Julius-Maximilians-Universität Würzburg
Institut für Informatik
Lehrstuhl für Informatik I
Effiziente Algorithmen und wissensbasierte Systeme

Masterarbeit

Symmetrieerkennung in Gebäudeumrissen

Hagen Schwaß

Eingereicht am 31. März 2013

Betreuer:
Prof. Dr. Alexander Wolff
Dr. Jan-Henrik Haunert

Inhaltsverzeichnis

1 Einleitung

Symmetrie ist ein fundamentaler Bestandteil in der Gestaltung von Objekten. In der Architektur wird Symmetrie in verschiedenen Größenordnungen verwendet. Diese reichen von der Form von Fassaden über die Form von Umrissen einzelner Gebäude bis hin zur Stadt- bzw. Landschaftsplanung. Gründe für die symmetrische Gestaltung sind unter anderem Funktionalität, Ästhetik sowie Kosten. Ein Beispiel ist das in Abbildung 1.1 dargestellte Pentagon, welches der Hauptsitz des US-amerikanischen Verteidigungsministeriums ist. Das Pentagon wurde mit regelmäßigen 5-Ecken entworfen, welche rotationssymmetrisch sind.

1.1 Motivation

Haunert [Hau12] beschreibt ein Verfahren zur Symmetrieerkennung für die Generalisierung in der Kartografie und zur Analyse von Stadtgebieten. Dort wird auch ein empirischer Test vorgestellt, in welchem 30 Probanden aufgefordert wurden, eine Gruppe von Gebäuden so zu skalieren, dass diese hinterher nur noch halb so groß sind. Dabei wurde es den Probanden erlaubt, die Gebäude vereinfacht darzustellen. Die Gebäude sollten nach der Skalierung aber möglichst originalgetreu aussehen. Unter allen Gebäuden, die sinnvoll vereinfacht wurden, also einfacher als das Original waren und aus mehr als vier Ecken bestanden, waren bei 81% die dominierenden Symmetrien erhalten geblieben. Dies zeigt, dass Symmetrie ein wichtiges Merkmal der Wiedererkennbarkeit ist.

Literaturzusammenfassungen zu psychologischen Studien bezüglich der menschlichen Wahrnehmung von Symmetrie geben Wagemans [Wag95] und Treder [Tre10]. Hieraus geht hervor, dass Menschen Symmetrien gut erkennen können.

1.1.1 Kartografische Generalisierung

In der Kartografie ist die Generalisierung die Vereinfachung einer Karte, so dass deren Lesbarkeit und Verständlichkeit erhalten bleibt. Dies ermöglicht die Hervorhebung wesentlicher Informationen. Unwichtige Informationen werden weggelassen. So kann die generalisierte Karte auch in kleinerem Maßstab noch verständlich dargestellt werden. Außerdem werden dann weniger Details benötigt. Liegt die Karte in elektronischer Form vor, so wird hierbei auch das benötigte Datenvolumen reduziert und damit die Lade- bzw. Verarbeitungszeit verringert.

Die Vereinfachung von Gebäuden ist dabei ein typischer Bestandteil der Generalisierung. Da die Bewahrung von Symmetrie die Verständlichkeit einer Vereinfachung verbessert, ist es hilfreich, die dominierenden Symmetrien der Gebäude zu kennen. Hierfür werden Verfahren zur Symmetrieerkennung benötigt.

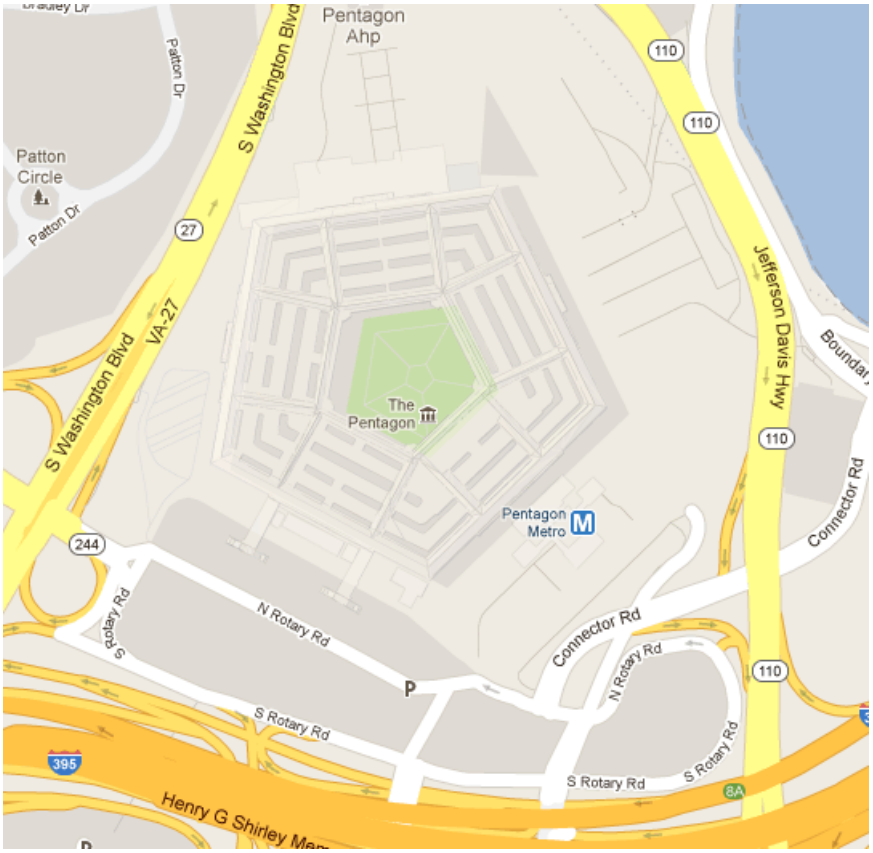

Abb. 1.1: Das Pentagon, der Hauptsitz des US-amerikanischen Verteidigungsministeriums, hat einen regelmäßig 5-eckigen Außenumriss mit einem regelmäßig 5-eckingen Loch. Quelle: Google Maps

1.1.2 Weitere Anwendungsgebiete

Neben der kartografischen Generalisierung kann Symmetrie auch für die automatische Auswahl von Orientierungspunkten für Wegbeschreibungen benutzt werden, siehe Peters et al. [PWW10]. Hat man hierbei Kenntnis über Symmetrien von Gebäuden, so kann man deren Wiedererkennbarkeitswert ausnutzen. Ebenso kann Symmetrie zur Klassifizierung von Gebäuden bezüglich deren Funktion eingesetzt werden, vergleiche Steiniger et al. [SLBW08] sowie Werder et al. [WKS10]. Gebäude, deren symmetrische Eigenschaften von höherer Ordnung sind, haben oft zentrale Funktionalität und sind wichtiger als andere Gebäude.

1.2 Ziel der Arbeit

In dieser Arbeit wird die Symmetrieerkennung in Gebäudeumrissen diskutiert. Symmetrien in Gebäudeumrissen werden oft durch kleine Details wie z.B. Aufgänge oder Laderampen unterbrochen. Abbildung 1.2 verdeutlicht dies. Daher kann es zur Erkennung von Symmetrie notwendig sein, den Gebäudeumriss zuerst zu verändern.

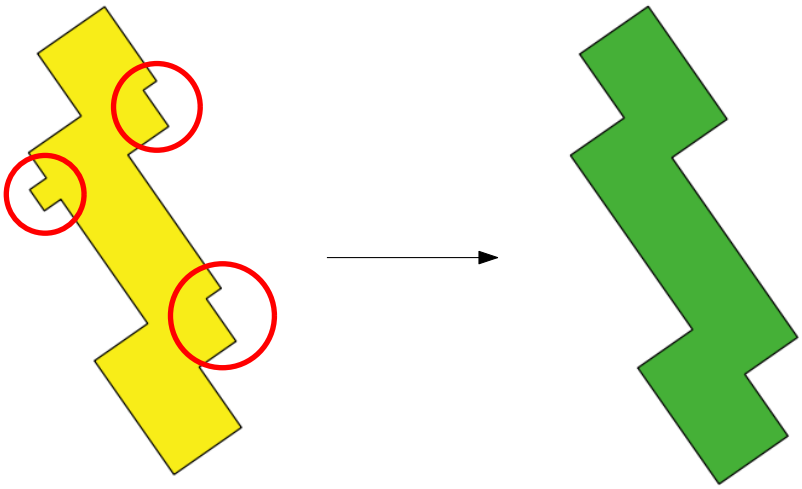

Abb. 1.2: Links ist ein Gebäude zu sehen, welches rotationssymmetrisch ist, wenn die markierten Details ausgefüllt bzw. entfernt werden. Rechts ist der entsprechende rotationssymmetrische Umriss abgebildet.

Bisher wurde Symmetrie als Eigenschaft eines Gebäudegrundrisses noch nicht bei dessen Vereinfachung berücksichtigt, zumal zur Erkennung von Symmetrie manchmal eine Vereinfachung vorliegen muss. In dieser Arbeit wird nach symmetrischen Vereinfachungen von Gebäuden gesucht. Ausgehend von einem einfachen Polygonring als Folge von Kanten wird dieser, aufbauend auf den Ansatz von Haunert und Wolff [HW08], durch Auswahl einer Teilfolge vergröbert. Indem für zwei aufeinanderfolgende Kanten einer

solchen Teilfolge eine neue Ecke dort eingeführt wird, wo sich die Geraden, auf welchen die Kanten liegen, schneiden, entsteht ein vereinfachter Polygonring.

1.3 Struktur der Arbeit

Zunächst wird in dieser Arbeit ein bestehender Ansatz von Lladós et al. [LBM97] zur Erkennung von Symmetrien in Umrissen von Objekten diskutiert. Dieses Verfahren betrachtet einen Umriss ebenfalls als Folge von Polygon-Kanten, wobei wiederholt zwei aufeinanderfolgende Kanten zu einer neuen Kante verschmolzen werden können. Durch dieses Verschmelzen werden die Umrisse vereinfacht, daher ist dieser Ansatz für Symmetrieerkennung in Gebäudeumrissen relevant.

Im weiteren Verlauf wird der Ansatz von Haunert und Wolff [HW08] zur Vereinfachung von Umrissen unter Verwendung von Abkürzungen vorgestellt. Darauf aufbauend werden dann Verfahren zur Erkennung von sich gleichenden Vereinfachungen mehrerer Gebäude sowie zur Erkennung von Spiegelsymmetrien zwischen verschiedenen Gebäuden vorgestellt. Schließlich werden Verfahren zur Erkennung von Symmetrien in einzelnen Umrissen diskutiert.

2 Symmetrieerkennung nach Lladós et al.

Das Verfahren *Cyclic String Matching* wurde von Wagner und Lowrance [WL75] eingeführt, um bei zwei gegebenen Strings durch Manipulation dieser den einen in den anderen überzuführen. Die Manipulation wurde dabei durch mit Kosten belegten Operationen definiert. Mit Hilfe Dynamischer Programmierung wird eine Sequenz von Operationen ermittelt, die minimale Kosten hat. Später wurde Cyclic String Matching von Bunke und Bühler [BB93] eingesetzt, um 2D Umrisse, welche als Strings repräsentiert sind, zu erkennen. Zur Symmetrieerkennung in 2D Umrissen wurde Cyclic String Matching von Lladós et al. [LBM97] benutzt, wobei die Umrisse aus Bilddaten polygonal approximiert extrahiert wurden. Dazu wurde eine Operation zum Manipulieren eingeführt, die zwei Symbole eines Strings zusammenfügt, und iterativ angewendet werden kann. Dieses Zusammenfügen von Symbolen kommt dabei unserem Ansatz mit Abkürzungen nahe. Daher ist eine Anwendung dieses Verfahrens auf Gebäudeumrisse eine Alternative zu dem hier vorgestellten Verfahren.

2.1 Funktionsweise

2.1.1 Rotationssymmetrien in Stringrepräsentation

Wie bei der Gebäudevereinfachung nach Haunert und Wolff [HW08] wird ein Gebäudeumriss durch einen Polygonring modelliert, welcher als Folge von Kanten vorliegt. Es werden aber im Prozess der Umriss-Vereinfachung vor der Symmetrieerkennung keine neuen geometrischen Objekte definiert, sondern es wird beim Vergleichen von Kanten auch deren Vereinigung mit Vorgänger-Kanten berücksichtigt. Hierzu wird eine Folge von Kanten als String betrachtet, dessen Symbole aus Winkeln und Länge bestehen. Sei $\Sigma = \{(l, \alpha, \beta) \mid l \in \mathbb{R}, \alpha, \beta \in [0; 2\pi]\}$ das Alphabet und $X = x_0 \dots x_n \in \Sigma^*$ ein String. Ein Symbol $x_i = (l_i, \alpha_i, \beta_i),\ 0 \leq i \leq n$ steht dann für eine Kante des Polygonzugs X, der entgegen des Uhrzeigersinnes verläuft. Mit l_i wird dabei die Länge der Kante gemessen, mit α_i der innere Winkel zwischen x_i und $x_{i-1 \bmod |X|}$, und β_i bezeichnet den Azimut von x_i, siehe Abbildung 2.1. Für einen Terminus der Art $i - j \bmod |X|$ wird in Zukunft eine einfachere Schreibweise benutzt:

$$i \ominus j \equiv i - j \bmod |X| \text{ und}$$

$$i \oplus j \equiv i + j \bmod |X|.$$

Das Quadrat von Abbildung 2.1 ist zur Ordnung 4 Rotationssymmetrisch, d.h. nach Drehung um 0°, 90°, 180° sowie 270° erhält man stets wieder das gleiche Quadrat. Bezeichne nun $X^i = x_i \dots x_n x_0 \dots x_{i \ominus 1},\ 0 \leq i \leq n$ den um i Stellen nach vorne rotierten

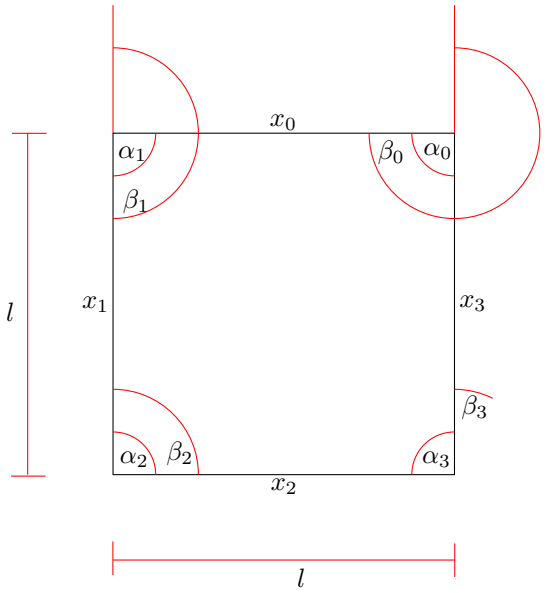

Abb. 2.1: Diese Abbildung zeigt ein Quadrat mit den zugehörigen Attributen für die Stringrepräsentation

String X. Dann soll Cyclic String Matching feststellen, dass X^0, X^1, X^2 und X^3 sich gleichen.

Auch Spiegelsymmetrien lassen sich so erkennen, indem X mit dem String X_{inv} verglichen wird, wobei X_{inv} das selbe Polygon wie X repräsentiert, allerdings die Kanten in Richtung Uhrzeigersinn geordnet sind.

2.1.2 Manipulationsoperationen

Um zu berechnen für welche $0 \leq i \leq n$ sich X und X^i ähnlich sind, werden vier verschiedene Manipulationsoperationen auf X angewendet, um X in X^i zu verwandeln. Löschen eines Symbols x aus X wird dabei mit $x \to \epsilon$ notiert, Einfügen mit $\epsilon \to x$, Austauschen von x_i und x_j mit $x_j \to x_i$ und Vereinigen der Symbole $x_i \dots x_{i \oplus k}$ zum Symbol x' mit $x_{i,i \oplus k} \to x'$. Das Vereinigen von zwei Symbolen $x_{i,i \oplus 1} \to x^1$ wird dabei wie folgt definiert:

$$l^1 = l_i + l_{i \oplus 1}$$

$$\alpha^1 = \alpha_i - \beta_i + \beta^1$$

$$\beta^1 = \frac{\beta_i \cdot l_i + \beta_{i \oplus 1} \cdot l_{i \oplus 1}}{l^1}$$

Die Vereinigung von k Symbolen entspricht der wiederholten Vereinigung zweier Symbole: $x_{i,i \oplus k} \to x^{k-1} \equiv x_i \dots x_{i \oplus k} \to x^{k-1} \equiv x^{k-2} x_{i \oplus k} \to x^{k-1}$.

Diese Operationen sind mit Kostenfunktionen belegt. Sei $l_X = \sum_{x_i \in X} l_i$ die Gesamtlänge des von X repräsentierten Polygonzugs. Dann betragen die Kosten

$$c(x_i \to \epsilon) = K_d + \frac{l_i}{l_X}, \;\; 0 \leq K_d \leq 1$$

für das Löschen von x_i mit einem zuvor festgelegten Löschkosten-Parameter K_d,

$$c(\epsilon \to x_i) = K_i + \frac{l_i}{l_X},\ 0 \le K_i \le 1$$

für das Einfügen von x_i mit einem zuvor festgelegten Einfügekosten-Parameter K_i,

$$c(y_j \to x_i) = \frac{|\alpha_i - \alpha_j|}{2\pi} + \left|\frac{l_j}{l_Y} - \frac{l_i}{l_X}\right|$$

für das Vertauschen von y_j mit x_i und

$$c(x_{i,i\oplus 1} \to x') = \frac{|\beta_i - \beta_{i\oplus 1}|}{\pi} \frac{l_i \cdot l_{i\oplus 1}}{(l')^2}$$

für das Vereinigen von x_i mit $x_{i\oplus 1}$.

2.1.3 Das Dynamische Programm

Das Dynamische Programm arbeitet auf der $(n+2) \times (2n+3)$-Matrix D. Deren Zeile 0 und Spalte 0 werden vorab initialisiert:

$$D(0,j) \ = \ 0\,,\ 0 \le j \le n+1$$

$$D(0,j) \ = \ \infty\,,\ n+2 \le j \le 2n+2$$

$$D(i,0) \ = \ D(i \ominus 1) \ + \ c(x_i \to \epsilon)\,,\ 1 \le i \le n+1$$

Die Werte $D(i,j),\ 1 \le i \le n+1,\ 1 \le j \le 2n+2$ können dann von links oben nach rechts unten berechnet werden, wie im folgenden näher erläutert wird. Es werden zunächst Gesamtkosten für das Vereinigen der Polygon-Kanten $x_{i\ominus k} \dots x_{i-1}$, das Vereinigen der Polygon-Kanten $y_{j\ominus l} \dots y_{j-1}$, sowie das Austauschen derer Resultate berechnet:

$$c(i,j,k,l) \ = \ c(x_{i\ominus k, i-1} \to x') \ + \ c(y_{j\ominus l, j-1} \to y') \ + \ c(x' \to y').$$

Für diese Kosten wird ein Minimum ermittelt, wobei M^2 viele Kombinationen von Vereinigungen ausprobiert werden. Hierbei ist M ein Festparameter, der bestimmt wie viele Symbole rückblickend zusammengefügt werden sollen:

$$c_r(i,j) \ = \ \min\{D(i-1\ominus k, j-1\ominus l) \ + \ c(i,j,k,l) \ \mid \ k,l = 1 \dots M\}.$$

Um das Einfügen von Symbol y_{j-1} in Betracht zu ziehen, legen wir auch für diese Operation einen Folgewert

$$c_d(i,j) \ = \ D(i \ominus 1, j) \ + \ c(x_{i-1} \to \epsilon)$$

fest, analog für das Löschen von Symbol x_{i-1}:

$$c_i(i,j) \ = \ D(i, j \ominus 1) \ + \ c(\epsilon \to y_{j-1}).$$

Abschließend wird der kleinste Wert für alle Operationen ermittelt, die zur aktuellen Position führen können:

$$D(i,j) \ = \ \min\{c_r(i,j),\, c_d(i,j),\, c_i(i,j)\}.$$

Die Symbole y_{j-1} kommen dabei im String XX vor, da X mit sich selbst verglichen werden soll. Außerdem wird gespeichert, welche Sequenz von Manipulationsoperationen verwendet wurde.

