abstand von $0.2mm$. Daraus ergibt sich ein Baurichtmaß beziehungsweise ein Rastermaß von $8mm \times 8mm$. In Abbildung 5.8 werden zur Veranschaulichung die Maße des populären 2×4 Steines aufgeschlüsselt. Die für ein dreidimensionales Maß noch fehlende Größe ist die Höhe der Steine. Diese beträgt $9.6mm$. Der Abstand zwischen zwei übereinander gestapelten Steinen hängt von dem Druck ab, der beim Zusammenstecken geleistet wurde. Dennoch kann dieser vernachlässigt, sprich als Abstand von $0.0mm$ gewertet werden.

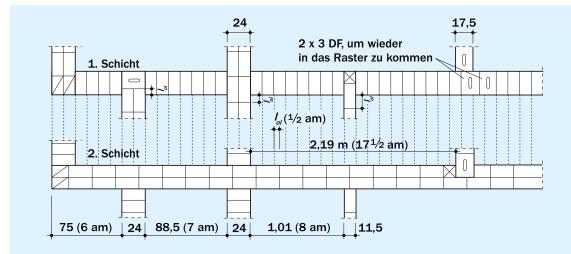


Abbildung 5.7: Lösung einer Kreuzung am Beispiel einer „Wand aus 2 DF im Kreuz- und Blockverband“ [31]

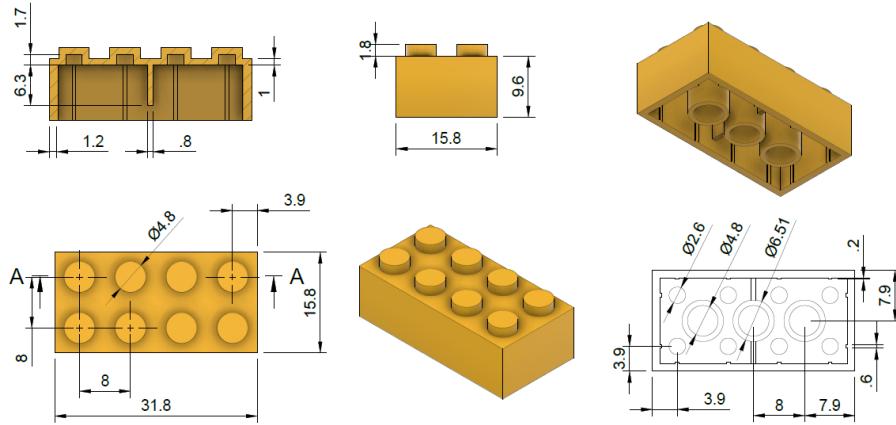


Abbildung 5.8: Maße des Standard 2×4 LEGO Steins [43]

5.7 BRep

Mal schauen

5.8 Ontologie

Der Begriff Ontologie stammt ursprünglich aus der Philosophie. Wörtlich übersetzt bedeutet er „Lehre des Seins“. Ein Definitionsverschlag lautet wie folgt (aus dem Englischen): „Die Ontologie als Teilgebiet der Philosophie ist die Wissenschaft von dem, was existiert, den Arten und Strukturen von Objekten, Eigenschaften, Ereignissen, Prozessen und Relationen in allen Bereichen der Wirklichkeit“ [7]. Eine Anmerkung, die oftmals im Anschluss eines Definitionsverschlags zu finden ist, besagt, dass dabei nicht nur das, *was ist*, sondern ebenfalls das, *was sein könnte* betrachtet werden müsse [7][20]. Dieser Gedanke spiegelt sich ebenfalls in dem Ontologiebegriff aus der Informatik wider und ist unter dem Begriff *Open World Assumption* bekannt. Das wird nach der Definition von Ontologien im Kontext der Informatik noch einmal ausführlicher aufgegriffen.

Im Fachbereich der Informatik wurde der Begriff Ontologie adaptiert und zunächst als „*explicit specification of a conceptualization*“ definiert [1]. Wörtlich übersetzt lautet diese Definition etwa „explizite Spezifizierung einer Konzeptualisierung“. Im Laufe

der Jahre wandelte sich diese Definition, sodass sich daraus zum Beispiel folgendes ergab: „An ontology is a formal, explicit specification of a shared conceptualization“ [2]. Hier wird eine Ontologie also als „formale, explizite Spezifizierung einer geteilten Konzeptualisierung“ definiert. Den wichtigsten Zusatz hierbei stellt das Wort „geteilt“ dar, da Ontologien unter anderem dem Zweck dienen, Informationen und Vokabulare mehrerer Wissensrepräsentation zu vereinen. Der Begriff Konzeptualisierung wird oft als „vereinfachte, abstrakte Sicht auf einen für einen bestimmten Nutzen relevanten Teil der Welt“ beschrieben [5].

Eine solche „Spezifizierung einer Konzeptualisierung“ einer gewissen Domäne kann heutzutage mithilfe der sogenannten *Web Ontology Language* (OWL), welche auf dem *Resource Description Framework* (RDF) aufbaut, realisiert werden [6][12]. OWL erlaubt es Klassen zu definieren und daran Eigenschaften anzuhängen. Außerdem können sogenannte Individuen beziehungsweise Instanzen erstellt werden. Diese realisieren eine oder mehrere Klassen. Mithilfe dieser Klassen, Eigenschaften und Individuen können sogenannte *Reasoner* logische Schlussfolgerungen über implizit angegebenes Wissen ziehen. Eine Ontologie kann damit einerseits etwa mit neuen Klassenzuordnungen oder implizit angegeben Vererbungen erweitert, andererseits auf Validität überprüft werden. Werden Verletzungen von aufgestellten Axiomem oder widersprüchliche Einschränkungen von Klassen oder Eigenschaften entdeckt, so kann ein Reasoner diese nicht nur melden, sondern auch mithilfe logischer Beweisführung herleiten. Damit stellen Reasoner ein wertvolles Werkzeug für die Entwicklung und Erweiterung von Ontologien dar. Drei weit verbreitete Reasoner sind:

- ELK Reasoner der Universitäten Ulm und Oxford [11].
- HermiT Reasoner von der *Information Systems Group* der Universität Oxford [10][19].
- Open Source Pellet Reasoner der Clark & Parsia LLC [4].

Aufgrund der oben genannten Idee der *Open World Assumption* unterscheidet sich die Art der Reasoner logische Schlussfolgerungen zu ziehen von der, mit welcher logische Aussagen von zum Beispiel Programmiersprachen bewertet werden. Im Grunde versteht man unter der *Open World Assumption*, das nicht Falsifizieren eines logischen Ausdrucks, falls Wissen existieren könnte, das den Ausdruck doch verifizieren würde. Ein Reasoner beziehungsweise eine Programmiersprache im Kontext der *Closed World Assumption* würde in so einer Situation davon ausgehen alle existierenden Informationen darüber bereits vorliegen zu haben und eventuell zu einem anderen Schluss gelangen. Definiert man zum Beispiel eine Klasse A als alle Individuen, die nur eine Eigenschaft E besitzen, die sie mit Individuen einer bestimmten anderen Klasse B verbindet, so kann ein Reasoner innerhalb dieser Ontologie ein Individuum, das zwar diese Anforderung erfüllt, nicht der Klasse A zuordnen, da es sein kann, dass zu einem späteren Zeitpunkt (oder an anderer Stelle) Informationen existieren, die diese Zuordnung fehlerhaft machen würden. In einer Umgebung, in der die *Closed World Assumption* gilt, wäre diese Zuordnung hingegen völlig korrekt. Diese Eigenschaft ermöglicht es ebenfalls die Idee des sogenannten „Semantic Web“ mithilfe von Ontologien zu realisieren [3]. Denn die Natur des Internets entspricht eher einer sich ständig ändernden und „offenen“ Welt, als einer Geschlossenen. Als

Semantic Web wird das Annotieren von sämtlichen Webinhalten mit semantischen Informationen bezeichnet. Damit wird das Ziel verfolgt, ein maschinenlesbares Internet zu schaffen.

5.8.1 Protégé

Protégé ist ein graphischer Editor zur Erstellung und Instandhaltung von Ontologien [13]. Entwickelt wird das Programm an der Universität Stanford und ist als Open Source Anwendung kostenfrei nutzbar. Die erste Version wurde bereits 1999 veröffentlicht. Durch seine Plugin-Struktur können neben zahlreichen Erweiterungen beispielsweise auch neue Reasoner an das Programm angeschlossen werden. Standardmäßig sind der ELK und der HermiT Reasoner integriert, aber es existieren Plugins um auch Pellet in Protégé zu integrieren. Das Arbeiten mit einem graphischen Editor erleichtert den Einstieg und ermöglicht später einen besseren Überblick über oftmals umfangreiche Ontologien, als es in einem herkömmlichen Texteditor möglich wäre. Darum wurde das Programm zur Erstellung einer Ontologie für diese Arbeit herangezogen.

5.8.2 Owlready2

Owlready2 ist eine Python 3 Bibliothek, die es ermöglicht „ontologie-orientiert“ zu programmieren [14]. Bisher mussten Ontologien mithilfe von Abfragesprachen (etwa SPARQL) oder APIs bearbeitet und ausgewertet werden [8]. Owlready2 bietet hingegen einen einfacheren Umgang mit Ontologien, eben damit den Einstieg in das Thema und ermöglicht gleichzeitig die Integration von Ontologien in jedes mit Python umsetzbare Projekt. Im Zusammenspiel mit Protégé bietet diese Bibliothek eine für Programmierer intuitive Möglichkeit, die oben genannten Werkzeuge für IFC und BIM durch Nutzung von Python 3 direkt mit der Technologie der Ontologien zu koppeln.

6 Konzept

Dieser Einleitungssatz leitet mit durchdachten Worten das Kapitel ein, in dem es um die Konzepte geht, die ich mir ganz sicher vor der Implementierung ausgedacht habe! Hier ist allerdings anzumerken, dass dies nur ein Platzhalter ist und mithilfe dieses TODOs als solcher gekennzeichnet ist. Ab jetzt sind wir aber alle mal wieder seriös.

6.1 Modellierung

Für gewöhnlich sind beim Planungsprozess eines Gebäudes oder anderer Infrastruktur eine Vielzahl an Experten aus unterschiedlichen Disziplinen beteiligt. Ein Ziel dieser Arbeit ist es allerdings eine intuitive Konstruktionsplanung zu ermöglichen, sodass eine Einzelperson mit relativ geringer Einlern- und Modellierungszeit in der Lage ist ein Gebäude zu entwerfen. Dieser Entwurf muss dennoch alle notwendigen Informationen für die anschließende Bauplandeduktion enthalten, ohne dass das Einpflegen dieser Daten spezielles Fachwissen voraussetzt.

