UNIVERSITÄT
KOBLENZ · LANDAU
Fachbereich 4: Informatik

Erkennen von Dominosteinen mit Hilfe von wissensbasierten Bildanalyse-Verfahren

Diplomarbeit
zur Erlangung des Grades
Diplom-Informatiker
im Studiengang Computervisualistik

vorgelegt von

Marcel Häselich

Betreuer: Dipl.-Math. (FH) Stefan Wirtz, Institut für Computervisualistik, Fachbereich Informatik, Universität Koblenz-Landau
Erstgutachter: Prof. Dr.-Ing. Dietrich Paulus, Institut für Computervisualistik, Fachbereich Informatik, Universität Koblenz-Landau
Zweitgutachter: Dipl.-Math. (FH) Stefan Wirtz, Institut für Computervisualistik, Fachbereich Informatik, Universität Koblenz-Landau

Koblenz, im Januar 2010

Kurzfassung

Das Ziel dieser Arbeit besteht darin, Dominosteine in 2-D Bildern robust zu erkennen und zu klassifizieren. Als Eingabedaten fungieren alle Arten von Intensitätsbildern, und die Ausgabe besteht aus klassifizierten Dominosteinen. Das Problem, das gelöst werden soll, besteht darin, bei so vielen Dominosteinen wie möglich exakt zu bestimmen, um welchen Dominostein es sich handelt. Zur Problemlösung werden Modellklassen verwendet, in denen explizites Wissen zur Merkmalsfindung und Objekterkennung enthalten ist. Dazu wird eine Segmentierung entwickelt, die einem Dominostein ermöglicht, seine Bestandteile im Bild zu lokalisieren. Bei der Zuordnung zwischen den im Bild gefundenen und im Modell vorhandenen Komponenten entstehen mehrere Hypothesen. Um diese zu bewerten, werden unterschiedliche Abstandsfunktionen entwickelt und evaluiert. Für die Zuordnung von Segmentierungs Objekten zu Modellbestandteilen wird die Ungarische Methode verwendet.

Abstract

The goal of this work is the solid recognition and classification of domino tiles in 2-D images. All kinds of intensity images can act as input data and the output consists of classified domino tiles. The problem to be solved is to determine as precisely as possible which domino tiles are present. In order to solve this problem, classes of models are used that contain explicit knowledge for feature extraction and object recognition. Therefore a segmentation is developed, allowing a domino tile to locate its components within the image. In the case of the assignment of components found within the image and those given by the model, several hypotheses emerge. To rate these hypotheses, different distance functions are developed and evaluated. Matching the segmentation objects to the model objects is accomplished by applying the Hungarian Method.

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Die Vereinbarung der Arbeitsgruppe für Studien- und Abschlussarbeiten habe ich gelesen und anerkannt, insbesondere die Regelung des Nutzungsrechts.

Mit der Einstellung dieser Arbeit in die Bibliothek bin ich einverstanden.	ja ☐	nein ☐
Der Veröffentlichung dieser Arbeit im Internet stimme ich zu.	ja ☐	nein ☐

Koblenz, den 26. Januar 2010

Danksagung

Mein Dank gilt allen Personen, die mich beim Erstellen dieser Arbeit unterstützt haben. Besonders bedanken möchte ich mich bei meinen beiden Betreuern Dipl.-Math. (FH) Stefan Wirtz und Prof. Dr. Dietrich Paulus. Danke für die hervorragende Betreuung, die intensive Förderung und die kreativen Anregungen.

Den Mitarbeitern der Arbeitsgruppe Aktives Sehen danke ich für ihre Hilfestellungen. Insbesondere gilt mein Dank Dipl.-Inform. Peter Decker und Dipl.-Inform. Detlev Droege, die bei Fragen der Bildverarbeitung immer ein offenes Ohr für mich hatten. Dipl.-Ing. Wolfram Hans möchte ich für seine Hilfestellung bei der Bedienung der Lichtbox, des Drehtellers und der Bilddatenbank danken. Für die Beratung zur Klassifikation bedanke ich mich bei Dr. Marcin Grzegorzek.

Bei Prof. Dr. Jürgen Ebert möchte ich mich für die privaten Treffen und Vorträge bedanken, die mir eine große Hilfe zum Verständnis der Graphentheorie waren.

Ganz besonders bedanken möchte ich mich bei meinen Eltern und meiner Ehefrau Sabrina, die mich während meines Studiums stets unterstützt haben und auf die ich immer zählen konnte.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Im Rahmen des Projekts ***S**oftware **T**echniques for **O**bject **R**ecognition* (STOR) [FE09b] [Fal09] entsteht ein komponentenorientiertes System zur Objekterkennung in Bildern und Bildfolgen. Das Projekt existiert seit 2007 und ist gemeinsam mit dem Projekt ***Po**se **Es**timation* (PoSe) [PPDS09] aus dem Enhanced Reality Projekt entstanden. Für das System werden Modelle verwendet, in denen explizites Wissen zur Merkmalsfindung und Objekterkennung enthalten ist. STOR ist ein gemeinsames Projekt der Arbeitsgruppe Aktives Sehen, Institut für Computervisualistik, und der Arbeitsgruppe Ebert, Institut für Softwaretechnik, der Universität Koblenz-Landau. Das Projekt ist in zwei Fallstudien gegliedert:

Erkennung von Dominosteinen (2-D) Die erste Fallstudie beschäftigt sich mit der Erkennung von Dominosteinen in 2-D Bildern und Bildfolgen. Als Eingabedaten fungieren alle Arten von Intensitätsbildern. Das Problem, das gelöst werden soll, ist, so viele Dominosteinen wie möglich exakt zu bestimmen.

Rekonstruktion und Modellierung von Gebäuden (3-D) Der zweite Teil beinhaltet die Generierung von urbanen Objektmodellen aus Bildern. Nähere Informationen zur Rekonstruktion und Modellierung von Gebäuden finden sich in [FE09a, FED+09].

Diese Arbeit beschäftigt sich mit dem Problem der ersten Fallstudie und realisiert eine Implementierung zur robusten Erkennung von Dominosteinen .[1] Das Ziel besteht darin, Dominosteine in 2-D Bildern zu Erkennen und möglichst exakt bestimmen zu können, um welchen Dominostein es sich handelt. Für Dominosteine existiert keine Norm, daher können sie in allen möglichen Farben und Seitenverhältnissen vorliegen. Eine präzise Beschreibung des verwendeten Modells befindet

[1] Die Diplomarbeit von Susanne Maur [Mau10] entsteht zeitgleich mit dieser Arbeit und beschäftigt sich mit der zweiten Fallstudie.

sich in Abschnitt 2.1. Das vorhandene Wissen über Dominosteine wird in Klassen modelliert. Ein Dominostein kennt dabei jede seiner Komponenten und ist in der Lage, diese im Bild zu finden. Dafür muss eine geeignete Segmentierung entwickelt und implementiert werden.

Bei der Zuordnung zwischen den im Bild gefundenen und im Modell vorhandenen Komponenten entstehen mehrere Hypothesen. Um diese zu bewerten, werden unterschiedliche Abstandsfunktionen entwickelt und evaluiert. Für die Zuordnung von Segmentierungsobjekten zu Modellbestandteilen werden Graph-Matching Algorithmen auf Matrizen verwendet.

Mittels Konsolen-Aufruf kann ein Ordner eingelesen werden und eine Klassifikation aller bekannten Modelle für alle im Ordner beinhalteten Bilddateien durchgeführt werden. Die Ergebnisse für alle Klassen werden für jedes Bild in einer Textdatei für diesen Ordner gespeichert. Auf diese Weise lässt sich eine Skript-basierte automatische Klassifikation realisieren.

Eine GUI (engl. **G**raphical **U**ser **I**nterface, grafische Benutzeroberfläche) stellt die Zuordnung von Modell- zu Bildbestandteilen grafisch dar und gibt Aufschluss über die Bewertung der Hypothesen. Darüber hinaus visualisiert eine Übersicht, wie einzelne Modelle bewertet und welche Zuordnungen anhand welcher Kriterien ausgewählt wurden.

Bei der Implementierung ist zu berücksichtigen, dass die entstehenden Softwaremodule im Rahmen des STOR Projekts wiederverwendet werden können und den Ansprüchen eines Komponentensystems gerecht werden.

Die Arbeit ist wie folgt gegliedert:

Kapitel 2 erklärt die theoretischen Grundlagen und die Theorie der Zuordnung von Modell- zu Bildteilen. Darüber hinaus wird das verwendete Dominomodell und die ausgewählte Bildverarbeitungsbilbiothek vorgestellt und die Arbeit in Relation zum Stand der Technik gesetzt.

Kapitel 3 gibt eine Übersicht aller Vorverarbeitungsschritte und beschreibt diese anschliessend an einem Beispielbild. Danach werden die entstandenen Segmentierungsobjekte mit ihren Attributen für die Zuordnung vorgestellt.

Kapitel 4 enthält die Vorgehensweise zum Lösen des Problems der Zuordnung. In diesem Kapitel wird die Ungarische Methode vorgestellt und die Vorteile einer grafischen Darstellung der Paarung von Graphen verdeutlicht.

Kapitel 5 beschäftigt sich mit der Datenakquisition unter Verwendung der Lichtbox und des Drehtellers sowie den durchgeführten Experimenten anhand der Testdaten. Anschliessend werden die Ergebnisse und evaluierten Abstandsfunktionen präsentiert.

Kapitel 6 reflektiert die Ergebnisse der im vorigen Kapitel beschriebenen Experimente und die Veränderungen durch die Evaluation. Außerdem werden weitere Optionen und Ansätze mit der Herangehensweise dieser Arbeit verglichen.

Kapitel 7 fasst die Ergebnisse der Experimente und Evaluation zusammen und enthält das Fazit sowie einen Ausblick über mögliche Erweiterungen.

Anmerkungen

Notation

Neu eingeführte Begriffe werden bei ihrer erstmaligen Verwendung *kursiv* gedruckt und anschliessend definiert.

UML

Sämtliche UML-Diagramme in dieser Arbeit wurden mit dem IBM Rational Software Architect [Cor09] erstellt.

Struktogramme

Für die Visualisierung von Programmstrukturen und -abläufen wurde das LaTeX-Paket Struktex [Tea09] verwendet.

Mathematische Symbole

Eine vollständige Übersicht aller in dieser Arbeit verwendeten mathematischen Symbole befindet sich in Anhang B.

Kapitel 2

Theoretische Grundlagen

Das folgende Kapitel beschreibt die theoretischen Grundlagen dieser Arbeit. Die Grundlagen lassen sich in drei Hauptgebiete unterteilen: Das Modell, die Extraktion der Bildelemente und die Zuordnung zwischen den beiden. Zunächst wird in Abschnitt 2.1 das verwendete Modell für die Dominosteine dargestellt und eine Übersicht des verwendeten Sets gegeben. Aufbauend auf dem Modell erfolgt anschliessend in Abschnitt 2.2 eine formale Spezifikation der gegebenen Problemstellung. In Abschnitt 2.4 wird eine Übersicht relevanter Bildverarbeitungsbibliotheken gegeben und die Gründe für die ausgewählte Bibliothek aufgeführt. Danach werden die Grundlagen der Zuordnung und die dazugehörigen Begriffe in Abschnitt 2.5 erklärt. Im letzten Abschnitt 2.6 wird diese Arbeit in Bezug zum aktuellen Stand der Technik gesetzt.

2.1 Modell

Für Dominosteine existiert keine Norm, es gibt Exemplare in sehr vielen Formen, Seitenverhältnissen, Größen und Materialien. Eine grafische Darstellung des gewählten Modells ist in Abbildung 2.1 zu sehen. Das Modell besitzt ein festes Seitenverhältnis von zwei zu eins, ein fixes Layout der Augen (*Pips*) und eine variable Größe. Die vierzehn möglichen Belegungen der Pips im Layout sind in der Grafik durch blaue Kreise repräsentiert, deren Mittelpunkte jeweils durch einen roten Punkt gekennzeichnet sind. Ein Pip bezeichnet einen Kreis innerhalb eines Dominosteins und besitzt die Attribute *x-Position*, *y-Position* und *Radius*. Die drei Attribute liegen in Modellkoordinaten vor, d. h. die Werte sind in Relation zur Größe des Dominosteins[1] angegeben. Die x-Position, in der Grafik grün eingezeichnet, beschreibt den Abstand zur kürzeren linken Seite des Steins. Analog beschreibt

[1] Da nicht alle Dominosteine ein festes Seitenverhältnis besitzen, wurde die längste Seite eines Steins als Metrik für die Größe gewählt.

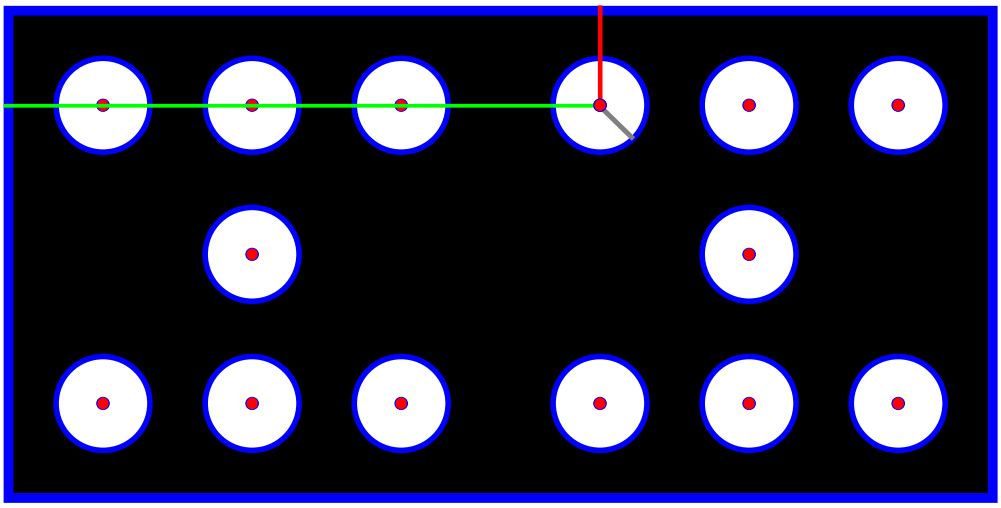


Abbildung 2.1: Grafisches Layout eines Dominosteins. Die vierzehn möglichen Belegungen der Pips im Layout sind in der Grafik durch blaue Kreise repräsentiert, deren Mittelpunkte jeweils durch einen roten Punkt gekennzeichnet sind. Der X-Wert eines Pips in Modellkoordinaten ist exemplarisch in Grün eingetragen, der Y-Wert in Rot und der Radius in Grau.

die rot eingefärbte y-Position den Abstand zur längeren oberen Seite des Stein. Das in Grau dargestellte Attribut Radius beinhaltet den Radius des Pips. Jeder Dominostein besteht aus zwei Hälften, von denen jede zwischen Null und Sechs Pips enthalten kann. Für gewöhnlich wird jeder Dominostein von einer Trennlinie in seine beiden Hälften geteilt. Diese Trennlinie ist jedoch nicht obligatorisch und wurde somit für das Modell nicht berücksichtigt.

Dominosteine sind in einem Set enthalten, das, ebenso wie die Steine selbst, keiner Norm unterliegt. Eine Unterscheidung der Sets richtet sich nach der höchsten Augenzahl eines einzelnen Steins. Es existieren u. a. Doppel-6er, Doppel-9er, Doppel-12er, Doppel-15er und Doppel-18er Dominosets. Bei dem für diese Arbeit verwendeten Sets handelt es sich um ein Doppel-6er, bestehend aus 28 unterschiedlichen Dominosteinen. Das Sets ist in Abbildung 2.2 dargestellt.

Das Dominomodell ist in der Lage, seine Bestandteile im Bild zu finden. Das Modell beinhaltet die Bestandteile mit ihren Attributen und deren Anordnung. Das Wissen, wie ein einzelnes Teil im Bild zu finden ist, wurde in einer separaten *Dominokontrolle* modelliert. Die Dominokontrolle kennt alle nötigen Bildverarbeitungsschritte auf einem Eingabebild, um die vorhandenen Bestandteile im Modell zu lokalisieren. Dabei handelt es sich um den Dominostein selbst, ein Rechteck mit einem Seitenverhältnis von zwei zu eins und den Pips eines Dominostein, Kreise

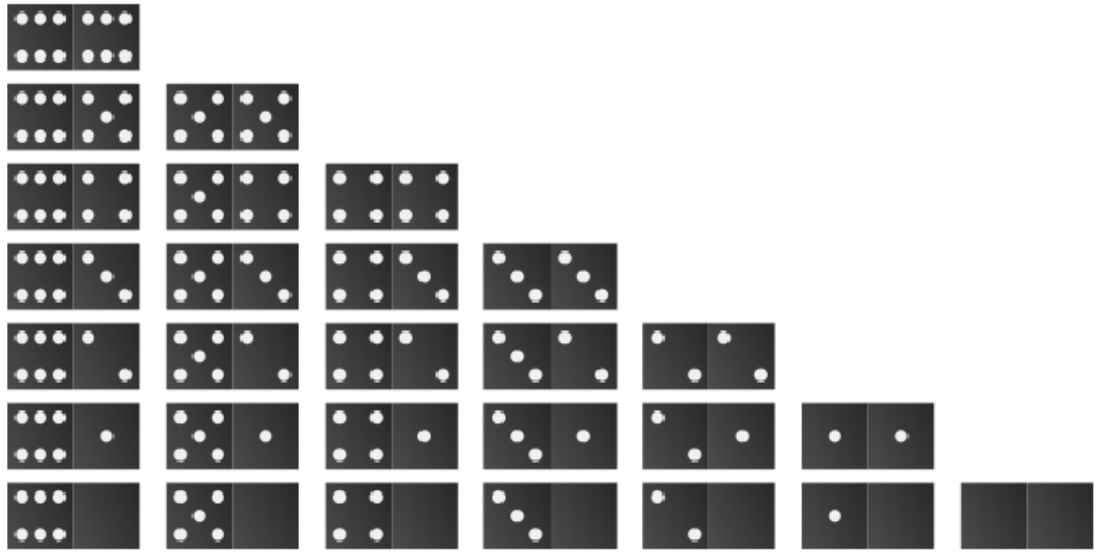


Abbildung 2.2: Doppel-6er Set von Dominosteinen. Das für die Modellierung verwendete Set besteht aus 28 verschiedenen Steinen, die zwischen Null und Sechs Pips pro Hälfte enthalten können (Quelle: [Wik09]).

innerhalb dieses Rechtecks. Das Verfahren der Unterteilung in eine Wissensbasis seitens des Modells und der Wissensnutzung seitens der Kontrolle wird auch als *wissensbasiertes Verfahren* bezeichnet [Qui97]. In der Dissertation von Quint wird ein wissensbasiertes System zur Interpretation monokularer Luftbilder vorgestellt. Das System erstellt eine automatische Beschreibung komplexer urbaner Szenen durch die Verwendung von Wissen aus topografischen Karten. Die Dominokontrolle verfügt neben dem im Modell vorhandenen Wissen zusätzlich über Strategien, wie unterschiedliche geometrische Objekte im Bild gefunden werden können. Eine genauere Beschreibung der Dominokontrolle und der verwendeten Strategien erfolgt in Abschnitt 3.1.

Aufbauend auf dem vorgestellten Modell wird den nächsten beiden Abschnitten 2.2 und 2.3 eine formale Spezifikation der Problemstellung erläutert. Die Spezifikation beinhaltet alle von der Problemstellung abhängigen Informationen und beschreibt diese mathematisch. Die von der Problemstellung unabhängige Extraktion der Segmentierungsdaten wird im darauf folgenden Abschnitt 2.4 behandelt, und in Abschnitt 2.4.1 erfolgt eine Darstellung des ausgewählten Datentyps.

2.2 Spezifikation

Die folgende Spezifikation beschreibt zunächst mathematisch die Zuordnung von Segmentierungsobjekten zu Modellbestandteilen. Anschliessend werden die zur Problemstellung korrespondierenden Ein- und Ausgabedaten des *Systems* spezifiziert. Der Begriff *System* bezieht sich dabei auf das Programm *DominoDiscoverer* (siehe Abschnitt 5.2.2), das während dieser Arbeit entstanden ist. Der letzte Teil der Spezifikation behandelt die Abstandsfunktion und die Basisversionen der Kosten- und Straffunktionen.

Gegeben:

- Ein Universum SEGMENT von Segmentierungsobjekten, wobei SEGMENT $= C \cup R, C \cap R = \varnothing$ mit einer Menge von Kreisen C und einer Menge von Rechtecken R
- Eine Menge $S \subseteq$ SEGMENT von im Bild gefundenen Segmentierungsobjekten
- Eine Menge M von Dominomodellen
- Ein konkretes Modell $m \in M$
- Ein konkreter Bestandteil eines Modells $k \in m$
- Ein Knoten (eng. **N**ode) N_k (siehe Abschnitt 2.5.1)
  - Eine Menge $N_{Dominostein} \subseteq N_k$
  - Eine Menge $N_{Pips} \subset N_k$

Gesucht:

- Eine partielle Funktion $\gamma : S \rightarrowtail \bigcup_{k \in m} N_k$ mit

  $\forall\, x \in S \cap R : \gamma(x) \in N_{Dominostein}$ und
  $\forall\, y \in S \cap C : \gamma(y) \in N_{Pips}$

  wobei gilt: $x \xrightarrow[\text{enthält}]{} y$

Abbildung 2.3: Fotografie möglicher Eingabedaten des Systems.

2.2.1 Problemstellung

Das System kann mit mehreren planaren Dominosteinen im Bild arbeiten und ist mit falschen Eingabedaten kompatibel. Abbildung 2.3 zeigt eine beispielhafte Darstellung möglicher Eingabedaten. Neben mehreren Dominosteinen kann das System auch mit Rechtecken und Kreisen im Bild umgehen, die nicht zu Dominosteinen gehören und somit ein sinnvolles und erwartungskonformes Verhalten zeigen. Bei Bildern, die keine detektierbare Geometrie enthalten, wird eine entsprechende Rückmeldung an den Benutzer des Systems gegeben.