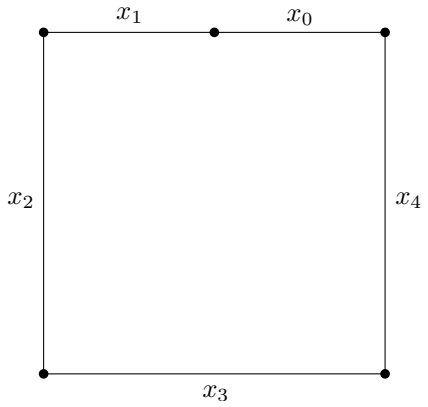

Abb. 2.2: Diese Abbildung zeigt ein Quadrat bei dem es eine Seite gibt, die durch zwei Kanten beschrieben wird.

2.1.4 Auswertung

Zur Auswertung des Dynamischen Programms benutzen Lladós et al. eine Partition P aus den Werten $D(n+1,j)\,,\ \ n+2 \leq j \leq 2n+2$, mit folgender Eigenschaft: Für zwei Mengen $A, B \in P$, $A \neq B$ gilt, dass die Sequenz von Manipulationsoperationen, die zu dem Wert $D(n+1,j_1) \in A$ führt, keine Operation enthält, die auch zu dem Wert $D(n+1,j_2) \in B$ führt. Danach werden diejenigen Symbole y_{j-1} als Startsymbol eines Rotationssymmetrischen Teilstrings verwendet, für die der Wert $D(n+1,j)$ minimal in einer Menge $A \in P$ ist, und $D(n+1,j)$ zudem kleiner als ein zuvor ausgewählter Schwellenwert T ist. Wurde im Dynamischen Programm $D(n+1,j)$ durch Vertauschen von $x_{n \ominus k+1,n}$ mit $y_{j \ominus l,j-1}$ festgelegt, so wird anstelle von y_{j-1} $y_{j \ominus l,j-1}$ als Startsymbol verwendet. Die Anzahl der so ausgewählten Startsymbole von rotationssymmetrischen Teilstrings bestimmt überdies die rotationssymmetrische Ordnung. Die ganze Prozedur kann in $O(n * m)$ Zeit berechnet werden, wobei im Fall $y_j \in XX$ $m = 2n$ gilt. Die Laufzeit beträgt demnach $O(n^2)$.

2.1.5 Startsymbol-Problematik

Abbildung 2.2 zeigt ein Quadrat, bei dem es eine Seite gibt, die durch die Kanten x_0 und x_1 beschrieben wird. Abbildung 2.3 zeigt die zugehörige Ausgabe des Dynamischen Programms, einmal mit x_0 als Startsymbol, darunter mit Startsymbol x_1. Die obere Tabelle wählt als Startsymbole für rotationssymmetrische Teilstrings $x_{0,1}$, x_2, x_3 sowie x_4 aus, was der Realität entspricht. Der untere Teil wählt x_0, x_2 und x_3 aus. Die Verwendung von x_1 führt also nicht zum gewünschten Ergebnis. Somit muss jedes Symbol einmal als Startsymbol ausprobiert werden. Dies erhöht die Gesamtlaufzeit auf $O(n^3)$. Wie dann das finale Startsymbol unter allen Ausgaben auszuwählen ist, bleibt unklar, daher schauen wir im folgenden stets auf die Ausgaben für alle Startsymbole.

	y_0	y_1	y_2	y_3	y_4	y_0	y_1	y_2	y_3	y_4
x_0	0	0,125	0,125	0,125	0,125	0	0,125	0,674	1,224	0,425
x_1	0,125	0	0	0	0	0,125	0	0,5	1	0,85
x_2	0,674	0,5	0	0	0	0,125	0	0	0,5	1
x_3	0,819	1	0,5	0	0	0,125	0	0	0	0,5
x_4	0,319	0,5	1	0,5	0	0,125	0	0	0	0

	y_1	y_2	y_3	y_4	y_0	y_1	y_2	y_3	y_4	y_0
x_1	0	0,375	0,375	0,375	0,25	0	0,528	1,078	1,146	0,425
x_2	0,528	0	0,319	0,319	0,389	0,319	0	0,5	1,028	0,975
x_3	1	0,5	0	0,319	0,444	0,319	0,319	0	0,5	0,819
x_4	0,5	1	0,5	0	0,319	0,319	0,319	0,319	0	0,319
x_0	0,125	0,625	0,819	0,319	0	0,125	0,444	0,444	0,319	0

Abb. 2.3: Hier sieht man die Ausgabe des Dynamischen Programms zum Umriss aus Abbildung 2.2 einmal mit Startsymbol x_0, unten mit Startsymbol x_1.

2.2 Anwendung

Auf der beiliegenden CD befindet sich eine Java-Implementierung des Verfahrens, welches die Matrix des Dynamischen Programms ausgibt und dessen Auswertung vornimmt. Abbildung 2.5 zeigt die Anwendung dieses Programms auf den Umriss aus Abbildung 2.4. Für einen hoch gewählten Löschkosten-Parameter K_d sollten die Symbole x_0 bis x_4 sowie die Symbole x_6 bis x_{10} vereinigt werden und rotationssymmetrische Teilstrings darstellen. Die entsprechenden Manipulationssequenzen sind mit schwarzen Pfeilen beschrieben. Das tatsächliche Ergebnis des Dynamischen Programms weicht aber vom erwünschten Ergebnis ab und ist unbrauchbar: Nachdem die gewünschte Vereinigung dreimal tatsächlich mit einer der Kanten x_5 bzw. x_{11} vertauscht worden ist, ist die Manipulationssequenz zu teuer geworden, und es wird an die billigen Sequenzen, welche die Drehung um 0° bzw. um 180° beschreiben, angefügt. Selbst probieren aller n^2 Startsymbol-Kombinationen kann hier nichts retten: Cyclic String Matching kann für keine Kombination von X^i und XX^j mit $0 \leq i, j \leq n$ eine Matrix mit mehr als drei unabhängigen Manipulationssequenzen berechnen. Auch das schrittweise herabsetzen der Lösch- bzw. Einfügekosten-Parameter K_d und K_i um 0.1-er Schritte von 1.0 bis 0.0 hat an dieser Tatsache nichts ändern können.

Dieses Verfahren eignet sich also nicht sehr gut, um Rotationssymmetrien in Gebäudeumrissen zu erkennen. Lladós et al. bemerken zudem, dass sich Cyclic String Matching auf ähnliche Art und Weise auch zur Erkennung von Spiegelsymmetrien verwenden lässt. Hierbei wird einer der beiden Eingabe-Strings durch dem Uhrzeigersinn entgegengesetztes Entlanglaufen am Umriss gewonnen, dieser String wird mit X_{inv} bezeichnet. Da für den Umriss aus Abbildung 2.4 $X^0 = X^1_{inv}$ gilt und alle Startpaare durchprobiert wurden, lässt sich für dieses Beispiel somit auch keine Ausgabe finden, die alle vier Spiegelachsen bestimmt. Diejenigen Sequenzen, die niemals zusammen in einer Matrix vorkommen, entsprechen dabei den Diagonalen.

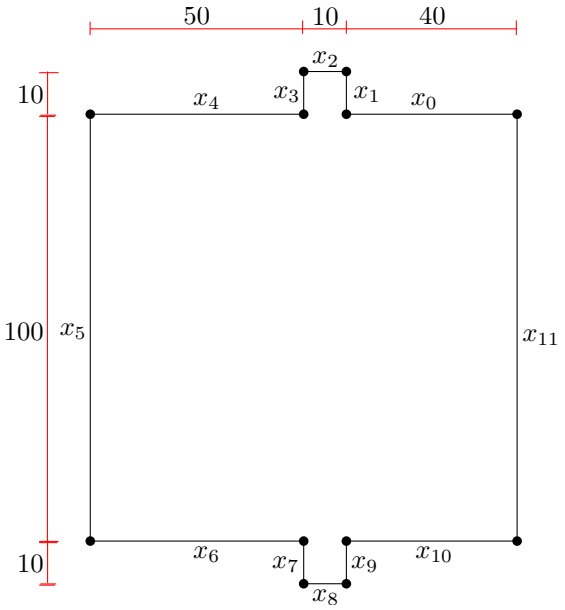

Abb. 2.4: Diese Abbildung zeigt einen quadratischen Gebäudeumriss, der zwei kleine Unregelmäßigkeiten aufweist.

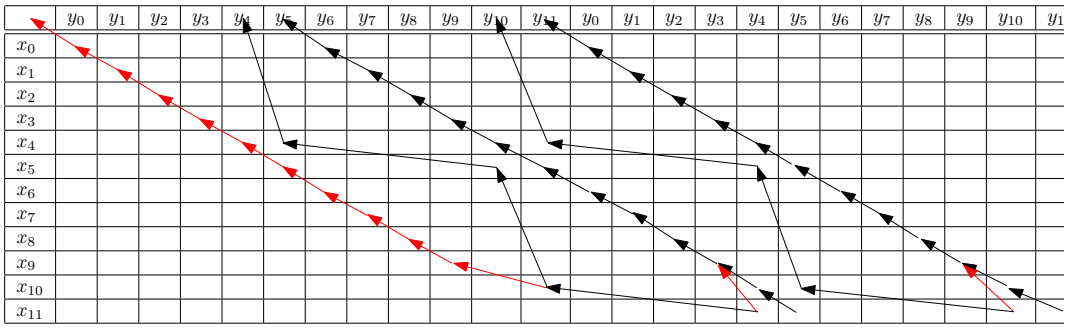

Abb. 2.5: Hier sieht man die Manipulationssequenz des Dynamischen Programms zum Gebäudeumriss aus Abbildung 2.4 wie sie zweckmäßig wäre. Die rot gezeichneten Manipulationen beschreiben die tatsächliche Abweichung von schwarz gezeichneten erwünschten Manipulationen.

2.3 Schlussfolgerung

Cyclic String Matching nach Lladós, Bunke und Martí eignet sich möglicherweise zum Erkennen von Symmetrien in 2D-Umrissen, die polygonal approximiert aus Bild-Daten gewonnen werden. Hat man aber Umrisse, die Gebäuden entsprechen, so möchte man ein Verfahren, welches Symmetrien erkennen kann, die durch kleine Unregelmäßigkeiten wie z.B. Schornsteine, Terrassen oder Treppen unterbrochen werden. Für das Beispiel aus Abbildung 2.4 sollte man eine Rotationssymmetrie der Ordnung 4 erkennen können. Hierzu ist Cyclic String Matching nicht geeignet.

3 Ansatz zur Vereinfachung nach Haunert und Wolff

Für die zur Erkennung von Symmetrien in Gebäudeumrissen notwendige Vereinfachung der Umrisse bauen wir auf den Ansatz von Haunert und Wolff [HW08] auf. Ein Gebäudeumriss hat einen einfachen Polygonring als Außenumriss und kann im Inneren mehrere einfache Polygonringe als Löcher haben. Ein Polygonring $P = \langle e_1, \ldots, e_n \rangle$ liegt dabei als Folge von Kanten vor.

Eine *Abkürzung* (engl. *Shortcut*) $s = (e_i, e_j)$ ist ein Paar von Kanten eines Polygonrings P. Die Anwendung der Abkürzung s auf P führt zu einem vereinfachten Polygonring. Die Kanten $e_{i+1}, \ldots, e_{j-1}$ fallen dabei weg. Für s wird eine neue Ecke dort eingeführt, wo die Geraden, auf welchen e_i und e_j liegen, sich schneiden. In Abbildung 3.1 ist ein Beispiel für das Prinzip von Abkürzungen skizziert.

Der Generalisierungs-Ansatz mit Abkürzungen stellt sicher, dass der Winkel zwischen den beiden Kanten einer Abkürzung im vereinfachten Polygon erhalten bleibt. So bleiben Eigenschaften wie Rechtwinkligkeit erhalten.

3.1 Auswahl geeigneter Abkürzungen

Damit eine Vereinfachung eines Gebäudes dem Original möglichst ähnlich sieht, wird eine Toleranz ϵ für die geometrische Abweichung der Abkürzung eingeführt.

3.1.1 Auswahl über Hausdorff-Abstand

Eine Abkürzung (a, b) hat die Eigenschaft, dass sie einen Polygonzug $p(a, b)$, der auf dem ursprünglichen Polygonring liegt, durch einen anderen Polygonzug $p'(a, b)$ ersetzt. Hierbei können $p(a, b)$ und $p'(a, b)$ so gewählt werden, dass beide am gleichen Punkt auf a anfangen und am gleichen Punkt in b enden. Abbildung 3.2 zeigt ein Beispiel hierfür. Der Polygonzug $p'(a, b)$ kann dabei immer mit drei Punkten beschrieben werden. Er besteht aus der Vereinigung von zwei Strecken. Die eine fängt am bereits gewählten Anfangspunkt an und endet an der Ecke, die (a, b) einführt. Die andere fängt an der neuen Ecke an und endet am bereits gewählten Endpunkt. Übersteigt der Hausdorff-Abstand zwischen $p(a, b)$ und $p'(a, b)$ die Toleranz-Schwelle ϵ nicht, so wird die Abkürzung akzeptiert.

3.1.2 Typisierung der Abkürzungen

Der in Abbildung 3.3 dargestellte Fall zeigt, dass der Endpunkt von $p(a, b)$ und $p'(a, b)$

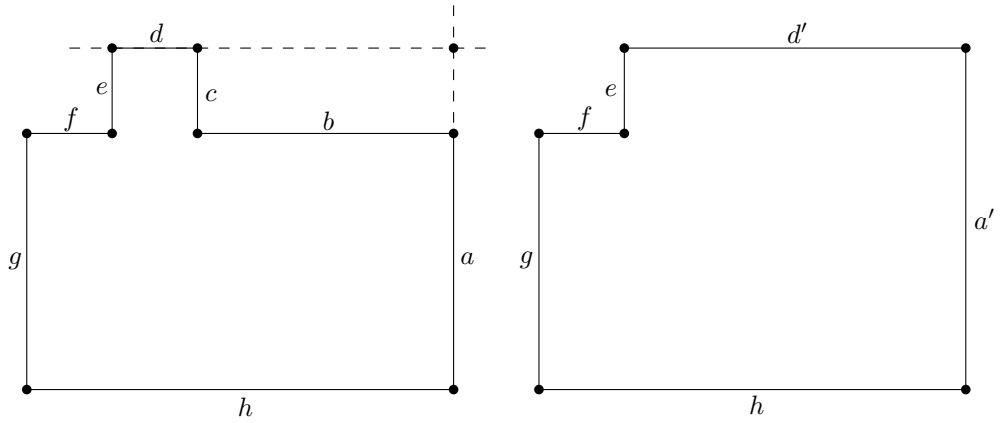

Abb. 3.1: Links ist das Eingabe-Polygon $\langle a, \ldots, h \rangle$ zu sehen. Rechts ist das Polygon $\langle a', d', e, \ldots, h \rangle$ zu sehen, das durch Anwendung der Abkürzung (a, d) entsteht. Dabei wurde auf dem Schnittpunkt der Geraden, auf welchen a und d liegen, eine neue Ecke eingeführt.

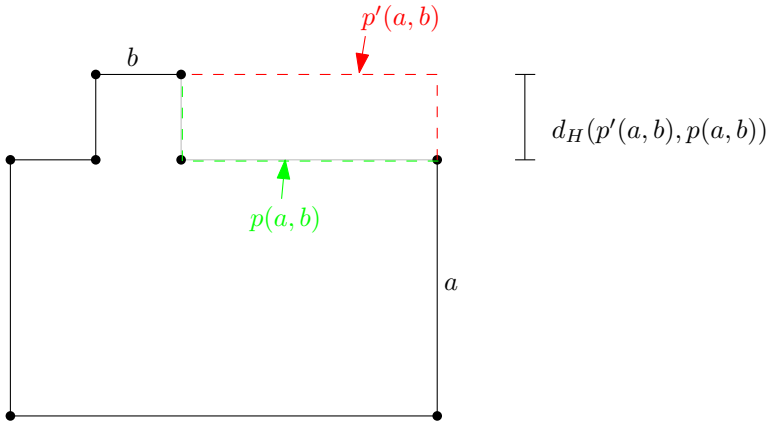

Abb. 3.2: Für das Eingabe-Polygon aus Abbildung 3.1 sieht man die Polygonzüge $p(a, b)$ und $p'(a, b)$. Es wird gefordert, dass der Hausdorff-Abstand d_H zwischen den beiden Polygonzügen ϵ nicht übersteigt.

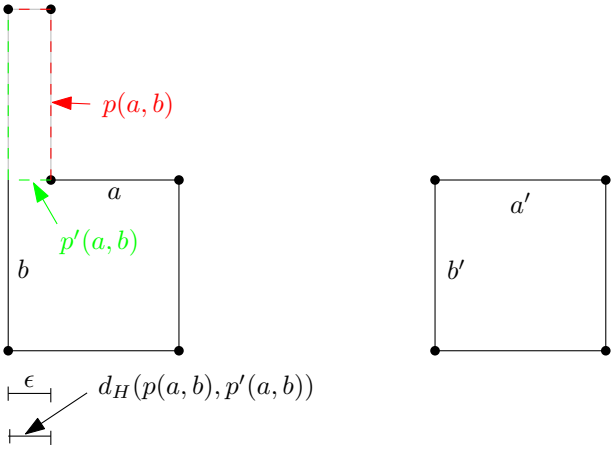

Abb. 3.3: Für die Abkürzung (a, b) des linken Polygonrings ist der Hausdorff-Abstand d_H zwischen $p(a, b)$ und $p'(a, b)$ nicht größer als ϵ. Angewendet ergibt sich daraus der rechte Polygonring, dessen längste Seite nur noch halb so lang ist wie die längste Seite des linken Polygonrings.

nicht beliebig auf b liegen darf. Denn nach Anwendung der Abkürzung (a, b) ist die längste Seite der resultierenden Vereinfachung, beschrieben durch Kante b', nur noch halb so lang wie die längste Seite des Originals. Da die Länge von b' aber immer noch ein vielfaches von ϵ beträgt, darf diese Abkürzung nicht verwendet werden. Somit müssen die Anfangs- und Endpunkte von $p(a, b)$ und $p'(a, b)$ so gewählt werden, dass $p'(a, b)$ keine weiteren Schnittpunkte mit a und b hat. Hierfür werden vier Fälle betrachtet.