Oftmals lässt sich die Komplexität einer Sache oder eines Vorgehens durch die Vorgabe von Einschränkungen reduzieren. Dabei ist es allerdings wichtig diese Einschränkungen so zu wählen, dass die damit erzielten Ergebnisse nach wie vor von Nutzen sind.

6.1.1 Raster

Im Fall von Gebäude- oder besser Gebilde-Konstruktionen existieren bereits einige Beispiele, die durch Einschränkungen so stark vereinfacht werden, dass sogar Kinder damit umgehen können. Das wohl bekannteste ist das Lego System (siehe Kapitel 5.6). Neben dessen nützlichen Steckverbindungen, die es ermöglichen ohne Anwenden von Klebstoff oder Schrauben Steine aneinander zu befestigen, ist für diese Arbeit das dadurch vorgegebene Raster ein interessantes Konzept zur Vereinfachung der Modellierung von Gebäuden. In Kapitel 5.5 wurden auch schon die Begriffe *Baunennmaß* und des *Baurichtmaß* eingeführt und das oktametrischen Maßsystem vorgestellt. Dies entspricht im Prinzip ebenfalls einem Raster, das aber in Realität durch die Möglichkeit des Zerschneidens von Bausteinen nicht zwingend eingehalten werden muss. Da das Vorgeben eines Rasters, dem sowohl die Größen der zu verwendenden Bausteine, als auch (im Fall des Lego Systems) deren Platzierung folgen müssen, eine Einschränkung darstellt, die im Einklang mit der Intuition vieler Menschen steht, wurde dies in das Modellierungskonzept dieser Arbeit integriert. Das Vorgeben eines einzigen, fest definierten Rasters stellt allerdings eine zu große Einschränkung dar, weshalb das Definieren verschiedener Rastergrößen möglich sein muss. So können Modelle erstellt werden, die sowohl genau dem Lego System oder dem oktametrischen Maßsystem entsprechen, als auch beliebig anderen Rastern.

Die Modellierungsumgebung kann mithilfe von Rasterinformationen eines Objektes für dessen korrekte Platzierung, Skalierung und Rotierung sorgen, indem eine solche Transformation auf die nächstliegende Größenordnung des Rasters gerundet wird. Im Beispiel eines Rasters von $[1.0m, 1.0m, 1.0m]$ würde demnach ein auf $[0.9m, -0.7m, 0.1m]$ zu translatierendes Objekt an die Position $[1.0m, -1.0m, 0.0m]$ versetzt werden. Da aber gewünschte valide Rotationen nicht aus einem derart vorgegebenen Raster interpretierbar sind, kann diese zusätzliche Information mithilfe eines kleinstmöglichen Winkelschritts angegeben werden. In den Modellen der Fallstudien aus Kapitel 3 werden zum Beispiel ausschließlich Rotationen eines Vielfachen von 90° verwendet, was neben den Rastern ebenfalls hinterlegt wurde. Damit kann die Modellierungsumgebung auch den Rotationen von Objekten durch Runden eine Art Raster aufzwingen.

6.1.2 Wandstück

Auch die schiere Menge verschiedener Bestandteile eines Hauses ist für eine Einzelperson ohne Vorkenntnisse nicht überblickbar. Da das Ziel dieser Arbeit das automatische Generieren von Legeplänen für Bausteine innerhalb der Wände eines Gebäudes darstellt, lässt sich diese Menge vorerst auf zwei wesentliche Objekttypen reduzieren: Wände und Öffnungen in Wänden. Öffnungen werden zum Beispiel für Fenster oder Türen benötigt. Da eine Wand im Prinzip ein arbiträrer geometrischer Körper sein kann, dies abzubilden aber wieder die Komplexität der Modellierung steigert, wird deren Form auf einen beliebig skalierten Quader beschränkt. Ein solcher Quader wird nachfolgend auch als *Wandstück* bezeichnet. Ein solches Wandstück besitzt demnach auch die Eigenschaften eines Quaders: eine Skalierung, eine Rotation um seinen eigenen Mittelpunkt und eine Translation im Raum. Nachfolgend werden die Begriffe Länge, Breite und Höhe zur einfachen Unterscheidung von einer Skalierung in X, Y und Z Richtung verwendet. Während Länge und Höhe dem Raster entsprechend beliebig gewählt werden können, ergibt sich die Breite aus dem gewählten Bausteinformat (auch als *Modul* bezeichnet) und dem geplanten Mauerwerksverband (siehe Kapitel ??). Der Verband und das Modul können jedem Wandstück als sogenannter *Wandtyp* zugewiesen werden. Dadurch ist es möglich, verschiedene Arten von Wänden innerhalb eines Gebäudemodells zu verwenden. Diese Informationen werden später dafür verwendet, die durch das Wandstück abgesteckten Dimensionen sinnvoll mit Bausteinen zu füllen.

6.1.3 Mauerwerksverband

In Kapitel 5.5.2 wurden bereits einige verschiedene Mauerwerksverbände behandelt. Um dem Wall Detailing Verfahren diese und beliebig andere Verbände in einer interpretierbaren Weise zur Verfügung stellen zu können, wird dafür eine mathematische Notation benötigt. Bei genauer Betrachtung handelt es sich bei den meisten Mauerwerksverbänden um ein sich wiederholendes Muster. Dabei wiederholen sich sowohl die Bausteintransformationen innerhalb einer Schicht des Verbandes, als auch dessen Schichten selbst. Daher sind folgende Informationen zur Beschreibung eines Mauerwerksverbands notwendig:

1. eine Menge einzigartiger Schichten, bis diese sich (mit einem Versatz) wiederholen.

2. Für jede Schicht eine Menge verschiedener Bausteintransformationen, bis diese sich (ebenfalls mit einem Versatz) wiederholen.

Ein Mauerwerksverband kann damit als Menge von Schichten verstanden werden, die sich wiederum aus einer Menge von Bausteintransformationen zusammensetzen. Diese Schichten besitzen jeweils einen festgelegten, darunter liegenden Vorgänger sowie einen darüber liegenden Nachfolger. Das bedeutet, die Reihenfolge in der Schichten eines Mauerwerksverbands auftreten, darf nicht verändert werden. Leicht zu sehen ist dies am Beispiel eines Kreuzverbandes, der durch Veränderung der Reihenfolge seiner Schichten, seine typische kreuzartige Form verlieren würde.

Eine Bausteintransformation wird in dieser Arbeit als Tupel mit folgendem Inhalt definiert:

1. Eine skalierbare Position $\vec{p} = [x, y, z]^T$.
2. Eine skalierbare Rotation $r = (x, y, z)$ angegeben in Euler-Winkel.
3. Einen festen translativen Versatz $\vec{p}_o = [x, y, z]^T$.
4. Einen festen rotierenden Versatz $r_o = (x, y, z)$ ebenfalls in Euler-Winkel.

Skalierbar bedeutet in diesem Fall, dass die Werte von beispielsweise der Position eines Bausteins in Abhängigkeit eines veränderbaren Faktors stehen. Zur endgültigen Platzierung eines Bausteins anhand einer Bausteintransformation aus einer Schicht eines Mauerwerksverbands wird lediglich noch ein Faktor benötigt, der angibt, die wievielte Wiederholung der Menge an Bausteintransformationen einer Schicht gerade Anwendung findet. Sei $B_1 = (\vec{p}, r, \vec{p}_o, r_o) = ([l, 0, 0]^T, (0, 0, 0), [0, 0, 0]^T, (0, 0, 0))$ eine Bausteintransformationen, $M = (l, b, h) = (2, 1, 1)$ ein Modul und $c \in \mathbb{N}$ ein Skalar, dann ergibt sich die Position P eines Bausteins zur Wiederholung $c = 1$ wie folgt: $P = \vec{p}_o + c * \vec{p} = [0, 0, 0]^T + 1 * [2, 0, 0]^T = [2, 0, 0]^T$. Im nächsten Wiederholungsschritt mit $c = 2$ ergibt dies eine Position $P = [4, 0, 0]^T$. Da die Rotationen als Euler-Winkel vorliegen, kann diese in ähnlicher Weise berechnet werden. Sie beschreibt die Rotation um den Mittelpunkt des Bausteins. In den meisten Fällen ist r allerdings gleich $(0, 0, 0)$ und r_o entweder ebenfalls $(0, 0, 0)$ oder $(0, 0, \frac{\pi}{2})$. Letzteres entspricht einer festen Rotation um 90° um die Z-Achse des Bausteins, unabhängig des Wiederholungsschritts r .

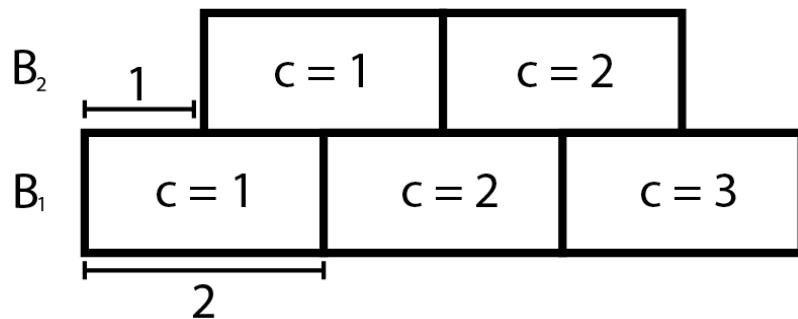


Abbildung 6.1: Der Läuferverband mit einem Versatz von 50%.

Mit $B_2 = ([l, 0, 0]^T, (0, 0, 0), [\frac{l}{2}, 0, 0]^T, (0, 0, 0))$ und je einer Schicht gebildet aus den Mengen $\{B_1\}$ bzw. $\{B_2\}$ erhält man bereits eine komplette Beschreibung für den Läuferverband mit einem Versatz von 50% der Steinlänge. Dieses Beispiel ist in Abbildung 6.1 schematisch dargestellt. Aber auch andere, komplexere Verbände können in dieser Form abgebildet und so dem nachfolgenden Wall Detailing in abstrahierter Weise übergeben werden.

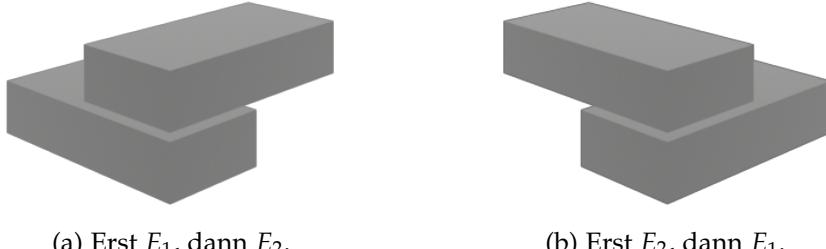


Abbildung 6.2: Eckpläne für den Läuferverband mit einem Versatz von 50%.