Eingabe:

- Eine Menge $S \subseteq$ SEGMENT von Segmentierungsobjekten, deren Elemente frei von perspektivischer Verzerrung sind.
  - $\| S \cap R \| = j, j \in \mathbb{N}$
  - $\| S \cap C \| = i, i \in \mathbb{N}$
  - Segmentierungsobjekte können rotiert oder skaliert vorkommen

  - Segmentierungsobjekte können zum Teil aus dem Bild herausragen beziehungsweise verdeckt sein
  - Nicht alle Kreise $S \cap C$ müssen in den Rechtecken $S \cap R$ enthalten sein
  - Segmentierungsobjekte untereinander können ein unterschiedliches Skalierungsverhältnis haben
- Ein konkretes Schema M der Dominosteine
- Eine Abstandsfunktion (siehe Abschnitt 2.3)

Ausgabe:

- Eine Menge $M' \subset M$ von Dominomodellen, wobei M' die Hypothesen der möglichen Dominosteine enthält
- Eine Bewertung J_{local}, $J_{local} \in [0..1] \subset \mathbb{R}$, wie gut diese Zuordnungen im Detail etabliert werden konnte
- Eine vollständige Bewertung $J \in [0..1] \subset \mathbb{R}$ der Korrespondenz

2.3 Abstandsfunktion

Die Zuordnung γ wird durch Anwendung einer Abstandsfunktion $\delta : M \times S \rightarrow \mathbb{R}$ optimiert. Die Abstandsfunktion setzt sich aus einer Kostenfunktion δ_r und einer Straffunktion δ_p zusammen. Die Funktion δ_r bewertet die Zuordnung der Segmentierungsobjekte zu den Elementen der Modelle. Die Funktion δ_p bestraft das Fehlen sowohl von Segmentierungsobjekten, als auch von den Elementen der Modelle. Es gilt:

- $S = s_1 \cup s_2, s_2 = \{s \in S \mid \nexists x \in M : (x, y) \in dom\ \delta_r\}$
- $M = m_1 \cup m_2, m_2 = \{m \in S \mid \nexists y \in S : (x, y) \in dom\ \delta_r\}$

Gesucht:

$$\underset{\gamma}{\operatorname{argmax}} \sum_{S \in SEGMENT} \delta_r(\gamma(s_1), s_1) + \delta_p(s_2, m_2)$$

2.3.1 Kostenfunktion

Die Kostenfunktion δ_r bewertet, mit welcher Genauigkeit die Elemente eines Modells zu den Elementen eines Segmentierungsobjekts zugeordnet werden können. Die Funktion wird im Rahmen dieser Arbeit evaluiert (siehe Abschnitt 5.4), die Basisversion ist wie folgt definiert:

Basisversion:

Gegeben ein $c = \begin{pmatrix} c_x & c_y & c_r \end{pmatrix}^T$ und ein $p = \begin{pmatrix} p_x & p_y & p_r \end{pmatrix}^T$ mit $c \in S \cap C$, $p \in N_{Pips}$, wobei (x, y) dem Mittelpunkt und r dem Radius entspricht.

$$\delta_{r1}(p, c) = \sum_i 50 \frac{min(c_{\mathrm{x}_i}, p_{\mathrm{x}_i})}{max(c_{\mathrm{x}_i}, p_{\mathrm{x}_i})} + 50 \frac{min(c_{\mathrm{y}_i}, p_{\mathrm{y}_i})}{max(c_{\mathrm{y}_i}, p_{\mathrm{y}_i})}$$

Rechteckbewertung

Der bisher vorgestellte Teil der Kostenfunktion bezieht sich ausschließlich auf die Kreise (Pips). Um die Dominosteine korrekt zu identifizieren und gegeneinander abzugrenzen, sind diese Informationen ausreichend. Da jedoch auch andere Objekte vorkommen können, wird die Bewertung des Rechtecks ρ als zusätzliches Kriterium herangezogen. Die Bewertung ρ fließt immer in die Abstandsfunktion mit ein, egal welche Version von Kosten- oder Straffunktion verwendet wird. Sie wird jedoch nicht mit evaluiert, da das Rechteck im Modell bei allen achtundzwanzig Steinen identisch ist und keinen Beitrag zur Unterscheidung leisten kann. Für die Bewertung des Rechtecks gilt:

Gegeben ein Rechteck r bestehend aus den vier Linien l_a, l_b, l_c und l_d. Eine Linie $l = \overline{AB}$ wird durch zwei Punkte $A(x_A, y_A)$ und $B(x_B, y_B)$ definiert, wobei A den Start- und B den Endpunkt beschreibt. Es gelten folgenden Eigenschaften:

- $r \in S \cap R$
- Sei $\overline{l} = \sqrt{(x_A - y_A)^2 + (y_A - y_B)^2}$
- $\overline{l_a} \approx \overline{l_c}$, $\overline{l_b} \approx \overline{l_d}$
- $S_1 \subset S, S_2 \subset S, S_3 \subset S, S_4 \subset S$
- Sei $\alpha(l_1, l_2)$ der Schnittwinkel zwischen den Linien l_1 und l_2 (für $l_1 \parallel l_2$: $\alpha(l_1, l_2) = 0$)
- Sei $\omega(l)$ der Winkel der Linie l zur x-Achse

Gesucht:

$$\rho(r) = \frac{F_1(l_a, l_b)}{4} + \frac{F_1(l_a, l_d)}{4} + \frac{F_2(l_a, l_c)}{4} + \frac{H(l_1, l_2, l_3, l_4)}{4}$$

wobei $l_a \perp l_b, l_a \perp l_d$ und $l_a \parallel l_c$
mit

$$F_1(x, y) : S_1 \times S_2 \rightarrow [0..1] \subset \mathbb{R} : \alpha(x, y)$$

$$F_2(x,y) : S_1 \times S_2 \rightarrow [0..1] \subset \mathbb{R} : \omega(x) - \omega(y)$$

$$H(w,x,y,z) : S_1 \times S_2 \times S_3 \times S_4 :\rightarrow [0..1] \subset \mathbb{R} : \frac{\frac{min(\overline{w},\overline{y})}{max(\overline{w},\overline{y})} + \frac{min(\overline{x},\overline{z})}{max(\overline{x},\overline{z})}}{2}$$

2.3.2 Straffunktion

Die Straffunktion δ_p bestraft das Fehlen von Elementen seitens des Modells oder des Segmentierungsobjekts. Die Funktion wird analog zur Kostenfunktion evaluiert (siehe Abschnitt 5.4). Die Grundversion ist folgendermassen definiert:

Basisversion:

Für $d = max(|s_2|, |m_2|) - min(|s_2|, |m_2|)$ gilt:

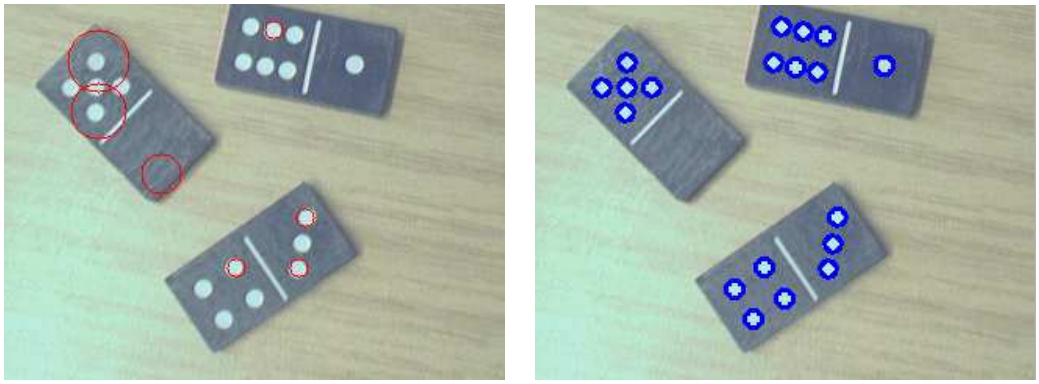
$$\delta_{p1}(m_2, s_2) = \sum_{i=1}^{d} 0 = 0$$

2.4 Bildverarbeitungsbibliothek

Eine *Bibliothek* (Programmbibliothek) bezeichnet in der Programmierung eine Sammlung von Datenstrukturen, Algorithmen und Funktionen. Im Unterschied zu Programmen handelt es sich bei Bibliotheken nicht um ausführbare Elemente, sondern um Funktionalität, die in eigenen Programmen verwendet werden kann. Bei einer *Bildverarbeitungsbibliothek* handelt es sich um eine Sammlung von speziellen Hilfsmodulen und -programmen zur Bildverarbeitung, die von anderen Programmen über eine Schnittstelle (API, engl. **a**pplication **p**rogramming **i**nterface) genutzt werden können.

Es existiert eine Vielzahl von Bildverarbeitungsbibliotheken. Für STOR wurden im Rahmen einer Diplomarbeit [Haa09] fünfzehn verschiedene Bibliotheken im Hinblick auf die Erkennung von Dominosteinen analysiert und die am besten geeigneten miteinander verglichen. Für diese Arbeit waren die folgenden beiden Bibliotheken besonders interessant.

OpenCV Die *Open Computer Vision Library* (OpenCV) [BK08] verfügt über eine weit verbreitete API. Diese bietet eine umfangreiche Sammlung von Algorithmen und Beispielprogrammen zur Bildverarbeitung. Ursprünglich wurde OpenCV von Intel entwickelt, seit 2008 ist die Bibliothek jedoch als Open-Source Projekt auf SourceForge erhältlich [Inc09].



(a) (b)

Abbildung 2.4: Vergleich einer Kreiserkennung von OpenCV und PUMA. Abbildung 2.4(a) zeigt das Ergebnisbild einer Kreiserkennung ohne manuelle Einstellung der Parameter unter Verwendung der OpenCV Bibliothek. Das Ergebnis der Kreiserkennung von PUMA ist in Abbildung 2.4(b) dargestellt.

PUMA Bei der *Programmierumgebung für die Musteranalyse* (PUMA) [Pau03] [AAS06] handelt es sich um eine interne API der Arbeitsgruppe Aktives Sehen der Universität Koblenz-Landau. Die PUMA-API bietet eine Vielzahl von Algorithmen, Datenstrukturen und feingranularen ausführbaren Programmen. PUMA entstand 1992 an der Universität Erlangen-Nürnberg und wird seit 2002 an der Universität Koblenz-Landau verwendet.

Aufgrund des Bekanntheitsgrades, des Umfangs und der detaillierten Dokumentation ist OpenCV auf den ersten Blick die bessere Wahl. Erste Experimente liessen eine einfache Handhabung erkennen, jedoch zeigten sich speziell bei den Problemstellungen zur Erkennung von Dominosteinen Schwächen. Besonders das Auffinden von Kreisen im Bild erweist sich mit OpenCV als problematisch, da die Größe der gesuchten Kreise nach Möglichkeit vorher angegeben werden sollte. Die Entscheidung der Bildverarbeitungsbibliothek fällt daher auf PUMA, da die Kreiserkennung dort auch ohne vorherige Eingabe der Suchgröße robuste Ergebnisse liefert. Ein Vergleich der Ergebnisse der jeweiligen Kreiserkennung von OpenCV und PUMA ist in Abbildung 2.4 zu sehen. In Abbildung 2.4(a) wurden von OpenCV gefundene Kreise in Rot dargestellt, von PUMA gefundene Kreise wurden in Abbildung 2.4(b) Blau markiert. Bei beiden Bibliotheken wurden die Algorithmen parameterlos auf das Eingabebild angewendet, d. h. die Größe der gesuchten Kreise war nicht bekannt.

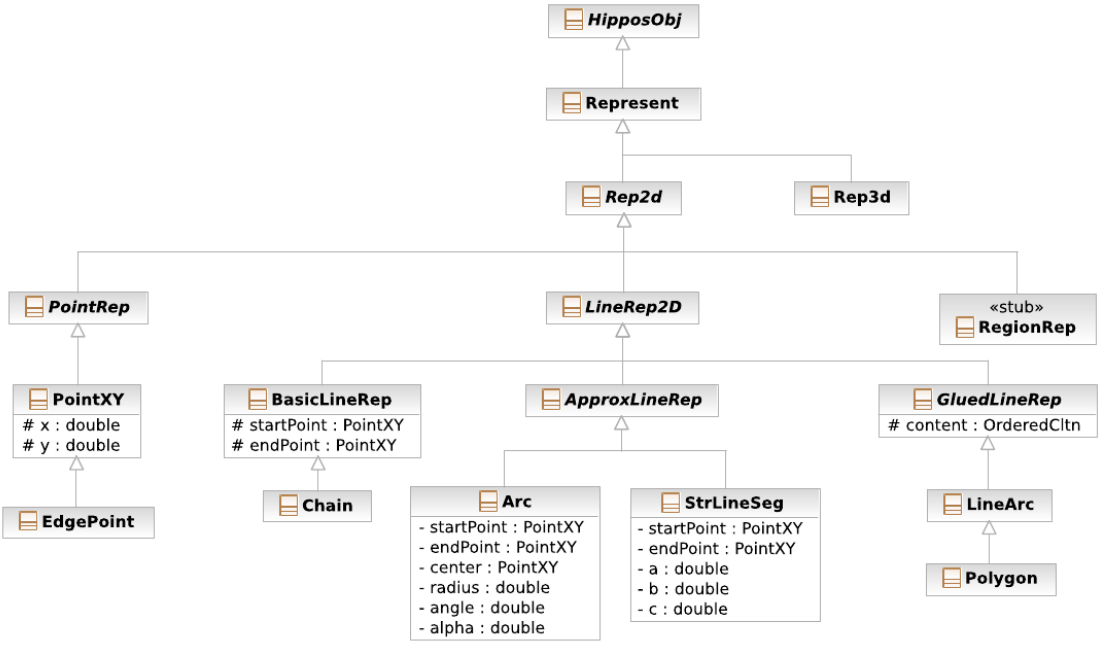


Abbildung 2.5: Hierarchische Darstellung der Geometrieklassen in PUMA. Die Darstellung wurde aus [Haa09] entnommen.

2.4.1 Segmentierungsobjekt

Mit PUMA als ausgewählter Bibliothek stehen eine Vielzahl an Algorithmen und Datenstrukturen für die Bildverarbeitung zur Verfügung. Die verwendeten Algorithmen werden in Kapitel 3 vorgestellt. Zunächst erfolgt eine Spezifikation der geometrischen Objekte in PUMA mit besonderem Hinblick auf das Segmentierungsobjekt. PUMA basiert auf der Bibliothek *NIHCL*[2], die vom National Institute of Health zur Objekt-orientierten Programmierung entwickelt wurde. NIHCL beinhaltet allgemein verwendbare Datentypen wie beispielsweise String, Date oder Time und bietet zudem ein umfangreiches Klassenset. Weitere Information zu NIHCL befinden sich in der Dokumentation [AAS05] [GOP90] oder in der Diplomarbeit von Haas [Haa09]. Innerhalb von PUMA bildet NIHCL[3] die Grundlage für die Datenstrukturen, die im Modul *Hippos* implementiert sind. Abbildung 2.5 zeigt eine Darstellung der Klassenhierarchie der geometrischen Objekte in PUMA. Wie in dem Diagramm zu erkennen ist, sind alle geometrischen Objekte Unterklassen der Klasse HipposObj und erben somit deren Eigenschaften und Funktionalität. Auf

[2]NIHCL steht für National-Institute-of-Health's-class-library.
[3]In der Variante KONIHCL (Koblenz-NIHCL).

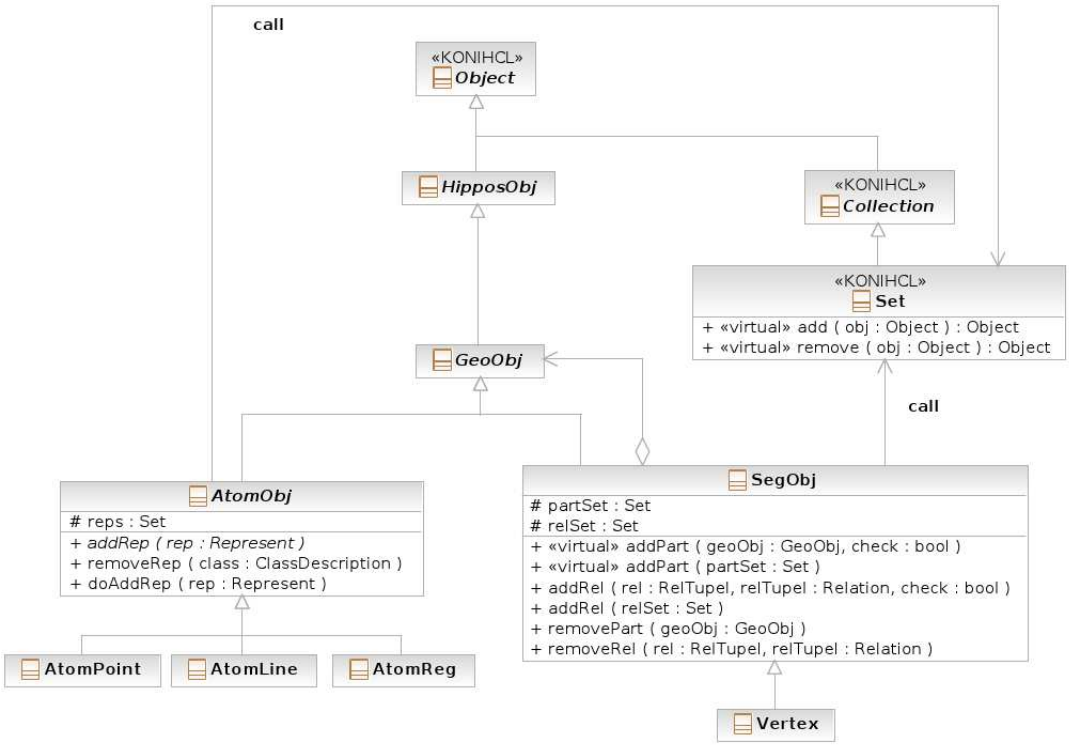


Abbildung 2.6: Das Segmentierungsobjekt in PUMA (Quelle: [Haa09]).

diese Weise werden Mechanismen wie Introspektion und Serialisierung aus NIHCL in PUMA ebenfalls nutzbar.
Ein weiterer großer Vorteil von PUMA besteht in der Datenstruktur *SegObj* (Segmentierungsobjekt), in der Segmentierungsergebnisse auf unterschiedliche Weise repräsentiert werden können. Beim SegObj handelt es sich um eine Containerklasse zur Verwaltung beliebiger geometrischer Objekte vom Typ GeoObj. Ein Modell der Klassenhierarchie des Segmentierungsobjekts in PUMA ist in Abbildung 2.6 dargestellt. Wie in der Abbildung zu sehen ist, besitzt ein Segmentierungsobjekt ein Attribut vom Typ Set. Durch das Kompositum-Pattern [GHJV95, Haa09] können somit sowohl atomare Objekte als auch weitere Segmentierungsobjekte einem einzelnen Objekt hinzugefügt werden. Das Segmentierungsobjekt lässt sich somit ideal für die Erkennung von Dominosteinen instrumetalisieren, da in einem einzigen Objekt ein vollständiger potentieller Kandidat für einen Dominostein abgespeichert werden kann. Ein einzelnes Segmentierungsobjekt beinhaltet also sämtliche zu einem im Bild gefundenen Objekt gehörenden Teile und kann diese ausgeben und sich selbst reflektieren.

2.5 Zuordnung

Nachdem das verwendete Modell in Abschnitt 2.1 und die Bildverarbeitungsbibliothek in Abschnitt 2.4 vorgestellt wurden, erfolgt nun der letzte Teil der Grundlagen, der die Terminologie der Zuordnung festlegt. Wie in Abschnitt 2.2 formal beschrieben wurde, besteht das Zuordnungsproblem (engl. matching problem) darin, in der Menge der Modelle dasjenige Element zu bestimmen, das zu einem Segmentierungsobjekt im Bild die größte Übereinstimmung besitzt. Die im nächsten Abschnitt definierten Begriffe bilden zum einen die Basis für die Zuordnung (siehe Abschnitt 4), zum anderen aber gewährleisten sie gleichzeitig die Konformität mit den in der Literatur verwendeten Begriffen.

2.5.1 Begriffsdefinitionen

Bei der Begriffsdefinition wurde versucht, einen möglichst großen Konsens mit den bereits in der Literatur vorhandenen Definitionen zu finden. Die jeweiligen Definitionen sind mit einem Verweis versehen, insofern sie wörtlich oder sinngemäss aus einem Literaturwerk entnommen wurden.

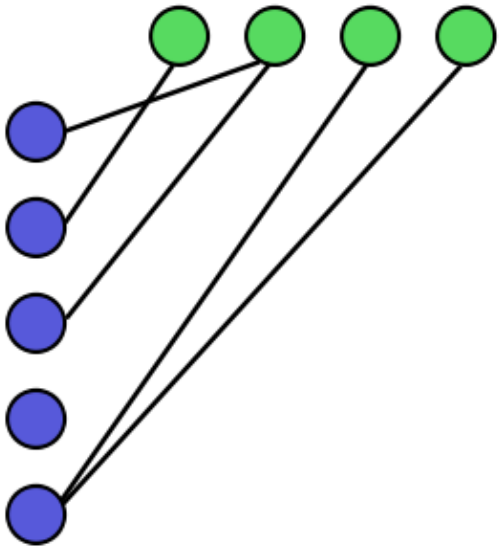
Graph Ein Graph $G = (N, E)$ besteht aus einer Menge N von Knoten, zwischen denen eine Menge E von Kanten (engl. **E**dges) verlaufen.

Bipartiter Graph Ein Graph $G = (N, E)$ mit $N = n_1 \cup n_2 \wedge n_1 \cap n_2 = \varnothing$ heisst bipartiter Graph, wenn für alle Kanten $e(a, b) \in E$ gilt: $a \in n_1 \wedge b \in n_2$. Eine Darstellung eines bipartiten Graphen befindet sich in Abbildung 2.7 (linke Seite).