Typ-0

Eine Abkürzung (a, b) ist von *Typ-0*, wenn diese trivial ist, d.h. a und b sind konsekutive Kanten im Ursprungs-Polygonring. Alle trivialen Abkürzungen werden in Betracht gezogen.

Typ-1

Unter einer *Typ-1*-Abkürzung verstehen wir eine Abkürzung (a, b), die eine Ecke neu einführt, welche auf der Kante a liegt. Abbildung 3.4 zeigt ein Beispiel für eine Typ-1-Abkürzung. Die Polygonzüge $p(a, b)$ und $p'(a, b)$ fangen in diesem Fall an der Ecke an, die durch (a, b) neu eingeführt wird. Sie enden am ersten Punkt, der auf b liegt.

Typ-2

In Analogie zu Typ-1-Abkürzungen stehen Abkürzungen vom *Typ-2*. Hier liegt die neu eingeführte Ecke auf Kante b. Entsprechend enden $p(a, b)$ und $p'(a, b)$ an der neu eingeführten Ecke und fangen am letzten Punkt an, der auf a liegt.

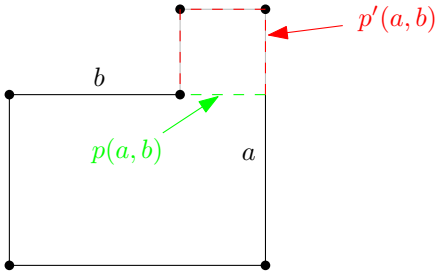

Abb. 3.4: Die Abkürzung (a, b) ist vom Typ-1. Die Polygonzüge $p(a, b)$ und $p'(a, b)$ fangen nicht am letzten Punkt von a an, sondern am Schnittpunkt der Geraden, auf denen a und b liegen.

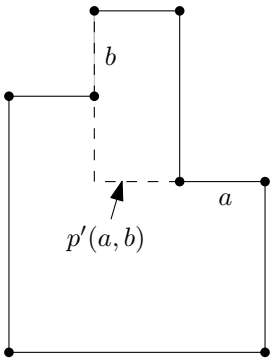

Abb. 3.5: Für die Abkürzung (a, b) liegt b auf $p'(a, b)$.

Typ-3

Bei *Typ-3*-Abkürzungen fangen $p(a, b)$ und $p'(a, b)$ am letzten Punkt von a an und enden am ersten Punkt von b. Dies entspricht dem Beispiel aus Abbildung 3.2. Es ist zu beachten, dass a und b selbst nicht auf $p'(a, b)$ liegen dürfen. Abbildung 3.5 zeigt ein Beispiel für eine Abkürzung, bei der b auf $p'(a, b)$ liegt. Im resultierenden Polygonring würde die Kante b komplett wegfallen, dafür würde es eine zu b kollineare Kante geben, die in Gegenrichtung zu b verläuft.

3.2 Abkürzungsgraph und Kombinationsgraph

Wie Haunert und Wolff gezeigt haben, ist das Problem der Umriss-Generalisierung NP-schwer, wenn man als Ausgabe die Vereinfachung mit der minimalen Anzahl an Kanten anstrebt, und zudem fordert, dass diese einfach ist. Für die Symmetrieerkennung selbst ist die Einfachheit der Ausgabe nicht so wichtig. Darum verzichten wir darauf, dass die vereinfachten Polygonringe einfach sind. In Abbildung 3.6 ist einer der Fälle zu sehen, die zugelassen werden. Somit sucht man eine Folge von Abkürzungen mit der Eigenschaft, dass keine Abkürzung mit einer Kante beginnt, die von einer anderen Abkürzung Übersprungen wird. Dies führt zum Abkürzungsgraph.

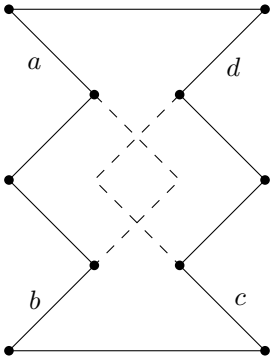

Abb. 3.6: Die Anwendung der beiden Abkürzungen (a, b) und (c, d) führt zu einem resultierenden Polygonring, der nicht einfach ist.

3.2.1 Abkürzungsgraph

Im *Abkürzungsgraph* $G = (V, E)$ werden Kanten des Eingabe-Polygonrings P durch Abkürzungen verbunden. Die Kanten von P sind die Knoten von G. Die Abkürzungen, die das Auswahl-Kriterium erfüllen, sind E. Eine Vereinfachung von P ist ein Kreis in G. Abbildung 3.7 zeigt ein Beispiel für einen Abkürzungsgraph.

Die Menge aller möglichen Abkürzungen ist schlimmstenfalls $O(n^2)$, wenn n die Anzahl der Kanten des Eingabe-Polygons ist. Die Berechnung des diskreten Hausdorff-Abstands benötigt dabei $O(n)$ Zeit, während die Typisierung in konstanter Zeit abläuft. Somit kann der Abkürzungsgraph in $O(n^3)$ Zeit aufgestellt werden.

3.2.2 Kombinationsgraph

Von einer Kante können mehrere Abkürzungen ausgehen, aber es können auch mehrere Abkürzungen zu dieser Kante hinführen. Somit muss auf jede einführende Abkürzung eine ausgehende Abkürzung folgen. Wie in Abbildung 3.8 zu sehen ist, wird im vereinfachten Polygonring zwischen den Ecken, die durch die Abkürzungen eingeführt werden, keine weitere Ecke liegen. Eine Kante in der Vereinfachung wird also über eine *Kombination* zweier Abkürzungen definiert, welche über eine Kante im Original-Polygon verbunden sind. Da man, wie wir später sehen werden, während der Suche nach einer Generalisierung Zugriff auf die Längen der resultierenden Kanten benötigt, definieren wir einen *Kombinationsgraph*. Die Knotenmenge des Kombinationsgraph besteht aus der Kantenmenge des Abkürzungsgraph, also Abkürzungen. Zwei Abkürzungen (a, b) und (c, d) sind im Kombinationsgraph über eine Kante verbunden, wenn $b = c$ gilt. Abbildung 3.7 zeigt einen Kombinationsgraph.

Richtungsumkehrung einer Polygon-Kante

Unter Umständen liegt die Ecke, die durch die ausgehende Abkürzung einer Kombination definiert wird, im vereinfachten Polygonring vor der Ecke, die durch die eingehende Abkürzung eingeführt wird, auf der Polygon-Kante. Dadurch würde die Polygon-Kan-

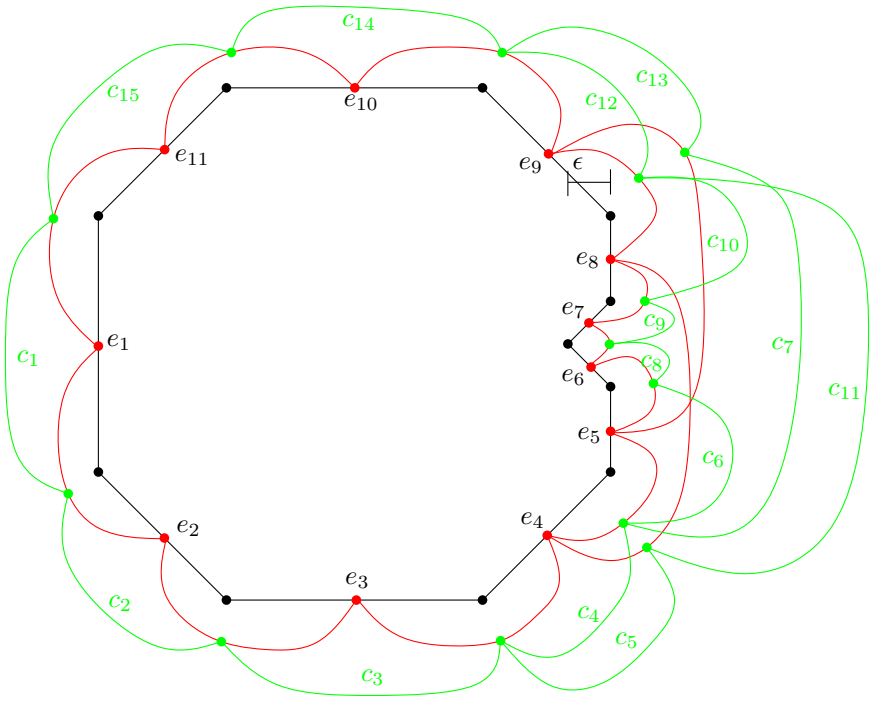

Abb. 3.7: Für den abgebildeten Polygonring ist der Abkürzungsgraph rot angedeutet. Der Kombinationsgraph ist grün eingezeichnet.

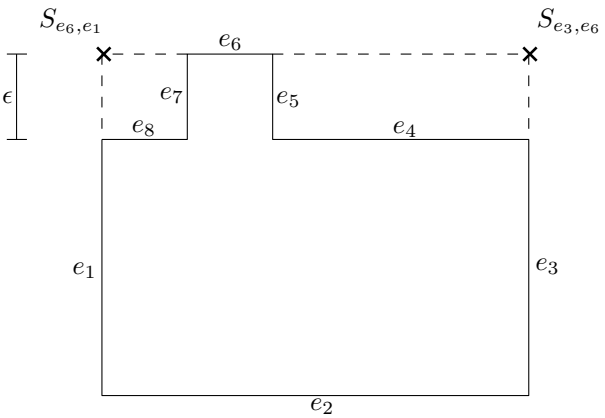

Abb. 3.8: Folgt nach Abkürzung (e_3, e_6) Abkürzung (e_6, e_1), so wären in dieser Vereinfachung S_{e_3,e_6} und S_{e_6,e_1} aufeinanderfolgende Ecken.

te in der Vereinfachung in umgekehrter Richtung zum Original durchlaufen werden. Dies hat die Benutzung der Innenseite einer Wand im Eingabe-Umriss als Außenseite im vereinfachten Umriss zur Folge, bzw. umgekehrt, wenn Löcher untersucht werden. Kombinationen, bei welchen dies der Fall ist, werden ausgeschlossen.

Pfade

Findet man nun einen Pfad im Kombinationsgraph, so findet man dabei einen Pfad im Abkürzungsgraph, während man bei der Suche Zugriff auf die Längen der Kanten in der resultierenden Vereinfachung hatte.

Topologische Ordnung

Eine Kombination eines Polygonrings P verbindet immer eine Abkürzung, die zu einer Kante e von P hinführt, mit einer Abkürzung, die von e wegführt. Dadurch lässt sich jede Kombination einer Kante zuordnen. Für zwei Kanten e_i und e_j mit $i < j$ ist klar, dass die Kombinationen, die e_j zugeordnet werden, niemals Vorgänger von einer Kombination sein können, welche e_i zugeordnet wird. Somit ist durch Zuordnung der Kombinationen zu Kanten von P eine topologische Ordnung der Kombinationen gegeben und muss nicht extra berechnet werden.

Generalisierungs-Menge

Der Kombinationsgraph ist eine für das Erkennungsverfahren ausreichend spezifizierte Menge von Generalisierungen. Man braucht noch den inneren Winkel zwischen zwei aufeinanderfolgenden Kombinationen. Dieser ist bereits durch die Abkürzung, über welche die Kombinationen verbunden sind, definiert.

Schlimmstenfalls gibt es $O(s \cdot s)$ Kombinationen, wobei s die Anzahl der ausgewählten Abkürzungen ist. Der Test, in welcher Reihenfolge die Ecken liegen, ist in konstanter Zeit durchführbar. Somit kann der Kombinationsgraph in $O(n^4)$ Zeit aus dem Abkürzungsgraph gewonnen werden.

4 Erkennung von Symmetrien zwischen verschiedenen Gebäuden

Neben der Erkennung von Symmetrien innerhalb eines Umrisses, ist auch die Erkennung von formgleichen Teilstücken in den Umrissen von verschiedenen Gebäuden interessant. Die Gebäude werden dabei paarweise verglichen. Mit *Formgleichheit* ist gemeint, dass ein Teilstück des einen Gebäudeumrisses durch Translation und Rotation so transformiert werden kann, dass es gleich dem entsprechenden Teilstück des anderen Gebäudes ist.

Definition 4.1. Sei $S = \langle s_i \rangle_{1 \leq i \leq n}$ ein Kreis im Abkürzungsgraph eines Polygonrings P. Die *vereinfachte Version* oder *Vereinfachung* von P ist der Polygonring, der durch die Folge von Ecken beschrieben wird, die zu den Abkürzungen s_i für $1 \leq i \leq n$ gehören.

Problem 1. Gegeben sind zwei einfache Polygonringe P_1 und P_2, die als Folge von Kanten in gleicher Drehrichtung vorliegen. Gesucht ist ein längster Polygonzug in einer vereinfachten Version von P_1, der formgleich in einer vereinfachten Version von P_2 vorkommt.

Die *Länge* eines solchen Polygonzugs kann dabei geometrisch oder kombinatorisch verstanden werden. Geometrisch könnte man z.B. die Summe der euklidischen Abstände zwischen den Ecken messen. Ebenso kann man die Anzahl der Ecken des gesuchten Polygonzugs als Länge verstehen.

4.1 Der Vergleichsgraph

Die Menge der zu vergleichenden Generalisierungen ist durch die Kombinationsgraphen $G_1 = (V_1, E_1)$ von P_1 und $G_2 = (V_2, E_2)$ von P_2 gegeben. Der *Vergleichsgraph* G_V wird ausgehend von einem Paar $(c_1, c_2) \in E_1 \times E_2$ von Kombinationen definiert. Das Paar (c_1, c_2) ist ein *Vergleich*. Wir nennen c_1 das *Urbild* und c_2 das *Bild*. Für G_V ist (c_1, c_2) das *Startpaar* oder auch der *Startvergleich*.

4.1.1 Knotenmenge und Kantenmenge

Die Knotenmenge des Vergleichsgraphen ist eine Teilmenge von $E_1 \times E_2$. Ein Vergleich (c_1, c_2) kann zulässig sein oder nicht. Zulässig ist er, wenn der innere Winkel von c_1 zu dessen Vorgängerkombination in G_1 gleich dem inneren Winkel von c_2 zu dessen Vorgängerkombination in G_2 ist. Da dieser Winkel durch die eingehenden Abkürzungen festgelegt wird, spielt es keine Rolle, welche Vorgängerkombination hierfür in Betracht gezogen wird. Zudem sollen die Längen von c_1 und c_2 gleich groß sein. Diese geometrischen Bedingen garantieren, dass c_1 und c_2 die Formgleichheit fortsetzen, wenn es bereits

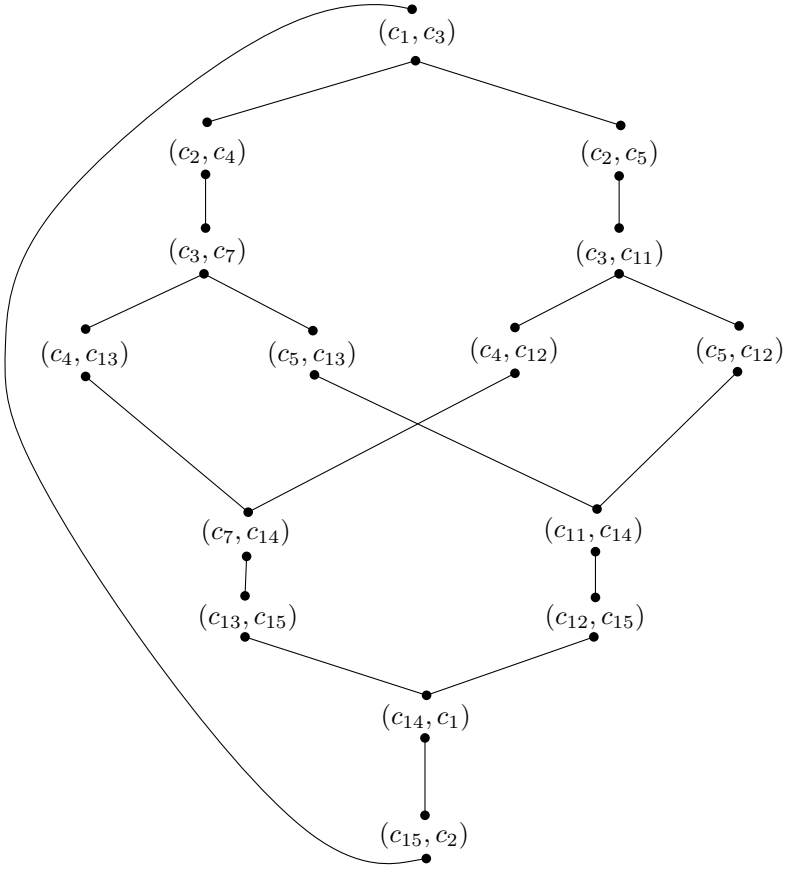

Abb. 4.1: Diese Abbildung zeigt den Vergleichsgraph für das Polygon aus Abbildung 3.7 als P_1 und als P_2. Der Startvergleich ist (c_1, c_3).

Vorgängerkombinationen von c_1 und c_2 gibt, die formgleich sind. Für den Startvergleich wird nur gefordert, dass die Längen von Urbild und Bild gleich sind.

Sei nun V_V^* die Menge aller Vergleiche, für welche die geometrischen Bedingungen erfüllt sind. Wir betrachten nun die Kantenmenge E_V^*. Zwei Vergleiche $(c_{1,1}, c_{2,1})$ und $(c_{1,2}, c_{2,2})$ sind eine in E_V^* liegende Kante, wenn $c_{1,2}$ in G_1 ein Nachfolger von $c_{1,1}$ ist, und $c_{2,2}$ in G_2 ein Nachfolger von $c_{2,1}$. Der Graph $G_V^* = (V_V^*, E_V^*)$ ist der *Supervergleichsgraph*. Der Vergleichsgraph G_V für einen Startvergleich v_0 ist die Zusammenhangkomponente in G_V^*, die v_0 enthält. Abbildung 4.1 zeigt ein Beispiel für einen Vergleichsgraph.

4.1.2 Topologische Ordnung

Mit der topologischen Ordnung der Urbilder von Vergleichen ist auch eine topologische Ordnung für die Vergleiche selbst gegeben. Denn wenn ein Vergleich weiter vorne auf einem Pfad liegt, als ein anderer Vergleich, so gilt dies für deren Urbilder erst recht.

4.2 Der reduzierte Vergleichsgraph

Um Problem 1 zu lösen, sucht man einen Kreis oder einen längsten Pfad in G_V ausgehend von dessen Startvergleich. Betrachtet man den Vergleichsgraph aus Abbildung 4.1, so sieht man, dass es Kreise und Pfade gibt, die mehrmals um den Vergleichsgraph herumlaufen. Solche Vereinfachungen müssen ausgeschlossen werden.