Die speziellen Legepläne zur korrekten Auflösung von kritischen Bereichen wie zum Beispiel Ecken können ebenfalls in dieser Form für jeden Mauerwerksverband angegeben werden. Für das vorangegangene Beispiel sähe der Eckplan wie folgt aus: Eine Nulltransformation $E_1 = ([0, 0, 0]^T, (0, 0, 0), [0, 0, 0]^T, (0, 0, 0))$ in der ersten Schicht und eine Bausteintransformationen $E_2 = ([0, 0, 0]^T, (0, 0, 0), [0, 0, 0]^T, (0, 0, \frac{\pi}{2}))$ mit einer festen 90° Rotation um die Z-Achse in der Zweiten. Dieser Eckplan ist in Abbildung 6.2 zu sehen. Ein Beispiel, in welchem dieser Eckplan angewandt wird, wurde bereits in Abbildung 5.6 in Kapitel 5.5.2 gezeigt. Der einzige Unterschied zwischen dem Plan eines Mauerwerksverbands für gerade Wände und dessen Spezialplänen ist deren Anwendung. Während aus dem Eckplan später für jede Schicht nur ein Baustein pro hinterlegter Bausteintransformation erzeugt werden muss, können beliebig lange gerade Wandabschnitte mithilfe des oben angesprochenen Wiederholungsschritts mit Bausteinen aufgefüllt werden.

6.1.4 Beziehungen zwischen Wandstücken

In dieser Arbeit stehen Wandstücke zueinander in einer Beziehung, sobald sie sich berühren oder schneiden. Diese Beziehungen sind relevant, um die gewählten Mauerwerksverbände ordnungsgemäß über mehrere voneinander abhängige Wandstücke anzuwenden, ohne dass es zu Verletzungen von Vorschriften und Normen wie etwa dem Überbindemaß aus Kapitel 5.5.2 kommt. Wie bereits in Abschnitt 3.1.1 und 5.5.2 angesprochen, treten verschiedene Arten von Beziehungen auf, die für das nachfolgende „Wall Detailing“ zu unterscheiden sind. Zur Modellierung von Gebäuden sind vor allem folgende Beziehungen unabdingbar: *Ecken*, *T-Kreuzungen* und *X-Kreuzungen*. Diese stellen wichtige Elemente bei der Planung von Gebäuden dar. Nur so ist es möglich abgeschlossene und aneinanderhängende Räume zu erstellen. Eine weitere, nicht ganz offensichtliche Beziehung besteht zwischen zwei Wandstücken, die in Verlängerung zueinander stehen, also gemeinsam einen größeren Wandbereich abdecken. Diese Beziehung wird nachfolgend *Wandstückverbund* genannt. Bei allen vier Beziehungen handelt es sich Sonderfälle,

für welche sich der Mauerwerksverband nicht ohne weiteres anwenden lässt, sondern teilweise explizit vorgegebene Legepläne angewandt werden müssen. Darum ist es notwendig diese Bereiche ausfindig zu machen und voneinander unterscheiden zu können. Es werden nun für jede dieser Beziehungen Eigenschaften aufgezeigt anhand derer man diese in einer Menge an Wandstücken erkennen kann.

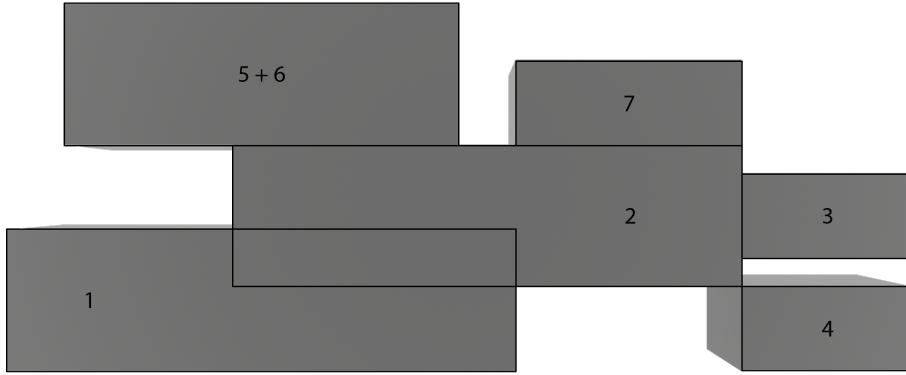


Abbildung 6.3: Modell einer Wand, die durch 7 einzelne Wandstücke des selben Typs gebildet wird, deren Mittelpunkte alle auf einer Ebene liegen.

Wandstückverbände sind alle Wandstücke, die zwar in dem Modellierungsprozess des Gebäudes durch mehrere einzelne Wandstücke realisiert wurden, eigentlich aber eine Einheit darstellen. Es ist notwendig alle Wandstücke zu identifizieren, die einen zusammenhängenden Wandbereich bilden, um den gewählten Mauerwerksverband korrekt über alle Wandstücke hinweg anzuwenden. Andernfalls würde jedes Wandstück den Mauerwerksverband unabhängig seiner Nachbarn neu beginnen und ihn dadurch zwischen Wandstückübergängen häufig fälschlicherweise unterbrechen. Ein umfangreiches Beispiel ist in Abbildung 6.3 dargestellt. Um zu überprüfen, ob zwei Wandstücke eine Einheit bilden, kann das Paar auf folgende Eigenschaften getestet werden:

1. Beide Wandstücke verwenden das selbe Modul und sind während der Modellierung mit den gleichen Wandtyp annotiert worden. Dies verhindert vor allem das Kombinieren unterschiedlich dicker Wände.
2. Die lokalen Z-Achsen beider Wandstücke sind parallel. Dies verhindert das Kombinieren unpassend rotierter Wandstücke. Die Überprüfung der Parallelität der Z-Achsen beider Wandstücke wird wie folgt durchgeführt:

Sei $\vec{v}_z = [0, 0, 1]^T$ eine Z-Achse und $v_z = (v_w, v_x, v_y, v_z) = (0, 0, 0, 1)$ das dazugehörige Quaternion. Außerdem seien q_1 und q_2 die beiden Rotationen der Wandstücke. Dann stellen $v_{z^1} = q_1 * v_z * q_1^{-1}$ und $v_{z^2} = q_2 * v_z * q_2^{-1}$ die jeweiligen „Z-Anteile“ dieser Rotationen dar. Daraus ergeben sich die „Z-Vektoren“ wie folgt:

$\vec{v}_{z1} = [v_{z_x^1}, v_{z_y^1}, v_{z_z^1}]^T$ und $\vec{v}_{z2} = [v_{z_x^2}, v_{z_y^2}, v_{z_z^2}]^T$. Ist nun $|\vec{v}_{z1} \cdot \vec{v}_{z2}| = 1$ gegeben, so sind die lokalen Z-Achsen der Wandstücke gleich oder um exakt 180° verdreht und damit Parallelität erfüllt.

3. Die lokalen X-Achsen beider Wandstücke sind ebenfalls parallel. Das Vorgehen zur Überprüfung entspricht dem von Punkt 2, nur dass v_z durch $v_x = (0, 1, 0, 0)$, also der X-Achse, ersetzt werden muss.
4. Sie stehen auf der selben Höhe oder versetzt um ein Vielfaches der gemeinsamen Modulhöhe.
5. Es liegt eine Berührung oder gar eine Überlappung vor.

Insgesamt bilden in dem Beispiel in Abbildung 6.3 demnach alle Wandstücke mit Ausnahme von Wandstück Nummer 4 eine Einheit, die alle obigen Eigenschaften erfüllt. Dies sind in der Tat all die Wandstücke, die den Start des gewählten Mauerwerksverbands ihren Nachbarn entsprechend anpassen müssen, sodass sich der Verband gleichmäßig über alle Wandstücke erstreckt.

Ecken, T-Kreuzungen und X-Kreuzungen stellen Beziehungen zwischen einzelnen Wandstücken oder den kombinierten Wandstückverbänden aus Abschnitt 6.1.4 dar. Wie schon zu Beginn dieses Abschnitts vorweggenommen, existieren für jeden Mauerwerksverband und je nach verwendetem Modul eigene Varianten für Eck- und Kreuzungslegepläne, an welchen sich Maurer orientieren. Einige Beispiele hierfür wurden schon in Kapitel 5.6 gegeben. Der Grund dafür ist die Komplexität in diesen Bereichen die vorgeschriebenen Normen (wie etwa dem in Abschnitt 5.5.2 genannten Überbindemaß) einzuhalten. Diese speziellen Pläne werden dann an den notwendigen Stellen angewandt - meistens bevor ein dazwischenliegender gerader Wandabschnitt mit Bausteinen aufgeführt wird. Es liegt deshalb nahe, die Eck- und Kreuzungslegepläne zur Beschreibung des Mauerwerksverbands hinzuzufügen, wie es schon im Abschnitt 6.1.3 definiert wurde. Durch das vorangegangene Kombinieren mehrerer Wandstücke zu einer Einheit, können nachfolgend Wandstücke auch das Ergebnis solcher Kombinationen sein, aber aufgrund deren Parallelität wie ein einzelnes behandelt werden. Da sich Ecken, T-Kreuzungen und X-Kreuzungen stark ähneln, besitzen sie einige geteilte Eigenschaften:

1. Beide Wandstücke verwenden das selbe Modul und sind während der Modellierung mit den gleichen Wandtyp annotiert worden.
2. Die lokalen Z-Achsen beider Wandstücke sind parallel. Das Vorgehen zur Überprüfung ist dasselbe wie in Abschnitt 6.1.4.
3. Sie stehen auf der selben Höhe oder versetzt um ein Vielfaches der gemeinsamen Modulhöhe.
4. Mindestens eines der beiden Wandstücke endet auf einem anderen.

Sind diese Eigenschaften erfüllt, werden die Schnittpunkte der Richtungsvektoren entlang der lokalen X-Achsen beider Wandstücke errechnet. Im Falle einer einfachen

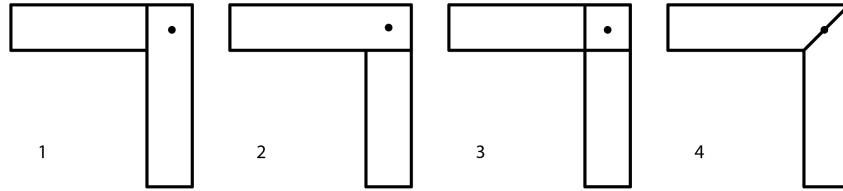


Abbildung 6.4: Draufsicht auf Varianten der Modellierung einer Ecke.