Gewichteter Graph Ein Graph $G = (N, E)$ heisst gewichteter Graph, wenn eine Gewichtsfunktion $w : E \to \mathbb{R}$ existiert, die jeder Kante $e \in E$ eine reelle Zahl zuordnet.

Zuordnung In einem Graphen $G = (N, E)$ ist eine Menge *unabhängiger* Kanten M eine Zuordnung (engl. Matching, auch Paarung genannt). Paarweise nicht benachbarte Kanten nennt man unabhängig. M ist eine Zuordnung von $U \subseteq N$, wenn jede Ecke aus U mit einer Kante aus M *indiziert*. Ein Knoten a und eine Kante e heissen indiziert, wenn $a \in e$ gilt. Diese indizierten Kanten heissen dann (in M) gepaart (zugeordnet). Mit keiner Kante aus M indizierte Knoten heissen ungepaart (nicht zugeordnet) (vgl. [Die00] Seite 1–19).

Gewichtsmaximale Zuordnung Sei $G = (N, E)$ gewichteter ein Graph mit einer Gewichtsfunktion $w : E \to \mathbb{R}$. Eine Zuordnung M ist eine *gewichtsma-*



$$\begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

Abbildung 2.7: Darstellung eines bipartiten Graphen mit einer korrespondierenden Adjazenzmatrix. Links ist der bipartite Graph zu sehen, mit den Teilmengen n_1 (blau) und n_2 (grün). Rechts befindet sich die zum Graph korrespondierende Adjazenzmatrix. Der Menge n_1 wurden dabei die Zeilen zugeordnet, der Menge n_2 die Spalten.

ximale Zuordnung (auch optimale Zuordnung/Korrespondenz genannt, siehe [Win94]), wenn gilt $w(M) \geq w(M')$.

Vollständige Zuordnung Eine Zuordnung M mit maximaler Mächtigkeit heisst *vollständige Zuordnung.*

Adjazenzmatrix Jeder bipartite Graph G lässt sich durch eine Adjazenzmatrix (auch Nachbarschaftsmatrix genannt) darstellen. Dabei werden die Knoten N durch Spalten beziehungsweise Zeilen der Matrix repräsentiert. Jedem Knoten wird dabei genau eine Zeile oder eine Spalte zugeordnet. Verläuft eine Kante e zwischen einem Knoten a und einem Knoten b, dann ist der Wert der Adjazenzmatrix an der Stelle (a, b) gleich 1, sonst 0. Ein Beispiel für eine Adjazenzmatrix zu einem bipartiten Graphen ist in Abbildung 2.7 (rechte Seite) zu sehen.

Pfad In einem Graphen $G = (N, E)$ mit zwei Knoten $a, b \in N$ ist ein Pfad p von a nach b eine Folge von Kanten $p = (e_0, e_1), (e_1, e_2), \cdots, (e_{k-1}, e_k)$ mit $a = e_0$, $b = e_k$ und $(e_i, e_{i+1}) \in E$ für $0 \leq i < k$ (vgl. [Lux04] Kapitel 2).

Alternierender Pfad Für einen Graphen $G = (N, E)$ ist ein einfacher Pfad $p = (e_0, e_1), (e_1, e_2), \cdots, (e_{k-1}, e_k)$ ein alternierender Pfad bezüglich einer

Zuordnung M, falls die Kanten von p abwechselnd in M und $E \backslash M$ liegen. D. h. $(e_i, e_{i+1}) \in M \Leftrightarrow (e_{i+1}, e_{i+2}) \notin M$ für $0 \leq i \leq k-2$ (vgl. [Lux04] Kapitel 2).

Augmentierender Pfad Für einen Graphen $G = (N, E)$ ist ein alternierender Pfad p ein augmentierender Pfad bezüglich einer Zuordnung M, falls die Endknoten von p ungepaart sind (vgl. [Lux04] Kapitel 2). Augmentierende Pfade werden in Abschnitt 4.1.2 für die Ungarische Methode noch einmal ausführlicher erläutert.

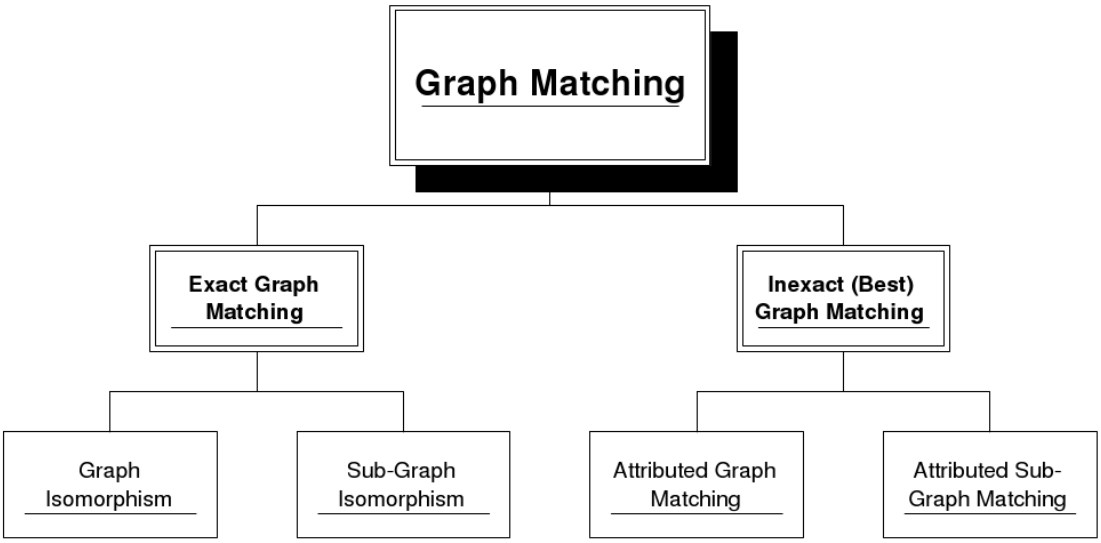


Abbildung 2.8: Klassifikation in exakte und bestmögliche Zuordnung nach Bengoetxea. Die Darstellung wurde aus [Ben02] entnommen.

Exakte und bestmögliche Zuordnung In [Ben02] unterscheidet Bengoetxea für die modellbasierte Bildverarbeitung zwei verschiedene Zuordnungsarten (siehe Abbildung 2.8). Die Unterscheidung unterteilt die Zuordnung in eine *exakte* oder eine *bestmögliche* Zuordnung (engl. exact and inexact (best) graph matching). Für einen Modellgraphen $G_M = (N_M, E_M)$ und einen Segmentierungsgraphen $G_S = (N_S, E_S)$ lassen sich für die Zuordnung zwei Möglichkeiten festlegen:

- Existiert für $|N_M| = |N_S|$ eine Zuordnungsmöglichkeit $f : N_S \rightarrow N_M$ mit $(u, v) \in E_S \Leftrightarrow (f(u), f(v)) \in E_M$, wird dies als *Isomorphismus* (engl. isomorphism) bezeichnet. Bei dieser Art von Zuordnung handelt es sich um eine exakte Zuordnung.

- Analog bedeutet dies für die bestmögliche Zuordnung, das diese dann vorliegt, wenn es nicht möglich ist, einen Isomorphismus zu finden, der eine exakte Zuordnung zwischen den beiden Graphen beschreibt.

Nach dieser Definition handelt es sich also bei der Problemstellung der Dominosteinerkennung aus dieser Sichtweise um eine bestmögliche Zuordnung. Denn gemäss der Spezifikation der Abstandsfunktion (siehe Abschnitt 2.3) können sowohl auf der Seiten des Modellgraphen G_M als auch auf der Seite des Segmentierungsgraphen G_S unterschiedliche Elemente vorliegen. D. h. die Bedingung $|N_M| = |N_S|$ ist nicht erfüllt.

Gewicht-maximales bipartites Zuordnungsproblem Sei $G = (N, E)$ ein bipartiter Graph mit einer Gewichtsfunktion $w : E \rightarrow \mathbb{R}$. Als *gewicht-maximales bipartites Zuordnungsproblem* bezeichnet man das Finden derjenigen Zuordnung M, für die $w(M) \geq w(M')$ gilt. Die Zuordnung M muss nicht zwangsläufig vollständig sein, daher handelt es sich um eine bestmögliche Zuordnung. Zur Lösung dieses Problems lässt sich die Ungarische Methode verwenden (siehe Abschnitt 4.1). Die Zuordnung umfasst dabei quantifizierbare Gemeinsamkeiten der beiden Teilmengen des bipartiten Graphen. D. h. eine gewichtete Zuordnung eines bipartiten Graphen besitzt einen vergleichbaren Wert, dem die Güte (engl. goodness) als Zuordnungs-Metrik zugrunde liegt (vgl. [BB82] Kapitel 11 Abschnitt 3.1).

2.6 Stand der Technik

In der Literatur existiert eine Reihe von Arbeiten, die mit dem vorliegenden Ansatz eine thematische Überschneidung aufweisen. Der folgende Abschnitt gibt einen Überblick der bereits vorhandenen Arbeiten und beschreibt zum einen, wie sich diese Arbeit in den aktuellen Stand der Technik einordnen lässt, und zum anderen, wie sich bereits vorhandene Lösungen aus der Literatur für einzelne Problemstellungen verwenden und erweitern lassen.

Für die Modellierung wurde das a priori vorhandene Wissen über die Dominosteine verwendet. Zur Erstellung der Wissensbasis wurden für das Doppel-6er Dominoset 28 Klassen modelliert, die insgesamt 42 Dominosteinen entsprechen. Die höhere Anzahl der Steine kommt daher, weil die Diagonalen in manchen Objekten von oben links nach unten rechts verlaufen, in anderen von unten links nach oben rechts, und somit für manche Steine zwei Repräsentationen notwendig sind. Auf diese Weise wird die Wissensbasis einmalig definiert und ist fortan ohne jeglichen weiteren Aufwand verfügbar. Bei dieser Art der Wissensrepräsentation handelt es sich um eine Modellierung geometrischer Informationen, wie sie auch in ERNEST (ERlanger semantisches NEtzwerk SySTem [NSSK90]) verwendet wurde. Wie bei

ERNEST wurden die von der Problemstellung unabhängigen Methoden von der Wissensbasis getrennt und nur die Strategie, wie ein Dominostein sich selbst im Bild lokalisieren kann, seitens der Wissensbasis in der Dominokontrolle modelliert. Im Gegensatz zu ERNEST wurden für die Wissensbasis jedoch ausschließlich die geometrischen Informationen der Objekte verwendet und keine Informationen über den Aufbau oder die Eigenschaften der in einem Eingabebild dargestellten Szene. Ein Vorteil gegenüber anderen Verfahren ist somit beispielsweise, dass es nicht notwendig ist, das Wissen für die Modellerstellung aus unterschiedlichen Ansichten und unter verschiedenen Beleuchtungen in einer Vielzahl an Aufnahmen zu generieren (siehe [GI92]).

Die Vorverarbeitung der Eingabebilder kann größtenteils durch Verwendung einer geeigneten Bildverarbeitungsbibliothek erfolgen. Die vorgestellte Bibliothek PUMA liefert ein gutes Repertoire an soliden Bildverarbeitungsfunktionen, deren Verwendung und Erweiterung im nächsten Kapitel 3 beschrieben wird. Um Verwechslungen vorzubeugen, wird an dieser Stelle darauf hingewiesen, dass der in dieser Arbeit verwendete Begriff *Vorverarbeitung* sich nicht auf eine Vorverarbeitung im Sinne der Bildverarbeitung bezieht und beispielsweise das Anwenden von Filtern beschreibt. Der Begriff umfasst im Rahmen dieser Arbeit stattdessen alle notwendigen Schritte, um die essentiellen Bild-Bestandteile zu extrahieren. Der innovative Ansatz dieser Arbeit beginnt, *nachdem* die Vorverarbeitung durchgeführt wurde. Indem durch die Vorverarbeitung Bestandteile aus dem Bild extrahiert wurden, ist es nun möglich auf den Segmentierungsobjekten zu arbeiten und diese zu betrachten. Ein Kreis ist nicht länger nur eine Kombination von Pixeln im Bild, sondern ein eigenes Objekt, das über Attribute und Funktionen verfügt.

Nach der Vorverarbeitung eröffnen sich unterschiedliche mögliche Herangehensweisen, wie die Zuordnung der Segmentierungsobjekte zu den Modellobjekten realisiert werden kann. In der Literatur werden zwei verschiedene Strategien beschrieben (vgl. [Nie90] Seite 240 ff.):

Daten-getriebene Strategie Bei der daten-getriebenen Strategie (engl. data-driven) oder bottom-up-Strategie dienen die im Bild gefundenen Segmentierungsobjekte als Start. Ausgehend von den Segmentierungsobjekten wird unter den vorhandenen Modellen das am besten geeignete herausgesucht.

Modell-getriebene Strategie Den umgekehrten Weg hingegen geht die modell-getriebene Strategie (engl. model-driven) oder top-down Strategie und beginnt mit den Modellobjekten als Start. Für jedes Modell wird versucht, seine Komponenten im Bild zu lokalisieren und herauszufinden, ob und wo es sich im Bild befindet.

Der in dieser Arbeit verwendete Ansatz basiert auf beiden Strategien und stellt eine Mischform dar. Zum einen wird anhand der Modelle bestimmt, welche Arten

von Segmentierungsobjekten für die Vorverarbeitung relevant sind. Dieser Teil entspricht einer modell-getriebenen Strategie und findet auf Ebene der Modellierung statt. Zum anderen werden jedoch beispielsweise nicht nur Rechtecke mit einem Seitenverhältnis von zwei zu eins betrachtet, wie in der reinen modell-getriebenen Strategie, sondern alle Rechtecke. Die Einschränkung über einen Schwellwert bezüglich des Seitenverhältnisses der Rechtecke erfolgt später durch die Dominokontrolle. D. h. nachdem die Arten der Segmentierungsobjekte eingeschränkt wurden, wird eine daten-getriebene Strategie verwendet, die alle Objekte von den in den Modellen vorhandenen Typen im Bild detektiert. Somit ist es möglich, dass die Abstandsfunktion ausschließlich auf den Segmentierungsobjekten arbeitet und die in den Modellen gesetzten Eigenschaften verwendet. Für die Evaluation der Abstandsfunktionen lässt sich seitens der Dominokontrolle weiteres a priori Wissen über die Dominosteine für die Bewertung verwenden. Die Ideen dazu, beispielsweise den zentralen Pip, der gerade von ungeraden Domino-Hälften unterscheidet, anders zu bewerten, werden in Kapitel 6 dargestellt.

Durch die Verwendung von explizitem Wissen eröffnen sich darüber hinaus zwei weitere interessante Aspekte für die Abstandsfunktionen. Zum einen das von Ballard und Brown vorgestellte Konzept der *Sprungfedern* (engl. springs[4]) (siehe [BB82] Kapitel 11.3). Die für die Dominosteine umgesetzte Idee sieht vor, dass eine Evaluation der Abstandsfunktionen nicht nur lineare Kosten beinhaltet, sondern eine Kostenfunktion realisiert, die sich exponentiell im Parameterbereich der Modellkoordinaten der Dominosteine verhält. Dadurch werden kleinere Ungenauigkeiten der Zuordnung nur noch minimal berücksichtigt, während große Ungenauigkeiten für die Zuordnung nahezu untragbar werden (siehe Kapitel 6). Der zweite interessante Aspekt betrifft die Straffunktion. Hier lässt sich das von Niemann vorgestellte Konzept des *pruning* anwenden, indem anhand der Erwartungen seitens des expliziten Wissens nicht erfüllte Zuordnungen unterschiedlich stark zur Bewertung beitragen können (vgl. [Nie90], Kapitel 6.5.3).

Nachdem eine Abstandsfunktion alle Korrespondenzen von Modell- zu Bildbestandteilen bewertet hat, gilt es, für jedes Gesamtmodell eine gewicht-maximale Zuordnung für jeden bipartiten Graph zu finden. Dabei wird für jedes Modell zu jedem passenden Segmentierungsobjekt eine Zuordnung erstellt. Für die Zuordnung der Modellbestandteile zu den Bestandteilen eines Segmentierungsobjekts existieren sehr schnelle Algorithmen, z. B. die Ungarische Methode, mit einem Aufwand von $O(x^3)$. Die Ungarische Methode wurde bereits in [Win94] von Winzen verwendet, um die Bestandteile von Segmentierungsobjekten aus unterschiedlichen Perspektiven einander zuzuordnen. Neben der Ungarischen Methode konnten in der Li-

[4] An dieser Stelle wurde keine wörtliche, sondern eine sinngemäße Übersetzung gewählt, da der Begriff Sprungfeder die Funktionsweise wesentlich besser beschreibt.

teratur keine weiteren adäquaten Alternativen zur Lösung des gewicht-maximalen bipartiten Zuordnungsproblems gefunden werden.

Das Ziel dieser Arbeit ist es, so viele Dominosteine wie möglich so exakt wie möglich in einem 2-D Eingabebild zu erkennen. Ebenso interessant wie die effiziente Umsetzung dieser Zielsetzung ist jedoch die Herangehensweise an die gegebene Problemstellung. Es existiert eine Vielzahl von Möglichkeiten, zu bestimmen, welcher Dominostein sich in einem Eingabebild befindet, zumal es sich nicht um ein komplexes geometrisches Modell handelt. So könnte man beispielsweise die Kreise in einem Rechteck einfach zählen - findet man auf der linken Seite zwei Kreise und auf der rechten vier, hat man den Dominostein 4-2 gefunden. Eine weitere Möglichkeit besteht darin, für die Dominosteine ein *Template Matching* (siehe [Bru09, BHJ+99]) durchzuführen, wie es z. B. für die Mustererkennung oder die Gesichtserkennung verwendet wird.

Ein weiterer Vorteil dieses Ansatzes besteht in der natürlichen und damit intuitiven Herangehensweise an die gegebene Problemstellung. Betrachtet ein Mensch ein Bild eines Dominosteins, so vergleicht er keinesfalls *alle* bereits in seinem Gedächtnis memorierten Bilder, wie es dem Template Matching entsprechen würde. Vielmehr interpretiert man Bilder um diese zu verstehen. Erkennen wir beispielsweise ein Rechteck in einem Bild, bauen wir auf dieser Information auf und versuchen weitere Information zu extrahieren. Befinden sich innerhalb des Rechtecks weitere Rechtecke, könnte es sich *vielleicht* um ein Fenster handeln oder Radiergummis auf einem Block. Gleiches gilt für Kreise innerhalb der Rechtecke, bei denen die Vermutung nahe liegt, dass es sich um einen Dominostein oder ein Buch mit darauf liegenden Münzen handeln *könnte*. Wir betrachten und vergleichen das Gefundene mit dem uns bereits Bekannten. Wenn wir nicht wissen, was ein Dominostein ist, ist es uns nicht möglich diesen in einem Bild zu erkennen. Aus algorithmischer Sicht stellen diese Vermutungen jeweils Hypothesen dar, die bewertet werden können. Viele Systeme in der Literatur bauen an dieser Stelle auf Ansätze, die Hypothesen für die Zuordnungen aufstellen, indem der Suchraum als Baum interpretiert wird. Innerhalb des Baumes werden den Korrespondenzen unterschiedliche Pfade zugeteilt [Gri90, Cas93]. Wurden Hypothesen gefunden, müssen diese verifiziert werden, ob sie ein konsistentes Ergebnis liefern oder nicht. Die Suche nach der bestmöglichen Zuordnung lässt sich nur dann als Pfadsuche realisieren, wenn nicht alle Zustände im Such- beziehungsweise Zustandsraum betrachtet werden. Dies lässt sich unter anderem durch Verwendung des A^*-Algorithmus [KSN92, SNHW92] umsetzen. Der A^*-Algorithmus wurde unter Verwendung einer Wissensbasis in [Bou00] dazu verwendet, ein Objekt aus unterschiedlichen Perspektiven zu erkennen und dessen Lage zu bestimmen. Für die Dominosteine hingegen wurden alle Hypothesen in Betracht gezogen und bewertet, was durch Anwendung der Ungarischen Methode einer Breitensuche entspricht (siehe [Win94]).

Neben der Herangehensweise liegt ein weiterer Vorteil dieses Ansatzes in der bereits erwähnten Wissensrepräsentation und der Tatsache, dass dieses Wissen zu jeder Zeit abrufbar vorliegt. Es ist nicht nur möglich zu bestimmen, um welchen Dominostein es sich handelt und wo sich dieser im Bild befindet, sondern alle Bestandteile können zugeordnet werden. D. h. für jedes im Bild gefundene Objekt wurde festgestellt, zu welchem Modell es am besten passt *und* welchem Modellteil es entspricht. Somit ist es beispielsweise möglich, den Bildobjekten die gleichen Farben wie den Modellobjekten zuzuordnen (siehe Abschnitt 4.2.2).

Kapitel 3

Vorverarbeitung

Dieses Kapitel beschreibt alle Vorverarbeitungsschritte zur Erkennung von Dominosteinen in Eingabebildern. Als erstes erfolgt eine Übersicht aller Schritte in Abschnitt 3.1. Anschliessend werden die einzelnen Teilschritte separat vorgestellt. Diese Arbeit richtet sich an einen mit der Bildverarbeitung vertrauten Leserkreis, daher werden einzelne Schritte nur kurz erklärt, wenn es sich um Aspekte handelt, die als bekannt vorausgesetzt werden können. Darunter fällt der Abschnitt 3.2 zur Kantendetektion, der Abschnitt 3.3, in dem die Linienextraktion beschrieben wird, und der Kreisfinder in Abschnitt 3.5. Danach folgt eine ausführlichere Beschreibung der restlichen Verarbeitungsschritte. Dabei handelt es sich um das Modul zum Zerteilen und Zusammenfügen von Segmentierungsobjekten in Abschnitt 3.4 und den Rechteckfinder in Abschnitt 3.6. Als letztes werden die durch die Vorverarbeitung entstandenen Segmentierungsobjekte in Abschnitt 3.7 vorgestellt.