Definition 4.2. Sei P ein Polygonring und $e_i \in P$ eine Kante. Eine Kombination $c = (s_1, s_2)$ *überspringt* e_i, wenn die Abkürzung $s_1 = (e_k, e_\ell)$ Kante e_i überspringt, d.h. es gilt $k < i < \ell$.

Im Beispiel aus Abbildung 4.1 überspringt Kombination c_{11} die Kanten $e_5 \dots e_7$.

Definition 4.3. Sei P ein Polygonring und $e_2 \in P$ eine Kante. Eine Kombination $c = (s_1, s_2)$ *benutzt* e_2, wenn $s_1 = (e_1, e_2)$ für eine Kante $e_1 \in P$ gilt.

Im Beispiel aus Abbildung 4.1 benutzen die Kombinationen c_4 und c_5 Kante e_4. Kombinationen, welche eine Kante $e \in P$ benutzen oder überspringen, heißen *Überspringer* von e.

Um zu verhindern, dass in G_V der längste Pfad länger als der längste einfache Kreis ist, auf dem der Startvergleich liegt, wird G_V reduziert. Für die Kante von P_1, zu welcher das Urbild des Startvergleichs zugeordnet wird, werden keine anderen Überspringer als Urbild für Vergleiche zugelassen. So ist auch gewährleistet, dass der Startvergleich auf jedem Kreis in G_V liegt.

4.2.1 Aufbau des reduzierten Vergleichsgraph

Algorithmus 1 zeigt, wie der reduzierte Vergleichsgraph aufgebaut wird. Wie bereits erwähnt wurde, sind die Kombinationen durch Zuordnung zu den Polygon-Kanten, die sie überbrücken, topologisch geordnet. Daher wird in der äußersten Schleife in Zeile 7 über die Polygon-Kanten, bzw. deren Indizes, iteriert. Dabei wird bei jener Polygon-Kante angefangen, zu welcher das Urbild des Startvergleichs zugeordnet wird. Die Mengen V_i entsprechen den Kombinationen, die zur i-ten Kante zugeordnet werden. Die Reihenfolge, in welcher in Zeile 8 anfangend über diese iteriert wird, spielt keine Rolle. In den inneren Schleifen wird jede Nachfolge-Kombinationen vom Urbild des gerade betrachteten Vergleichs v_1 mit jeder Nachfolge-Kombination vom Bild von v_1 verglichen. Falls diese passen, wird eine Kante zum Vergleichsgraph hinzugefügt. Insgesamt wird für jeden Vergleich einmal jeder Nachfolgevergleich betrachtet. Der Geometrie-Test für diesen ist in konstanter Zeit berechenbar. Sei n die Anzahl der Kanten von P_1 und m die Anzahl der Kanten von P_2. Dann gibt es maximal $O(n^4 \cdot m^4)$ Vergleiche, mit maximal genauso vielen Nachfolgevergleichen. Somit benötigt der Aufbau des reduzierten Vergleichsgraph im schlimmsten Fall $O((m \cdot n)^8)$ Zeit.

4.2.2 Pfadsuche im reduzierten Vergleichsgraph

Im reduzierten Vergleichsgraph $G'_V = (V'_V, E'_V)$ kann man schließlich für eine Längenfunktion length: $V'_V \to \mathbb{R}$ z.B. durch Relaxierung, vergleiche Cormen et al. [CLR04],

Algorithmus 1: Aufbau des reduzierten Vergleichsgraphen

Eingabe : Kombinationsgraphen G_1 und G_2; Startvergleich v_0; Kantenanzahl n des Eingabe-Rings P_1

Ausgabe : Reduzierter Vergleichsgraph für Startvergleich v_0

```
1  E_V = ∅
2  for 1 ≤ i ≤ n do
3      V_i = ∅
4  e = Kante, zu der das Urbild von v_0 zugeordnet wird
5  topology = Index von e
6  V_topology = V_topology ∪ {v_0}
7  for topology − 1 ≤ i < topology − 1 + n do
8      foreach v_1 ∈ V_((i mod n)+1) do
9          c_1,1 = Urbild von v_1
10         c_2,1 = Bild von v_1
11         foreach c_1,2 als Nachfolger von c_1,1 in G_1 do
12             if c_1,2 ∈ Überspringer(e) ∧ c_1,2 ≠ Urbild von v_0 then
13                 continue;
14             target = Index der Kante, zu der c_1,2 zugeordnet wird
15             foreach c_2,2 als Nachfolger von c_2,1 in G_2 do
16                 if Längen und Winkel zu Vorgängern von c_1,2 und c_2,2 passen nicht
                   then
17                     continue
18                 v_2 = (c_1,2, c_2,2)
19                 V_target = V_target ∪ {v_2}
20                 E_V = E_V ∪ {(v_1, v_2)}
21 return (⋃_{i=1}^{n} V_i, E_V)
```

einen längsten Pfad berechnen. Algorithmus 2 zeigt dies. Die äußeren Schleifen, die in den Zeilen 8 und 9 beginnen, iterieren dabei wieder wie beim Aufbau des Vergleichsgraph über alle Vergleiche in topologischer Ordnung. In der innersten Schleife wird dann über alle ausgehenden Kanten des betrachteten Vergleichs iteriert. Wenn der Zielvergleich v_2 über diese Kante günstiger erreicht werden kann, als dies bisher der Fall ist, so wird dies im Zeiger $p(v_2)$ gespeichert. Dies passiert in den Zeilen 24 bis 30. Führt die betrachtete Kante zum Startvergleich, so gibt es einen Kreis. Für diesen Fall gibt es in den Zeilen 11 bis 21 eine Sonderbehandlung, da die Distanz nicht weiter berechnet werden muss. Zudem wird noch ein Kosten-Kriterium $\text{cost}\colon V'_V \to \mathbb{R}$ angewendet, falls die Distanz gleich groß sein sollte. Besonders wenn als Länge die Anzahl der Vergleiche verwendet wird, ist dies ein sinnvolles Zusatz-Kriterium. In den Zeilen 31 bis 34 wird der Variable *tail* der am weitesten vom Startvergleich entfernte Vergleich zugewiesen. Nach Relaxierung über alle Kanten kann beginnend bei *tail* über die Pfad-Zeiger $p(v)$ zu v_0 zurück iteriert werden. Hieraus ergibt sich der längste Pfad.

Unter der Voraussetzung, dass die Längen- sowie die Kostenfunktion in konstanter Zeit berechenbar ist, hat die Pfadsuche die gleiche Laufzeit wie der Aufbau des reduzierten Vergleichsgraph, also $O((m \cdot n)^8)$, für die Anzahlen der Kanten n von P_1 und m von P_2.

Längenfunktion

Die Längenfunktion

$$\text{length}\colon V'_V \to \mathbb{R}$$

berechnet für jeden Vergleich $v \in V'_V$ eine Länge. Wie bereits erwähnt können unterschiedliche Maße als Länge aufgefasst werden. Sei $c = (s_1, s_2)$ das Urbild von v. Der euklidische Abstand zwischen den Ecken, die durch s_1 und s_2 eingeführt werden, wäre ein möglicher Funktionswert. Ein weiteres Beispiel für die Länge eines Vergleichs wäre die Zahl 1. So würde die Länge des resultierenden Polygonzugs anhand der Anzahl dessen Kanten gemessen werden.

Kostenfunktion

Bei gleichen Längen zweier Lösungs-Möglichkeiten kann ein Kosten-Kriterium

$$\text{cost}\colon V'_V \to \mathbb{R}$$

zur Entscheidung herangezogen werden. Auch hierfür existieren mehrere Möglichkeiten. Sei $c = (s_1, s_2)$ das Urbild eines Vergleichs $v \in V'_V$. Ein möglicher Funktionswert wäre die Summe der Hausdorff-Abstände von s_1 und s_2, wie sie in Abschnitt 3.1.1 beschrieben wurden. Des weiteren können die Längen und Winkel-Differenzen von Urbild und Bild verwendet werden. Zudem können Kosten auch für die Anzahl der Kanten im Eingabe-Ring, die c überspringt, auferlegt werden.

Algorithmus 2: Längster Pfad im reduzierten Vergleichsgraph

Eingabe : Reduzierter Vergleichsgraph $(\bigcup_{i=1}^{n} V_i, E_V)$; Startvergleich v_0; Kantenanzahl n des Eingabe-Rings P_1

Ausgabe : Längster Pfad im reduzierten Vergleichsgraph

```
1  e = Kante, zu der das Urbild von v_0 zugeordnet wird
2  topology = Index von e
3  foreach v ∈ ⋃_{i=1}^{n} V_i do
4      length(v) = 0
5  length(v_0) = lengthFunc(v_0)
6  cost(v_0) = costFunc(v_0)
7  tail = v_0
8  for topology − 1 ≤ i < topology − 1 + n do
9      foreach v_1 ∈ V_{(i mod n)+1} do
10         foreach v_2 : (v_1, v_2) ∈ E_V do
11             if v_2 = v_0 then
12                 if p(v_0) = NIL then
13                     p(v_0) = v_1
14                     tail = v_0
15                 else if length(v_1) > length(p(v_0)) then
16                     p(v_0) = v_1
17                     tail = v_0
18                 else if length(v_1) = length(p(v_0)) ∧ cost(v_1) < cost(p(v_0)) then
19                     p(v_0) = v_1
20                     tail = v_0
21                 continue
22             length = length(v_1) + lengthFunc(v_2)
23             cost = cost(v_1) + costFunc(v_2)
24             if length > length(v_2) then
25                 p(v_2) = v_1
26                 length(v_2) = length
27                 cost(v_2) = cost
28             else if length = length(v_2) ∧ cost < cost(v_2) then
29                 p(v_2) = v_1
30                 cost(v_2) = cost
31             if length > length(tail) then
32                 tail = v_2
33             else if length = length(tail) ∧ cost < cost(tail) then
34                 tail = v_2
35 return reverse(⟨tail, p(tail), p(p(tail)), ..., v_0⟩)
```

4.3 Startvergleiche

Der Vergleichsgraph muss für jedes Startpaar $v_0 \in E_1 \times E_2$ der Kombinationsgraphen $G_1 = (V_1, E_1)$ von P_1 und $G_2 = (V_2, E_2)$ von P_2 aufgestellt werden. Für jedes Startpaar wird im reduzierten Vergleichsgraph ein längster Pfad ermittelt. Die Lösung von Problem 1 ergibt sich aus dem Maximum über den Lösungen aller Startpaare. Startpaare, die in der Lösung eines anderen Startpaares vorkommen, können ignoriert werden. Algorithmus 3 demonstriert dies. Es wird dabei über $O((n \cdot m)^4)$ Startvergleiche iteriert,

Algorithmus 3: Lösung von Problem 1

Eingabe : Kombinationsgraphen $G_1 = (V_1, E_1)$ und $G_2 = (V_2, E_2)$ zweier Polygonringe P_1 und P_2
Ausgabe : Längster Polygonzug, der als Teilstück in P_1 formgleich zu einem Teilstück in P_2 vorkommt

```
result = ⟨⟩
ignored = ∅
foreach v_0 ∈ E_1 × E_2 do
    if v_0 ∈ ignored then
        continue
    G'_V = reduzierter Vergleichsgraph für v_0
    p = längster Pfad in G'_V
    if lengthFunc(p) > lengthFunc(result) then
        result = p
    else if
    lengthFunc(p) = lengthFunc(result) ∧ costFunc(p) < costFunc(result)
    then
        result = p
    ignored = ignored ∪ p
return result
```

wobei n wie gehabt die Anzahl der Kanten von P_1 ist und m die Anzahl der Kanten von P_2. Die meiste Zeit innerhalb dieser Schleife benötigen gleichermaßen der Aufbau des reduzierten Vergleichsgraph sowie die Pfadsuche in diesem mit $O((n \cdot m)^8)$. Somit ergibt sich eine Gesamtlaufzeit von $O((n \cdot m)^{12})$. Diese ist in der Praxis aber wesentlich geringer, da meistens weit weniger als n^4 Kombinationen von Abkürzungen existieren. Unabhängig davon existieren im Vergleichsgraph auch deutlich weniger als $(n \cdot m)^4$ Vergleiche und auch weniger als $(n \cdot m)^8$ Kanten. Im nächsten Abschnitt werden Abweichungstoleranzen thematisiert. Geht man einmal von perfekten Vergleichen aus, d.h. ohne Tolerierung von Abweichungen, so kann man davon ausgehen, dass für eine Kombination c_1 die Vergleichsgraphen für die Startvergleiche (c_1, c_2) über allen $c_2 \in E_2$ disjunkte Kontenmengen haben. Dadurch reduziert sich die Laufzeit auch für den schlimmsten Fall auf $O(n^{12} \cdot m^8)$.

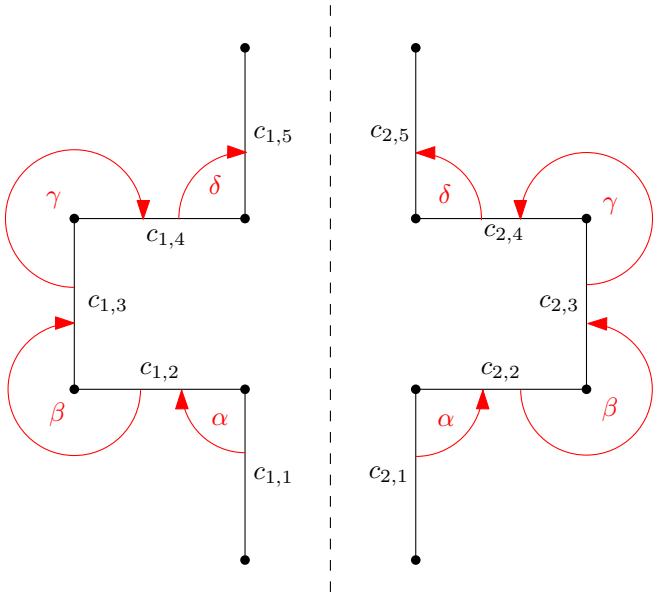

Abb. 4.2: Hier sieht man zwei in entgegengesetzter Drehrichtung verlaufende Polygonzüge, die zueinander spiegelsymmetrisch sind.

4.4 Abweichungstoleranz

Dieses Verfahren ist robust gegen Abweichungen bei Winkel- und Längen-Vergleichen zwischen Urbild und Bild eines Vergleichs, da nach Aufbau des reduzierten Vergleichsgraphen keine geometrischen Eigenschaften im Verfahren ausschlaggebend sind. Beispielsweise könnte man einen Vergleich $v = (c_1, c_2)$ mit ℓ_1 als Länge von c_1 und ℓ_2 als Länge von c_2 im Vergleichsgraph berücksichtigen, wenn

$$\frac{\ell_1 - \ell_2}{\ell_1 + \ell_2} \leq .05$$

gilt, also beide Längen nicht mehr als 5% von deren Mittelwert abweichen. Ähnlich kann man mit Winkeln arbeiten.

4.5 Spiegelsymmetrie

Abbildung 4.2 zeigt ein Teilstück einer Vereinfachung, welches spiegelsymmetrisch zu einem Teilstück einer anderen Vereinfachung ist. Im reduzierten Vergleichsgraph für das Startpaar $(c_{1,1}, c_{2,1})$ wäre diese Symmetrie ein Pfad, der von Algorithmus 2 zurückgegeben wird. Hierbei ist zu beachten, dass P_2 in entgegengesetzter Drehrichtung zu P_1 durchlaufen wird. Liegen P_1 und P_2 in gleicher Drehrichtung vor, so genügt es, G_2 in entgegengesetzter Drehrichtung zu durchlaufen. Einen entsprechenden reduzierten Vergleichsgraph liefert Algorithmus 1, wenn dort in Zeile 15 *Nachfolger* durch *Vorgänger* ersetzt wird.

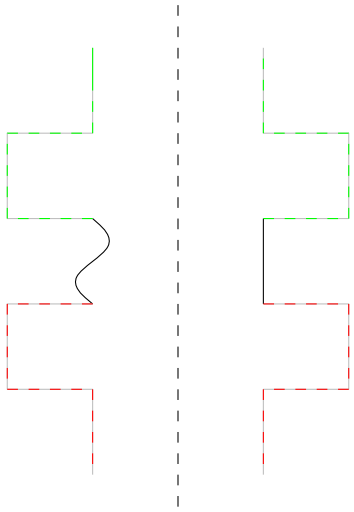

Abb. 4.3: Mehrere spiegelsymmetrische Teilstücke.

4.5.1 Finden von Spiegelsymmetrien

Formuliert man das Problem des längsten gemeinsamen Teilstücks zweier Vereinfachungen so um, dass P_2 bzw. G_2 in zu G_1 entgegengesetzter Drehrichtung durchlaufen wird, muss dessen Lösung nicht unbedingt spiegelsymmetrisch sein, auch wenn spiegelsymmetrische Teilstücke in den Vereinfachungen existieren. Zudem können Spiegelsymmetrien zwischen mehreren Paaren von Teilstücken bestehen, wie in Abbildung 4.3 angedeutet ist. Somit wird nicht ein längster Pfad gesucht, sondern alle Pfade, die Spiegelsymmetrien enthalten. Die Aggregation von spiegelsymmetrischen Teilstücken wird von Haunert [Hau12] behandelt.

Problem 2. Gegeben sind zwei Polygonringe P_1 und P_2, die als Folge von Kanten in gleicher Drehrichtung vorliegen. Gesucht sind alle längsten Teilstücke in Vereinfachungen von P_1, die zu einem Teilstück einer Vereinfachung von P_2 spiegelsymmetrisch sind.

Wie bei Problem 1 kann die Länge z.B. die Summe der euklidischen Abständen zwischen allen konsekutiven Ecken der Lösung sein. Ebenso kann die Länge der Lösung anhand der Anzahl deren Ecken gemessen werden.

Algorithmus 4 berechnet die in Problem 2 gesuchten Pfade. Analog wie beim Suchen des längsten gemeinsamen Polygonzugs beträgt die Laufzeit schlimmstenfalls $O((n{\cdot}m)^{12})$ Zeit, wobei n die Anzahl der Kanten von P_1 und m die Anzahl der Kanten von P_2 ist.

Algorithmus 4: Alle spiegelsymmetrischen Teilstücke

Eingabe : Kombinationsgraphen $G_1 = (V_1, E_1)$ und $G_2 = (V_2, E_2)$ zweier Polygonringe P_1 und P_2

Ausgabe : Alle spiegelsymmetrischen Teilstücke

$ignored = \emptyset$

$resultset = \emptyset$

foreach $v_0 \in E_1 \times E_2$ **do**

  if $v_0 \in ignored$ **then**

    continue

  G'_V = reduzierter Vergleichsgraph für v_0 mit entgegengesetzter Drehrichtung von G_1 und G_2

  p = längster Pfad in G'_V

  if p beinhaltet Spiegelsymmetrie **then**

    $resultset = resultset \cup \{p\}$

  $ignored = ignored \cup p$

return $resultset$

5 Erkennung von Symmetrien in einem Umriss

In diesem Kapitel geht es um Symmetrie als Eigenschaft eines Umrisses, während es in Kapitel 4 um Symmetrie als Beziehung zwischen Paaren von Umrissen ging. Symmetrie als Eigenschaft eines Umrisses ist eine Beziehung des Umrisses zu sich selbst. Daher wird auf die Methodik aus Kapitel 4 aufgebaut.