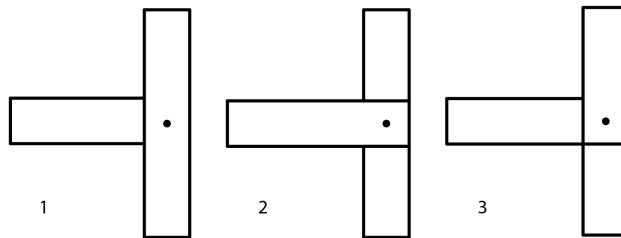


Abbildung 6.5: Draufsicht auf Varianten der Modellierung einer T-Kreuzung.

Ecke oder einer T-Kreuzung, existiert nur ein einzelnes Paar an Wandstücken, die sich im Mittelpunkt der Ecke oder der T-Kreuzung schneiden. Der Unterschied ist lediglich die Stelle des Schnittpunkts relativ zu den beiden Wandstücken. Sei w_1 die Breite von einem und w_2 die des anderen Wandstücks, so gilt nach Punkt 1 $w_1 = w_2$. Liegt der errechnete Schnittpunkt bei beiden Wandstücken näher als $\frac{w_1}{2}$ entlang der lokalen X-Achse an einer der beiden Außenkanten handelt es sich um eine Ecke, da beide Wandstücke an diesem Punkt enden. Falls der Schnittpunkt bei einem der beiden Wandstücke allerdings mindestens $\frac{w_1}{2}$ innerhalb des Wandstücks liegt, so handelt es sich um eine T-Kreuzung. Ein Wandstück steht also auf einem Anderen. Dies ist in den Abbildungen 6.4 und 6.5 zu sehen. Werden für zwei unterschiedliche Paare die selben Schnittpunkte errechnet, so kann es sich entweder um eine mit drei Wandstücken modellierte T-Kreuzung oder X-Kreuzung handeln. Treffen sich an dem Schnittpunkt zwei Ecken, ergibt sich daraus eine T-Kreuzung (siehe Fall 2 in Abbildung 6.5). Teilen sich aber zwei T-Kreuzungen die selben Schnittpunkte, so sind alle Voraussetzungen für eine eine X-Kreuzung gegeben (siehe Fall 1 in Abbildung 6.6). Einen Sonderfall für Punkt 4 stellt eine X-Kreuzung dar -

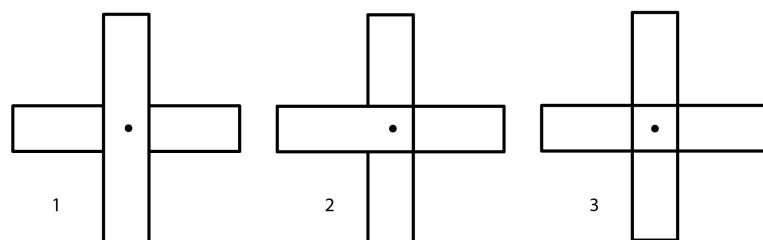


Abbildung 6.6: Draufsicht auf Varianten der Modellierung einer X-Kreuzung.

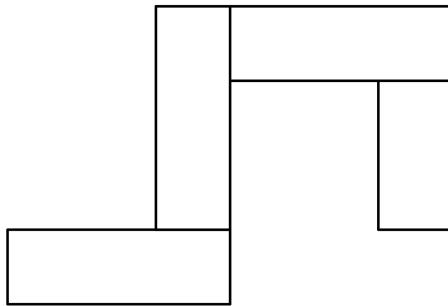


Abbildung 6.7: Draufsicht auf eine Verkettung von 4 Wandstücken durch 3 dazwischenliegende Ecken.

modelliert aus zwei sich schneidenden Wandstücken. Dieser Fall ist in Abbildung 6.6 mit der Variante Nummer 3 dargestellt. Hier liegt der Schnittpunkt des Wandstück-Paars mindestens $\frac{w_1}{2}$ innerhalb beider Wandstücke und keines der Wandstücke endet auf dem Anderen. Durch das vorhergehende Kombinieren von Wandstücken werden die Fälle 3 aus Abbildung 6.5 und 2 aus Abbildung 6.6 eliminiert, denn dabei erfüllen jeweils zwei Wandstücke alle Eigenschaften, um nach Abschnitt 6.1.4 zu einer Einheit verschmolzen zu werden. So entsteht aus Fall 3 aus Abbildung 6.5 der Fall Nummer 1 und auch Fall Nummer 2 aus Abbildung 6.6 wird zu deren 1. Fall.

6.1.5 Lösen von Beziehungen

Durch Ecken, T- und X-Kreuzungen werden ansonsten unabhängige Wandstücke miteinander verkettet. Das kann zu Situationen führen, in welchen das einfache Anwenden eines Mauerwerksverbands Lücken in manchen Wandstücken hinterlässt. Das Beispiel in Abbildung 6.7 soll helfen diese Problematik zu veranschaulichen. Zu sehen ist eine Konstellation von 4 Wandstücken, die in einer Kette durch 3 Ecken miteinander verbunden sind. Das anzuwendende Modul hat die Dimensionen [2, 1, 0.5]. Die Höhe des Moduls ist für dieses Problem allerdings irrelevant, weshalb die folgenden Skizzen ausschließlich die Draufsicht auf die Situationen zeigen. Wie in Kapitel 6.1.3 am Beispiel des Läuferverbands gezeigt, gibt es mehrere Möglichkeiten einem Eckplan zu folgen. Solange die Reihenfolge weiterhin eingehalten wird, kann mit jeder Schicht des Eckplans begonnen werden einen Eckbereich aufzufüllen. Das führt in dem Beispiel aufgrund des zwei-schichtigen Eckplans des Läuferverbands dazu, dass für jede der 3 Ecken zwei mögliche Startschichten existieren. Dies resultiert demnach in insgesamt 8 unterschiedlichen Varianten, von welchen allerdings nicht alle zielführend sind. In Abbildung 6.8 ist eine Variante dargestellt, die Lücken in der Wand hinterlassen würde. Das zunächst naheliegend wirkende Füllen der Löcher durch kleinere Bausteine ist nicht zulässig, da damit das Überbindemaß (siehe Kapitel 5.5.2) verletzt wäre. Darum ist es notwendig eine Kombination an Startschichten über alle miteinander verbundenen Eckbereiche zu finden, die es ermöglicht die verbleibenden geraden Wandstücke im Anschluss durch einfaches Anwenden des Läuferverbands aufzufüllen, ohne dass Lücken entstehen.

Ein Verfahren, welches für das vorliegende Beispiel des halbversetzten Läuferverbands,

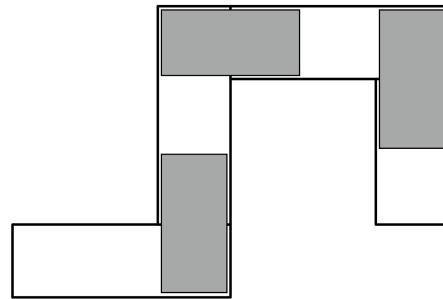


Abbildung 6.8: Draufsicht auf eine unzulässige Lösung der Eckplankonfiguration.

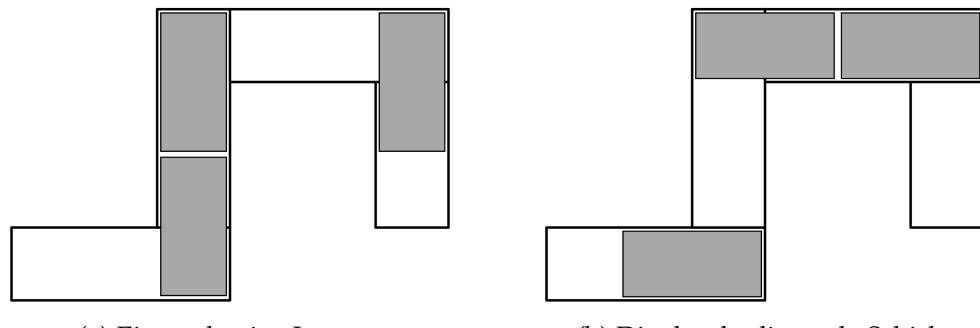


Abbildung 6.9: Eine mögliche Lösung und die durch die Eckpläne entstehende zweite Lösung für die darüber liegende Schicht.

dessen sehr einfachen Eckplans und dem durch seine praktische Form gewählten Moduls Lösungen findet, löst nicht zwangsläufig auch eine beliebig andere Situation. Ebenfalls führt das Miteinbeziehen der 3. Dimension, also der Höhe der beteiligten Wandstücke, zu einer erheblichen Vervielfachung möglicher Eck-Konstellationen. Zieht man etwa das Beispiel aus Abbildung 6.3 heran und erweitert es durch Einfügen weiterer Wandstücke um Ecken an den Außenkanten von Bereich 1, 3, 4, 5+6 und 7, an welchen beliebig weitere verkettete Wandstücke in sich geschlossene Kreise bilden, so können sogar nicht lösbare Fälle auftreten. Durch die Anzahl der Schichten des Eckplans eines Mauerwerksverbands und die Anzahl der miteinander verketteten Wandstücke baut sich schnell ein großer Lösungsraum auf. Aufgrund der Einschränkung eine mögliche Lösung lediglich mit zulässig oder unzulässig bewerten zu können, sind herkömmliche Optimierungsverfahren nicht in der Lage das Problem zu lösen. Diese benötigen meistens eine sogenannte *Fitness Funktion*, welche die Güte der derzeitigen Lösung angibt und dem Optimierungsverfahren sozusagen eine Richtung weist, in der ein lokales Minimum oder Maximum dieser Funktion zu erwarten ist. Wählt man die *Fitness Funktion* sinnvoll, kann damit ein kontinuierlicher Lösungsraum effizient nach einer passablen Lösung durchsucht werden. In dem vorliegenden Fall gibt es allerdings, wenn überhaupt, nur einige wenige zulässige Lösungen, die an den Startindizes der Pläne der jeweiligen Eckbereiche gekoppelt sind. Diese Startindizes weisen aber nicht dieselben Eigenschaften auf, wie kontinuierliche Parameter, die durch kleine Veränderungen eine etwas bessere oder schlechtere Lösung

erzeugen. Das Ändern eines Startindex einer einzelnen Ecke kann zwar eine Lösung mit einer reduzierten Anzahl an Lücken liefern, gibt aber deswegen keine Richtung vor in der man noch bessere Lösungen erwarten kann. Außerdem ist auch eine Lösung mit einer einzigen Lücke nicht zulässig, da dies die Stabilität eines Wandstücks stark beeinflussen könnte.