3.1 Übersicht

Ein Dominostein kennt seine Komponenten und soll in der Lage sein, Segmentierungsobjekte in Form von Kreisen und Rechtecken im Bild finden zu können. In den Klassen der Dominosteine ist ausschließlich das Wissen über die vorhandenen Pips und die eigene Geometrie gespeichert. Die Strategie, wie die Bestandteile zu finden sind, wurde abgekapselt in einer separaten Dominokontrolle modelliert. Jeder Dominostein kann auf die Dominokontrolle zugreifen und sich somit im Bild lokalisieren. Abbildung 3.1 zeigt eine Darstellung der Dominokontrolle mit dem Dominomodell. Bei den Dominosteinen handelt es sich um geometrische Objekte, die wiederum zur Geometrie des Systems gehören. Jedes geometrische Objekt muss seitens der Modellierung zwangsweise über eine Kontrolle verfügen. Jeder Dominostein besitzt zwei Hälften, in dem die Pips enthalten sind. Nach aussen ist das Modell jedoch als eine Einheit definiert, d. h. es existiert keine Unterscheidung

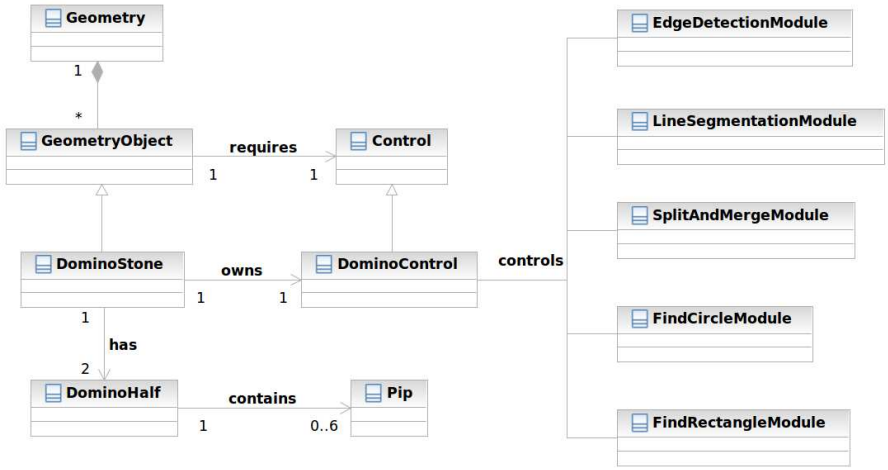


Abbildung 3.1: Grafische Übersicht der Vorverarbeitung. Das Bild zeigt das Zusammenspiel des Dominomodells mit der Dominokontrolle und gibt eine Übersicht der einzelnen Vorverarbeitungsschritte.

zwischen dem Dominostein 6-3 und dem Dominostein 3-6. Aus den vorhandenen Modulen wählt die Dominokontrolle die zur Suchstrategie passenden aus. Zuerst wird eine Kantendetektion angewendet, danach die Linien extrahiert und zum Schluss die Segmentierungsobjekte erstellt. Die folgenden Abschnitte beschreiben die Inhalte der Module der Dominokontrolle. Exemplarisch wird für Abbildung 3.2 dabei die Ausgabe der Schritte an einem Bild demonstriert.

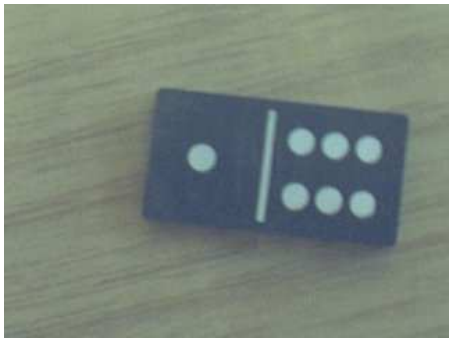


Abbildung 3.2: Beispielbild eines Dominosteins für die Vorverarbeitung

100	100	100	100	100
100	100	100	100	100
0	0	0	0	0
-100	-100	-100	-100	-100
-100	-100	-100	-100	-100

100	100	100	100	100
100	100	100	78	-32
100	92	0	-92	-100
32	-78	-100	-100	-100
-100	-100	-100	-100	-100

100	100	100	-32	-100
100	100	92	-78	-100
100	100	0	-100	-100
100	78	-92	-100	-100
100	32	-100	-100	-100

Abbildung 3.3: Masken des Nevatia–Babu–Operators. Links ist die Maske für 0° zu sehen, in der Mitte für 30° und rechts diejenige für 60° Die dargestellten Masken wurden aus [Har96] entnommen.

3.2 Kantendetektion

Rechtecke und Kreise bestehen aus Linien beziehungsweise Liniensegmenten. Um die benötigten Linien im Bild zu finden, müssen geeignete Merkmale durch Kantendetektion ausfindig gemacht werden. Das Modul zur Kantendetektion verwendet die Klasse *Nevedge* aus PUMA. Diese ist im Rahmen der Dissertation von Harbeck [Har96] entwickelt worden und wendet den Nevatia–Babu Maskenoperator [NB80] auf ein Bild an. Abbildung 3.3 zeigt eine grafische Darstellung der verwendeten Masken. Insgesamt können durch Drehung der drei Masken um 90°, 180° und 270° zwölf verschiedene Kantenrichtungen ermittelt werden. Das Resultat dieser Kantendetektion auf das Eingabebild aus Abbildung 3.2 ist in Abbildung 3.4 zu sehen.

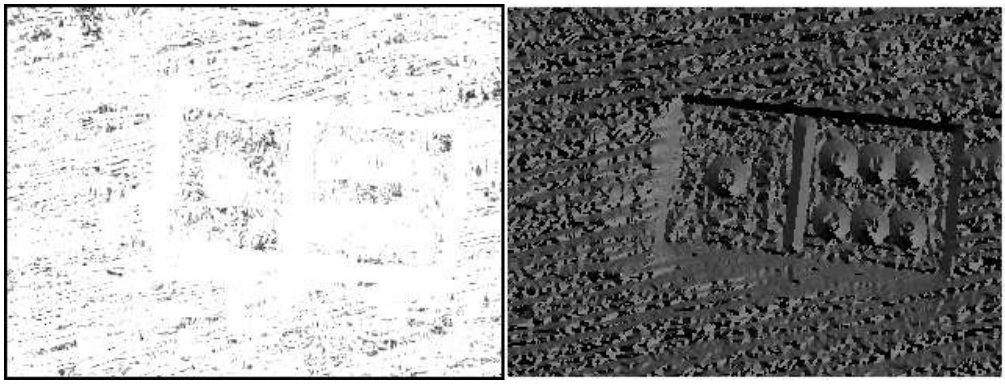


Abbildung 3.4: Resultat der PUMA Kantendetektion. Als Eingabebild wurde Abbildung 3.2 verwendet.

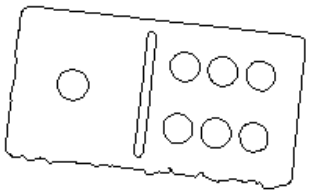


Abbildung 3.5: Resultat der PUMA Linienextraktion. Die Abbildung zeigt die Extraktion der Linien exemplarisch für das Eingabebild aus Abbildung 3.4.

3.3 Linienextraktion

Das Modul zur Linienextraktion verwendet die Funktion *HystLine* aus PUMA. In einem interaktiven Programm-Modus wird eine Kantenverfolgung durchgeführt und das Ergebnis in ein Segmentierungsobjekt gespeichert. Ein Beispiel für die aus dem Kantenbild in Abbildung 3.4 ermittelten Linien ist in Abbildung 3.5 dargestellt. Die gefundenen Linien liegen als zusammengesetzte Repräsentationen vor, deren Teile miteinander verbunden sind (engl. glued lines, in PUMA die Klasse *GluLineRep*). Eine Detailansicht der Liniensegmente ist in Abbildung 3.5 zu sehen.

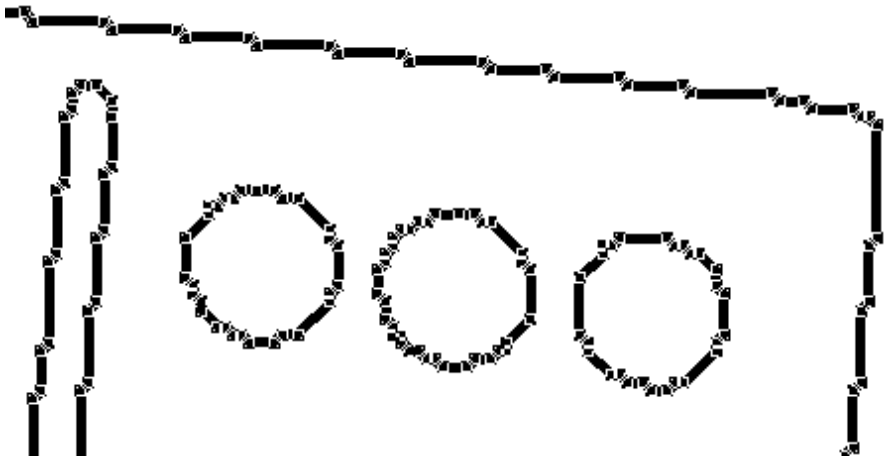


Abbildung 3.6: Detailansicht der zusammengesetzten Linienrepräsentationen.

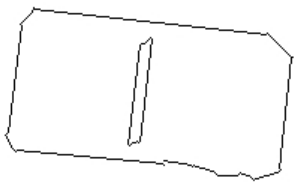


Abbildung 3.7: Resultat des Split & Merge-Moduls. Das Bild zeigt die zerteilten/zusammengesetzten Linien für die Liniensegmente aus Abbildung 3.5.

3.4 Split & Merge

Die Ergebnisse der Linienextraktion liegen als zusammengesetzte Repräsentationen in den Segmentierungsobjekten vor. Die Aufgabe des Split & Merge-Moduls besteht anschliessend darin, die einzelnen Bestandteile der Segmentierungsobjekte zu separieren (engl. split), zusammenzufügen (engl. merge) oder zu verwerfen. Abbildung 3.7 zeigt das Resultat des Split & Merge-Moduls angewendet auf die Liniensegmente aus Abbildung 3.5. Eine Detailansicht des gleichen Ausschnitts wie in Abbildung 3.6 nach Anwendung des Split & Merge-Moduls ist in Abbildung 3.8 zu sehen. Verworfene Segmentierungsobjekte wurden in Rot eingezeichnet, grüne

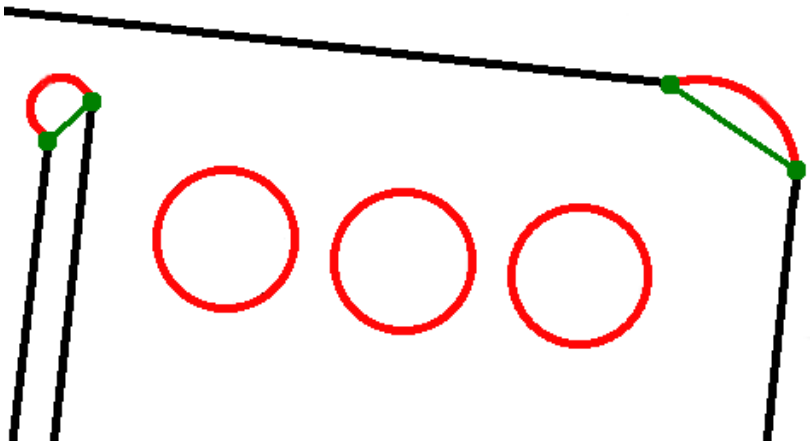


Abbildung 3.8: Detailansicht der Split & Merge-Segmente. Verworfene Objekte sind in Rot eingezeichnet, neu hinzugefügte Objekte in Grün.

Objekte repräsentieren neu hinzugefügte Elemente. In der Abbildung ist erkennbar, das sämtliche geschlossenen Kreise entfernt und nicht geschlossene Kreiselemente durch Linien ersetzt werden. Zur Verdeutlichung der Arbeitsweise des Moduls ist

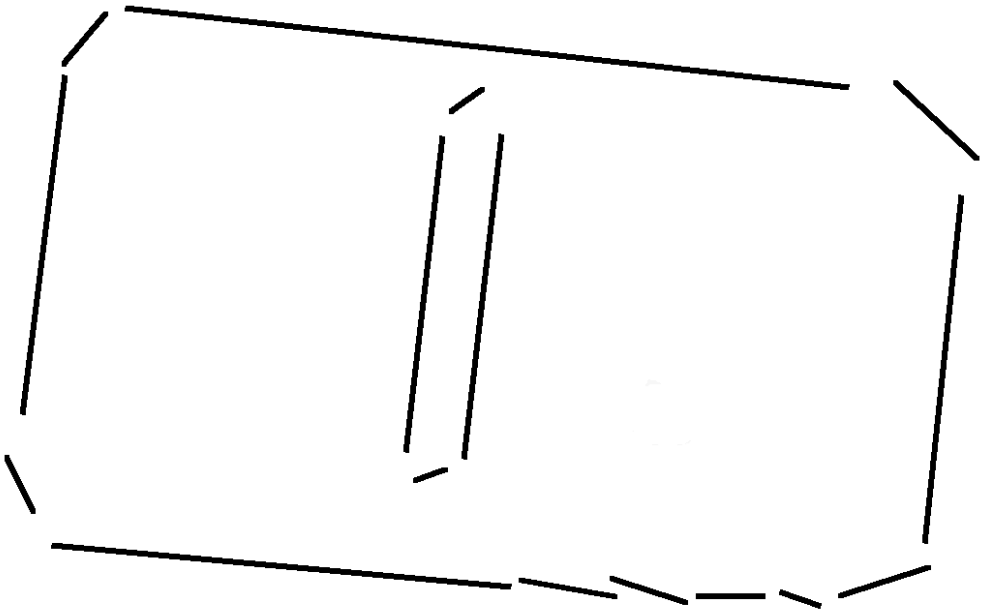


Abbildung 3.9: Explosionszeichnung des Resultats des Split & Merge-Moduls. Die Abbildung zeigt die einzelnen Bestandteile des Segmentierungsergebnisses aus Abbildung 3.7.

in Abbildung 3.9 eine Explosionszeichnung des Resultats zu sehen. In der Darstellung ist erkennbar, wie innerhalb des Moduls die kleinen Liniensegmente zu vollständigen Seitenkanten des ursprünglichen Dominosteins zusammengefasst wurden. Auffallend ist zudem, dass sich die untere Kante des Dominosteins auch nach Anwendung des Moduls noch in mehrere kleinere Linien aufteilt, deren Richtungen unterschiedlich stark von der Hauptrichtung der eigentlichen Kante abweichen. In den Experimenten mit dem Modul hat sich herausgestellt, dass diese zerlegten Linienbereiche mit dem Schattenwurf des Dominosteins und der Textur der Oberfläche zusammenhängen und sich auch durch Einstellung der Schwellwerte nicht eliminieren lassen. Daher wurde im Rechteckfinder eine zusätzliche Hilfsfunktion implementiert, die diesen Spezialfall behandeln kann.

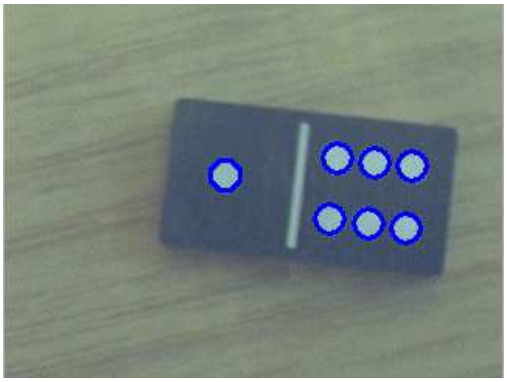


Abbildung 3.10: Resultat des Kreisfinder-Moduls. Das Bild zeigt die gefundenen Kreise auf den segmentierten Linien (vgl. Abbildung 3.5).

3.5 Kreisfinder

Das Kreisfinder-Modul arbeitet auf den Segmentierungsergebnissen der Linienextraktion aus Abschnitt 3.3. Über eine in Puma bereits implementierte Funktion lässt sich überprüfen, ob es sich bei einem vorliegenden Liniensegment um einen Kreis handelt. Ein Ergebnisbild des Moduls ist in Abbildung 3.10 dargestellt. Bei unterschiedlichen Farben und Formen liefert die Funktion gute Ergebnisse (Recall = 95.53% und Precision = 93.66%), jedoch existieren auch Dominosteine mit reflektierenden Pips. Diese stellen je nach vorhandenen Lichtverhältnissen ein Problem dar, wie in Abbildung 3.11 zu sehen ist. Daher wurde für die Experimente eine Lichtbox verwendet (siehe Abschnitt 5.1).

Abbildung 3.11: Links: Eingabebild. Rechts: Ausgabe des Kreisfinder-Moduls. Gefundene Kreise sind mit blauer Farbe dargestellt. Schwarze Dominosteine mit goldenen Pips stellen in Abhängigkeit von der jeweiligen Beleuchtung ein Problem dar.

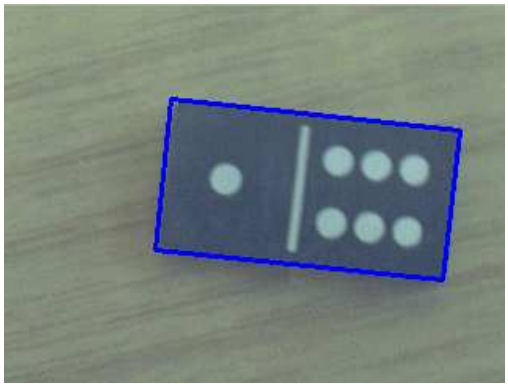


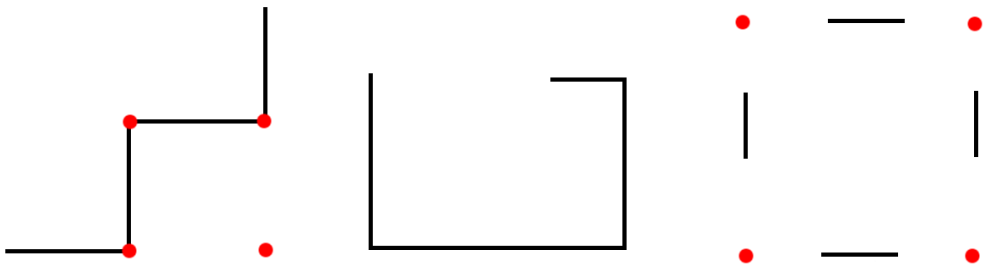
Abbildung 3.12: Resultat des Rechteckfinder-Moduls. Das Bild zeigt die auf den zerteilten segmentierten Linien (vgl. Abbildung 3.7) gefundenen Rechtecke.

3.6 Rechteckfinder

Bei der Literaturrecherche fand sich keine adäquate Lösung für einen Rechteckfinder. Weder in PUMA, noch in einer anderen Bibliothek fanden sich Ansätze zu dieser Problemstellung auf Basis von Segmentierungsobjekten, daher wurde ein eigener Ansatz entwickelt, implementiert und in PUMA integriert. Ein Ergebnisbild des Rechteckfinder-Moduls ist in Abbildung 3.12 dargestellt. Der Rechteckfinder iteriert über alle segmentierten Linien und versucht, vier Linien zu finden, die den Kriterien eines Rechtecks entsprechen. Ein Struktogramm des Rechteckfinders befindet sich in Abbildung C.1 aus Anhang C. Als erstes wird eine Hilfsfunktion *combineSmallLines* (Abbildung C.2 in Anhang C) aufgerufen, welche die eventuell vorhandenen kleinen Liniensegmente des Split & Merge Moduls zusammenfassen kann (vgl. Abschnitt 3.4).

Die combineSmallLines-Funktion iteriert über alle Linien, die in Relation zu Bildbreite kleiner sind als ein Schwellwert. Dabei werden Linienpaare gesucht, die parallel sind und deren Abstand zueinander innerhalb eines festgelegten Radius liegt. Wurde ein solches Linienpaar gefunden, wird eine neue Linie aus denjenigen beiden Punkten der beiden Linien gebaut, die den weitesten Abstand zueinander haben. Die neue längere Linie wird der Menge der Linien hinzugefügt und die beiden kürzeren Linien entfernt. Die Funktion ruft sich solange rekursiv auf, bis keine Linienpaare mehr gefunden werden.

Anschliessend beginnt die eigentliche Arbeit des Rechteckfinder-Moduls und es werden zu einer als Start ausgewählten Linie eine Parallele und zwei Senkrechte gesucht. Da in diesem Prozess sehr viele Linien mehrfach betrachtet und verglichen werden, ist der Aufwand der Funktion entsprechend hoch. Um dem entgegen zu wirken, wurden zahlreiche Abbruchkriterien festgelegt, von denen nur ein geringer



(a) (b) (c)

Abbildung 3.13: Abbruchkriterien des Rechteckfinder-Moduls. Links: Vergleich der Halbräume mit den in Rot eingefärbten Schnittpunkten. Mitte: Längenvergleich gegenüberliegender Linien. Rechts: Abstand der Schnittpunkte zu den Linien.

Teil im Struktogramm in Abbildung C.1 dargestellt sind. Neben den dargestellten Abbruchkriterien für die Parallelität der ersten zur zweiten Linie, der Orthogonalität der ersten zur dritten Linie und der Orthogonalität der ersten zur vierten Linie wurden noch folgende Kriterien hinzugefügt:

- Um ein Rechteck aus treppenförmig angeordneten Linien zu vermeiden, wurden zusätzlich die Halbräume der Linien mit den berechneten Schnittpunkten verglichen und als Abbruchkriterium hinzugezogen. Abbildung 3.13(a) zeigt ein Beispiel für eine treppenförmige Konstellation. Das ohne Abbruchkriterium resultierende Rechteck ist durch die rot eingetragen Punkte dargestellt.