5.1 Rotationssymmetrie

Ein Polygonring P ist rotationssymmetrisch, wenn der Polygonring, der durch Drehung von P um einen nicht-trivialen Winkel mit dem Schwerpunkt von P als Drehzentrum entsteht, gleich P ist. Der kleinste nicht-triviale Winkel α_0, für den eine Rotationssymmetrie vorliegt, bestimmt die *Ordnung*

$$o = \frac{2\pi}{\alpha_0}$$

der Symmetrie. Man nennt P dann *o-rotationssymmetrisch*. Abbildung 5.1 zeigt einen 3-rotationssymmetrischen Polygonring.

Problem 3. Gegeben ist ein Polygonring P als Folge von Kanten. Gesucht sind alle rotationssymmetrischen Vereinfachungen von P sowie deren Ordnung.

5.1.1 Rotationssymmetrie als Beziehung einer Vereinfachung zu sich selbst

Wie in Abbildung 5.2 demonstriert, hat eine rotationssymmetrische Vereinfachung eines Polygonrings P die Eigenschaft, dass sie einem Kreis im Vergleichsgraph für P mit sich selbst entspricht. In diesem Beispiel ist (c_1, c_3) der Startvergleich. So ein Kreis entspricht einer Drehabbildung, deren Bild hier gleich deren Urbild ist.

5.1.2 Die Mächtigkeit der Lösungsmenge

Die Mächtigkeit der Lösungsmenge von Problem 3 kann exponentiell mit der Anzahl der Kanten von P wachsen: Unterteilt man eine Kante in einem rotationssymmetrischen Polygon in zwei kollineare Kanten, kann jede dieser beiden neuen Kanten per Abkürzung übersprungen werden, so dass sich die Anzahl der rotationssymmetrischen Vereinfachungen verdoppelt. Unterteilt man so in einem n-regulären Polygon jede Kante einmal in zwei Kanten, erhält man schließlich $2^{\frac{n}{2}} = \sqrt{2}^n$ rotationssymmetrische Vereinfachungen.

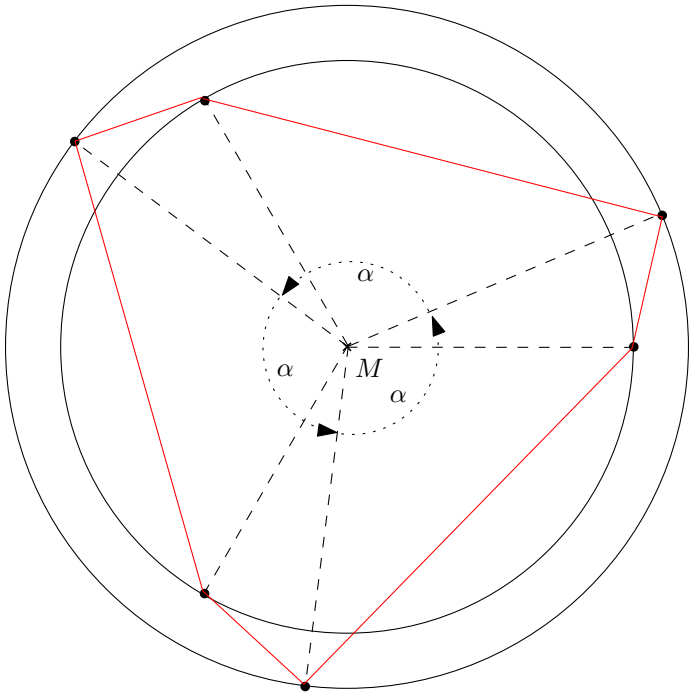

Abb. 5.1: Ein 3-rotationssymmetrischer Polygonring.

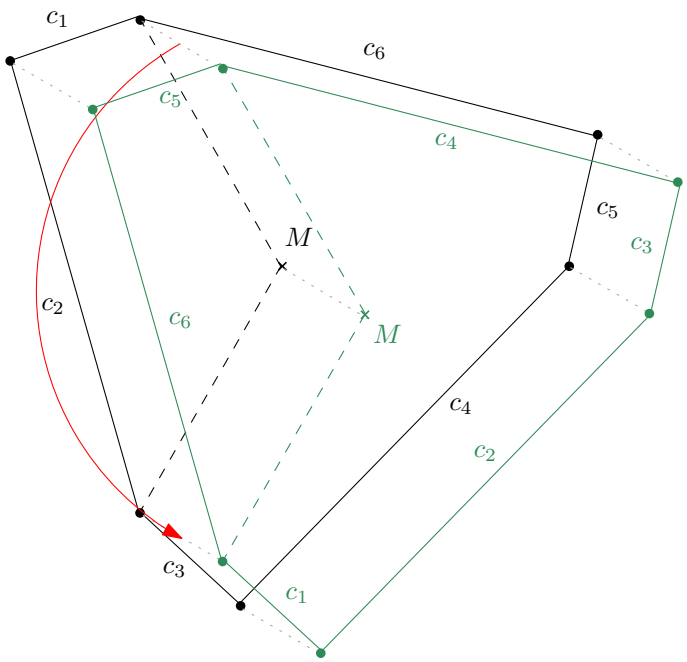

Abb. 5.2: Die abgebildete Drehung des Polygons P aus Abbildung 5.1 entspricht einem Kreis im Vergleichsgraph von P und P mit Startvergleich (c_1, c_3).

5.2 Eine Heuristik zur Erkennung von Rotationssymmetrie

Das im Folgenden diskutierte heuristische Polynomialzeit-Verfahren „Pentagon“ wurde im praktischen Teil dieser Arbeit implementiert und ist auf der beiliegenden CD als Java-Programm-Bibliothek mit Quellcode und Java-Doc enthalten.

5.2.1 Der Rundvergleichsgraph

Eine vollständige rotationssymmetrische Vereinfachung kann im Vergleichsgraph nur als Urbild (oder Bild) eines Kreises gefunden werden. Daher ist es sinnvoll, Knoten, die nicht auf einem Kreis liegen, von Anfang an auszuschließen.

Definition 5.1. Sei $G_V = (V_V, E_V)$ ein reduzierter Vergleichsgraph für den Startvergleich v_0. Der *Rundvergleichsgraph* $G_R = (V_R, E_R)$ existiert als Teilgraph von G_V dann, wenn v_0 auf einem Kreis in G_R liegt. Die Knoten V_R des Rundvergleichsgraph sind die Knoten $v \in V_V$, die in G_V auf einem Kreis liegen. Entsprechendes gilt für die Kanten von E_R.

Der in Abbildung 4.1 dargestellte Vergleichsgraph ist ein Beispiel für einen Rundvergleichsgraph.

Algorithmus 5: Erzeugung des Rundvergleichsgraphen aus einem durch Algorithmus 1 gewonnenen reduzierten Vergleichsgraphen

Eingabe : Reduzierter Vergleichsgraph (V_V, E_V); Startvergleich v_0; Kantenanzahl n des Eingabe-Polygons
Ausgabe : Rundvergleichsgraph für Startvergleich v_0

```
1  if {(v1, v2) ∈ E_V | v2 = v0} = ∅ then
2      return NIL
3  E_R = ∅
4  for 1 ≤ i ≤ n do
5      V_i = ∅
6  e = Kante, zu welcher das Urbild von v0 zugeordnet wird
7  topology = Index von e
8  V_topology = {v0}
9  for topology − 1 + n ≥ i > topology − 1 do
10     foreach v2 ∈ V_(i mod n)+1 do
11         foreach v1 : (v1, v2) ∈ E_V do
12             E_R = E_R ∪ {(v1, v2)}
13             target = Index der Kante, zu welcher das Urbild von v1 zugeordnet wird
14             V_target = V_target ∪ {v1}
15 return (⋃_{i=1}^{n} V_i, E_R)
```

Algorithmus 5 liefert den Rundvergleichsgraph für einen aus Algorithmus 1 gewonnenen reduzierten Vergleichsgraph, wenn dieser existiert. Hier wird, wie dies bei der

Pfadsuche im reduzierten Vergleichsgraph auch der Fall ist, für jeden Vergleich in topologischer Ordnung jede ausgehende Kante betrachtet, nur diesmal umgekehrt. Ausgehend vom Startvergleich werden die Vergleiche in umgekehrter topologischer Ordnung durchlaufen, und es werden deren eingehende Kanten betrachtet. Da jeder Vergleich v beim Aufbau des Graphen gleichzeitig mit eingehender Kante aufgenommen wurde, existiert immer ein rückwärts laufender Pfad von v zum Startvergleich. Da v nur betrachtet wird, wenn ein rückwärts laufender Pfad vom Startvergleich zu v existiert, liegt v auf gefordertem Kreis. Die Laufzeit entspricht dabei der des Aufbaus des reduzierten Vergleichsgraphen und ist somit $O((n \cdot n)^8)$, wobei n die Anzahl der Kanten des Eingabe-Polygons ist. Zusammengefasst ergibt sich also eine Laufzeit von $O(n^{16})$.

5.2.2 Der Symmetriegraph

Abbildung 5.3 zeigt für das dort abgebildete Polygon P einen Teil dessen Vergleichsgraph G_R für das Startpaar (lab, cde). In G_R existiert neben einem Kreis, dessen Urbild rotationssymmetrisch ist, ein anderer Kreis, dessen Urbild nicht rotationssymmetrisch ist. Um rotationssymmetrische Kreise zu identifizieren, wird bei diesem Ansatz der *Symmetriegraph* aufgestellt.

Definition 5.2. Der *Supersymmetriegraph* $G_S^* = (V_S^*, E_S^*)$ für einen Rundvergleichsgraph $G_R = (V_R, E_R)$ hat die Knotenmenge $V_S^* = V_R$. Ein Paar von Vergleichen $(v_1, v_2) \in V_S^* \times V_S^*$ liegt dann in E_S^*, wenn das Bild von v_1 das Urbild von v_2 ist. Der Symmetriegraph G_S für einen Vergleich v ist die Zusammenhangkomponente in G_S^*, in der v liegt.

In Abbildung 5.4 ist der Symmetriegraph für das Beispiel aus Abbildung 5.3 zu sehen. Wie man sieht, hat ein rotationssymmetrischer Kreis im Rundvergleichsgraph für einen Startvergleich v_0 die Eigenschaft, dass im Symmetriegraph von v_0 ein Kreis existiert, der eine Teilfolge des rotationssymmetrischen Kreises ist. Dies gilt allerdings nur für v_0, wie man an Abbildung 4.1 sehen kann. Dort existiert im Symmetriegraph für Vergleich (c_2, c_5) ein Kreis, der über die Vergleiche (c_5, c_{13}), (c_{13}, c_{15}) und (c_{15}, c_2) geht. Diese liegen aber nicht auf einem Kreis im Vergleichsgraph.

Rotationssymmetrische Startvergleiche

Definition 5.3. Sei G_V ein Rundvergleichsgraph mit Startvergleich v_0, G_S der Symmetriegraph für v_0 und C ein Kreis in G_S. Die Vergleiche, die auf C liegen, heißen *rotationssymmetrische Startvergleiche*.

Dies kommt daher, dass die Pfade eines rotationssymmetrischen Kreises in G_V, die zueinander rotationssymmetrisch sind, mit solchen Vergleichen beginnen. Diese liegen aber nicht unbedingt in topologischer Ordnung vor. Würde man das grüne Polygon aus Abbildung 5.2 nochmals um 120° gegen den Uhrzeigersinn drehen, so entspräche die Drehabbildung einem Kreis im Rundvergleichsgraph für (c_1, c_5). Im Symmetriegraph wäre der Kreis dann $((c_1, c_5), (c_5, c_3), (c_3, c_1), (c_1, c_5))$.

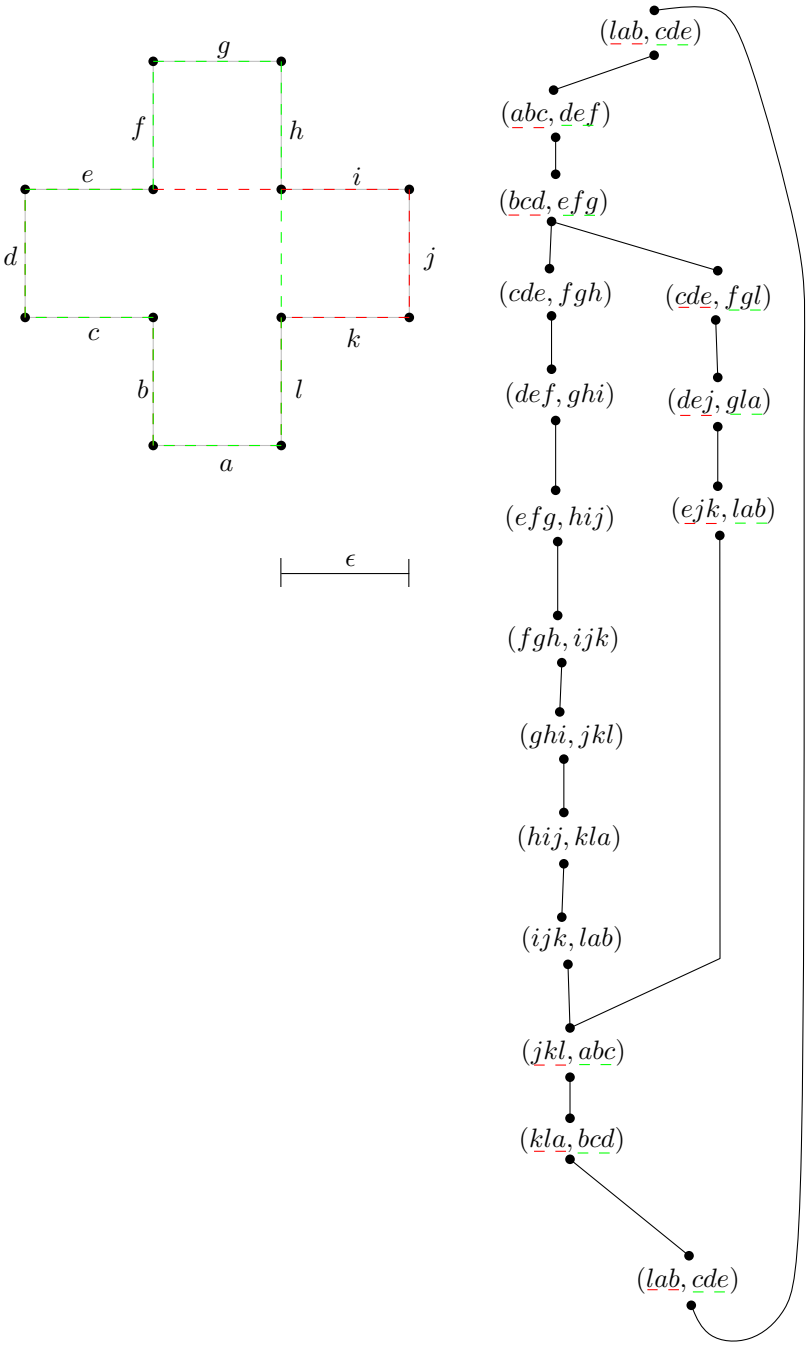

Abb. 5.3: Für das Polygon links gibt es rechts einen Kreis in einem Vergleichsgraph, dessen Urbild rotationssymmetrisch ist, sowie im gleichen Vergleichsgraph einen Kreis, dessen Urbild nicht rotationssymmetrisch ist.

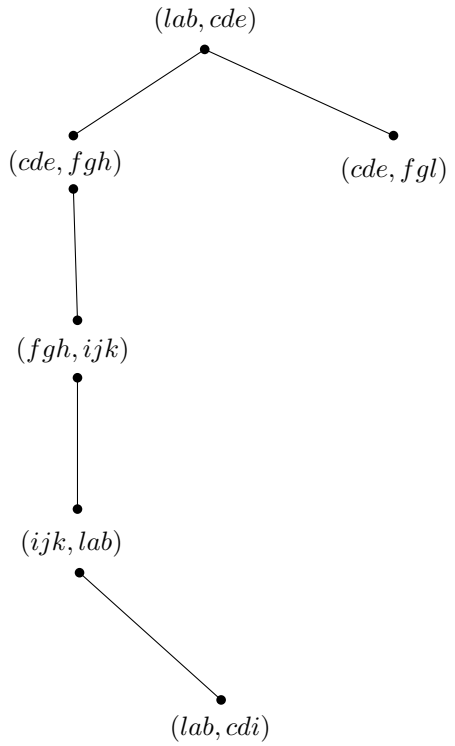

Abb. 5.4: Für den Rundvergleichsgraph und dessen Startvergleich aus Abbildung 5.3 ist hier der Symmetriegraph zu sehen.

Aufbau des Symmetriegraphen

Algorithmus 6: Aufbau des Symmetriegraphen

Eingabe : Rundvergleichsgraph $G_R = (V_R, E_R)$; Vegleich v_S
Ausgabe : Symmetriegraph für v_S
foreach $v \in V_R$ **do**
  $visited(v) =$ **false**
$Q =$ neue FIFO-Warteschlange
put v_S into Q
$visited(v_S) =$ **true**
$E_S = \emptyset$
$V_S = \emptyset$ **while** $Q \neq$ **empty do**
  entnehme v_1 aus Q
  $V_S = V_S \cup \{v_1\}$
  foreach v_2: Urbild von v_2 ist Bild von v_1 **do**
    $E_S = E_S \cup \{(v_1, v_2)\}$
    if $visited(v_2) =$ **true then**
      continue
    put v_2 into Q
    $visited(v_2) =$ **true**
return (V_S, E_S)

Algorithmus 6 zeigt, wie der Symmetriegraph aufgebaut werden kann. Hierbei wird höchstens jeder Knoten aus V_R einmal in die Warteschlange eingereiht, da diese entsprechend markiert und geprüft werden. Für jeden Knoten in der Warteschlange wird jedes Abbild einmal betrachtet. Somit ist die Laufzeit maximal $O(|V_R|^2)$ Zeit, wobei $|V_R|$ für die Kantenanzahl n des Eingabe-Polygons maximal n^8 beträgt.

Pfadsuche im Symmetriegraph

An dieser Stelle des Verfahrens fällt jetzt die erste Entscheidung, die das Verfahren zu einer Heuristik macht, nämlich die Wahl der rotationssymmetrischen Startvergleiche. Abbildung 5.5 zeigt ein Beispiel, bei welchem man sich für die falschen rotationssymmetrischen Startvergleiche entscheiden kann. Im Rundvergleichsgraph für den Startvergleich $(abc, k\ell m)$ kommt die nicht rotationssymmetrische Drehung unten als Kreis ebenso wie eine Drehung der oben eingezeichneten rotationssymmetrischen Vereinfachung vor. Die nicht rotationssymmetrische Drehung hat dabei auch einen Kreis im Symmetriegraph für den Startvergleich, nämlich $(abc, k\ell m), (k\ell m, nop), (nop, xyz), (xyz, abc)$. Dennoch wird jetzt nach dem Kosten-Kriterium, wie es auch schon im Abschnitt 35 diskutiert wurde, ein Pfad im Symmetriegraph für den Startvergleich des Rundvergleichsgraphen gesucht.