Darum bleibt zunächst nur die Brute-Force-Methode (auch als Exhausionsmethode bekannt), also das Berechnen sämtlicher Konfigurationen bis eine zulässige gefunden wurde. Allerdings fällt schnell auf, dass das Wählen eines einzigen Startindex für eine Ecke, alle direkt und indirekt damit verbundene anderen Ecken beeinflusst. Wählt man in dem obigen Beispiel in Abbildung 6.9a etwa die untere Ecke aus und legt deren Startindex auf die abgebildete Situation fest, so verbleibt ohnehin nur noch eine zulässige Konfiguration für die darüber liegende Ecke. Andernfalls befände man sich in der Situation aus Abbildung 6.8, welche keine zulässige Lösung darstellt. Setzt man nun den Eckplanindex für diese Ecke ebenfalls fest, zwingt man deren zweiten Nachbarn zwangsläufig in die abgebildete Situation, da der zweite Fall wieder zu einer unzulässigen Lösung führen würde. Da sich an einem Wandstück übereinander mehrere Ecken befinden können, bildet sich damit ein Abhängigkeitsgraph zwischen allen miteinander verbundenen Wandstücken. Dieser kann wie folgt erstellt werden:

1. Wähle eine Startecke und finde alle darüber- und darunterliegenden Ecken, die durch Festlegen eines Index einer Ecke in einer dieser Schichten direkt davon betroffen sind. Dies stellt den ersten Knoten des Abhängigkeitsgraphen dar.
2. Finde alle Nachbarn aller zuvor gefunden Ecken.
3. Wiederhole für jeden Nachbarn das Vorgehen aus dem ersten Schritt. Daraus resultieren die nächsten Knoten, welche mit dem Vorgängerknoten verbunden werden.
4. Stößt man dabei auf einen schon existierenden Knoten, so bricht man diesen Pfad an der Stelle ab. Dieser Fall tritt zum Beispiel auf, wenn in dem Modell ein abgeschlossener Raum existiert.

Nun kann das Modell mithilfe des Graphen einfach ähnlich einer Breitensuche durchlaufen und ausgehend von den Startindex des Eckplans eines Knotens die Indizes von dessen Nachbarn in passender Weise gewählt werden. Dies wiederholt man für alle Nachbarn des derzeitigen Knotens, im nächsten Schritt für deren Nachbarn, solange bis alle Knoten einen Startindex zugewiesen bekommen haben. In ungünstigen Situationen hängt das Finden einer passenden Lösung von dem Startknoten ab. Dies tritt manchmal auf, wenn mehrere Kreise im Abhängigkeitsgraphen existieren. Wird für den gewählten Startknoten mithilfe des Verfahrens keine zulässige Lösung gefunden, so muss ein anderer Startknoten ausprobiert werden. Findet man für keinen Startknoten eine valide Lösung, so existiert für das Modell und den Modulen der gewünschten Mauerwerksverbände keine naheliegende, lückenlose Bausteinkonfiguration.

6.1.6 Öffnungen

Neben Wänden sind Öffnungen ein integraler Bestandteil bei der Planung von Gebäuden. Öffnungen sind Bereiche an welchen das Mauerwerk unterbrochen wird, um Lücken zu schaffen, in die später zum Beispiel Fenster oder Türen eingesetzt werden können. Da eine Öffnung nicht ohne eine Wand existieren kann, liegt es nahe die dazugehörigen Informationen in der Wand zu hinterlegen, die von der Öffnung betroffen ist. Eine Öffnung kann ebenfalls als quaderförmiger Körper angesehen werden, der einen Bereich innerhalb eines Wandstücks definiert, an dem später keine Bausteine gelegt werden dürfen. In dieser Arbeit besitzt dieser Quader die selbe Orientierung wie das dazugehörige Wandstück und schneidet zunächst immer durch die gesamte Breite der Wand. Daher sind nur Position, Länge und Höhe des Quaders von Bedeutung für das Wall Detailing. Da es allerdings üblich ist einen sogenannten Sturz über der eigentlichen Öffnung eines Fenster oder einer Tür anzubringen, muss auch für dieses Objekt Platz im Wandstück geschaffen werden. Der Sturz ist etwas länger als die Länge der Öffnung und liegt an beiden Seiten auf dem Mauerwerk auf, um den Bausteinen darüber eine stabile Grundlage zu bieten. So bleibt die Stabilität der Wand trotz Öffnungen erhalten.

6.1.7 Wandenden

Stehen eine oder beide Kanten eines Wandstücks allein, ohne dort über eine Beziehung mit einem anderen verbunden zu sein, muss der Mauerwerksverband zu einem geraden Abschluss gebracht werden. Häufig ist dies nur möglich, indem man für manche Bausteine die Länge des verwendeten Moduls anpasst.

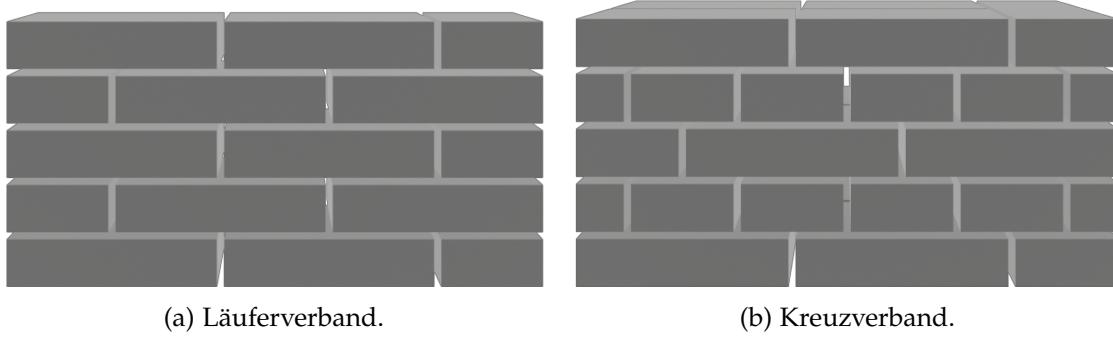


Abbildung 6.10: Wandenden verschiedener Mauerwerksverbände.

In Abbildung 6.10a werden für den Läuferverband Bausteine mit halber Modullänge und für den Kreuzverband in der Abbildung daneben zwei angepasste Versionen des Moduls benötigt: Erneut ein Bausteinformat mit halber Modullänge um die Läuferverbandschichten des Kreuzverbands abzuschließen und ein Bausteinformat, dessen Länge sich aus der halben Modulbreite bildet und für den Abschluss der Kopfverbandschichten notwendig ist. Aber nicht nur an Wandenden muss ein Mauerwerksverband mit einer geraden Kante abgeschlossen werden. Auch an den zuvor angesprochenen Öffnungen ist dies für die Schichten erforderlich, die durch die Öffnung geteilt werden. Für den Fall, dass das vollständige Detailing eines Modells das Anpassen des gewünschten Moduls

notwendig macht, muss diese Information neben den errechneten Transformationen der Bausteine ebenfalls in das Ergebnis integriert werden. Somit erhält jeder Baustein aus diesem Grund zusätzlich eine Beschreibung über dessen Dimensionen.

6.2 Wall Detailing

Als „Wall Detailing“ (sprich das „Detailieren von Wänden“) wird in dieser Arbeit der Vorgang bezeichnet, ein als geometrischer Körper definiertes Wandstück in konkretes Mauerwerk zu überführen. Dieser Vorgang wendet die zuvor behandelten Konzepte an, um Lösungen für beliebig komplexe Gebäudemodelle zu errechnen. Als Lösung sollen nur diejenigen Bausteinmengen gelten, die sämtliche von Wandstücken abgedeckten Bereiche lückenlos mit Bausteinen füllen, während gleichzeitig die gewünschten Mauerwerksverbände eingehalten werden. Das nachfolgende schrittweise Verfahren weist aufgrund der ähnlichen Zielsetzung zwangsläufig Ähnlichkeiten zu dem von Usmanov et al. auf, das bereits in Kapitel 4.2.1 zusammengefasst wurde.

1. Extrahieren relevanter Informationen wie Geometrie und Typ-Annotationen aus dem vorgegebenen Gebäudemodell. Dabei interessieren in erster Linie Wände und deren Öffnungen sowie gewünschte Module und Mauerwerksverbände (siehe Abschnitt 6.1.3).
2. Anwenden des vorgegebenen Moduls für jedes gefundene Wandstück. Das führt zu einer schichtweisen Repräsentation jedes Wandstücks. Dies erleichtert es später Operationen an und zwischen mehreren Wandstücken durchzuführen.
3. Finden und Kombinieren von Wandstückverbänden nach Abschnitt 6.1.4.
4. Finden und Lösen von anderen Beziehungen wie Ecken T- und X-Kreuzungen nach Abschnitt 6.1.4 (Finden) und Abschnitt 6.1.5 (Lösen).
5. Schichtweises Anwenden der Öffnungen jedes Wandstücks. Dabei werden alle betroffenen Schichten in passender Weise geteilt und deren Länge reduziert.
6. Schichtweises Anwenden der den Wandstücken zugewiesenen Mauerwerksverbände. Dazu gehören sowohl die besonderen Bereiche aus Punkt 4, als auch gerade Wandabschnitte und allein stehende Wandenden.
7. Finden von (direkten) Nachbarschaftsbeziehungen zwischen Bausteinen.
8. Konvertieren in das für die nachfolgende Bauplandeduktion notwendige Format.

Das Ergebnis dieser Schritte ist eine Menge aus Bausteinen, nachfolgend auch als *Konstruktionsplan* bezeichnet. Durch die Berechnungen in Schritt 7 werden bereits die grundlegendsten Beziehungen zwischen den Bausteinen hergestellt. Eine Aussage darüber in welcher Reihenfolge die Bausteine gesetzt werden müssen, um das geplante Gebäude unter Berücksichtigung etwaiger physikalischer oder anderer Einschränkungen zu errichten, wird allerdings noch nicht getroffen. Dazu müssen Schritt für Schritt Teilmengen

aus der Gesamtmenge an Bausteinen extrahiert werden, die nur diejenigen Bausteine beinhalten, die in einer bestimmten Situation gelegt werden können. Welche das sind, soll anhand vorgebener Regeln ausgewertet werden können. Diese Regeln können dafür verwendet werden bestimmte Voraussetzungen an einen Baustein zu setzen, um zum Beispiel das Ablegen eines Bausteins zu verhindern, unter dem sich weder fester Boden, noch ausreichend andere Bausteine befinden. Denn setzt man einen Baustein mitten in der Luft ab, so fällt er in der echten Welt aufgrund der auf ihn einwirkenden Gravitation zu Boden. Außerdem weisen unterschiedliche Bausteinarten und Umgebungen, in die man das Gebäude mithilfe des zuvor errechneten Konstruktionsplans errichten möchte, womöglich unterschiedliche Einschränkungen auf. So ist es notwendig je nach Situation voneinander abweichende Regelsets auf dem Konstruktionsplan anzuwenden zu können. Die Möglichkeit der nachträglichen Definition dieser Regeln verhindert zusätzlich die Notwendigkeit den Konstruktionsplan für sich unterscheidene Situationen neu berechnen zu müssen.