- Damit ein Rechteck nicht aus allen möglichen Linien mit korrekten Winkeln konstruiert wird, wurde ein weiteres Kriterium hinzugefügt, dass die Länge der Linien berücksichtigt. In Abbildung 3.13(b) ist ein Beispiel zu sehen, in dem Winkel-konforme Linien trotzdem kein Rechteck bilden, da die gegenüberliegende Linie zu kurz beziehungsweise zu lang ist. Als Schwellwerte für dieses Kriterium wurden 25 % beziehungsweise 400 % gewählt.

- Ein weiteres Kriterium behandelt Linien, die sowohl von der Länge der gegenüberliegenden Linien, als auch von den Winkeln zueinander passende Kandidaten wären. Es existieren Konstellationen, die diese Eigenschaften aufweisen, bei denen allerdings die Linien zu weit voneinander entfernt sind. Ein Beispiel dafür ist in Abbildung 3.13(c) dargestellt. Das Abbruchkriterium vergleicht daher die Abstände der Schnittpunkte, in der Abbildung rot eingezeichnet, mit den Linien. Überschreitet einer dieser acht Werte einen Schwellwert, wird die aktuelle Iteration abgebrochen.

Der Rechteckfinder liefert somit alle Rechtecke, die anhand der im Bild vorliegenden Linien gefunden werden konnten. Die Entscheidung, ob ein gefundenes Rechteck über spezielle Eigenschaften verfügt, wie beispielsweise ein Seitenverhältnis von zwei zu eins, und sich somit als Teil eines Dominosteins eignet, wird später durch die Dominokontrolle getroffen.

3.7 Segmentierungsobjekte

Nachdem beschrieben wurde, wie die Segmentierungsobjekte im Bild gefunden und extrahiert werden, folgt nun eine Darstellung der wichtigsten Segmentierungsobjekte. Von den vielen Segmentierungsobjekten sind zwei besonders interessant, da sie für die Erkennung der Dominosteine essentiell sind. Dabei handelt es sich um Kreise und Rechtecke, die den Pips und den Dominosteinen zugeordnet werden müssen.

3.7.1 Kreis

Die Klasse *Arc* dient in PUMA zur Darstellung von Kreisbogensegmenten beziehungsweise Vollkreisen und wurde als Datenstruktur für die segmentierten Kreise gewählt. Für die Unterscheidung der beiden existiert eine Funktion *isClosed*, die genau dann "Wahr" zurück liefert, wenn der Arc ein Vollkreis ist. Bei den Vollkreisen handelt es sich immer um echte Kreise, d. h. Ellipsen werden durch Kreise approximiert. Ein Beispiel dafür ist in Abbildung 3.14 zu sehen. Jeder Kreis verfügt

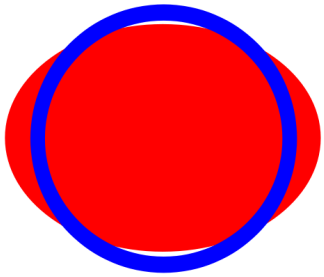


Abbildung 3.14: Einpassen eines Kreises in eine Ellipse.

über die Attribute *centerPoint*, *radius* und *circulation*. Das Attribut centerPoint beinhaltet den Mittelpunkt des Kreises und der Radius ist im Attribut radius gespeichert. Zusätzlich beschreibt das Attribut circulation die Umlaufrichtung des

Kreises. Es existieren noch weitere Attribute wie Flächeninhalt, Umfang uvm., die jedoch für die Dominosteine nicht relevant sind.

3.7.2 Rechteck

Als Datenstruktur für die Rechtecke wurde die Hippos-Klasse *H_Rectangle* verwendet. Die Klasse H_Rectangle war ursprünglich nur für achsenparallele Rechtecke ausgelegt und wurde daher um einen Wert erweitert, der die Orientierung des Rechtecks beinhaltet. Eine geometrische Darstellung der erweiterten Klasse ist in Abbildung 3.15 zu sehen. Der in der Abbildung dargestellte Winkel alpha bein-

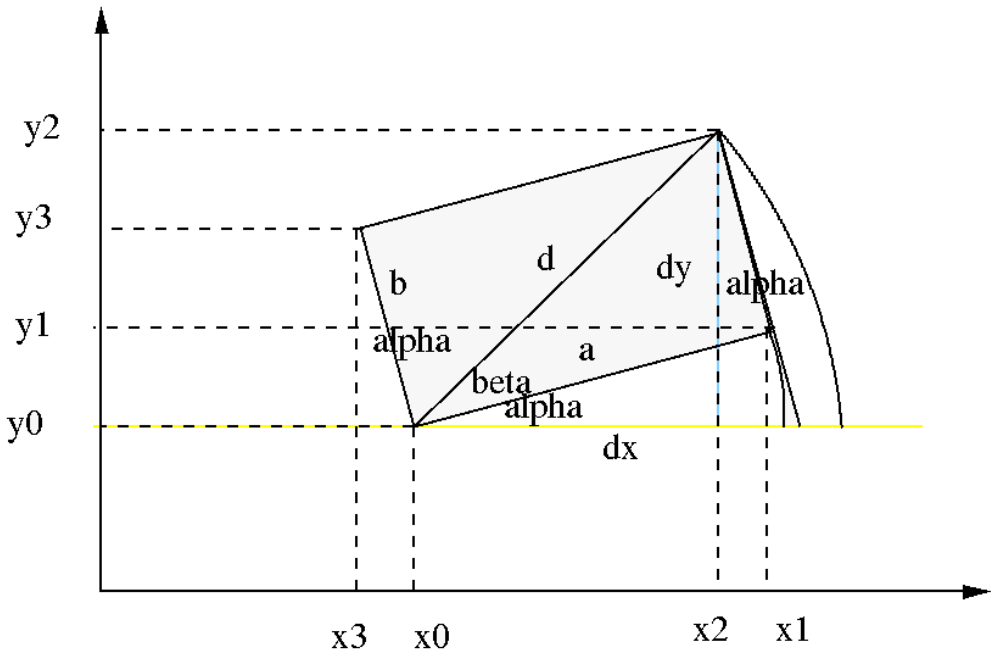


Abbildung 3.15: Geometrische Darstellung der Klasse H_Rectangle.

haltet die Orientierung des Rechteck, die in dem Attribut *orientation* gespeichert ist. Darüber hinaus existiert noch ein Attribut *ratio* für das Seitenverhältnis der längeren zur kürzeren Seite und zwei Attribute *width* (in der Abbildung als *a* eingetragen) und *height* (in der Abbildung als *b* eingetragen), welche die Breite und Höhe des Rechtecks beinhalten. Das Attribut *area* enthält den Flächeninhalt. Die wichtigsten Funktionen sind der []-*Operator*, die Funktion *rotate* und die Funktion *contains*. Der []-Operator erlaubt den Zugriff auf die vier Eckpunkte des Rechtecks. Die rotate-Funktion ermöglicht für einen gegeben Winkel eine Rotation des

Rechtecks um seinen Mittelpunkt und die contains-Funktion eignet sich, um festzustellen, ob ein anderes geometrisches Objekt im Rechteck enthalten ist.

Kapitel 4

Zuordnung

Das folgende Kapitel beschreibt den Lösungsansatz des gewicht-maximalen bipartiten Zuordnungsproblems und dessen Darstellung in der grafischen Benutzeroberfläche. Zur Lösung des Problems wird in Abschnitt 4.1 die Ungarische Methode auf Basis von Zuordnungs-Matrizen (Abschnitt 4.1.1) vorgestellt. In der Literatur wird die Idee des Algorithmus oftmals nur kurz behandelt und ist für Leser ohne Hintergrundwissen der Graphentheorie und -paarung nur schwer nachvollziehbar. Daher wird in Abschnitt 4.1.2 das Konzept des Algorithmus separat aufgeführt und erklärt. Anschliessend erfolgt in Abschnitt 4.2 eine Beschreibung der grafischen Benutzeroberfläche und eine Darstellung der Bewertung der Hypothesen.

4.1 Ungarische Methode

Das Problem der Bestimmung einer Zuordnung mit maximaler Bewertung bezüglich einer Abstandsfunktion (siehe Abschnitt 2.3) kann direkt auf das gewicht-maximale bipartite Zuordnungsproblem zurückgeführt werden (siehe [Win94] Kapitel 5.6.1). Zur Lösung des gewicht-maximalen bipartiten Zuordnungsproblems existiert die Ungarische Methode (engl. hungarian algorithm), auch Kuhn-Munkres-Algorithmus [Kuh55, Mun57] genannt.

Die Ungarische Methode wurde in [Win94] von Winzen dazu verwendet, Korrespondenzen in verschiedenen Segmentierungsobjekten aus unterschiedlichen Ansichten eines Objekts zu finden. Abbildung 4.1 zeigt ein Beispiel für die Korrespondenzzuordnung mit der Ungarischen Methode. Die Implementierung erfolgte auf Grundlage von [Jun90] und ein Nachweis der Korrektheit und die Bestimmung der Komplexität $O(n^3)$ sind in [Jun90] Kapitel 11 zu finden. Das entwickelte Verfahren von Winzen verwendet dabei die Datenstruktur *Dictionary* aus NIHCL als Basis und ist in der Programmiersprache C++ geschrieben. Bei einem Dictionary han-

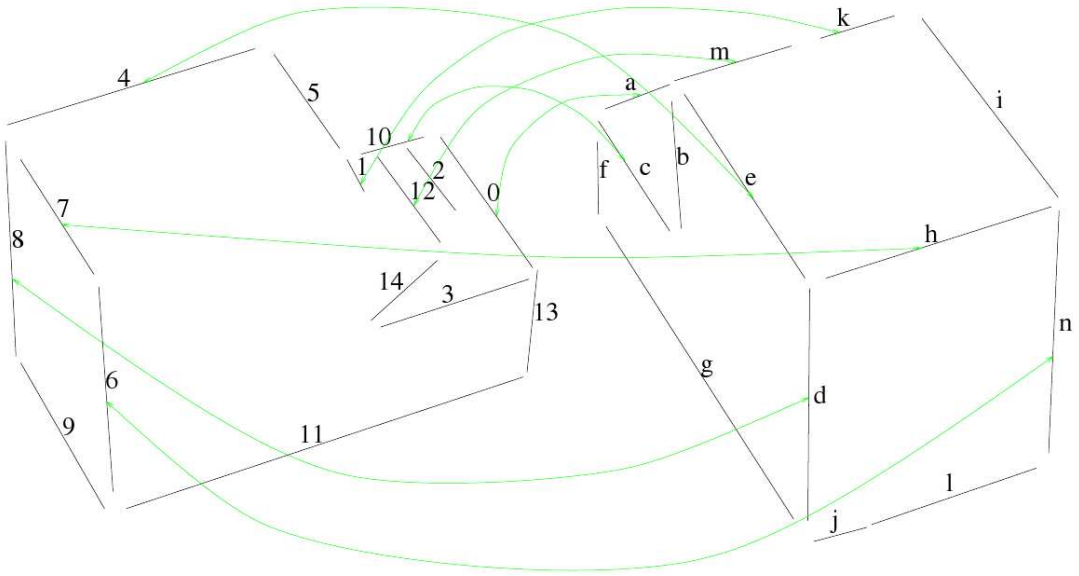


Abbildung 4.1: Korrespondenzzuordnung mit der Ungarischen Methode. Die Abbildung zeigt zwei Ansichten eines L-förmigen Polyeders, deren Korrespondenz von Winzen mit dem Ungarischen Algorithmus bestimmt wird. (Quelle: [Win94]).

delt es sich um eine Menge von Assoziationen, die aus Key-Value-Paaren des Typs Object (siehe Abbildung 2.5) bestehen.

Ein sehr gutes Tutorial zum Verständnis der Ungarischen Methode befindet sich in [Top09]. Das Tutorial erklärt schrittweise die einzelnen Phasen des Algorithmus und illustriert diese mit Bildern. Die dort beschriebene Implementierung verwendet ausschließlich C++ und besitzt ebenfalls eine Komplexität von $O(n^3)$. D. h. sollten PUMA und damit auch NIHCL im weiteren Verlauf des Projekts durch eine andere Bildverarbeitungsbibliothek ersetzt werden, müssten keine Änderungen an der Ungarischen Methode vorgenommen werden. Daher wurde diese Implementierung anstatt der NIHCL-Variante von Winzen für die Zuordnung verwendet.

4.1.1 Zuordnungs-Matrizen

Wie bereits in Abschnitt 2.5.1 beschrieben und in Abbildung 2.7 grafisch dargestellt wurde, lässt sich jeder bipartite Graph durch eine Matrix repräsentieren. Die Werte dieser Zuordnungs-Matrix werden dabei von der Kosten- beziehungsweise Straffunktion berechnet. Die Aufgabe der Zuordnung und damit der Ungarischen Methode besteht demnach darin, für eine gegebene Zuordnungs-Matrix das Ergebnis in Form einer Adjazenzmatrix zu bestimmen. Abbildung 4.2 zeigt ein Beispiel

$$\begin{pmatrix} 33 & 67 & 91 & 3 & 11 & 54 & 81 \\ 92 & 12 & 24 & 77 & 4 & 0 & 13 \\ 6 & 81 & 43 & 95 & 60 & 11 & 12 \\ 22 & 42 & 21 & 0 & 54 & 90 & 3 \\ 35 & 97 & 80 & 75 & 31 & 5 & 75 \\ 11 & 57 & 7 & 5 & 50 & 63 & 89 \\ 54 & 23 & 62 & 14 & 97 & 13 & 43 \end{pmatrix} \quad \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Abbildung 4.2: Darstellung einer Zuordnungs-Matrix mit ihrer korrespondierenden Adjazenzmatrix.

für eine beliebige Zuordnungs-Matrix auf der linken Seite und ihrer korrespondierenden Adjazenzmatrix auf der rechten Seite.

Für jedes im Bild gefundene geeignete Segmentierungsobjekt muss für jedes Modell eine solche Zuordnungsmatrix durch die Abstandsfunktion berechnet werden und anschliessend mit der Ungarischen Methode zugeordnet werden. Somit ergibt sich für den Aufwand eine Komplexität $O(|S|\ |M|)$, wobei $|S|$ die Anzahl der geeigneten Segmentierungsobjekte ist und $|M|$ die Anzahl der Modelle[1]. Aus der Adjazenzmatrix lässt sich die Bewertung des Modells für das Segmentierungsobjekt einfach errechnen, in dem alle in der Adjazenzmatrix mit 1 belegten Positionen den Werten der Zuordnungs-Matrix zugeordnet, aufaddiert und durch die Dimension der Matrix dividiert werden. Für das Beispiel aus Abbildung 4.2 entspricht dies einer Bewertung von $\frac{91+92+95+90+97+89+97}{7} = 93\,\%$.

4.1.2 Augmentierung

Augmentierende Pfade wurden bereits in Abschnitt 2.5.1 definiert und sind der Dreh- und Angelpunkt der Ungarischen Methode. Zur Erklärung der Augmentierung (Vergrößerung) fehlen jedoch noch zwei weitere Definition, die für das Konzept des Algorithmus benötigt werden:

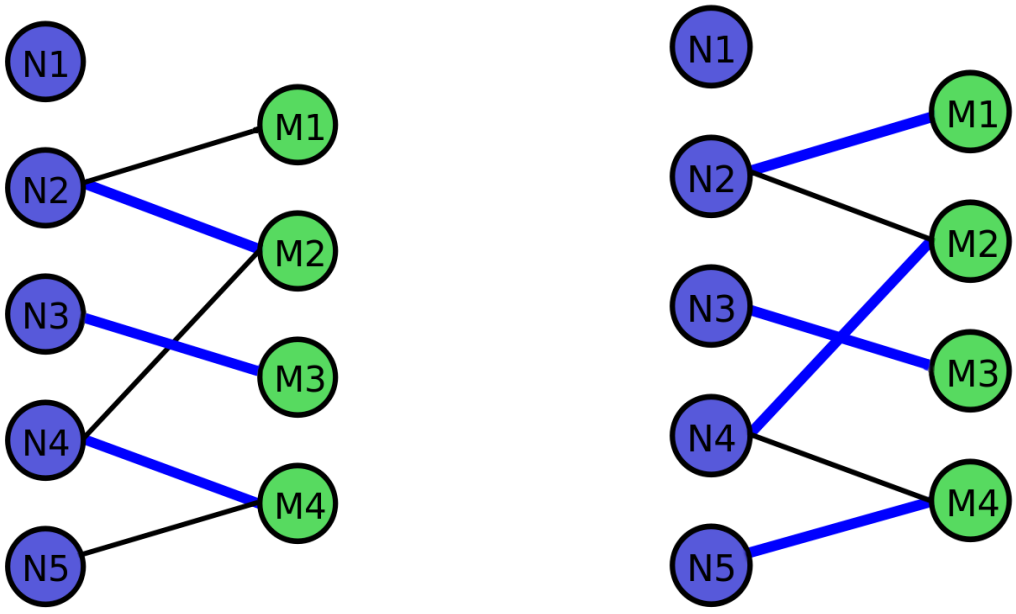
Kennzeichnung Sei $G = (N, E)$ ein bipartiter Graph mit $N = n_1 \cup n_2 \wedge n_1 \cap n_2 = \varnothing$ und einer Gewichtsfunktion $w : E \to \mathbb{R}$. Eine *Kennzeichnung* (engl. labeling) ist eine Funktion $l : E \to \mathbb{R}$, die jedem Knoten von G eine *Kennzahl* (engl. label) zuordnet. Eine Kennzeichnung heisst genau dann *zulässig*, wenn gilt: $\forall\, a \in n_1\ \forall\, b \in n_2 : l(a) + l(b) \geq w(e(a, b))$.

Gleichheitsgraph Sei $G = (N, E)$ ein bipartiter Graph mit einer Gewichtsfunktion $w : E \to \mathbb{R}$ und $G_t = (N, E_t)$ ein Teilgraph von G. Dann ist

[1] Für den Fall, dass der Geometrie nur Dominosteine bekannt sind ist $|M| = 28$.

G_t genau dann ein *Gleichheitsgraph*, wenn gilt: $\forall\, e(a,b) \in E_t : e(a,b) \in E \wedge l_a + l_b = w(e(a,b))$ Anders ausgedrückt: Ein Gleichheitsgraph G_t enthält nur diejenigen Kanten eines bipartiten Graphen G, die den Knoten eine zulässige Kennzeichnung erlauben.

Zu Beginn der Ungarischen Methode werden ein initialer Gleichheitsgraph und eine initiale Zuordnung erstellt. Handelt es sich bereits um eine vollständige Zuordnung, terminiert der Algorithmus mit einer bestmöglichen Zuordnung. Falls die Zuordnung nicht vollständig ist, kommt die zentrale Idee der Ungarischen Methode zum Einsatz. Die Augmentierung setzt sich aus zwei Phasen zusammen. In der ersten Phase werden augmentierende Pfade gesucht. Wurde ein augmentierender Pfad gefunden, kann der Gleichheitsgraph vergrößert werden, indem alle Belegungen entlang des augmentierenden Pfades invertiert werden. Dadurch entsteht eine zusätzliche Kante im Gleichheitsgraph. Anhand des Beispiels in Abbildung 4.3 kann die Erweiterung eines augmentierenden Pfades nachvollzogen werden. Belegte



(a) (b)

Abbildung 4.3: Darstellung der Erweiterung eines augmentierenden Pfades anhand eines Beispiels. Durch Invertieren aller Belegungen des augmentierenden Pfades $M_1, N_2, M_2, N_4, M_4, N_5$ in Abbildung 4.3(a) vergrößert sich die Anzahl der belegten Kanten um eins (siehe Abbildung 4.3(b)).

Kanten sind in der Abbildung in Blau eingezeichnet, freie Kanten in Schwarz. Die

zweite Phase des Algorithmus kommt dann zum Einsatz, wenn kein augmentierender Pfad gefunden werden konnte und die Zuordnung noch nicht vollständig ist. Die Idee der Methode besteht darin, die Kennzeichnung der Knoten solange zu verändern, bis mindestens ein neuer augmentierender Pfad gefunden werden konnte. Die Aktualisierung der Kennzeichnung erfolgt über einen Parameter Δ. Sei $G = (N, E)$ ein bipartiter Graph mit $N = n_1 \cup n_2 \wedge n_1 \cap n_2 = \varnothing$. Dann ist

$$\Delta = \min_{a \in n_1, b \in n_2} l(a) + l(b) - w(e(a, b))$$

Für die Kennzahl $l(x) = l'(x)$ eines Knoten $x \in N$ gilt

$$l'(x) = \begin{cases} l(x) + \Delta, & x \in n_1 \\ l(x) - \Delta, & x \in n_2 \\ l(x), & sonst \end{cases}$$

Dadurch, dass sich die Kennzahl in beiden Knoten der Teilgraphen identisch vergrößert beziehungsweise verkleinert, bleibt die Kennzeichnung zulässig, da nach wie vor die Bedingung $\forall\, a \in n_1\ \forall\, b \in n_2 : l(a) + l(b) \geq w(e(a, b))$ erfüllt ist.

Im Gegensatz zu vielen anderen Algorithmen findet also keine direkte Wertebestimmung statt, sondern eine Schätzung, die ausschließlich der Generierung von neuen Erweiterungsmöglichkeiten des Gleichheitsgraphen dient. Eine Demonstration der Anwendung der Ungarischen Methode auf einen bipartiten Graphen befindet sich in Anhang D in Abbildung D.2.

4.2 Grafische Benutzeroberfläche

Um die Vorteile des in dieser Arbeit entwickelten Verfahrens optimal zur Geltung zu bringen, eignet sich besonders die grafische Darstellung in einer GUI. Sämtliche in einem Eingabebild gefundenen Segmentierungsobjekte lassen sich konvertieren und mit Qt darstellen. Die Segmentierungsobjekte können auch von PUMA wieder in ein Bild eingetragen werden, jedoch ermöglicht die Umwandlung in Qt-Objekte diverse Vorteile, die in Abschnitt 4.2.1 vorgestellt werden.