Dabei ist folgendes zu beachten: wie in Abbildung 5.4 zu sehen ist, geht jeder Kreis durch den Vergleich v, für den der Symmetriegraph aufgestellt worden ist. Bei Berück-

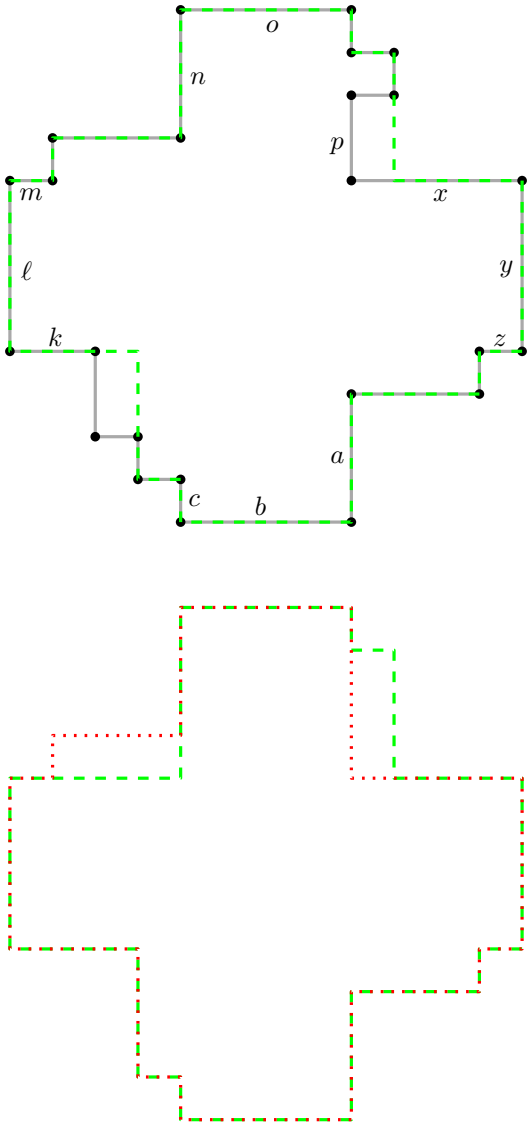

Abb. 5.5: Oben ist für das originale Polygon die einzige rotationssymmetrische Vereinfachung grün gestrichelt eingezeichnet. Unten ist eine Drehung der grün gestrichelten Vereinfachung auf die rot gepunktete Vereinfachung eingezeichnet.

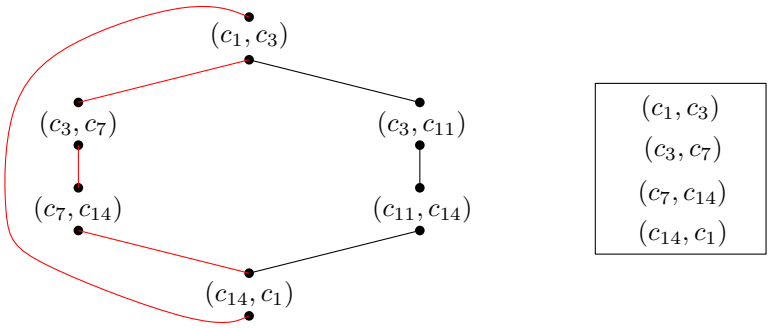

Abb. 5.6: Für den Rundvergleichsgraph aus Abbildung 4.1 ist hier der links Symmetriegraph für dessen Startvergleich (c_1, c_3) zu sehen. Für einen Kreis sind dessen Kanten rot gefärbt. Rechts ist die resultierende Rotationsklasse dargestellt.

sichtigung von Vergleichs-Toleranzen kann der Symmetriegraph trotz Auftrennung an v zyklisch sein. Daher müsste man eigentlich schon einen Algorithmus wie Dijkstra verwenden, um die kostengünstigsten rotationssymmetrischen Startvergleiche auszuwählen, vgl. etwa Cormen et al. [CLR04]. Hier wird dieser Pfad aber „gierig“ ausgewählt. Beim Aufbau des reduzierten Vergleichsgraphen bietet sich die Gelegenheit, für jeden Vergleich die Gesamtkosten für den günstigsten Pfad vom Startvergleich her zu speichern. Beim Reduzieren zum Rundvergleichsgraph werden dann noch die Gesamtkosten des günstigsten Pfads zum Startvergleich hin gespeichert. So stehen die Gesamtkosten des günstigsten Kreises im Rundvergleichsgraph zur Verfügung, auf dem der Vergleich liegt. Aufgrund dieser Kosten wird sukzessive die Sequenz der rotationssymmetrischen Startvergleiche ausgewählt. Landet man dabei auf einem Zyklus, der nicht durch v geht, so wird angenommen, dass für einen anderen Rundvergleichsgraph eine passendere rotationssymmetrische Vereinfachung existiert. Somit kann die Pfadsuche im Symmetriegraph in $O(|V_S|)$ Zeit erfolgen, wobei $|V_S|$ im schlimmsten Fall in $O(|V_R|)$ liegt, also $O(n^8)$ in der Anzahl n der Kanten des Eingabe-Polygons.

5.2.3 Rotationsklassen

Wir fassen nun Kreise in Symmetriegraphen zu Klassen zusammen.

Definition 5.4. Sei $G_R = (V_R, E_R)$ ein Rundvergleichsgraph. Ein einfacher Kreis

$$(v_1, \ldots, v_o, v_1)$$

im Symmetriegraph eines Vergleiches $v \in V_R$ ist eine *Rotationsklasse*, wobei $v_1 = v$ gefordert wird.

In Abbildung 5.6 ist ein Beispiel für eine Rotationsklasse zu sehen.

Rotationsklassengraph

Definition 5.5. Sei $G_R = (V_R, E_R)$ ein Rundvergleichsgraph. Sei V_K^* die Menge aller Rotationsklassen von G_R und E_K^* die Menge aller Paare $(v_1, v_2) \in V_K^* \times V_K^*$, so dass E_R

in $v_1 \cup v_2$ ein perfektes Matching festlegt, d.h. jedes Element von v_1 mit genau einem Element von v_2 verbunden ist. Der Graph $G_K^* = (V_K^*, E_K^*)$ ist der *Rotationsklassengraph.*

Aus einem Kreis im Rotationsklassengraph lässt sich ein rotationssymmetrischer Kreis im Rundvergleichsgraph ableiten.

Satz 5.6. *Die Anzahl der Rotationsklassen kann exponentiell mit der Anzahl der Kanten des Eingabe-Polygons wachsen.*

Beweis. In Abschnitt 5.1.2 wird eine Klasse von Polygonen definiert, die $\sqrt{2}^n$ viele rotationssymmetrische Vereinfachungen haben. Ist n dabei eine Primzahl, so bildet jede dieser Vereinfachungen eine eigene Rotationsklasse. □

Topologische Ordnung

Für den Rotationsklassengraph wird die topologische Ordnung der ersten Elemente der Rotationsklassen verwendet.

5.2.4 Der Symmetrische-Kanten-Graph

Die Rotationsklasse für den Startvergleich des Rundvergleichsgraph wurde wie bereits geschildert durch Pfadsuche im Symmetriegraph aufgestellt. Um eine Kante im Rotationsklassengraph ausgehend von der Startvergleichs-Klasse zu finden wird der *Symmetrische-Kanten-Graph* aufgestellt.

Definition 5.7. Sei $G_R = (V_R, E_R)$ ein Rundvergleichsgraph, $(v_1, v_2) \in E_R$ eine Kante in diesem, und v_K eine Rotationsklasse, in welcher v_1 enthalten ist. Die Knotenmenge V des Symmetrische-Kanten-Graphen entsteht iterativ für $1 \leq i < |v_K|$:

$$V^1 = \{(v_1, v_2)\}$$

$$V^{(i \bmod |v_K|-1)+1} = \{(v_3, v_4) \in E_R \mid v_3 \in v_K \wedge \exists (v_5, v_6) \in V^i \colon \text{Bild}(v_6) = \text{Urbild}(v_4)\}.$$

Am Ende ist $V = \bigcup_{i=1}^{|v_K|} V^i$. Die Kantenmenge E besteht aus Paaren $((v_3, v_4), (v_5, v_6))$, für die $\text{Bild}(v_4) = \text{Urbild}(v_6)$ gilt.

Den Symmetrische-Kanten-Graph für eine Kante $(v_1, v_2) \in E_R$ kann man sich wie den Symmetriegraph für v_2 vorstellen, für den zusätzlich die Bedingung gilt, dass jeder Knoten das Ziel einer Kante in E_R ist, deren Quelle in v_K liegt. Oder auch wie den Symmetriegraph für v_2, der die Menge der Rotationsklassen spezifiziert, die eine eingehende Kante von v_K haben. Abbildung 5.7 zeigt aufbauend auf das Beispiel aus Abbildung 5.6 einen Symmetrische-Kanten-Graph.

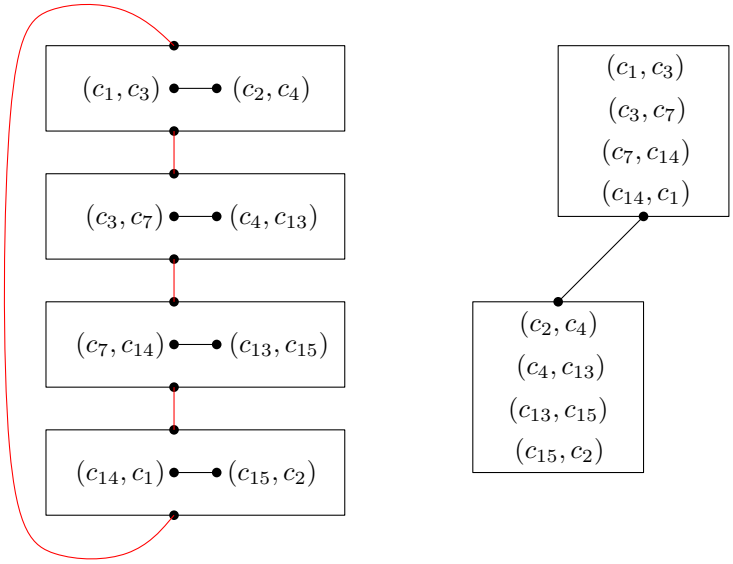

Abb. 5.7: Ausgehend von der in Abbildung 5.6 dargestellten Rotationsklasse im Rundvergleichsgraph aus Abbildung 4.1 und für Kante $((c_1, c_3), (c_2, c_4))$ ist hier links der Symmetrische-Kanten-Graph und rechts die resultierende Kante im Rotationsklassengraph zu sehen.

Aufbau des Symmetrische-Kanten-Graphen

Ausgehend davon, dass v_K als Folge von Vergleichen vorliegt, die in der gleichen Reihenfolge besteht, wie der Kreis im Symmetriegraph, muss einfach nur durch diese Folge durch-iteriert werden, wie dies in der äußeren Schleife (Zeilen 5 bis 10) in Algorithmus 7 passiert. Für jeden dieser Vergleiche wird jede ausgehende Kante einmal mit jeder ausgehenden Kante des folgenden Vergleiches in v_K verglichen. Dies ergibt maximal $O(|V_R| \cdot |E_R|^2)$ Zeit.

Kreise im Symmetrische-Kanten-Graph

Ein Kreis im *Symmetrische-Kanten-Graph* bestimmt für eine Kante $(v_1, v_2) \in E_R$ und eine Rotationsklasse v_K, in welcher v_1 liegt, eine Kante im Rotationsklassengraph, die (v_1, v_2) in E_R repräsentiert. Alle anderen Rotationsklassen, die das auch könnten, werden außer Acht gelassen. Genau wie beim Symmetriegraph wird hier „gierig“ der kostengünstigste Kreis ausgewählt. Anders als beim Symmetriegraph gibt es hier keine Kreise, die nicht durch (v_1, v_2) gehen, da Bindungen zu v_K bestehen. Daher müsste man hier nicht auf eine Pfad-Suche wie Dijkstra zurückgreifen, wenn man aufgrund anderer Kosten-Definitionen den exakt günstigsten Pfad erhalten möchte. Wie dem auch sei, ein Beispiel, bei welchem die „gierige“ Selektion der Rotationsklassen zu einem nicht exakten Ergebnis führt, ist bis zur Abgabe der Arbeit nicht bekannt geworden.

Algorithmus 7: Aufbau des Symmetrische-Kanten-Graphen

Eingabe : Rundvergleichsgraph $G_R = (V_R, E_R)$; Kante $(v_1, v_2) \in E_R$; Rotationsklasse v_K, die v_1 enthält
Ausgabe : Symmetrische-Kanten-Graph

```
1  for 2 ≤ i < |v_K| do
2      V^i = ∅
3  V^1 = {(v_1, v_2)}
4  E = ∅
5  for 1 ≤ i < |v_K| do
6      v_3 = (i + 1)-tes Element in v_K
7      foreach (v_5, v_6) ∈ V^i do
8          foreach (v_3, v_4) ∈ E_R: Urbild(v_4) = Bild(v_6) do
9              V^((i mod |v_K|−1)+1) = V^((i mod |v_K|−1)+1) ∪ {(v_3, v_4)}
10             E = E ∪ {((v_5, v_6), (v_3, v_4))}
```

5.2.5 Aufbau des Rotationsklassengraphen

Ausgehend vom Startvergleich v_0 des Rundvergleichsgraphen wird die erste Rotationsklasse durch den Symmetriegraph von v_0 gewonnen. Dies passiert in Zeile 3 von Algorithmus 8. Nach dem üblichen Muster wird jede Rotationsklasse in topologischer Ordnung besucht und für deren erstes Element in den Zeilen 11 bis 19 alle ausgehenden Kanten (v_1, v_2) betrachtet. Mit Hilfe eines Kreises im Symmetrische-Kanten-Graph wird eine (v_1, v_2) repräsentierende Kante im Rotationsklassengraph ausgewählt und übernommen. Wenn v_2 der Startvergleich ist, so wird der Kreis geschlossen. Das Besuchen aller Kanten in E_R benötigt dabei $O(|E_R|)$ Zeit, das Bestimmen von v_K^2 benötigt $O(|V_R| \cdot |E_R|^2)$ Zeit für den Aufbau des Symmetrische-Kanten-Graphen und $O(|E_R|)$ Zeit für das Finden des kostengünstigsten Pfades. Somit benötigt der Aufbau des Rotationsklassengraphen $O(|V_R| \cdot |E_R|^3)$ Zeit.

Beispiel

Das Polygon aus Abbildung 3.7 besitzt offenbar eine 4-rotationssymmetrische Vereinfachung, die für den Startvergleich (c_1, c_3) als Kreis im Rundvergleichsgraph existiert. Der entsprechende Rundvergleichsgraph ist in Abbildung 4.1 zu sehen. Als Rotationsklasse v_K^0 kann das Beispiel aus Abbildung 5.6 verwendet werden. Die ausgehenden Kanten von v_0 sind $((c_1, c_3), (c_2, c_4))$ und $((c_1, c_3), (c_2, c_5))$. Für erstere ergibt sich die in Abbildung 5.7 gezeigte Kante für den Rotationsklassengraph. Für zweitere erweitert sich der Rotationsklassengraph wie in Abbildung 5.8 dargestellt. Die nächsten betrachteten Kanten sind $((c_2, c_4), (c_3, c_7))$ und $((c_2, c_5), (c_3, c_{11}))$. Beide erfüllen die Bedingung von Zeile 13 im Algorithmus, erstere zudem die Bedingung von Zeile 14. Somit ergibt sich der Rotationsklassengraph nach Abbildung 5.9. Dieser hat einen Kreis und wird damit beachtet.

Algorithmus 8: Aufbau des Rotationsklassengraphen

Eingabe : Rundvergleichsgraph G_R; Startvergleich v_0; Kanten-Anzahl n des Eingabe Polygons

Ausgabe : Zyklischer Rotationsklassengraph oder **NIL**

```
1  for 1 ≤ i ≤ n do
2      V_i = ∅
3  v_K^0 = Rotationsklasse für v_0
4  topology = Index der Kante, zu welcher das Urbild von v_0 zugeordnet wird
5  V_topology = V_topology ∪ {v_K^0}
6  E = ∅
7  e = Kante, zu welcher das Bild von v_0 zugeordnet wird
8  v_x = zweites Element in v_K nach topologischer Ordnung
9  for topology − 1 ≤ i < topology − 1 + n do
10     foreach v_K^1 ∈ V_(i mod n)+1 do
11         v_1 = erstes Element von v_K^1
12         foreach v_2 : (v_1, v_2) ∈ E_R do
13             if Urbild(v_2) ∈ Überpringer(e) then
14                 if v_2 = v_x then
15                     E = E ∪ {(v_K^1, v_K^0)}
16                 continue
17             v_K^2 = Rotationklasse für v_2 mit (v_K^1, v_K^2) ∈ E_K^*
18             t = Index der Kante zu welcher das Urbild von v_2 zugeordnet wird
19             V_t = V_t ∪ {v_K^2}
20             E = E ∪ {(v_K^1, v_K^2)}
21 V = ⋃_{i=1}^{n} V_i
22 if ∄ v_K^1 ∈ V : (v_K^1, v_K^0) ∈ E then
23     return NIL
24 return (V, E)
```

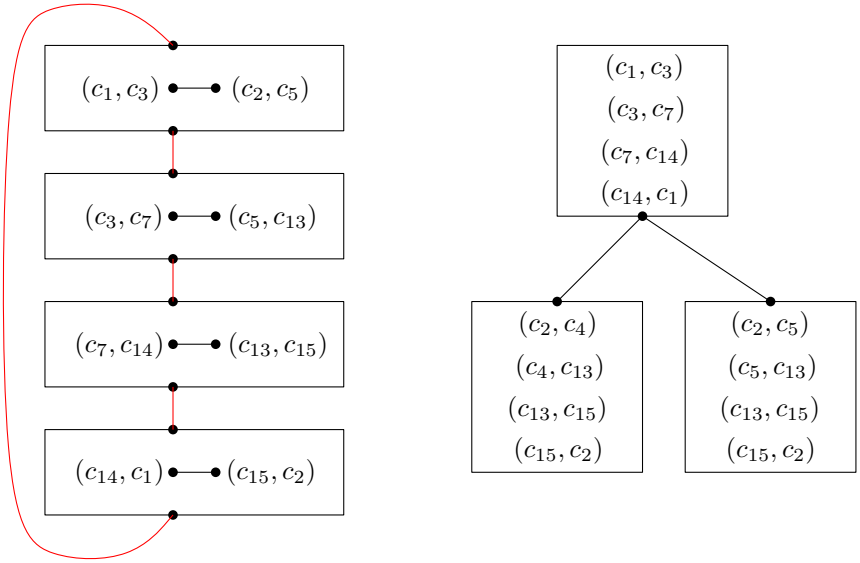

Abb. 5.8: Analog zu Abbildung 5.7 ergibt sich hier für Kante $((c_1, c_3), (c_2, c_5))$ links der Symmetrische-Kanten-Graph und rechts die resultierende Kante im Rotationsklassengraph.