6.3 Regelbasierte Bauplandeduktion

Ontologien bieten verschiedene Werkzeuge an, mithilfe derer auf streng definierten Objekttypen und deren Relationen zueinander konkrete Instanzen gruppiert und gefiltert werden können.

1. Klassendefinition aufstellen
2. Relationen definieren
3. Bild Einfügen
4. Regel einführen (und versuchen diese nachträglich mit owlready reinzuhauen)
5. Reasoner ausprobieren und schrittweise bausteine als gesetzt markieren um neue Menge an möglichen steinen rauszubekommen

7 Realisierung

Mit dem in Kapitel 5.1 vorgestellten *IFC Standard* existiert ein umfangreiches Framework zur Beschreibung sämtlicher für die Baubranche relevanter Daten. Im Zusammenspiel mit dem darauf aufbauenden *Building Information Modelling* wird ein einheitliches Verwaltungskonzept für das Planen, Bauen und Bewirtschaften von Infrastruktur vorgeschlagen (siehe Kapitel 5.3). Darum stellen im IFC Format vorliegende Gebäudepläne den Ausgangspunkt dieser Arbeit dar.

7.1 Modellierung

Dank den in Kapitel 5.2.3 vorgestellten Open Source Projekten zu den Technologien *IFC* und *BIM* sowie deren Anbindungen an die ebenfalls öffentliche 3D-Grafiksoftware Blender (siehe Kapitel 5.2) ist es möglich auf einen komplett kostenlosen Technologiestack zur Modellierung von Gebäuden im IFC Format zurückzugreifen. Die Option Blender mithilfe sogenannter *Addons* an spezielle projektabhängige Anforderungen anzupassen, macht diese Modellierungsumgebung noch zweckdienlicher. Ein solches Addon ist zum Beispiel *blenderbim*. Dieses ermöglicht es ein IFC Projekt direkt in Blender entweder neu zu beginnen oder ein vorhandenes zu bearbeiten. Für diese Arbeit relevant ist zunächst das Erstellen und Annotieren von Wänden beziehungsweise Wandtypen, sowie das Anbringen von Öffnungen. Beide Konzepte sind Teil des IFC Standards und wurden bereits ausführlich in Kapitel 5.1 vorgestellt. Blenderbim ermöglicht es in wenigen Schritten einen neuen *IFCWallType* zu definieren und daraus *IFCWall* Objekte zu erstellen. Diese können mit den von Blender nativ angebotenen Werkzeugen angepasst werden. Zusätzlich gibt es weitere hilfreiche von Blenderbim eingeführte Modellierungsmöglichkeiten. Dank der Möglichkeit eigene *IFCProperties* (siehe Kapitel 5.1.2) zu definieren, lassen sich die in Kapitel 6.1.1 konzeptionierten Raster- und Modulinformationen zu den neu definierten Wandtypen hinzufügen. Bei Generierung eines *IFCWall* Objekts werden diese Properties ebenfalls an das neue konkrete Wandstück geknüpft. Damit können diese Informationen sowohl an das nachfolgende Wall Detailing übermittelt, als auch zur intuitiveren Modellierung des 3D Modells herangezogen werden, indem man Nutzern durch ein für diese Arbeit entwickeltes Addon das für jeden Wandtyp zugeordnete Raster bei sämtlichen Transformationsvorgängen aufzwingt. Dadurch werden kleine Modellierungsfehler von vornherein vermieden.

7.2 Wall Detailing

In Abschnitt 6.2 wurde das Wall Detailing als der Vorgang, ein als geometrischer Körper definiertes Wandstück in ein konkretes Mauerwerk zu überführen, bezeichnet. Innerhalb

des IFC Standards werden einige mathematische/geometrische Repräsentationen der sogenannten *IFCWall* unterstützt, um neben einfachen Boxen auch komplexere Formen abbilden zu können. Beispielsweise ist es möglich in das Modell eines Hauses zunehmend dünner werdende Wandstücke, kurvige Wandstücke oder Wandstücke, welche nur durch ein arbiträres Vieleck beschrieben werden können zu integrieren. Standardmäßig haben *IFCWalls* aber einen einfachen Quader als Grundform, was für die Fallstudien dieser Arbeit ausreichend ist. Die Unterkapitel dieses Abschnitts folgen den in Kapitel 6.2 genannten Schritten, um von einer Menge solcher Wände hin zu einer Menge an Bausteinen zu gelangen, die den gewünschten Mauerwerksverband darauf anwenden.

7.2.1 Konvertieren des IFC zu BREP

Den ersten Schritt stellt das Extrahieren aller notwendigen Daten aus dem vorliegenden IFC Modell dar. Für die Fallstudien dieser Arbeit sind sowohl alle Objekte des Typs *IFCWall* als auch die, etwa durch Fenster oder Türen entstehenden, daran angeknüpften Objekte vom Typ *IfcOpeningElement* (siehe 5.1.4) von Interesse. Zusätzlich werden aus den, in den *IfcPropertySets* der Wandstücke hinterlegten Daten, Informationen über das zu verwendende Modul und Raster ausgelesen. Mithilfe der Werkzeuge der in Kapitel 5 vorgestellten Python Bibliothek *ifcopenshell* (siehe 5.2.3) ist dies in wenigen Schritten möglich.

¹ TODO Sätze zum Code, Code TODO kleines Klassendiagramm von
² Wall bzw WallLayerGroup und Opening und BrickInformation

Listing 7.1: Extraktion relevanter Informationen aus einem IFC-File.

Da sich diese Arbeit zunächst ausschließlich mit quaderförmigen Wandstücken beschäftigt, müssen alle zuvor aus dem IFC Modell extrahierten Wandstücke auf diese Eigenschaft geprüft werden. Somit ist gewährleistet, dass lediglich passende Wandstücke an die nachfolgenden Schritte weitergegeben werden.

¹ TODO Satz zum Code, Code `iscubic`

Listing 7.2: Überprüfen der extrahierten Wandstücke auf Quaderförmigkeit.

7.2.2 Anwenden des Moduls

Mit dem zu jedem Wandstück festgelegten Modul werden nun alle Wandstücke in Schichten aufgeteilt. Deren Höhe entspricht im Normalfall der Höhe des jeweiligen Moduls. Lediglich die oberste Schicht kann durch falsch modellierte Wandstücke eine niedrigere Schichthöhe aufweisen. Dies ist der Fall, wenn die Gesamthöhe des Wandstücks nicht exakt einem Vielfachen der Höhe des Moduls entspricht und ein nicht aufzuteilender Rest existiert. Das Aufteilen in Schichten erleichtert es im Anschluss Berechnungen an Wandstücken durchzuführen und Beziehungen zwischen ihnen zu finden.

7.2.3 Kombinieren passender Wandstücke

Eine solche Beziehung stellen Wände dar, die, wie bereits in Kapitel 6.1.4 definiert, durch mehrere einzelne Objekte modelliert wurden, eigentlich aber eine Einheit bilden.

Daher werden in diesem Schritt alle Wandstücke miteinander verglichen und eventuell kombiniert, sodass jeweils ein gefundenes Paar durch ein einzelnes Wandstück repräsentiert wird. Um zwei Wandstücke zu sinnvoll kombinieren zu können, müssen die Eigenschaften aus Kapitel 6.1.4 gelten. Allerdings kann Punkt 5, welcher Berührung oder Überlappung voraussetzt mittlerweile wie folgt verschärft werden:

5. Mindestens eine Schicht des einen Wandstücks berührt, überlappt oder befindet sich exakt eine Modulöhe ober- oder unterhalb einer Schicht des anderen Wandstücks.

In Abbildung 6.3 treten verschiedene Konstellationen von Wandstücken auf, die miteinander kombiniert werden müssen. Das aus Wandstück 1 und 2 gebildete Paar erfüllt alle oben genannten Eigenschaften und weist eine Teil-Überlappung auf. Somit müssen beide Wandstücke miteinander kombiniert werden. Den obersten Bereich füllen sowohl Wandstück 5 als auch Wandstück 6, sodass dort eine komplette Überlappung vorliegt. Auch diese beiden Wandstücke werden kombiniert, wobei dabei im Prinzip einfach eines verworfen wird. Zusätzlich muss dieser Bereich, wie auch Wandstück 7, mit Wandstück 2 kombiniert werden, da die beiden Paare sich an Ober- und Unterkante berühren. Einen weiteren Fall stellen seitliche, nicht überlappende Berührungen dar, wie sie zwischen Wandstück 2 und 3 zu sehen ist. Auch für dieses Paar sind alle Voraussetzungen zur Fusion erfüllt. Lediglich Wandstück 4 muss nicht mit dem Rest vereint werden, da dafür kein Paar existiert, das die obrigen Eigenschaften erfüllt. Anhand von Abbildung 7.1 kann man erkennen, dass Wandstück 4 im weiteren Verlauf des Detailings tatsächlich unabhängig betrachtet werden kann.

Während dem Kombinieren von zwei Wänden, wird ein Wandstück schichtweise in das Andere überführt. Dabei werden alle Schichten paarweise miteinander verglichen, um diejenigen Paare zu finden, die die Eigenschaften aus Punkt 5 erfüllen. Ein solches Paar wird dann wie folgt miteinander verschmolzen:

```

1 def combine(layer1, layer2):
2     # get the local positions of both edges of layer1 as 3D array
3     left_edge1 = layer1.get_left_edge(relative=True)
4     right_edge1 = layer1.get_right_edge(relative=True)
5
6     # get the position of both edges of layer 2 relative to the coordinate
7     # system of layer1 as 3D array
8     left_edge2 = layer2.get_left_edge(relative=False)
9     left_edge2 = layer1.relative_position_of(left_edge2)
10
11    right_edge2 = layer2.get_right_edge(relative=False)
12    right_edge2 = layer1.relative_position_of(right_edge2)
13
14    # get the total length both layers cover by subtracting max - min of the x
15    # coordinates
16    max_x = max(left_edge1[0], right_edge1[0], left_edge2[0], right_edge2[0])
17    min_x = min(left_edge1[0], right_edge1[0], left_edge2[0], right_edge2[0])
18    total_length = max_x - min_x
19
20    # calculate the center of the resulting layer
21    left = min([left_edge1, right_edge1, left_edge2, right_edge2], key=lambda
22      x: x[0])

```

```
20     local_center = left.copy()
21     local_center[0] += total_length / 2.0
22
23     # convert to global coordinates and return new layer
24     center = layer1.global_position_of(local_center)
25     return Layer(center, total_length)
```

Listing 7.3: Pseudocode zur Vereinigung zweier sich berührender oder überlappender Schichten von zwei nach den Voraussetzungen aus Abschnitt 6.1.4 kombinierbaren Wandstücken.