Für die Zuordnung ergibt sich durch die Verwendung einer grafischen Benutzeroberfläche die Möglichkeit, für ein Segmentierungsobjekt alle Hypothesen, deren Darstellung als Zuordnungs-Matrizen und seine Bewertung übersichtlich darzustellen. Ein Beispiel dafür ist in Abbildung 4.4 zu sehen. Für jedes Modell wird dabei für alle Segmentierungsobjekte eine separate Darstellung erstellt, so dass es dem Benutzer des Systems möglich ist, die unterschiedlichen Hypothesen miteinander zu vergleichen und die Bewertung des Systems zu verstehen.

Das Wissen, welches Segmentierungsobjekt welchem Modellbestandteil zugeordnet wurde, lässt sich zudem farblich visualisieren. Diese Möglichkeit wird in Abschnitt 4.2.2 vorgestellt.

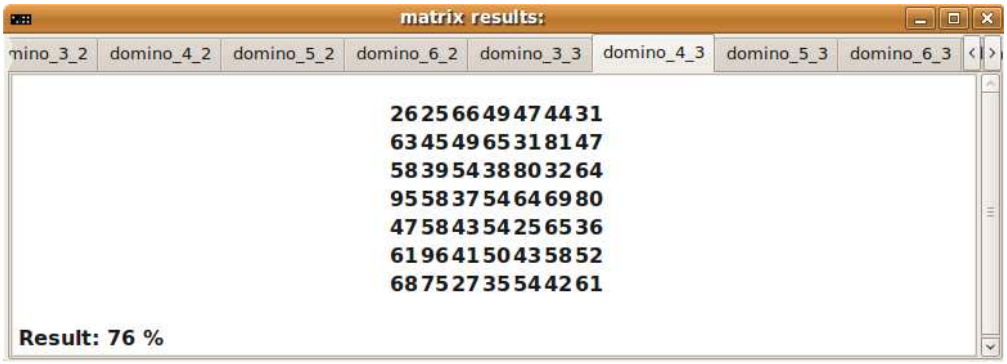


Abbildung 4.4: Grafische Darstellung der Übersicht der Hypothesen in der GUI. Zu jeder Hypothese ist eine Darstellung der Zuordnungs-Matrix und des Ergebnisses der Ungarischen Methode verfügbar.

4.2.1 Repräsentationen der Segmentierungsobjekte

Wie bereits erwähnt, ist es auch mit der Bildverarbeitungsbibliothek PUMA möglich, Segmentierungsobjekte wieder in ein Bild zu übertragen oder mit einem entsprechenden Programm anzuzeigen[2]. Der Vorteil der Konvertierung der Segmentierungsobjekte in Qt-Objekte liegt darin, dass nicht mehr auf Bildpixeln sondern auf geometrischen Objekten gearbeitet wird und trotzdem kein zusätzliches Programm zur Darstellung notwendig ist. Darüber hinaus lassen sich sämtliche Eigenschaften der Segmentierungsobjekte übernehmen, sodass zu jedem Objekt alle wichtigen Details erhalten bleiben. Außerdem ist es nun möglich, für jedes Objekt seine Eigenschaften in der GUI anzuzeigen.

Für die Umsetzung in Qt wurden die Klassen QGraphicsView und QGraphicsScene verwendet. Der QGraphicsView stellt die QGraphicsScene dar, in der die Objekte enthalten sind. Die wichtigsten Objekte sind in Tabelle 4.1 aufgeführt. Die Ergebnisse des Rechteckfinders werden, falls der Benutzer die Darstellung von Zwischenergebnissen wünscht, als *GraphicsPath* gezeichnet. Beim GraphicsPath handelt es sich um einen geschlossenen rechteckigen Linienzug, der seine Farbe von Blau in Rot ändert wenn die Maus über ihm schwebt aber sonst über keine weiteren Eigenschaften verfügt. Ein weiteres Zwischenergebnis sind die Ergebnisse des Kreisfinders in Form der *GraphicsEllipse*. Diese sind analog zum GraphicsPath aufgebaut, verwenden jedoch Kreise anstelle der rechteckigen Linienzüge. Wurde einem Objekt eine oder mehrere Hypothesen zugeordnet, beispielsweise im Falle eines potentiellen Dominosteins, wird die Klasse *GraphicsPolygon* verwendet. Ähnlich zur Klasse GraphicsPath handelt es sich um einen geschlossenes rechtecki-

[2]Beispielsweise das Programm X-Fig [Xfi09].

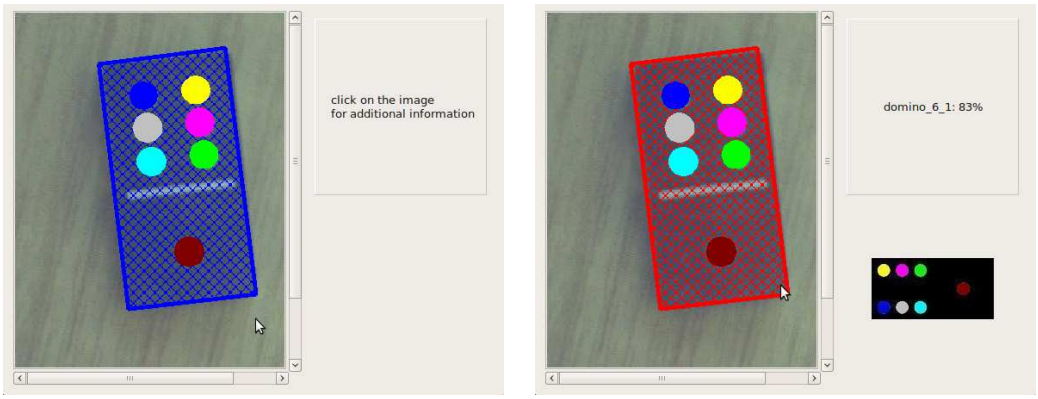
Segmentierungsobjekt	Repräsentation in der GUI
Rechteck	GraphicsPath
Kreis	GraphicsEllipse
Dominostein	GraphicsPolygon
Pip	FilledGraphicsEllipse

Tabelle 4.1: Repräsentationen der Segmentierungsobjekte in der GUI.

ges Polygon, der die Bounding-Box eines geometrischen Objekts darstellt. Analog zu den beiden vorangegangenen Klassen ändert das Polygon seine Farbe, zeigt jedoch zusätzlich die am besten bewertete Hypothese an. Außerdem reagiert jedes Objekt der Klasse auf Maus-Klicks und konstruiert ein neues Fenster mit allen zu dem Objekt vorhandenen Details, darunter die bereits oben erwähnten Hypothesen-Bewertungen und Zuordnungs-Matrizen. Als letztes existiert noch die Klasse *FilledGraphicsEllipse*, deren Verwendung in Abschnitt 4.2.2 dargestellt wird. Interagiert der Benutzer mit der GUI, werden die Ereignisse an den jeweiligen Teil der GUI weitergegeben. Bewegt der Benutzer beispielsweise die Maus über die QGraphicsScene, überprüft diese, ob ein in ihr enthaltenes Objekt von der Aktion betroffen ist. Falls dies der Fall ist und das Objekt die Aktion verarbeiten kann, startet das Objekt selbst dann die entsprechende implementierte Reaktion.

4.2.2 Farbliche Zuordnung

Die Wissensrepräsentation durch die Klassenobjekte bringt den Vorteil, dass jedem Pip seine Position innerhalb des Dominosteins bekannt ist und er somit die eindeutige Farbe der Layout-Position abrufen kann. Eine Übersicht aller Layout-Farben befindet sich in Abbildung D.1 in Anhang D. Somit ist es für jeden im Bild gefundenen Dominostein möglich, dessen Kreise im Bild entsprechend der Zuordnung einzufärben. Dadurch ist es dem Benutzer des Systems möglich, die Zuordnung und die Auswahl der Hypothese nachzuvollziehen. Abbildung 4.5 zeigt ein Beispiel für die farbliche Darstellung der Zuordnung der Kreise des Bildes zu den Kreisen des am besten bewerteten Modells. Als Datenstruktur für die farbigen Kreise dient die Klasse *FilledGraphicsEllipse*. Im Gegensatz zu den anderen grafischen Repräsentation der Segmentierungsobjekte findet keine Farbänderung statt. Auf diese Weise ist es möglich, auch bei mehreren im Bild gefundenen Dominosteinen allein anhand der Farben die Zuordnung zu erkennen.



(a) (b)

Abbildung 4.5: Farbliche Zuordnung der Elemente eines Dominosteins. Abbildung 4.5(a) zeigt das Ergebnisbild für einen 6-1er Dominostein in der GUI. Erkannte Objekte sind durch blaue Bounding Boxen gekennzeichnet. Bewegt der Benutzer die Maus über eine solche Bounding Box, wird das Klassifikationsergebnis für dieses Objekt angezeigt (siehe Abbildung 4.5(b)).

Kapitel 5

Experimente und Evaluation

Dieser Teil der Arbeit beschäftigt sich mit der Datenakquisition, den durchgeführten Experimenten und den Ergebnissen der Abstandsfunktionen. Als erstes wird in Abschnitt 5.1 die Beschaffung der Testdaten beschrieben und der Aufnahmeprozess dargestellt. In Abschnitt 5.2 erfolgt eine Beschreibung des Versuchsträgers und anschliessend wird in Abschnitt 5.3 die Durchführung der Experimente erörtert. Im letzten Abschnitt 5.4 werden die evaluierten Abstandsfunktionen vorgestellt, aufgeteilt in Kosten- und Straffunktionen.

5.1 Datenakquisition

Sowohl für die Validierung der Ergebnisse als auch für die Evaluation der Abstandsfunktion wird eine große Anzahl an Aufnahmen von Dominosteinen benötigt. Die Erstellung von umfangreichem Bildmaterial ist für viele Verfahren erforderlich, beispielsweise für die Gewinnung von Trainingsdaten für die Objekterkennung. Zu diesem Zweck wurde 2008 an der Universität Koblenz ein Verfahren zur automatischen Objektaufnahme vorgestellt [HP08]. Um von einem Objekt mehrere Aufnahmen aus unterschiedlichen Ansichten zu erhalten, bietet sich die Verwendung eines Drehtellers an. Da für die manuelle Lageveränderung eines Dominosteins ein hoher zeitlicher Aufwand nötig ist, bringt der Einsatz eines Drehtellers eine deutliche Zeitersparnis. Zusätzlich befand sich der Drehteller in einer Lichtbox, um unterschiedliche Beleuchtungsverhältnisse auf den Dominosteinen zu simulieren. Bei der verwendeten Lichtbox handelt es sich um eine *JUST Pantone© Color Viewing box*[1]. Eine dem verwendeten Aufbau ähnliche Konstellation ist in Abbildung 5.1 zu sehen. Für die Datenakquisition der Dominosteinaufnahmen wurde die Lichtbox jedoch vollständig verdunkelt und der dargestellte Roboterarm nicht verwendet. Somit konnten die Daten der verwendeten Lichtquellen ohne zusätzliche

[1] Weitere Information zur Lichtbox finden sich auf http://www.just-normlicht.de

Abbildung 5.1: Darstellung der Lichtbox und des Drehtellers. Die Grafik wurde aus [HP08] entnommen.

Einflüsse gespeichert werden. Zur Erstellung der Aufnahmen existiert eine Software, die als Schnittstelle zum Drehteller, zu den Lichtquellen und zu den Kameras fungiert. Für die Dominosteine wurde eine handelsübliche USB-Kamera verwendet und eine Einstellung von 22–45° pro Rotationsschritt (8, 12 oder 16 Aufnahmen pro Lichtquelle) für den Drehteller gewählt. Die Aufnahmen werden direkt nach dem Entstehungsprozess in die Bilddatenbank der AGAS geladen. Dort wurde ein extra Datensatz für die Dominosteine angelegt der mehrere hundert Aufnahmen umfasst. Eine Übersicht der so entstandenen Aufnahmen ist in Abbildung 5.2 zu sehen. Zu jeder Aufnahme wurde automatisch eine Beschreibung generiert, die neben vielen anderen Attributen die verwendeten Lichtquellen beinhaltet, die Daten der Kamera und die Eigenschaften des Drehtellers. Neben den automatisch generierten Aufnahmen wurden zusätzlich noch manuell erstellte Aufnahmen bei Tageslicht oder Raumbeleuchtung dem Datensatz hinzugefügt.

Abbildung 5.2: Beispielaufnahmen von Dominosteinen unter Verwendung des Drehtellers und der Lichtbox.

5.2 Versuchsträger

Zur Erprobung des entwickelten Systems wurde ein handelsüblicher PC verwendet. Der als Versuchsträger verwendete PC ist mit einem Intel® Core™2 Quad 2.40 GHz Prozessor und 4 Gigabyte Arbeitsspeicher ausgestattet. Als Betriebssystem kommt Ubuntu 9.04 zum Einsatz. Eine weiterführende und ausführliche Beschreibung der Hardware des Versuchsträgers und der installierten Pakete befindet sich im Ordner `/Sonstiges/Versuchsplattform` auf der beiliegenden DVD.

5.2.1 Softwareentwicklungsumgebung

Zur Implementierung des Systems wurde die Programmiersprache C++ verwendet. Für die grafische Darstellung in Form einer GUI wurde zusätzlich Qt ([Sub09], [SB06]) verwendet. Die Bildverarbeitung erfolgte mit Hilfe der Bibliothek PUMA[2] (siehe Abschnitt 2.4). C++ wurde mit dem gcc-Compiler Version 3.4.6 [FSF09] verwendet, Qt in der Version 4 mit qmake Version 4.5.0. Die Installationshinweise zum Gebrauch der Software befinden sich in Anhang A. Eine Dokumentation der entstandenen Klassen und des entwickelten Frameworks befinden sich im Doxygen-Format im Ordner `/Programmcode/DoxygenDoku` auf der beiliegenden DVD.

5.2.2 DominoDiscoverer

Das für diese Arbeit geschaffene Programm *DominoDiscoverer* verfügt über drei unterschiedliche Programm-Modi. Mit dem Konsolen-Aufruf `./DominoDiscoverer -help` lässt sich eine Übersicht der Parameter und Optionen anzeigen. Der erste Modus entspricht dem Standard-Fall und startet die grafische Benutzeroberfläche. Zusätzlich kann dem Programm bereits eine Bilddatei angegeben werden, beispielsweise durch den Aufruf `./DominoDiscoverer Images/real3.jpg`. Die einzelnen Programmabläufe sich grafisch durch die GUI dargestellt, eine erweiterte Ausgabe lässt sich durch aktivieren des Häkchens "show intermediate results" anzeigen. Der zweite und der dritte Programm-Modus dienen der Verarbeitung und Klassifikation von ganzen Ordnern und deren enthaltenen Bildern. Beide Modi werden im folgenden Abschnitt 5.3 separat erklärt und arbeiten nach dem gleichen Verarbeitungsschema wie die GUI, jedoch ohne die Einstellungsmöglichkeiten der GUI für Einzelbilder.

[2] Revision 36561.

5.3 Versuchsdurchführung

Mit dem Konsolen-Aufruf `./DominoDiscoverer -batch <Ordnerpfad> -i3` lässt sich ein ganze Gruppe aller im relativen Ordner-Pfad angegebenen Bilder verarbeiten. Mit der Option *-i* lässt sich zusätzlich die zu verwendende Abstandsfunktion angeben, im obigen Beispiel würde die dritte Funktion verwendet werden. Das Ergebnis des Aufrufs wird in einer Textdatei mit dem gleichen Namen wie der Ordner gespeichert. Hierbei gilt es zu beachten, das der Name des Ordners der Identifikation seines Inhalts dient. Beispielsweise sind alle Dominosteine des Typs 6-3 im Ordner domino_6_3 abgelegt. Der Ablauf der Verarbeitung ist dabei für jedes Bild folgendermassen: Zuerst wird die in Kapitel 3 vorgestellte Vorverarbeitungs-Strategie auf die Bilder angewandt. Anschliessend werden die extrahierten Segmentierungsobjekte in Modellkoordinaten umgerechnet, d. h. rotiert und skaliert, und von der ausgewählten Abstandsfunktion bewertet. Im letzten Schritt führt die Ungarische Methode eine Zuordnung auf Basis von Adjazenzmatrizen durch und das Ergebnis wird für jedes Modell pro Segmentierungsobjekt als einzelne Zeile in den besagten Ordner mit dem gleichen Namen wie die Klasse des Bildes gespeichert. Die Zeilen der Textdateien haben dabei das Format:

```
P1000414.JPG domino_6_5 0 7 14 13 21 27 20 28 34 42 26 34 41 [..]
P1000415.JPG domino_6_5 0 6 13 12 19 25 19 26 31 38 25 32 38 [..]
P1000417.JPG domino_6_5 0 5 12 10 16 22 15 22 28 34 20 27 33 [..]
P1000418.JPG domino_6_5 0 6 14 12 20 27 19 26 33 41 25 33 39 [..]
[..]
```

Als erstes beinhalten die Zeilen den Namen des untersuchten Bildes und an zweiter Stelle einen Identifikator der zugehörigen Klasse. Danach folgen die sortierten Bewertungen der jeweiligen Klassen. Dieser Ansatz bietet zum einen die Möglichkeit, die Bewertung jederzeit nachzuvollziehen, zum anderen werden Verwechslungen aufgedeckt, indem innerhalb einer Datei mehrfach auftauchende falsche Klassen beobachtet werden können. Mit dem Aufruf `./DominoDiscoverer -results` wird der letzte Programm-Modus gestartet. Dieser liest alle vorhandenen Ergebnisse aus den Textdateien ein und bewertet die Klassifikationsergebnisse sowohl für jedes Modell als auch für den kompletten Datensatz.

5.4 Ergebnisse & Evaluation

Der folgende Abschnitt beschreibt die Ergebnisse der einzelnen Abstandsfunktionen. Die Interpretation der Ergebnisse und die Darlegung der Gründe für die jeweils evaluierten Versionen findet im nächsten Kapitel 6 statt. Sämtliche Ergebnisse befinden sich im Ordner `/Sonstiges/Ergebnisse` auf der beiliegenden DVD. Für

die Ergebnisse und die Evaluation wurden mehrere Datensätze verwendet und unterschiedliche Bewertungskriterien verwendet:

Datensatz DS_0 Der Datensatz DS_0 umfasst 517 Bilder von einzelnen Dominosteinen auf größtenteils homogenem Untergründen. Bei den Aufnahmen handelt es sich um Bilder, bei denen die Kamera einen ungefähr senkrechten Winkel zum Objekt hat.

Datensatz DS_1 Der Datensatz DS_1 umfasst 314 Bilder von einzelnen Dominosteinen auf unterschiedlichen Untergründen. Die Kamera weicht teilweise um bis zu 20 Grad von einem senkrechten Betrachtungswinkel der Objekte ab.

ER Die Erkennungsrate *ER* gibt an, in wie vielen untersuchten Bildern ein Dominostein gefunden wurde. Dabei zählen sowohl die richtig erkannten, als auch die falsch erkannten Dominosteine.

TQ Die Trefferquote *TQ* (engl. recall) gibt an, wie viele von den erkannten Dominosteinen dem richtigen Modell zugeordnet wurden.

RE Das Kriterium *RE* beinhaltet die Anzahl der richtig erkannten Dominosteine.

HV Das Kriterium *HV* beschreibt, falls vorhanden, die häufigste Verwechslung eines Dominosteins. Als Schwellwert wurden 20% gewählt. Die häufigste Verwechslung ist nur in der Detailansicht der Ergebnisse in Anhang E verfügbar.

Verwendete Parameter Für die Experimente wurden die Standardwerte $theta = 0.50$, $epsilon = 7$ und $min\_length = 11$ verwendet. theta beeinflusst die Toleranz des Kreisfinder, epsilon die des Rechteckfinders und min_length stellt einen Schwellwert für das Split & Merge Modul dar.

Sei #G die Gesamtzahl der Bilder, #TP die Anzahl der richtig erkannten Bilder und #FP die Anzahl der falsch erkannten Bilder. Dann ergeben sich für die Erkennungsrate, die Trefferquote und die richtig Erkannten folgende Formeln:

$$\text{ER} \quad \frac{\#\text{TP} + \#\text{FP}}{\#\text{G}}$$

$$\text{TQ} \quad \frac{\#\text{TP}}{\#\text{G}}$$

$$\text{RE} \quad \frac{\#\text{TP}}{\#\text{TP} + \#\text{FP}}$$

5.4.1 Abstandsfunktionen

Gegeben ein $c = \begin{pmatrix} c_x & c_y & c_r \end{pmatrix}^T$ und ein $p = \begin{pmatrix} p_x & p_y & p_r \end{pmatrix}^T$ mit $c \in S \cap C$, $p \in V_{Pips}$, wobei (x, y) dem Mittelpunkt und r dem Radius entspricht. Sei $d = max(|S|, |M|) - min(|S|, |M|)$.

Eine Detailansicht der Ergebnisse befindet sich in Anhang E.

Kostenfunktion δ_{r1} und Straffunktion δ_{p1}:

$$\delta_{r1}(p, c) = \sum_i 50 \frac{min(c_{\mathrm{x}_i}, p_{\mathrm{x}_i})}{max(c_{\mathrm{x}_i}, p_{\mathrm{x}_i})} + 50 \frac{min(c_{\mathrm{y}_i}, p_{\mathrm{y}_i})}{max(c_{\mathrm{y}_i}, p_{\mathrm{y}_i})}$$

$$\delta_{p1}(m_2, s_2) = \sum_{i=1}^{d} 0 = 0$$

	ER	TQ	RE
Ergebnis DS_0	96.63%	88.18%	91.16%
Ergebnis DS_1	95.11%	83.06%	87.26%

Tabelle 5.1: Ergebnisse der ersten Abstandsfunktion.