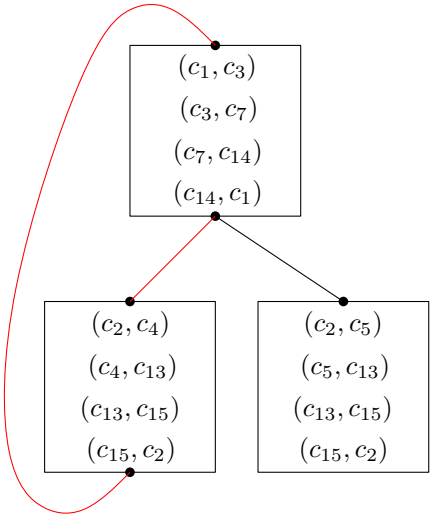

Abb. 5.9: Folgt man der Beispiel-Kette nach Abbildung 5.8, so ergibt sich der hier dargestellte Rotationsklassengraph.

5.2.6 Synthese einer zusammengesetzten rotationssymmetrischen Vereinfachung

Definition 5.8. Sei P ein einfacher Polygonring und G dessen Kombinationsgraph. Eine *zusammengesetzte o-rotationssymmetrische Vereinfachung* von P ist eine Folge

$$\langle c_0 \rightsquigarrow c'_0, c_1 \rightsquigarrow c'_1, \ldots, c_{o-1} \rightsquigarrow c'_{o-1}, c_0 \rangle$$

von Kombinationen aus G mit der Eigenschaft, dass für ein $1 \leq k < o$ und für alle $0 \leq i < o$ der Pfad

$$\langle c_i, c_{i+k \bmod o}) \rightsquigarrow (c'_i, c'_{i+k \bmod o} \rangle$$

ein Pfad im Rundvergleichsgraph für Startvergleich (c_0, c_k) ist und zudem c_1 in G ein Nachfolger von c'_0 ist. Die Teilfolge

$$\langle c_0, \ldots, c_{o-1} \rangle$$

liegt dabei in topologischer Ordnung vor.

Sei $V = (v_K, \ldots, v'_K, v_K)$ ein Kreis im Rotationsklassengraph. Die Teilpfade $\langle c_i \rightsquigarrow c'_i \rangle$ können aus den Urbildern der i-ten Elemente von V gewonnen werden und müssen dann nur noch topologisch geordnet werden. In Abbildung 5.9 ist ein Kreis rot markiert. Für die ersten Elemente der Rotationsklassen bekommt man die Folge $\langle c_1, c_2 \rangle$, für die zweiten Elemente $\langle c_3, c_4 \rangle$, dann $\langle c_7, c_{13} \rangle$ und schließlich $\langle c_{14}, c_{15} \rangle$. Ein Konkatenation dieser ist eine rotationssymmetrische Vereinfachung des Polygons aus Abbildung 3.7. Der Kreis V kann nach dem üblichen Muster in $O(|E_R|)$ Zeit berechnet werden, die Synthese der Vereinfachung in $O(n)$ Zeit, wobei n die Anzahl der Kanten des Eingabe-Polygons ist.

5.2.7 Wahl der Startvergleiche für den Rundvergleichsgraph

Ziel ist es, jeden Kreis im Kombinationsgraph mit jedem anderen zu Vergleichen, um so eine rotationssymmetrische Vereinfachung zu finden. Hierbei genügt es, die Kreise einmal an den Überspringern der ersten Kante des Eingabe-Polygons festzuhalten. Die zu vergleichenden Kreise müssen an jeder Kombination einmal festgehalten werden. Sei $G = (V, E)$ also der Kombinationsgraph und e die erste Kante des Eingabe-Polygons. Die Menge der Startvergleiche ist formal

$$\{(c_1, c_2) \in E \times E \mid c_1 \in \text{Überspringer}(e)\}.$$

Diese Menge umfasst im schlimmsten Fall $O(n^8)$ Elemente, wobei n die Anzahl der Kanten im Eingabe-Polygon ist. Somit liegt die Gesamtlaufzeit in $O(n^8 \cdot |V_R| \cdot |E_R|^3) = O(n^{68})$. Dies ist eine sehr grobe Abschätzung, ebenso wie in Abschnitt 4.3 gesehen ergeben sich in der Praxis geringere Laufzeiten. Im einem idealen Fall gibt es so viele Kombinationen wie Kanten in Eingabe-Polygon, dadurch verringert sich die Laufzeit schon auf $O(n \cdot |V_R| \cdot |E_R|^3)$. Ein Vergleichsgraph hat dann genauso viele Kanten wie Vergleiche, somit beträgt die Laufzeit dann nur noch $O(n \cdot |V_R|^4)$. Letztlich kommt dann jeder Vergleich nur noch höchstens einmal in einem Symmetriegraph oder in einem Symmetrische-Kanten-Graph vor, dadurch verringert sich die Laufzeit abermals zu $O(n \cdot |V_R|^2)$. Dabei gibt es dann so viele Vergleiche wie Kombinationen, also ist die Laufzeit $O(n^3)$.

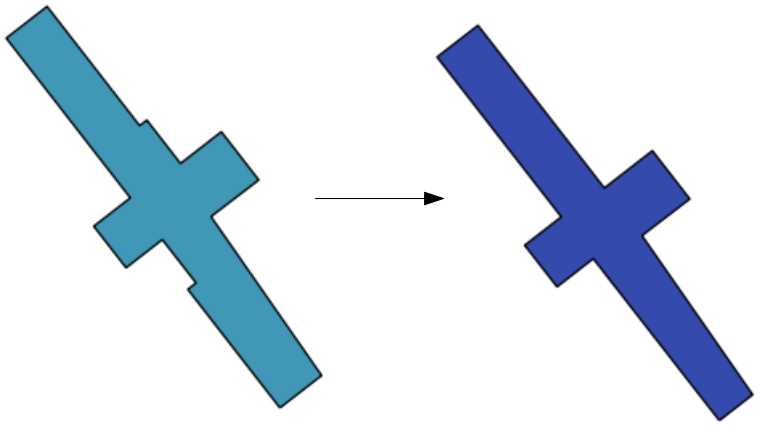

Abb. 5.10: Ein weiteres Beispiel für eine rotationssymmetrische Vereinfachung.

5.2.8 Anwendung

Der Testdatensatz beinhaltet 4553 Gebäude aus dem Stadtgebiet von Boston. In einem Testlauf wurden dabei über 16.000 rotationssymmetrische Vereinfachungen von Außenumrissen oder Löchern gefunden. Unter diesen wurde der originale Ring durch die Vereinfachung mit dem geringsten Hausdorff-Abstand ersetzt, ohne dabei Rücksicht auf die rotationssymmetrische Ordnung der Vereinfachung zu nehmen. Als Toleranz für Abkürzungen wurde 5 Meter gewählt. Ein Vergleich (c_1, c_2) mit l_1 und l_2 als Längen von c_1 bzw. c_2 und α_1 und α_2 als Winkel zu deren Vorgänger-Kombinationen wurde zugelassen, wenn

$$\frac{|l_1 - l_2|}{l_1 + l_2} \leq .15 \text{ und}$$

$$\frac{|\alpha_1 - \alpha_2|}{2\pi} \leq .01$$

erfüllt war. Die Synthese der Vereinfachung erfolgte nach dem Kriterium der billigsten Vergleichskosten. Die Berechnungszeit auf einem 32-Bit System mit 4 GB Arbeitsspeicher und einem Intel i7 Prozessor bei 2.8 GHz dauerte knapp 20 Sekunden. Aus diesem Testlauf stammt das in Abbildung 1.2 gezeigte Beispiel. Ein weiteres Beispiel aus diesem Testlauf ist in Abbildung 5.10 zu sehen.

In einem anderen Testlauf wurde ein Gebäude mit 30 Kanten ausgewählt. Die Parameter wurden nicht verändert. Es wurden 163 zyklische Rotationsklassengraphen gefunden. Der Lauf dauerte ca. 450 ms. Abbildung 5.11 zeigt exemplarisch für jede gefundene Ordnung ein Beispiel einer rotationssymmetrischen Vereinfachung.

5.3 Exakte Ansätze zur Erkennung von Rotationssymmetrie

Um Problem 3 zufriedenstellend zu lösen, wäre ein exakter Polynomialzeit-Algorithmus, der robust gegen Vergleichs-Toleranzen funktioniert, von Vorteil. Dessen Existenz konnte

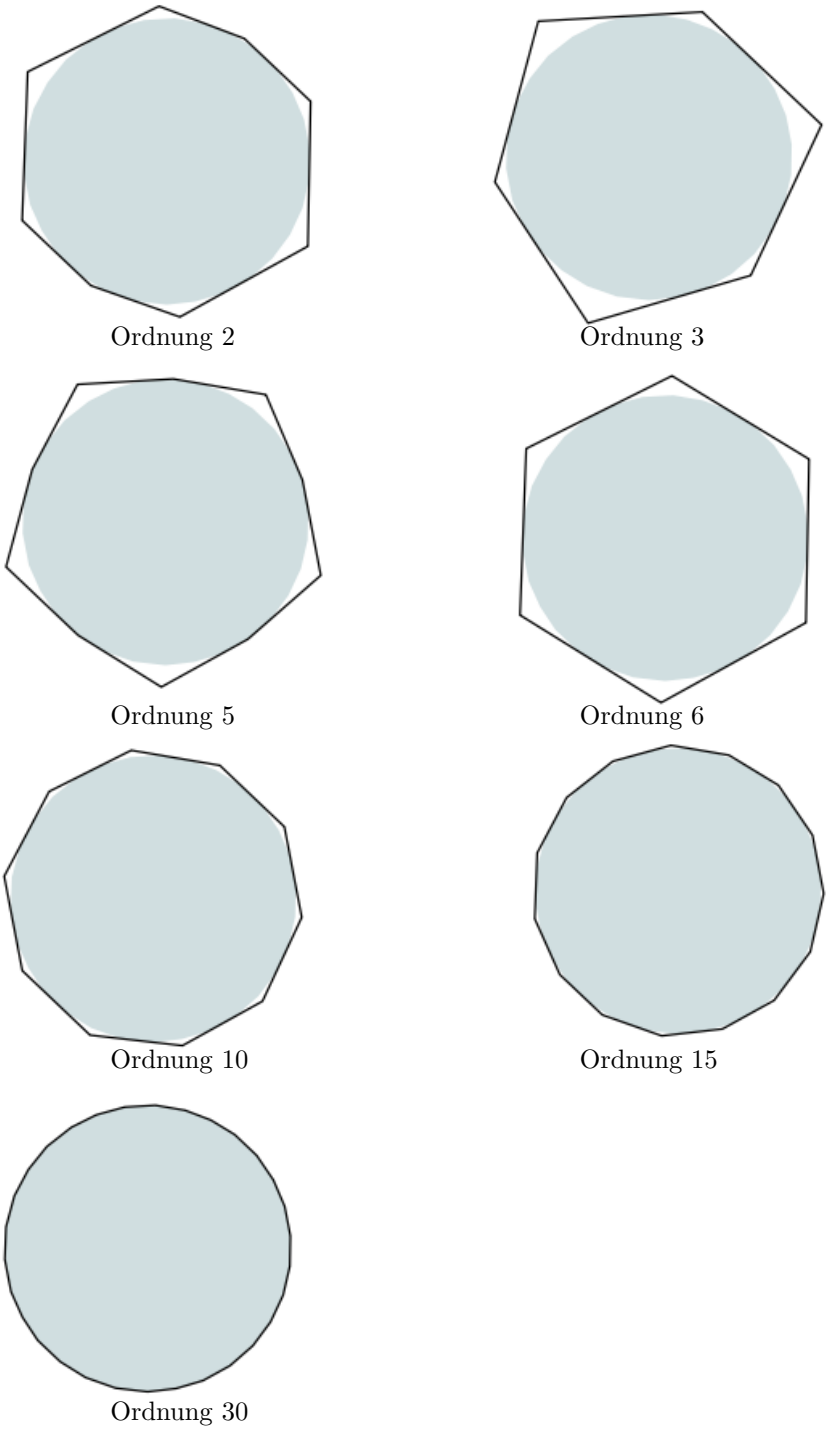

Abb. 5.11: Für ein Gebäude mit 30 Kanten wurde aus einem Testlauf mit der Heuristik exemplarisch für jede gefundene Ordnung ein Beispiel zufällig ausgewählt.

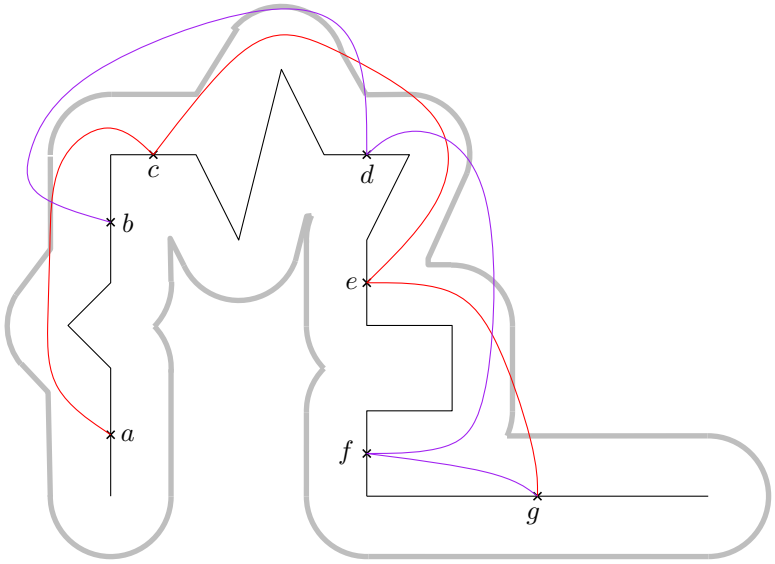

Abb. 5.12: Die durch die Folgen $\langle ace, ceg\rangle$ (rot) und $\langle bdf, dfg\rangle$ (lila) beschriebenen Polygonzüge haben die gleiche Punktemenge. Diese liegen im grauen Puffer.

in dieser Arbeit weder bewiesen noch widerlegt werden. Im Folgenden werden einige Ansätze diskutiert, die zur exakten Lösung beitragen können.

5.3.1 Anpassung des Auswahl-Kriteriums für Abkürzungen

Die im Abschnitt 5.2 vorgestellte Heuristik kann ein exaktes Ergebnis liefern, wenn man die Voraussetzungen etwas anpasst. Hierbei muss zunächst gelten, dass keine Vergleichs-Toleranzen zugelassen werden. Nun wählt man nicht Abkürzungen aus, sondern Kombinationen. Das Auswahl-Kriterium muss dabei berücksichtigen, dass eine Kombination c immer dann zugelassen wird, wenn es eine andere Kombination c' gibt, deren Punktemenge gleich der Punktemenge von c ist, und die nach einem anderen Kriterium zugelassen ist. Dies könnte z.B. durch Überdeckung von einem Puffer um das Eingabe-Polygon geregelt werden, wie es in Abbildung 5.12 angedeutet ist. Wie man sieht, würde auch der Polygonzug $\langle bde, deg\rangle$ die gleiche Punktemenge haben. Hierbei wurde die Kombination bde so gewählt, dass deren Abkürzungen bd und de unter allen Kombinationen mit gleicher Punktemenge die wenigsten Kanten im Eingabe-Polygon überspringen. Wenn es also eine rotationssymmetrische Vereinfachung gibt, die ace oder bdf benutzt, so gibt es in jedem Fall auch eine, die bde benutzt. Die Heuristik wählt dann jene Rotationsklassen, deren Elemente die Bild-Kombinationen mit den am wenigsten übersprungenen Kanten des Eingabe-Polygons haben. Dann kann sie keine rotationssymmetrische Vereinfachung mehr verpassen.

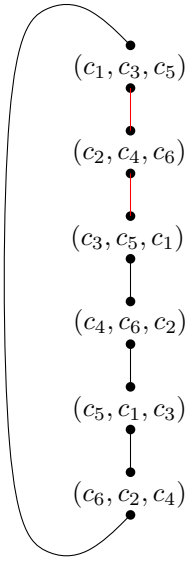

Abb. 5.13: Für das Beispiel aus Abbildung 5.2 ist der multiple Rundvergleichsgraph mit Startvergleich (c_1, c_3, c_5) zu sehen. Aus dem rot markierten Pfad lässt sich die rotationssymmetrische Vereinfachung synthetisieren.

5.3.2 Multiple Vergleiche

Ein exakter Ansatz, der robust gegen Vergleichs-Toleranzen ist, ist die Verwendung von multiplen Vergleichen. Analog zum Vergleichsgraph, wie er in Abschnitt 4.1 beschrieben wird, wird hierbei ein Vergleichsgraph $G_V = (V_V, E_V)$ erstellt, dessen Vergleiche mehrere Kombinationen berücksichtigen. Die Anzahl der Kombinationen, die verglichen werden, bestimmen dabei die Ordnung der Symmetrie. Das Beispiel aus Abbildung 5.2 ist 3-rotationssymmetrisch. In Abbildung 5.13 ist der zugehörige multiple Rundvergleichsgraph für den Startvergleich (c_1, c_3, c_5) zu sehen. Dieser liegt in topologischer Ordnung vor. Dadurch lässt sich eine rotationssymmetrische Vereinfachung synthetisieren. Diese wird aus dem Pfad vom Startvergleich v_0 zu einem Vergleich v gewonnen. Für den Vergleich v muss dabei gelten, dass dessen i-te Kombination die $((i \bmod |v|) + 1)$-te Kombination von v_0 ist für alle $1 \leq i \leq |v|$. Hier ist dies der Vergleich (c_3, c_5, c_1). Hieraus lassen sich die Pfade $c_1 \rightsquigarrow c_3$, $c_3 \rightsquigarrow c_5$ und $c_5 \rightsquigarrow c_1$ extrahieren, welche sich problemlos konkatenieren lassen. Dieses Verfahren benötigt für das Vergleichen $O(|E_V|^{|v_0|})$ Zeit. Hierbei kann die Ordnung $o = |v_0|$ der gesuchten Symmetrie so groß wie die Kanten-Anzahl n des Eingabe-Polygons sein. Dadurch ist die Laufzeit dieses Verfahrens exponentiell.

5.3.3 Einfach zusammengesetzte Rotationssymmetrie

Wie bereits erwähnt wurde, beschreibt jeder Kreis im Rundvergleichsgraph die Drehung einer Vereinfachung auf eine andere Vereinfachung wenigstens dann, wenn das Bild des Startvergleichs im Urbild des Kreises enthalten ist. Betrachtet man den Kreis im Vergleichsgraph aus Abbildung 5.3, der die nicht rotationssymmetrische Vereinfachung

dreht, so sieht man, dass diese Vereinfachung in einem Vergleichsgraph nicht beliebig oft auf eine andere Vereinfachung gedreht werden kann.