TODO erklärung Code

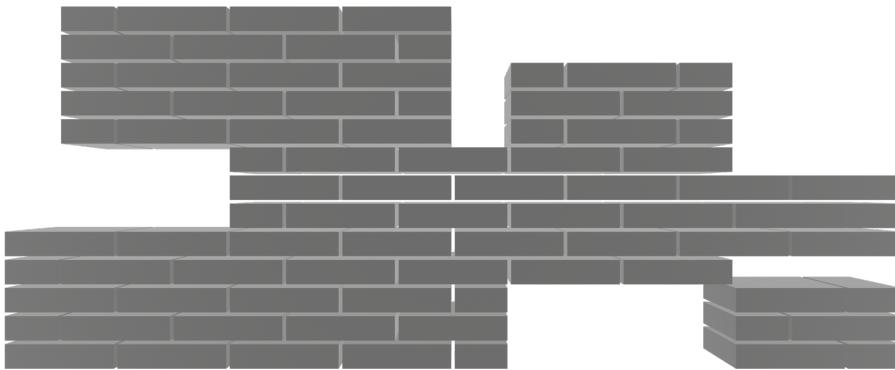


Abbildung 7.1: Ergebnis mit berücksichtigtem x_offset .

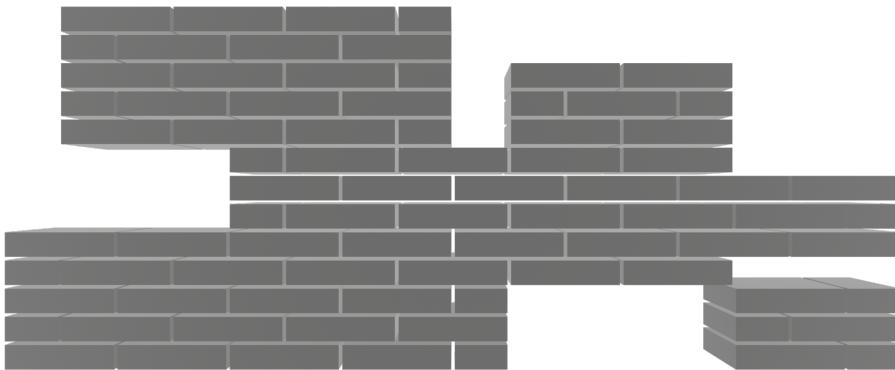


Abbildung 7.2: Ergebnis mit ignoriertem x_offset .

Steht ein Wandstück in X-Richtung versetzt auf einem anderem, so ist es notwendig diesen Versatz während dem nachfolgenden Detailing zu berücksichtigen. Ignoriert

man dies, kann das zu den in Abbildung 7.2 gezeigten Fehlern (zum Beispiel zwischen Wandstück 2 und 7) innerhalb des Mauerwerksverbands und damit ebenfalls zu Verletzungen des vorgeschriebenen Überbindemaßes führen (siehe Abschnitt 5.5.2). Dieser, nachfolgend als x_offset bezeichnete Versatz ist definiert durch die Differenz zwischen der kleinsten lokalen X-Koordinate aller Schichten eines Wandstückes und der lokalen X-Koordinate der zu betrachtenden Schicht. Der daraus resultierende Wert wird später dazu verwendet den anzuwendenden Mauerwerksverband erst an der passenden Stelle zu beginnen. Dadurch erzielt man einen einheitlichen Verband über das gesamte Wandstück und verhindert den in Abbildung 7.2 gezeigten Fehlerfall. Eine weitere Eigenschaft, die aus dem Kombinieren mehrerer Wandstücke entstehen kann, ist das vorhandensein unterbrochender Schichten oder, anders ausgedrückt, mehrerer Schichten auf einer Höhe innerhalb des resultierenden Wandstückes. Dies ist ebenfalls in Abbildung ?? zu sehen. Zwischen drei Schichten des Bereichs der Wandstücke 5 und 6 und des Wandstücks 7 ist eine Lücke. Durch das Einbeziehen des x_offset können derartige Situationen jedoch ebenfalls gelöst werden, da für jedes Teilstück einer Schicht ein eigener x_offset berechnet wird.

Nach Vereinigung aller passenden Paare reduziert sich die Zahl der ursprünglichen Wandstücke aus dem Modell in Abbildung 6.3 von 7 auf 2.

AB HIER ÜBERARBEITEN, DA TEILE INS KONZEPT GEWANDERT SIND. @CONSTITUTION hier aufhören zu lesen :)

7.2.4 Finden und Lösen von Beziehungen

Nun wird die aus dem vorherigen Schritt entstandene neue Menge an Wandstücken auf weitere Beziehungen untersucht. Für das Wall Detailing relevante Beziehungen stellen Ecken, T-Kreuzungen und X-Kreuzungen dar (siehe 5.5.2). Der Grund dafür ist die Komplexität in diesen Bereichen die vorgeschriebenen Normen (wie etwa dem in Abschnitt 5.5.2 genannten Überbindemaß) einzuhalten. Deswegen existieren für jeden Mauerwerksverband eigene spezielle Eck- und Kreuzungslegepläne, die an diesen Stellen eingefügt werden müssen. Es werden wie oben nur die Wandstücke miteinander verglichen, die mit dem selben Wandtyp annotiert wurden und das gleiche Modul verwenden. Da die drei Beziehungen sich stark ähneln, besitzen sie einige geteilte Eigenschaften:

1. Die lokalen Z-Achsen beider Wandstücke sind parallel. Das Vorgehen zur Überprüfung ist dasselbe wie in Abschnitt 7.2.3.
 2. Sie stehen auf der selben Höhe oder versetzt um ein Vielfaches der gemeinsamen Modulhöhe.
 3. Mindestens eines der beiden Wandstücke endet auf einem anderen, sodass mindestens eine Schicht das andere Wandstück direkt berührt.
-

Sind diese Eigenschaften erfüllt, werden für jede durch die einzelnen Schichten der Wandstücke vorgegebene Höhe Schnittpunkte zwischen den beiden Wandstücken errechnet. Im Falle einer einfachen Ecke oder einer T-Kreuzung, existiert nur ein einzelnes Paar an Wandstücken, deren Schichten sich in jedem Höhenschritt des Moduls im Mittelpunkt der Ecke schneiden. Der Unterschied ist lediglich die Stelle des Schnittpunkts relativ zu den beiden Wandstücken. Liegt der errechnete Schnittpunkt bei beiden Wandstücken näher als Wandbreite/2 an einer der beiden Außenkanten handelt es sich um eine Ecke. Falls der Schnittpunkt bei einem der beiden Wandstücke allerdings mindestens Wandbreite/2 innerhalb des Wandstücks liegt, so handelt es sich um eine T-Kreuzung. Werden für zwei unterschiedliche Paare die selben Schnittpunkte errechnet, so kann es sich entweder um eine mit drei Wandstücken modellierte T-Kreuzung oder X-Kreuzung handeln. Teilen sich zwei T-Kreuzungen die selben Schnittpunkte, so sind alle Voraussetzungen für eine X-Kreuzung gegeben. Liegt der Schnittpunkt des Wandstück-Paars mindestens Wandbreite/2 innerhalb beider Wandstücke, so handelt es sich um eine X-Kreuzung - modelliert aus zwei sich schneidenden Wandstücken. TODO Bilder

Lösen der Beziehungen

Je nach gewählten Verband müssen z.B. Ecken (deren Baupläne im vornherein definiert wurden) so angeordnet werden, dass die dazwischenliegenden Wandstücke lückenlos eingefüllt werden können. Dafür muss ein sogenannter "plan_offset" für jede Wand und jede Ecke gefunden werden. Dieser gibt an, an welchem Index der Bauplandefinition das Wandstück (von unten) beginnen muss, um insgesamt einen einheitlichen Wandkörper ohne Lücken zu bilden. Voraussetzung dafür ist natürlich ein passender Eckplan zu dem gewählten Verband.

Schwierigkeiten: Eckpläne insgesamt, Ecken ragen in die sie bildenden Wände hinein. Diese müssen dementsprechend verkleinert werden.

7.2.5 Anwenden der Öffnungen

Blablabla

7.2.6 Anwenden der Mauerwerksverbände

Ablaufen aller Ecken und Wände und einsetzen der Ziegel gemäß den gefundenen Lösungen.

7.2.7 Nachbarschaftsbeziehungen zwischen Bausteinen berechnen

7.2.8 Export

Abhängigkeitsgraph und Ontologie für Regelwerk

TODOs

1. reduzieren der layer length abhängig vom corner plan

2. aufteilen der layer bei öffnungen
3. einpflegen der Dimensionen der Bausteine und Rasterinformationen