Kostenfunktion δ_{r2} und Straffunktion δ_{p2}:

$$\delta_{r2}(p, c) = \sum_i 100 - 50 \sqrt{(c_{\mathrm{x}_i} - p_{\mathrm{x}_i})^2 + (c_{\mathrm{y}_i} - p_{\mathrm{y}_i})^2} - 50 \sqrt{(c_{\mathrm{r}_i} - p_{\mathrm{r}_i})^2}$$

$$\delta_{p3}(m_2, s_2) = \sum_{i=1}^{d} -25 (i - 1)$$

	ER	TQ	RE
Ergebnis DS_0	96.63%	89.68%	92.68%
Ergebnis DS_1	95.11%	85.11%	89.39%

Tabelle 5.2: Ergebnisse der zweiten Abstandsfunktion.

Kostenfunktion δ_{r3} und Straffunktion δ_{p3}:

$$\delta_{r3}(p, c) = \sum_i 100 - 200^k - 50\sqrt{(c_{\mathrm{r}_i} - p_{\mathrm{r}_i})^2}$$

mit $k = \sqrt{(c_{\mathrm{x}_i} - p_{\mathrm{x}_i})^2 + (c_{\mathrm{y}_i} - p_{\mathrm{y}_i})^2}$

$$\delta_{p3}(m_2, s_2) = \sum_{i=1}^{d} -25\, i$$

Ergebnisse:

	ER	TQ	RE
Ergebnis DS_0	96.63%	90.08%	93.08%
Ergebnis DS_1	95.11%	85.94%	90.22%

Tabelle 5.3: Ergebnisse der dritten Abstandsfunktion.

Kostenfunktion Version δ_{r4} und Straffunktion Version δ_{p4}:

$$\delta_{r4}(p, c) = \sum_i 100 - 150^k - 100\sqrt{(c_{\mathrm{r}_i} - p_{\mathrm{r}_i})^2}$$

mit $k = \sqrt{(c_{\mathrm{x}_i} - p_{\mathrm{x}_i})^2 + (c_{\mathrm{y}_i} - p_{\mathrm{y}_i})^2}$

$$\delta_{p4}(m_2, s_2) = \sum_{i=1}^{d} h(i)$$

mit $h(x) \begin{cases} -25\,(x+1), & |S| > |M| \\ -10\,(x+1), & |M| > |S| \end{cases}$

Ergebnisse:

	ER	TQ	RE
Ergebnis DS_0	96.63%	89.84%	92.84%
Ergebnis DS_1	95.11%	85.49%	89.78%

Tabelle 5.4: Ergebnisse der vierten Abstandsfunktion.

5.4.2 Übersicht der Ergebnisse

	TQ	RE
Ergebnis δ_{r1}, δ_{p1} für DS_0	88.18%	91.16%
Ergebnis δ_{r2}, δ_{p2} für DS_0	89.68%	92.68%
Ergebnis δ_{r3}, δ_{p3} für DS_0	90.08%	93.08%
Ergebnis δ_{r4}, δ_{p4} für DS_0	89.84%	92.84%
Ergebnis δ_{r1}, δ_{p1} für DS_1	83.06%	87.26%
Ergebnis δ_{r2}, δ_{p2} für DS_1	85.11%	89.39%
Ergebnis δ_{r3}, δ_{p3} für DS_1	85.94%	90.22%
Ergebnis δ_{r4}, δ_{p4} für DS_1	85.49%	89.78%

Tabelle 5.5: Übersicht der Ergebnisse aller Abstandsfunktionen.

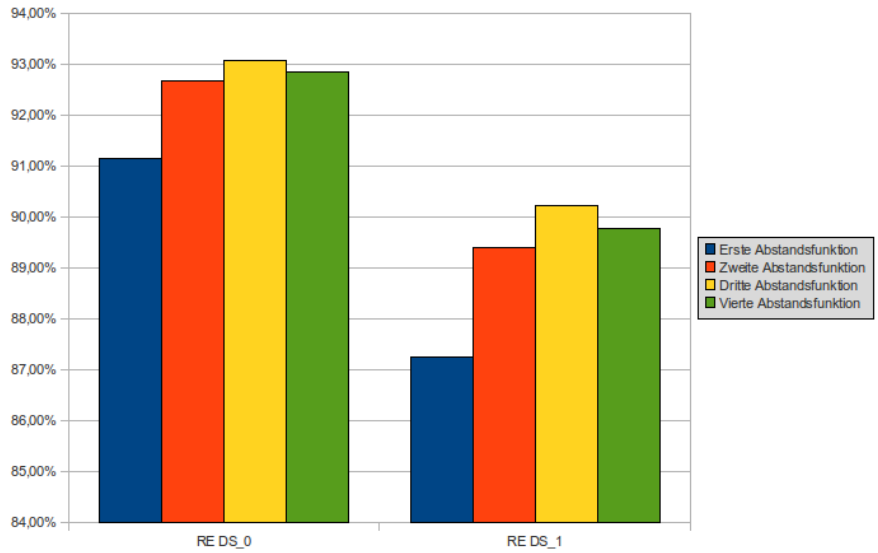


Abbildung 5.3: Säulendiagramm der Ergebnisse aller Abstandsfunktionen. Das Diagramm zeigt die Ergebnisse der richtig erkannten Dominosteine der vier Abstandsfunktionen angewendet auf den Datensatz DS_0 auf der linken Seite und DS_1 auf der rechten Seite.

Kapitel 6

Diskussion

Nachdem die Ergebnisse präsentiert wurden, erfolgen in diesem Kapitel die Begründungen und Kriterien für die gewählten Abstandsfunktionen. Des weiteren werden die Ergebnisse der Funktionen reflektiert und die Beobachtungen bei den Experimente beschrieben.

Die Bewertungskriterien lassen sich in zwei Kategorien unterteilen: In Kriterien, die ausschließlich die Vorverarbeitung betreffen und in Kriterien, welche die Qualität der Abstandsfunktionen widerspiegeln. Die Erkennungsrate ER dient der Bewertung der Vorverarbeitung. Sie beeinflusst die Trefferquote insofern, dass Bilder, in denen keine oder unzureichende Geometrie gefunden werden konnte, die Trefferquote verschlechtern. Die Abstandsfunktionen arbeiten direkt auf den Segmentierungsobjekten, die im Fall der Dominosteine aus Rechtecken und Kreisen im Bild konstruiert werden. Findet die Vorverarbeitung keine der benötigten Bild-Bestandteile, ist es demnach nicht möglich, diese von einer Funktion bewerten zu lassen oder Modellbestandteilen zuzuordnen. Das Kriterium RE stellt eine direkte Bewertung der Qualität der Abstandsfunktion dar. Je höher dieser Wert ist, desto besser ist die Abstandsfunktion. Die Anzahl der richtig erkannten Dominosteine RE beeinflusst ebenfalls die Trefferquote, indem Bilder, in den zwar Geometrie gefunden werden konnte, diese jedoch falsch zugeordnet wurde, die Trefferquote verschlechtern. Dabei werden allerdings nur diejenigen Bilder in die Bewertung mit einbezogen, in denen Geometrie gefunden wurde, d. h. dieser Wert basiert nicht auf allen Eingabebildern, sondern nur auf solchen, in denen eine Zuordnung stattfinden konnte (Die Anzahl der richtig Erkannten RE berechnet sich ausschließlich aus den Werten TP und FP). Somit sind die Kriterien RE und ER unabhängig voneinander und eine Veränderung der Abstandsfunktion wirkt sich nicht auf die Erkennungsrate aus. Dies ist auch in den Ergebnistabellen ablesbar, für den Datensatz DS_0 liegt die Erkennungsrate konstant bei 96.63%, und für den Datensatz DS_1 bei 95.11%. Die häufigsten Verwechslungen eines Dominosteins waren besonders für die Entwicklung der Abstandsfunktionen von Interesse, da durch den Wert

HV direkt abgelesen werden konnte, wo eine Abstandsfunktion häufig den gleichen Fehler begeht beziehungsweise die gleiche Fehlentscheidung trifft.

Bei der ersten Abstandsfunktion wurde als Basis für die weitere Evaluation eine einfache Summe der Abstände in x- beziehungsweise in y-Richtung für die Kostenfunktion δ_{r1} gewählt. Bei den Abständen handelt es sich immer um Werte im Bereich von 0 bis 2, da die im Bild gefundenen Segmentierungsobjekte in Modellkoordinaten umgerechnet werden und anschliessend mit den Faktor 50 multipliziert werden um das Ergebnis in % um zurechnen. Die Basisversion der Straffunktion δ_{p1} wurde ebenfalls einfach gehalten, indem jedem nicht zugeordnete Element seitens der Segmentierungs- oder der Modellobjekte eine Bewertung von 0 zugeordnet wurde. Die zweite Abstandsfunktion verwendet für die Kostenfunktion δ_{r2} die Summe der quadratischen Abstände in Modellkoordinaten und berücksichtigt zusätzlich das Verhältnis des Kreisradius der Pips zum Kreisradius des Segmentierungsobjekts. Die Straffunktion δ_{p2} bewertet die erste fehlende Zuordnung mit 0 und jede weitere fehlende Zuordnung mit zusätzlichen -25%, um Zuordnungen mit mehr als zwei oder drei fehlenden Elementen zu verhindern.

Basierend auf den Ergebnissen der ersten beiden Abstandsfunktion erfolgt nun ein Umbruch in der Herangehensweise für die Evaluierung der Abstandsfunktionen. Die beiden ersten Versionen liefern mit Raten der richtig erkannten Dominosteinen von 91.16% beziehungsweise 87.26% für die erste Version und 92.68% beziehungsweise 89.39% bereits sehr gute Werte für die Zuordnung. Eine Tatsache die sich jedoch nicht in der Ergebnistabellen widerspiegelt ist der Umstand, dass zwischen der ausgewählten besten Korrespondenz und der zweitbesten Korrespondenz oft nur wenige Prozent liegen. In [BB82] beschreiben Ballard und Brown in Kapitel 11.3 einen Ansatz, der neben den Bewertungs- und Strafkosten eine weitere Komponente enthält. Dabei handelt es sich um die sogenannten Sprungfedern. Ein Beispiel für die Verwendung der Sprungfedern ist in Abbildung 6.1 dargestellt. Die Idee besteht darin, das zwischen den Elementen Sprungfedern agieren, deren "Spannung" zusätzlich in die Bewertung mit einbezogen wird. Je weiter zwei Elemente von einander entfernt sind, desto stärker wird die korrespondierende Sprungfeder die Bewertung negativ beeinflussen. Der Ansatz von Ballard und Brown konnte aufgrund der unterschiedlichen Systemstrukturen nicht eins zu eins für die Anwendung auf den Dominosteinen verwendet werden, lieferte jedoch die grundlegende Idee für eine weitere Verbesserung der Abstandsfunktionen. Die bisherigen Versionen waren linear realisiert worden, d. h. wurde einem Pip ein Kreis zugeordnet, der doppelt so weit entfernt ist wie ein anderer Kreis, dann erhält dieser Kreise eine halb so gute Bewertung. Durch die Idee der Sprungfedern ergab sich daraus die Möglichkeit Abstandsfunktionen zu verwenden, die sich im Parameterraum der Modellkoordinaten exponentiell verhalten. Dieser Ansatz brachte eine Vielzahl von Vorteilen mit sich. Zum einen wirkt sich der Bereich von 0.00 bis ca. 0.2 nur noch

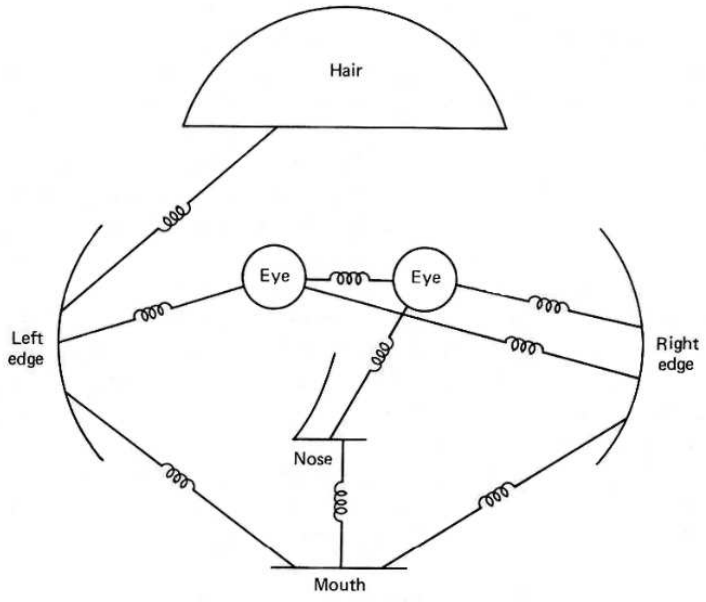


Abbildung 6.1: Darstellung der Sprungfedern am Beispiel eines Gesichts. Die Grafik wurde aus [BB82] entnommen.

sehr schwach auf das Ergebnis aus. Dies hat den Vorteil, dass geringe Ungenauigkeiten im Bild nur noch minimal gewichtet werden. Zum anderen kann ein durch den Kurvenverlauf festlegbarer Bereich völlig ausgeblendet werden. Bei den Experimenten hat sich ein Bereich bei ca. 0.80 als gute Wahl herauskristallisiert. Somit ist es möglich, die Auswahl der Abstandsfunktion auf eine Hälfte des Dominostein zu beschränken, da weiter entfernte Objekte wesentlich stärker bestraft werden. Gepaart mit der Eigenschaft des Systems, auch negative Werte in allen Schritten unterstützen zu können, liefern die exponentiellen Abstandsfunktion ab einer wählbaren Distanz Werte, die aufgrund ihrer Größe nicht mehr tragbar sind und vermieden werden müssen, da sie für das Gesamtergebnis der Korrespondenz untragbar geworden sind. Ein Vergleich der linearen ersten und zweiten Abstandsfunktion mit der exponentiellen dritten und vierten Abstandsfunktion ist in Abbildung 6.2 zu sehen. Für den Vergleich wurde der Kreisradius vernachlässigt und die Kostenfunktion δ_{r1} auf einen Parameter reduziert. Für die dritte Abstandsfunktion wurde die Kostenfunktion δ_{r3} nach den gerade beschriebenen Kriterien angepasst. Der Funktionsteil $100 - 200^x$ hat bei $x \approx 0.869$ den Wert 0 erreicht und schränkt somit die Auswahlmöglichkeiten der Abstandsfunktion enorm ein. Die Straffunktion δ_{p3} wurde ebenfalls verändert. Das erste fehlende Element wird nun direkt mit -25% bestraft, anstatt wie bei der vorangegangenen Version bei 0 zu beginnen.

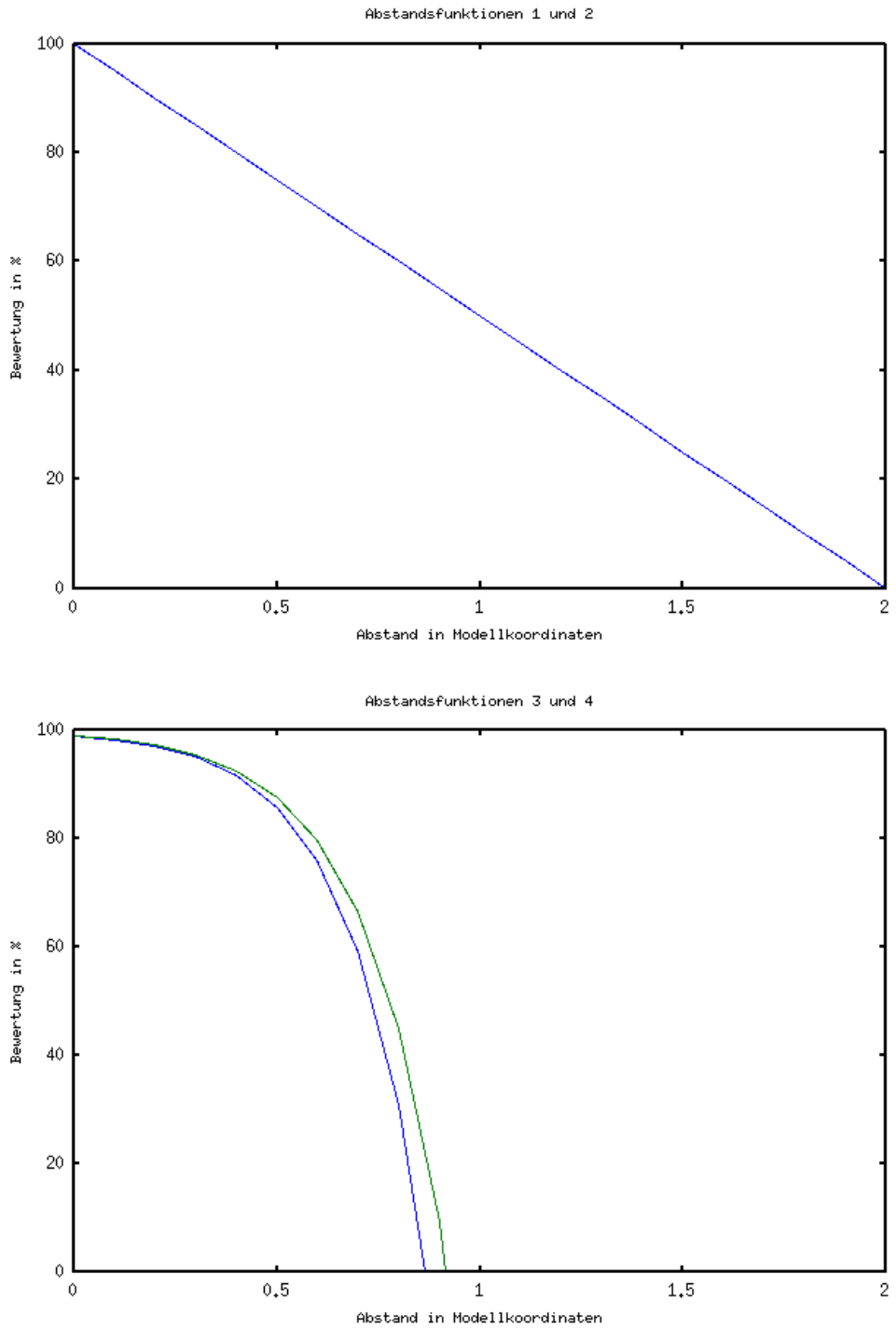


Abbildung 6.2: Darstellung der Kostenfunktionen. Die beiden Abbildungen stellen den Verlauf der vier Kostenfunktionen in vereinfachter Form dar. Dabei wurde bei allen Funktionen der Kreisradius für die Darstellung vernachlässigt und für die Kostenfunktion δ_{r1} ist die Aufteilung für unterschiedliche x- und y-Parameter auf einen Wert reduziert worden. Die obere Abbildung zeigt die identisch verlaufenden Funktionen δ_{r1} und δ_{r2}. Die untere Abbildung zeigt in Blau die Funktion δ_{r3} und in Grün die Funktion δ_{r4}.

Neben den bereits erwähnten Verbesserungen bestand auch die Möglichkeit, eigene Ideen und Ansätze auszuprobieren und mit in die Abstandsfunktionen einfließen zu lassen. Eine Idee bestand darin, bestimmten Positionen der Pips im Layout eine stärkere Gewichtung zu geben. Ein Kandidat hierfür wäre Beispielsweise der Pip an der zentralen Position einer Domino-Hälfte, da er gerade von ungeraden Augenzahlen unterscheidet und somit eine gesonderte Rolle spielt. Er könnte dazu genutzt werden, bei korrekter Erkennung die Auswahl der Modelle einzuschränken oder gewisse Modelle zu bevorzugen. An dieser Stelle hat sich jedoch in den Experimenten herausgestellt, das die Abstandsfunktion beziehungsweise die Zuordnung kontraintuitive Ergebnisse liefert. Bei einer doppelten Gewichtung des zentralen Dominostein für passende Kreise im Bild und ungerade Modellhälften sank die Rate der richtig erkannten Dominosteine auf 63% für den Datensatz DS_1. Eine Ursache dafür liegt bei der Erkennung der Segmentierungsobjekte. Wird bei einem Dominostein beispielsweise der Schatten mit zum Rechteck eines Steins hinzu gezählt, kommt es zu einer Verschiebung aller Werte der Kreise. Bei einfach gewichteten Steinen bleibt die Konstellation weitestgehend erhalten und die Abstandsfunktion trifft dennoch die korrekte Entscheidung. Werden einzelne Pips jedoch besser bewertet, kann es durch die Verschiebung dazu kommen, dass Kreise durch ungenaue Segmentierung diesen Pips nahe genug kommen, um von der besseren Wertung zu profitieren. Die Absicht, auch beim Ausfall einiger Pips noch das richtige Modell zu finden, kommt an dieser Stelle erschwerend hinzu, da es der Abstandsfunktion somit möglich ist, einen Kreis zu einer höher gewichteten Position zuzuordnen und bei einer Zuordnung eine geringfügige Strafe in Kauf zu nehmen. Dieser Fall lies sich in den Experimenten anhand des Wertes HV beobachten und trat besonders häufig bei der Domino-Hälfte mit zwei Pips auf, die sehr oft mit der Hälfte mit drei Pips verwechselt wurde. Ein weiteres Problem trat auf, wenn das zu einem Dominostein korrespondierende Rechteck schräg über dem eigentlichen Stein detektiert wurde. In diesem Fall kam es zu Fehlzuordnungen der Domino-Hälfte mit vier Pips, die des öfteren mit der Hälfte mit drei Pips verwechselt wurde. Aufgrund dieser Beobachtungen hat sich für die vierte Abstandsfunktion eine Kostenfunktion δ_{r4} ergeben, die der vorangegangen Version wesentlich ähnlicher ist als ursprünglich geplant. Die Restriktion des maximalen Abstands wurde um ca. 10% vermindert und gleichzeitig die Gewichtung des Kreisradius noch einmal verdoppelt, um falsche Kreis gänzlich auszublenden. In Verbindung mit der Straffunktion δ_{p4}, die als erste zwischen fehlenden Modellelementen und fehlenden Bildelementen unterscheidet und fehlende Kreise im Bild milder bestraft als fehlende Pips, konnte eine vergleichbare Trefferquote erreicht werden.