Der weiter-drehende Rundvergleichsgraph

Definition 5.9. Sei $G_R = (V_R, E_R)$ ein Rundvergleichsgraph für einen Startvergleich v_0. Sei $C_c(G_R)$ die Menge aller Kreise in G_R, deren Bild-Kreis auch als Urbild-Kreis eines Kreises in G_R enthalten ist. Für einen Kreis $c \in C_c(G_R)$ sei $V(c) \subset V_R$ die Menge aller Vergleiche, die auf c liegen, und $E(c) \subset E_R$ die Menge aller Kanten auf c.

Definition 5.10. Sei $G_R = (V_R, E_R)$ ein Rundvergleichsgraph für einen Startvergleich v_0. Der *weiter-drehende Vergleichsgraph* G_R^r für v_0 wird iterativ definiert:

$$G_R^0 = G_R \text{ und}$$

$$G_R^{r+1} = (\bigcup_{c \in C_c(G_R^r)} V(c), \bigcup_{c \in C_c(G_R^r)} E(c)).$$

Zur Berechnung dieses Graphen schaut man sich jede Kante (v_1, v_2) in $G_R = (V_R, E_R)$ einmal in topologischer Ordnung an, und übernimmt diese, wenn es eine Kante (v_3, v_4) gibt, so dass die Bilder von v_1 und v_2 die Urbilder von v_3 und v_4 sind, und zudem ein Pfad p_1 von v_1 nach v_0 über bereits gewählte Kanten sowie ein Pfad p_2 von v_3 ausgehend existiert, so dass der Bild-Pfad von p_1 der Urbild-Pfad von p_2 ist. Der Pfad von v_1 nach v_0 kann nach dem üblichen Muster in $O(|E_R|)$ Zeit berechnet werden, ebenso der passende Pfad p_2 (mit einer Kostenfunktion, die jeder passenden Kante Kosten 0 und allen anderen Kosten ∞ zuweist). Somit kann G_R^r in $O(r \cdot |E_R|^4)$ Zeit berechnet werden.

Abschlusseigenschaft des weiter-drehenden Rundvergleichsgraphen

Satz 5.11 (Abschlusseigenschaft von G_R^r)**.** *Sei $m = |E_R^0|$ die Anzahl der Kanten im ungedrehten Rundvergleichsgraph. Dann ist G_R^m entweder nicht existent, oder es gilt $G_R^m = G_R^{m+k}$ für jedes $k \in \mathbb{N}$.*

Beweis. Geht bei einer Drehung $G_R^r \to G_R^{r+1}$ keine Kante von E_R^r verloren, so gilt $G_R^r = G_R^{r+1}$. Wenn somit $G_R^m = G_R^{m+1}$ nicht gilt, sind nach m-maliger Drehung von G_R^0 insgesamt m Kanten verloren gegangen, d.h. E_R^m ist leer. Daraus folgt, dass G_R^m nicht existiert. □

Für das kleinste r', für welches $G_R^{r'} = G_R^{r'+1}$ gilt, heißt $G_R^{r'}$ der *abgeschlossene Rundvergleichsgraph*. Dieser kann in $O(|E_R|^4)$ Zeit berechnet werden. Für einen Startvergleich der Form (c, c) existiert der abgeschlossene Rundvergleichsgraph immer.

Satz 5.12. *In einem abgeschlossenen Rundvergleichsgraph $G_R^{r'}$ gibt es für jeden Kreis c einen Kreis c', so dass der Bild-Kreis von c der Urbild-Kreis von c' ist.*

Beweis. Für jeden Kreis c in $G_R^{r'+1}$ gibt es per Definition einen Kreis c' in $G_R^{r'}$, so dass der Bild-Kreis von c der Urbild-Kreis von c' ist. Da $G_R^{r'} = G_R^{r'+1}$ ist, gilt die Behauptung. □

Rotationssymmetrie im abgeschlossenen Rundvergleichsgraph

Definition 5.13. Sei P ein einfacher Polygonring, G dessen Kombinationsgraph und

$$C = \langle c_0 \rightsquigarrow c'_0, c_1 \rightsquigarrow c'_1, \ldots, c_{o-1} \rightsquigarrow c'_{o-1}, c_0 \rangle$$

Folge von Kombinationen aus G, so dass $(c_i, c_{(i-1+k \bmod o)+1}) \rightsquigarrow (c'_i, c'_{(i-1+k \bmod o)+1})$ für alle $0 \leq i < o$ und für ein $1 \leq k < o$ ein Pfad im abgeschlossenen Rundvergleichsgraph für Startvergleich $(c_i, c_{(i-1+k \bmod o)+1})$ ist. Man nenne C eine *einfach zusammengesetzte rotationssymmetrische Vereinfachung* von P und k deren *Sprung*.

Satz 5.14. *In jedem abgeschlossenen Rundvergleichsgraph $G_R^{r'}$ für einen nicht-trivialen Startvergleich v_0 existieren Vergleiche $(v_0, \ldots, v_{o-1})$ mit der Eigenschaft, dass das Bild von v_i das Urbild von $v_{(i \bmod o)+1}$ für alle $0 \leq i < o$ ist.*

Beweis. Sei c'_0 ein beliebiger Kreis in $G_R^{r'}$ und c_1 der Kreis, dessen Urbild-Kreis der Bild-Kreis von c'_0 ist. Dann liegt v_1 auf c_1. Sei n die Länge von c'_0 und ℓ der Abstand von v_1 zu v_0 auf c_1. Des weiteren seien für $2 \leq i \leq o$ der Kreis c_i sowie v_i entsprechend gewählt, dass der Bild-Kreis von c_i der Urbild-Kreis von $c_{(i \bmod o)+1}$ und das Bild von v_i das Urbild von $v_{(i \bmod o)+1}$ ist. Dies ist im abgeschlossenen Rundvergleichsgraph nach Satz 5.12 möglich. Der Abstand von v_i zu v_0 auf c_i beträgt $i \cdot \ell \bmod n$, was durch die Weitergabe von Bild-Kreis an Urbild-Kreis sichergestellt wird. Ist

$$o = \tfrac{n}{\mathrm{ggT}(n,\ell)},$$

so ist der Abstand von v_o zu v_0 auf c_o gerade

$$\tfrac{n \cdot \ell}{\mathrm{ggT}(n,\ell)},$$

d.h. dass $v_o = v_0$ ist. □

Satz 5.15. *In jedem abgeschlossenen Rundvergleichsgraph $G_R^{r'}$ für einen nicht-trivialen Startvergleich v_0 existiert eine einfach zusammengesetzte rotationssymmetrische Vereinfachung.*

Beweis. Seien $(v_0, v_1, \ldots, v_{o-1})$ die Vergleiche aus Satz 5.14 in topologischer Ordnung (d.h. dass deren Indizes ggf. vertauscht wurden). Der Pfad $v_0 \rightsquigarrow v_1$ existiert in $G_R^{r'}$ mit Existenz von v_1. Die Folge $c_0 \rightsquigarrow c'_0$ der einfach zusammengesetzten Vereinfachung mit Sprung k ist der Urbild-Pfad von $v_0 \rightsquigarrow v_1$ ohne v_1, $c_k \rightsquigarrow c'_k$ ist dessen Bild-Pfad und so weiter. Da so die c_i für alle $0 \leq i < o$ an der gleichen Position auf deren Kreis beginnen, wie die v_i auf deren Kreis, ergibt sich die einfach zusammengesetzte Vereinfachung. □

5.4 Spiegelsymmetrie

So wie bei der Erkennung von Rotationssymmetrie vergleicht man auch bei der Suche nach Spiegelsymmetrien das Eingabe-Polygon mit sich selbst, bzw. wie in Abschnitt 4.5 bereits geschildert mit sich selbst in entgegengesetzter Drehrichtung. Für spiegelsymmetrische Teilstücke kann hierfür das Verfahren aus Abschnitt 4.5 auch verwendet werden,

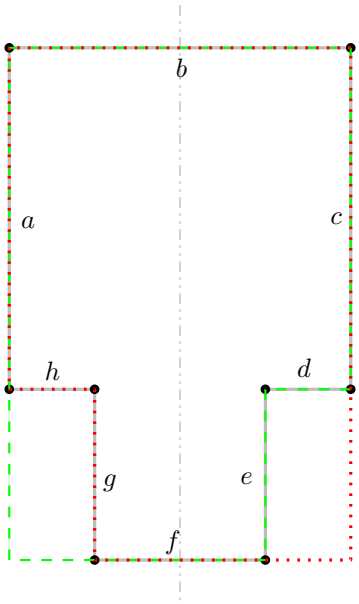

Abb. 5.14: Das Abgebildete Polygon ist zur grauen Achse spiegelsymmetrisch. Dabei ergibt die grüne Vereinfachung zusammen mit der roten Vereinfachung wie in Abschnitt 4.5 besprochen einen Kreis im reduzierten Vergleichsgraph für Startvergleich (abc, cba).

die in Abbildung 4.2 gezeigten Polygonzüge können Teile des gleichen Polygonrings sein. Allerdings kann man dabei mehr finden, als man möchte. Abbildung 5.14 verdeutlicht dies. Die Spiegelsymmetrie wird zwar gefunden, jedoch ist weder die grüne noch die rote Vereinfachung für sich spiegelsymmetrisch.

5.4.1 Suche einer Vereinfachung mit einer einzigen Spiegelachse

Ein Pfad im Vergleichsgraph, wie er in Abschnitt 4.5 beschrieben wird, ist immer spiegelsymmetrisch, wenn der Startvergleich aus zwei identischen Kombinationen besteht, da die durch diese Kombination repräsentierte Kante immer zu sich selbst spiegelsymmetrisch ist. Das gleiche gilt, wenn die beiden Kombinationen des Startvergleichs eine Abkürzung gemeinsam haben, und von dieser weglaufend betrachtet werden, vgl. Abbildung 5.15. Somit ist

$$\{(abc, cba)\} \cup \{(bcd, cba)\}$$

die Menge der zu betrachtenden Startvergleiche für alle Kanten a, b, c, d des Eingabe-Rings, so dass ab, bc und cd ausgewählte Abkürzungen sind. Hierfür gibt es nicht mehr als $O(n^6)$ Möglichkeiten wenn n die Anzahl der Kanten im Eingabe-Polygon ist. Wie in Abschnitt 4.5 besprochen, wird für diese Startvergleiche der reduzierte Vergleichsgraph $G_V = (V_V, E_V)$ in $O(|E_V|)$ Zeit aufgebaut. Ein Pfad p in diesem Graph muss dann an einem Vergleich aus der Menge

$$\{(abc, cba)\} \cup \{(abc, dcb)\}$$

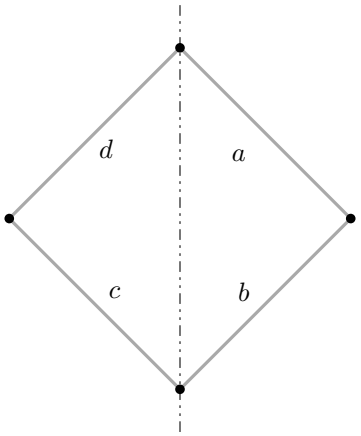

Abb. 5.15: Die spiegelsymmetrische Vereinfachung des Polygons steckt im Pfad von (dab, adc) nach (abc, dcb).

enden. Im Beispiel aus Abbildung 5.14 würde das mit Startvergleich (abc, cba) und Zielvergleich (efg, gfe) funktionieren. Abbildung 5.15 zeigt dagegen den Fall der Kombinationen, die sich eine Abkürzung teilen. Diesen Pfad kann man nach dem in dieser Arbeit üblichen Muster in $O(|E_V|)$ Zeit finden. Dann kann man den Bild-Pfad von p mit dem Urbild-Pfad von p zu einer spiegelsymmetrischen Vereinfachung konkatenieren. Somit beträgt die Laufzeit zur Entdeckung aller Vereinfachungen, die zu mindestens einer Achse spiegelsymmetrisch sind, $O(n^6 \cdot |E_V|) = O(n^{14})$.

6 Zusammenfassung und Ausblick

In dieser Arbeit wurde nach Symmetrien in Gebäudeumrissen gesucht. Für diesen Zweck ist es vielversprechend, die Symmetrien nicht im originalen Umriss zu suchen, sondern in einer vereinfachten Version. Somit können auch Symmetrien gefunden werden, die durch kleine Details wie z.B. Laderampen etc. unterbrochen wurden.

Ein Ansatz, der von Lladós et al. [LBM97] beschrieben wird, nutzt Cyclic String Matching mit einer Operation zum Zusammenfügen von Kanten des Eingabe-Rings, um trotz solcher Unregelmäßigkeiten Symmetrien erkennen zu können. Dieses Verfahren wurde dazu gedacht, um auf polygonal approximierte Umrisse angewendet zu werden, welche aus Bild-Daten extrahiert werden. In dieser Arbeit wurde gezeigt, dass dieses Verfahren zur Anwendung auf Gebäudeumrisse nicht sehr gut geeignet ist.

Weiter wird in dieser Arbeit der Ansatz zur Gebäude-Vereinfachung mit Abkürzungen von Haunert und Wolff [HW10] genutzt, um mit Hilfe eines Vergleichsgraphen Teilstücke von vereinfachten Versionen von Gebäuden miteinander zu vergleichen. Hieraus sind Verfahren zum Vergleichen von Vereinfachungen verschiedener Gebäude sowie zum Finden von Spiegelsymmetrien zwischen diesen entstanden. Des weiteren wird eine Heuristik zur Erkennung von rotationssymmetrischen Vereinfachungen eines Gebäudes beschrieben, welche auch als Java-Programm-Bibliothek implementiert wurde. Außerdem wurden verschiedene exakte Ansätze zur Erkennung von rotationssymmetrischen Vereinfachungen diskutiert. Ein weiteres in dieser Arbeit vorgestelltes Verfahren dient zur Erkennung von spiegelsymmetrischen Vereinfachungen eines Gebäudes zu einer einzigen Symmetrieachse.

Einige grundlegende Fragen sind vorerst offen geblieben. Zur Erkennung von Rotationssymmetrie konnte nicht geklärt werden, ob diese in Gegenwart von Vergleichstoleranzen exakt und in Polynomialzeit lösbar ist. Vermutlich ist dies aber mit einer der im Abschnitt 5.3.3 vorgestellten Behandlung des Rundvergleichsgraphen ähnlichen Methode möglich. Die Erkennung von spiegelsymmetrischen Vereinfachungen wurde kaum behandelt. Hier wurde ein Verfahren vorgestellt, welches alle Vereinfachungen eines Gebäudes findet, die wenigstens zu einer Achse spiegelsymmetrisch sind. Will man von diesem Ansatz ausgehend alle Vereinfachungen mit allen Symmetrieachsen haben, so wäre die Laufzeit exponentiell zur Anzahl der Kanten im Eingabe-Ring, da es analog wie bei der Rotationssymmetrie exponentiell viele spiegelsymmetrische Vereinfachungen gibt. Daher ist offen, ob man alle Vereinfachungen mit allen Spiegelachsen in Polynomialzeit finden kann.

Literaturverzeichnis

[BB93] H. Bunke und U. Bühler: Applications of approximate string matching to 2D shape recognition. *Pattern Recognition*, 26(12):1797 – 1812, 1993, ISSN 0031-3203. http://www.sciencedirect.com/science/article/pii/003132039390177X.

[CLR04] T. H. Cormen, C. E. Leiserson und R. L. Rivest: *Algorithmen - Eine Einführung.* Oldenbourg Wissensch.Vlg, 1. Auflage, 2004, ISBN 3486275151. http://www.amazon.de/Algorithmen-Einf%C3%BChrung-Thomas-H-Cormen/dp/3486275151%3FSubscriptionId%3D13CT5CVB80YFWJEPWS02%26tag%3Dws%26linkCode%3Dxm2%26camp%3D2025%26creative%3D165953%26creativeASIN%3D3486275151.

[Hau12] J. H. Haunert: A Symmetry Detector for Map Generalization and Urban-Space Analysis. *ISPRS Journal of Photogrammetry and Remote Sensing*, 74:66–77, 2012.

[HW08] J. H. Haunert und A. Wolff: Optimal simplification of building ground plans. In: *Proceedings of the XXIst ISPRS Congress, July 3–11, 2008, Beijing, China*, Band XXXVII(Part B2) der Reihe *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, Seiten 373–378. ISPRS, 2008. http://www.isprs.org/proceedings/XXXVII/congress/tc2.aspx#wg3.

[HW10] J. H. Haunert und A. Wolff: Optimal and Topologically Safe Simplification of Building Footprints. In: *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '10, Seiten 192–201, New York, NY, USA, 2010. ACM, ISBN 978-1-4503-0428-3. http://doi.acm.org/10.1145/1869790.1869819.

[LBM97] J. Lladós, H. Bunke und E. Martí: Finding rotational symmetries by cyclic string matching. *Pattern Recognition Letters*, 18(14):1435–1442, 1997. http://dblp.uni-trier.de/db/journals/prl/prl18.html#LladosBM97.

[PWW10] D. Peters, Y. Wu und S. Winter: Testing Landmark Identification Theories in Virtual Environments. In: Christoph Hölscher, Thomas F. Shipley, Marta Olivetti Belardinelli, John A. Bateman und Nora S. Newcombe (Herausgeber): *Spatial Cognition*, Band 6222 der Reihe *Lecture Notes in Computer Science*, Seiten 54–69. Springer, 2010, ISBN 978-3-642-14748-7. http://dblp.uni-trier.de/db/conf/spatialCognition/spatialCognition2010.html#PetersWW10.

[SLBW08] S. Steiniger, T. Lange, D. Burghardt und R. Weibel: An Approach for the Classification of Urban Building Structures Based on Discriminant Analysis Techniques. *Transactions in GIS*, 12(1):31–59, 2008, ISSN 1467-9671. http://dx.doi.org/10.1111/j.1467-9671.2008.01085.x.

[Tre10] M. S. Treder: Behind the Looking-Glass: A Review on Human Symmetry Perception. *Symmetry*, 2(3):1510–1543, 2010, ISSN 2073-8994. http://www.mdpi.com/2073-8994/2/3/1510.

[Wag95] J. Wagemans: Detection of visual symmetries. *Spatial Vision*, 9(1):9–32, 1995. http://www.ingentaconnect.com/content/vsp/spv/1995/00000009/00000001/art00002.

[WKS10] S. Werder, B. Kieler und M. Sester: Semi-automatic interpretation of buildings and settlement areas in user-generated spatial data. In: *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '10, Seiten 330–339, New York, NY, USA, 2010. ACM, ISBN 978-1-4503-0428-3. http://doi.acm.org/10.1145/1869790.1869836.

[WL75] R. A. Wagner und R. Lowrance: An Extension of the String-to-String Correction Problem. *Journal of The ACM*, 22:177–183, 1975.

Erklärung zur Masterarbeit

Hiermit versichere ich, dass ich die Arbeit selbstständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und die Arbeit bisher oder gleichzeitig keiner anderen Prüfungsbehörde vorgelegt habe.

Würzburg, den 31.03.2013 Hagen Schwaß