8 Proof of Concept

9 Fazit und Ausblick

Literatur

- [1] Thomas R. Gruber. „A translation approach to portable ontology specifications“. In: *Knowledge Acquisition* 5 (2 Juni 1993), S. 199–220. issn: 10428143. doi: 10.1006/knac.1993.1008.
- [2] Rudi Studer, V.Richard Benjamins und Dieter Fensel. „Knowledge engineering: Principles and methods“. In: *Data and Knowledge Engineering* 25 (1-2 März 1998), S. 161–197. issn: 0169023X. doi: 10.1016/S0169-023X(97)00056-6.
- [3] Tim Berners-Lee, James Hendler und Ora Lassila. „The Semantic Web“. In: *Scientific American* 284 (5 Mai 2001), S. 34–43. issn: 0036-8733. doi: 10.1038/scientificamerican0501-34. URL: <https://www.scientificamerican.com/article/the-semantic-web>.
- [4] Evren Sirin u. a. „Pellet: A practical OWL-DL reasoner“. In: *Journal of Web Semantics* 5 (2 Juni 2007), S. 51–53. issn: 15708268. doi: 10.1016/j.websem.2007.03.004.
- [5] Nicola Guarino, Daniel Oberle und Steffen Staab. „Handbook on Ontologies“. In: Springer Berlin Heidelberg, 2009, S. 1–17. doi: 10.1007/978-3-540-92673-3.
- [6] *OWL 2 Web Ontology Language Document Overview (Second Edition)*. W3C Recommendation. <https://www.w3.org/TR/2012/REC-owl2-overview-20121211/>. W3C, Dez. 2012.
- [7] Barry Smith. „Ontology“. In: Leiden, Niederlande: Brill, 2012, S. 47–68. isbn: 9789401207799. doi: https://doi.org/10.1163/9789401207799_005.
- [8] Steven Harris und Andy Seaborne. *SPARQL 1.1 Query Language*. W3C Recommendation. <https://www.w3.org/TR/2013/REC-sparql11-query-20130321/>. W3C, März 2013.
- [9] Lieyun Ding, Ying Zhou und Burcu Akinci. „Building Information Modeling (BIM) application framework: The process of expanding from 3D to computable nD“. In: *Automation in Construction* 46 (2014). Wie mit BIM + Zeit (= 4D Bim) Kosten, Emmisionen und Sicherheitsaspekte berechnet werden können. Schlägt angeblich ein Verfahren vor von 3D zu nD zu gelangen, aber ist eigentlich unwichtig für mich, S. 82–93. issn: 09265805. doi: 10.1016/j.autcon.2014.04.009.
- [10] Birte Glimm u. a. „HermiT: An OWL 2 Reasoner“. In: *Journal of Automated Reasoning* 53 (3 Okt. 2014), S. 245–269. issn: 0168-7433. doi: 10.1007/s10817-014-9305-1.
- [11] Yevgeny Kazakov, Markus Krötzsch und František Simančík. „The Incredible ELK“. In: *Journal of Automated Reasoning* 53 (1 Juni 2014), S. 1–61. issn: 0168-7433. doi: 10.1007/s10817-013-9296-3.
- [12] Markus Lanthaler, Richard Cyganiak und David Wood. *RDF 1.1 Concepts and Abstract Syntax*. W3C Recommendation. <https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/>. W3C, Feb. 2014.

- [13] Mark A. Musen. „The protégé project“. In: *AI Matters* 1 (4 Juni 2015), S. 4–12. ISSN: 2372-3483. doi: 10.1145/2757001.2757003. URL: <https://dl.acm.org/doi/10.1145/2757001.2757003>.
- [14] Jean Baptiste Lamy. „Owlready: Ontology-oriented programming in Python with automatic classification and high level constructs for biomedical ontologies“. In: *Artificial Intelligence in Medicine* 80 (Juli 2017), S. 11–28. ISSN: 18732860. doi: 10.1016/j.artmed.2017.07.002.
- [15] José Luis Moro. „MASSORDNUNG“. In: *Baukonstruktion – vom Prinzip zum Detail: Band 1 Grundlagen*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2021, S. 65–97. ISBN: 978-3-662-64155-2. doi: 10.1007/978-3-662-64155-2_4. URL: https://doi.org/10.1007/978-3-662-64155-2_4.
- [16] Vjačeslav Usmanov, Jan Illetško und Rostislav Šulc. „Digital Plan of Brickwork Layout for Robotic Bricklaying Technology“. In: *Sustainability* 13.7 (Apr. 2021), S. 3905. doi: 10.3390/su13073905. URL: <https://doi.org/10.3390/su13073905>.
- [17] Chengran Xu u.a. „Optimal brick layout of masonry walls based on intelligent evolutionary algorithm and building information modeling“. In: *Automation in Construction* 129 (Sep. 2021), S. 103824. ISSN: 0926-5805. doi: 10.1016/J.AUTCON.2021.103824.
- [18] Kathrin Dörfler u.a. „Additive Manufacturing using mobile robots: Opportunities and challenges for building construction“. In: *Cement and Concrete Research* 158 (Aug. 2022), S. 106772. ISSN: 0008-8846. doi: 10.1016/J.CEMCONRES.2022.106772.
- [19] *Hermit Reasoner: Home*. <http://www.hermit-reasoner.com>. (Accessed on 12/03/2023). Dez. 2023.
- [20] *Ontologie - Fakultät für Philosophie, Wissenschaftstheorie und Religionswissenschaft - LMU München*. <https://www.philosophie.uni-muenchen.de/fakultaet/schwerpunkte/ontologie/index.html>. (Accessed on 12/02/2023). Dez. 2023.
- [21] *Python.org*. <https://www.python.org>. (Accessed on 11/30/2023). Nov. 2023.
- [22] *05_maurerfibel_kap-4.pdf*. https://www.kalksandstein.de/media/08_downloadcenter/05_maurerfibel_kap-4.pdf. (Accessed on 06/29/2023).
- [23] *API Overview - Blender Python API*. <https://ifc43-docs.standards.buildingsmart.org/IFC/RELEASE/IFC4x3/HTML/lexical/IfcOpeningElement.htm>. (Accessed on 11/30/2023).
- [24] *Bemessung von Ziegelmauerwerk nach DIN EN 1996-3/NA:2019-12*. https://www.wienerberger.de/content/dam/wienerberger/germany/marketing/documents-magazines/instructions-guidelines/wall/DE_MKT_DOC_POR_Bemessung_Ziegelmauerwerk.pdf. (Accessed on 06/29/2023).
- [25] *BIM for Health and Safety in Construction | Autodesk University*. <https://www.autodesk.com/autodesk-university/article/BIM-Health-and-Safety-Construction-2017>. (Accessed on 04/18/2023).
- [26] *blender.org - Home of the Blender project - Free and Open 3D Creation Software*. <https://www.blender.org/>. (Accessed on 02/16/2023).

- [27] *BlenderBIM Add-on - beautiful, detailed, and data-rich OpenBIM*. <https://blenderbim.org/>. (Accessed on 02/16/2023).
- [28] *Building Information Modeling – Wikipedia*. https://de.wikipedia.org/wiki/Building_Information_Modeling. (Accessed on 02/16/2023).
- [29] Charles M. Eastman u. a. „BIM handbook : a guide to building information modeling for owners, managers, designers, engineers and contractors“. In: Kap. 3.4.4. ISBN: 9781119287544.
- [30] *Fachkräftemangel und Rohstoffpreise – Die Deutsche Bauindustrie*. <https://www.bauindustrie.de/zahlen-fakten/bauwirtschaft-im-zahlenbild/fachkraeftemangel-und-rohstoffpreise>. (Accessed on 03/31/2023).
- [31] Kalksandstein-Dienstleistung GmbH. *Mauern von Stößen und Kreuzungen*. <https://www.ks-maurerfibel.de/maurerfibel/4-mauerwerksverbaende/4-7-mauern-von-stoessen-und-kreuzungen/>. (Accessed on 12/01/2023).
- [32] *IFC Formats - buildingSMART Technical*. <https://technical.buildingsmart.org/standards/ifc/ifc-formats/>. (Accessed on 02/16/2023).
- [33] IfcOpenShell. *IfcOpenShell - The open source IFC toolkit and geometry engine*. <https://ifcopenshell.org/>. (Accessed on 05/05/2023).
- [34] *Industrial automation systems and integration — Product data representation and exchange — Part 21: Implementation methods: Clear text encoding of the exchange structure*. <https://www.iso.org/standard/63141.html>. (Accessed on 05/05/2023).
- [35] *Industry Foundation Classes – Wikipedia*. https://de.wikipedia.org/wiki/Industry_Foundation_Classes. (Accessed on 02/16/2023).
- [36] *Industry Foundation Classes (IFC) - buildingSMART Technical*. <https://technical.buildingsmart.org/standards/ifc>. (Accessed on 02/16/2023).
- [37] buildingSMART International. *IFC 4.3.1.0 OpeningElement*. <https://ifc43-docs.standards.buildingsmart.org/IFC/RELEASE/IFC4x3/HTML/lexical/IfcOpeningElement.htm>. (Accessed on 11/30/2023).
- [38] buildingSMART International. *IFC 4.3.1.0 Pset_WallCommon*. https://ifc43-docs.standards.buildingsmart.org/IFC/RELEASE/IFC4x3/HTML/lexical/Pset_WallCommon.htm. (Accessed on 11/30/2023).
- [39] buildingSMART International. *IFC 4.3.1.0 Spezification*. <https://ifc43-docs.standards.buildingsmart.org/>. (Accessed on 05/05/2023).
- [40] buildingSMART International. *IFC 4.3.1.0 Spezification Chapter 1 Scope*. <https://ifc43-docs.standards.buildingsmart.org/IFC/RELEASE/IFC4x3/HTML/content/scope.htm>. (Accessed on 05/05/2023).
- [41] ISO - ISO 16739-1:2018 - *Industry Foundation Classes (IFC) for data sharing in the construction and facility management industries — Part 1: Data schema*. <https://www.iso.org/standard/70303.html.norm>. (Accessed on 02/16/2023).
- [42] ISO - ISO 2848:1984-04 - *Building construction; Modular coordination; Principles and rules*. <https://www.iso.org/standard/7846.html.norm>. (Accessed on 12/01/2023).

Literatur

- [43] *LEGO Brick Dimensions and Measurements - Christoph Bartneck, Ph.D.* <https://www.bartneck.de/2019/04/21/lego-brick-dimensions-and-measurements/>. (Accessed on 07/07/2023).
- [44] *Mauerwerksverband - Baulexikon.* <https://baulexikon.beuth.de/MAUERWERKSVERBAND.HTM>. (Accessed on 06/29/2023).
- [45] *Microsoft Word - ConstructionLifecycleManagementwithBIM_080109.* https://www.researchgate.net/profile/Yusuf-Arayici/publication/243972464_Building_information_modelling_BIM_for_Construction_Lifecycle_Management/links/54f4456c0cf2f9e34f094781/Building-information-modelling-BIM-for-Construction-Lifecycle-Management.pdf. (Accessed on 02/16/2023).
- [46] Mohd Nasrun u. a. „Impact of Fragmentation Issue in Construction Industry: An Overview; Impact of Fragmentation Issue in Construction Industry: An Overview“. In: (). doi: 10.1051/C. URL: <http://dx.doi.org/10.1051/matecconf/20141501009>.
- [47] *Revit-Software | BIM-Software | Autodesk.* <https://www.autodesk.de/products/revit/>. (Accessed on 02/16/2023).
- [48] *Steinformat – Wikipedia.* <https://de.m.wikipedia.org/wiki/Steinformat>. (Accessed on 07/25/2023).
- [49] *The open source BIM collective.* <https://github.com/opensourceBIM>. (Accessed on 02/16/2023).
- [50] *Top 10 Benefits of BIM in Construction.* <https://bim360resources.autodesk.com/connect-construct/top-10-benefits-of-bim-in-construction>. (Accessed on 04/18/2023).
- [51] *DIN 4172:2015-09 Maßordnung im Hochbau.* Norm. 2015.
- [52] *Eurocode 6: Bemessung und Konstruktion von Mauerwerksbauten - Teil 1-1: Allgemeine Regeln für bewehrtes und unbewehrtes Mauerwerk.* Norm. 2012.