Die Ergebnisse aller vorgestellten Abstandsfunktionen liefern vergleichbare Ergebnisse, eine Verbesserung der Anzahl der richtig erkannten Dominosteine von bis zu 3% konnte erreicht werden. Bei einem initialer Qualität der Basisversion

mit einem Wert von 91.12% für den Datensatz DS_0 und 87.26% für den Datensatz DS_1 der richtig erkannten Steine existiert keine hohe Steigerungsfähigkeit. Dies wird besonders deutlich, wenn einige Beispiele der falsch zugeordneten Dominosteine im Detail betrachtet werden. Abbildung 6.3 zeigt einige Beispiele, in denen es durch Anwendung einer Abstandsfunktion überhaupt nicht möglich ist, die korrekte Entscheidung zu treffen, da essentielle Elemente für die Separierung der Klassen nicht detektiert wurden.

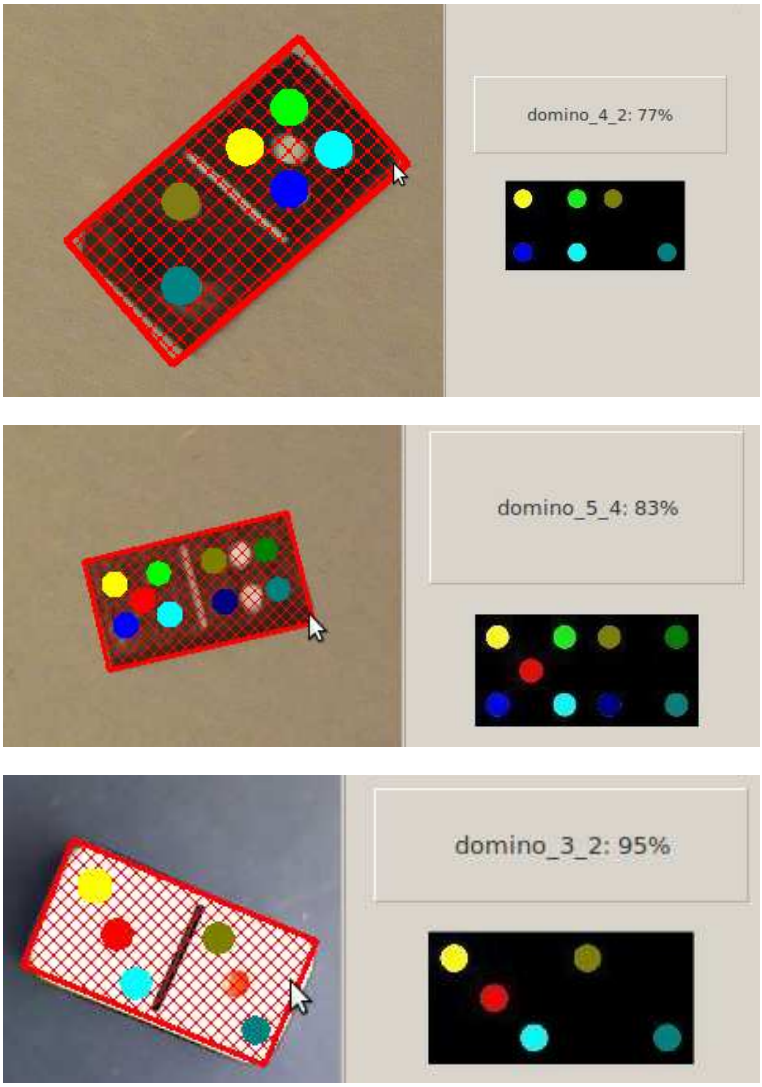


Abbildung 6.3: Beispiele nicht korrigierbarer Fehler. Eine Korrektur des falschen Ergebnisses ist durch Modifikation der Abstandsfunktionen nicht realisierbar.

Aufgrund der Ergebnisse und den Beobachtungen bei den Experimenten haben sich weitere Ansätze und Erweiterungsmöglichkeiten eröffnet, die im folgenden Kapitel 7 vorgestellt werden.

Kapitel 7

Zusammenfassung

Der in dieser Arbeit verwendete wissensbasierte Ansatz stellt in Kombination mit einem Graph-Matching Algorithmus eine solide und akkurate Lösung für die Erkennung von Dominosteinen in 2-D Eingabebildern dar. Die entwickelte Segmentierungsstrategie unter Anwendung und Erweiterung der Bildverarbeitungsbibliothek PUMA liefert auf planaren Eingabebildern mit homogenen Hintergründen sehr gute Ergebnisse. Der Rechteckfinder liefert ca. 96% (recall) aller Rechtecke und der Kreisfinder ca. 95% (recall) aller Kreise. Zur Bewertung der Hypothesen für die Zuordnung hat sich der Einsatz und die Evaluation einer Abstandsfunktion als exzellentes Werkzeug erwiesen. Auf den beiden verwendeten Datensätzen DS_0 (517 Bilder) und DS_1 (314 Bilder) wurden mit der dritten Abstandsfunktion für DS_0 93.08% und für DS_1 90.22% der Dominosteine richtig zugeordnet (RE). Die Trefferquote (recall) von 83% bis 90% für alle Abstandsfunktionen auf beiden Datensätzen liegt in einem akzeptablen Intervall.

Unter Ausnutzung des in Klassen modellierten Wissens konnten unterschiedliche Abstandsfunktionen und deren Eigenschaften erprobt und evaluiert werden. Dabei hat sich die Verwendung von exponentiellen Abstandsfunktionen als besonders effizient erwiesen. Durch die nicht-lineare Skalierung der Parameter im Modellkoordinatensystem konnten sehr gute Ergebnisse erzielt werden. Im Gegensatz dazu hat die unterschiedliche Gewichtung bestimmter Layoutpositionen, beispielsweise der zentralen Position, die ungerade von geraden Augenzahlen separiert, nicht zu einer Verbesserung des Gesamtergebnisses beitragen können. Bei den Experimenten hat sich gezeigt, dass die entwickelte Segmentierung und die Funktionalität aus PUMA, bedingt durch Schattenwurf oder andere Störfaktoren, zwar immer noch gute Ergebnisse liefert, die Resultate jedoch mit einer minimalen Ungenauigkeit behaftet sind. Durch die unterschiedliche Gewichtung gepaart mit leicht rotierten oder vergrößerten Rechtecken kam es vermehrt zu fehlerhaften Zuordnungen, je stärker die Gewichtung zugunsten einzelner Positionen variiert wurde. Aus diesen Beobachtungen und den Resultaten der Experimente lässt sich

zusammenfassend erkennen, dass eine unterschiedliche Gewichtung einzelner Layoutpositionen durchaus sinnvoll ist, jedoch auf Kosten der numerischen Stabilität des Systems geht und auf den Testdaten keine besseren Ergebnisse liefern konnte. Aus dieser Erkenntnis lässt sich wiederum ableiten, dass die Qualität einer Abstandsfunktion bereits bei planaren Aufnahmen von Objekten vor homogenem Hintergrund zu einem großen Teil von ihrer Fähigkeit bestimmt wird, wie sie mit ungenauen Eingabedaten umzugehen vermag.

Die Aufteilung der Abstandsfunktionen in Kosten- und Straffunktion hat sich bewährt, da es auf diese Weise möglich war, einzelne Erkenntnisse, wie beispielsweise eine Auffälligkeit unter den häufigsten Verwechslungen, individuell für den Kosten- oder Strafteil zu evaluieren.

Die Zuordnung und die Auswahl der bewerteten Hypothesen lassen sich mit Hilfe der Ungarischen Methode einwandfrei realisieren. Unter Verwendung von Adjazenzmatrizen konnte das Graph-Matching ausschließlich auf Basis von Matrizen durchgeführt werden. Durch die Verwendung einer grafischen Benutzeroberfläche konnte erfolgreich demonstriert werden, dass das entwickelte System nicht nur in der Lage ist die Hypothesen zu bewerten, sondern darüber hinaus auch fähig ist, das Wissen zu akquirieren, welches im Bild gefundene Element am besten zu welchem Element unter den Modellen korrespondiert. Innerhalb der grafischen Benutzeroberfläche hat sich besonders die Möglichkeit des direkten Vergleichs der einzelnen Vorverarbeitungsschritte sowie die Übersicht der Bewertungen der Zuordnungen bewährt. Durch die Umwandlung der Segmentierungsobjekte in Qt-Objekte konnte zudem ein intuitives und optisch ansprechendes Design realisiert werden, das vor allem von den Effekten auf den Segmentierungsobjekten selbst profitiert.

7.1 Ausblick

Nachdem die Leistungsfähigkeit des Systems am Beispiel von Dominosteinen erfolgreich demonstriert wurde besteht nun die Möglichkeit, die entwickelten Strategien auf variierende und komplexere Problemfelder anzuwenden. Aufbauende Arbeiten könnten beispielsweise zum Thema haben, Fenster beziehungsweise Fensterscheiben in 2-D Bildern von Gebäudefassaden zu erkennen.

Eine andere Erweiterungsmöglichkeit besteht in der Erkennung von Dominosteinen in perspektivisch verzerrten Bildern. Diesbezüglich müsste zum einen ein Austausch des Kreisfinders durch einen Ellipsenfinder stattfinden, zum anderen ein Austausch des Rechteckfinders durch einen Viereckfinder. Im Rahmen dieser Erweiterung bietet sich zudem ein Austausch des Split & Merge Moduls an. Durch einen iterativen Programm-Modus könnte erreicht werden, dass der komplette Prozess parameterlos abläuft und völlig unabhängig von der Bildgröße ist.

Bei näherer Betrachtung der Einzelergebnisse zeigt sich in einigen Fällen, dass trotz adäquater Abstandsfunktion und solider Segmentierung die Abstände zwischen den Bewertungen der einzelnen Modelle nicht besonders hoch sind. An dieser Stelle bieten sich ebenfalls weitere Möglichkeiten an, diesen Abstand zu vergrößern. Beispielsweise könnte ein erneuter Durchlauf mit veränderten Optionen in solchen Fällen Abhilfe schaffen. Außerdem könnten, für diese speziellen Fälle oder allgemein, zusätzlich Farb- oder Texturinformationen in die Bewertung mit einfließen, was besonders zu einer besseren Erkennung des 0-0er Dominosteins beitragen könnte.

Literaturverzeichnis

[BB82] Ballard, Dana H. ; Brown, Chris. M.: *Computer Vision*. Englewood Cliffs, NJ : Prentice-Hall, 1982 `http://www.dai.ed.ac.uk/homes/rbf/BANDB/`

[Ben02] Bengoetxea, E.: *Inexact Graph Matching Using Estimation of Distribution Algorithms*. Paris, France, Ecole Nationale Supérieure des Télécommunications, Diss., Dec 2002

[BHJ+99] Bollmann, M. ; Hoischen, R. ; Jesikiewicz, M. ; Justkowski, C. ; Mertsching, B.: Playing Domino: A Case Study for an Active Vision System. In: *Computer Vision Systems*, 1999, S. 392 ff.

[BK08] Bradski, Gary ; Kaehler, Adrian: *Learning OpenCV: Computer Vision with the OpenCV Library*. Cambridge, MA : O'Reilly, 2008

[Bou00] Bouattour, Sahla ; Universität Erlangen-Nürnberg, Institut für Mathematische Maschinen und Datenverarbeitung (Hrsg.): *Objekterkennung und Lageschätzung mit geometrischen Modellen*. Universität Erlangen-Nürnberg, Institut für Mathematische Maschinen und Datenverarbeitung, Jan 2000. – Studienarbeit

[Bru09] Brunelli, R.: *Template Matching Techniques in Computer Vision: Theory and Practice*. Wiley, 2009

[Cas93] Cass, Todd A.: *Polynomial-time geometric matching for object recognition*. Cambridge, MA, USA, Diss., 1993

[Die00] Diestel, Reinhard: *Graphentheorie (elektronische Ausgabe)*. Heidelberg : Springer Verlag, 2000 `http://www.math.uni-hamburg.de/home/diestel/books/graphentheorie/`. – Ausführliche Einführung in die Graphentheorie

[FE09a] Falkowski, Kerstin ; Ebert, Jürgen: Graph-based urban object model processing. In: Stilla, Uwe (Hrsg.) ; Rottensteiner, Franz

(Hrsg.) ; PAPARODITIS, Nicolas (Hrsg.) ; International Society for Photogrammetry and Remote Sensing (Veranst.): *Object Extraction for 3D City Models, Road Databases and Traffic Monitoring - Concepts, Algorithms and Evaluation (CMRT) 2009* Bd. 38 - 3 / W4 International Society for Photogrammetry and Remote Sensing, 2009, S. 115 – 120

[FE09b] FALKOWSKI, Kerstin ; EBERT, Jürgen: The STOR Component System / Institut für Softwaretechnik, Universität Koblenz-Landau. 2009 (14/2009). – Forschungsbericht. – Arbeitsbericht STOR

[FED+09] FALKOWSKI, Kerstin ; EBERT, Jürgen ; DECKER, Peter ; WIRTZ, Stefan ; PAULUS, Dietrich: Semi-automatic generation of full CityGML models from images. In: *Geoinformatik 2009* Bd. 35, Institut für Geoinformatik Westfälische Wilhelms-Universität Münster, 2009, S. 101–110

[GHJV95] GAMMA, Erich ; HELM, Richard ; JOHNSON, Ralph E. ; VLISSIDES, John M.: *Design patterns: elements of reusable object-oriented software.* Boston, MA, USA : Addison-Wesley Longman Publishing Co., Inc., 1995 (Addison-Wesley Professional Computing Series). – ISBN 0–201–63361– 2

[GI92] GREMBAN, K.D. ; IKEUCHI, K.: Appearance-Based Vision and the Automatic Generation of Object Recognition Plans. In: *Three-Dimensional Object Recognition Systems* Bd. 1, 1992

[GOP90] GORLEN, K. E. ; ORLOW, S. ; PLEXICO, P. S.: *Data Abstraction and Object-Oriented Programming in C++.* Chichester : John Wiley and Sons, 1990

[Gri90] GRIMSON, William Eric L.: *Object recognition by computer: the role of geometric constraints.* Cambridge, MA, USA : MIT Press, 1990 (Artificial intelligence). – ISBN 0–262–07130– 4

[Haa09] HAAS, Judith: *Analyse, Evaluation und Vergleich von Bildverarbeitungsbibliotheken aus Sicht der Softwaretechnik*, Universität Koblenz-Landau, Institut für Softwaretechnik, Arbeitsgruppe Softwaretechnik, Diplomarbeit, 2009

[Har96] HARBECK, Michael: *Objektorientierte linienbasierte Segmentierung von Bildern*, Friedrich-Alexander-Universität Erlangen-Nürnberg, Diss., 12 1996

[HP08] Hans, Wolfram ; Paulus, Dietrich: Automatisierte Objektaufnahme für Bilddatenbanken. In: Helling, Stephan (Hrsg.) ; Brauers, Johannes (Hrsg.) ; Hill, Bernhard (Hrsg.) ; Aach, Til (Hrsg.): *14. Workshop Farbbildverarbeitung*. RWTH Aachen : Shaker, http://www.shaker.de, 10 2008. – ISBN 978–3–83227578– 5, 143-151

[Jun90] Jungnickel, Dieter: *Graphen, Netzwerke und Algorithmen*. Wissenschaftsverlag Mannheim/Wien/Zürich, 1990. – ISBN 3–411–14262–6

[KSN92] *Kapitel* 5. In: Kummert, F ; Sagerer, G. ; Niemann, H.: *A problem-independent control algorithm for image understanding*. Universität Bielefeld, 1992, S. 297–301

[Kuh55] Kuhn, Harold W.: The Hungarian Method for the assignment problem. In: *Naval Research Logistics Quarterly* 2 (1955), S. 83–97

[Lux04] Lux, Reiner: *Gewichtete Matchings mit Anwendungen in der kombinatorischen Optimierung, Implementierung und Vergleich verschiedener Algorithmen*, Universität Bayreuth, Diss., 2004

[Mau10] Maur, Susanne: *Modellbasierte Gebäudeerkennung*, Universität Koblenz-Landau, Institut für Computervisualistik, Arbeitsgruppe Aktives Sehen, Diplomarbeit, Feb 2010. – vorraussichtlicher Abgabetermin

[Mun57] Munkres, J.: Algorithms for the Assignment and Transportation Problems. In: *Journal of the Society of Industrial and Applied Mathematics* 5 (1957), Nr. 1, S. 32–38

[NB80] Nevatia, R. ; Babu, R.: Linear Feature Extraction and Description. In: *Computer Graphics and Image Processing* 13 (1980), S. 257–269

[Nie90] Niemann, Heinrich: *Springer Series in Information Sciences*. Bd. 4: *Pattern Analysis and Understanding*. Heidelberg : Springer Verlag, 1990

[NSSK90] Niemann, H. ; Sagerer, G. ; Schröder, S. ; Kummert, F.: ERNEST: A Semantic Network System for Pattern Understanding. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence, http://www.computer.org/tpami/* 9 (1990), 883-905. `https://www.uni-koblenz.de/~agas/Pool/Niemann1990EAS.pdf`

[Pau03] Paulus, Dietrich ; Universität Koblenz-Landau, http://www.uni-koblenz.de (Hrsg.): *PUMA (Programmier-Umgebung für die Musteranalyse)*. 1. Campus Koblenz, Postfach 201 602, 56070 Koblenz, Germany: Universität Koblenz-Landau, http://www.uni-koblenz.de, 1 2003

[PPDS09] Paulus, Dietrich ; Priese, Lutz ; Decker, Peter ; Schmitt, Frank: Pose-Tracking Forschungsbericht / Institut für Computervisualistik, Universität Koblenz- Landau. 2009 (17/2009). – Forschungsbericht. – Arbeitsbericht STOR

[Qui97] Quint, Franz: *Kartengestützte Interpretation monokularer Luftbilder*. Karlsruhe, Universität Fridericiana zu Karlsruhe (TH), Diss., 1997

[SB06] Summerfield, Mark ; Blanchette, Jasmin: *C++ GUI Programming with Qt 4*. Pap/Cdr. Prentice Hall International, 2006

[SNHW92] Salzbrunn, R. ; Niemann, H. ; Harbeck, M. ; Winzen, A.: Object Recognition by a Robust Matching Technique. In: Bunke, H. (Hrsg.): *Advances in Structural and Syntactic Pattern Recognition: Proc. of the International Workshop*. Singapore : World Scientific, 1992, S. 481–495

[Win94] Winzen, Andreas: *Automatische Erzeugung dreidimensionaler Modelle für Bildanalysesysteme*. Erlangen, Friedrich-Alexander-Universität Erlangen-Nürnberg, Univ. Erlangen-Nürnberg, Diss., 1994

Internetquellen

[AAS05] Arbeitsgruppe Aktives Sehen, Universität K.: *Dokumentation - Programmierumgebung für die Musteranalyse (PUMA).* 2005. – http://www.uni-koblenz.de/FB4/Institutes/ICV/AGPaulus/puma/Dokumentation/ [letzter Abruf: Dez 2009]

[AAS06] Arbeitsgruppe Aktives Sehen, Universität K.: *Programmierumgebung für die Musteranalyse (PUMA).* 2006. – http://www.uni-koblenz.de/FB4/Institutes/ICV/AGPaulus/puma [letzter Abruf: Dez 2009]

[Cor09] Corporation, IBM: *Rational Software Architect Standard Edition, Part of the Rational Software Architect family.* 2009. – http://www-01.ibm.com/software/awdtools/swarchitect/standard/ [letzter Abruf: Dez 2009]

[Fal09] Falkowski, Kerstin: *Homepage Software Techniques for Object Recognition (STOR).* 2009. – http://er.uni-koblenz.de [letzter Abruf: Dez 2009]

[FSF09] Free Software Foundation, Inc.: *GCC, the GNU Compiler Collection.* 2009. – http://gcc.gnu.org/ [letzter Abruf: Dez 2009]

[Inc09] Inc., Geeknet: *Open Computer Vision Library.* 2009. – http://sourceforge.net/projects/opencvlibrary/ [letzter Abruf: Dez 2009]

[Sub09] Subsidiaries, Nokia C. i.: *Qt - Cross-platform application and UI framework.* 2009. – http://qt.nokia.com/products [letzter Abruf: Dez 2009]

[Tea09] Team, The C.: *CTAN - The Comprehensive TeX Archive Network.* 2009. – http://www.ctan.org/tex-archive/macros/latex/contrib/struktex/ [letzter Abruf: Dez 2009]

[Top09] TopCoder, Inc: *Algorithm Tutorials - Assignment Problem and Hungarian Algorithm.* 2009. – http://www.topcoder.com/tc?module=

Static\&d1=tutorials\&d2=hungarianAlgorithm [letzter Abruf: Nov 2009]

[Wik09] WIKIPEDIA: *Domino (Spiel)*. Dec 2009. – http://de.wikipedia.org/wiki/Domino_(Spiel) [letzter Abruf: Nov 2009]

[Xfi09] XFIG.ORG, MCJ C.: *Xfig Drawing Program for the X Windows System*. 2009. – http://www.xfig.org/ [letzter Abruf: Dez 2009]

Tabellenverzeichnis

Abbildungsverzeichnis

Anhang A

Installationshinweise

SVN

Die URL zum SVN lautet: `https://svn.uni-koblenz.de/mhaeselich/domino/`.

PUMA

Die Software benötigt eine funktionsfähige PUMA Version für die Bildverarbeitung. Entsprechende Anleitungen und Installationshinweise finden sich unter [AAS06, AAS05].

Pakete

Eine Liste der benötigten Pakte kann der Datei *Install.sh* aus dem Ordner `30_prog/` entnommen werden. Unter der Linux Distribution Ubuntu kann das Skript wahlweise auch ausgeführt werden, um alle benötigten Pakte bei Bedarf zu installieren. Das Skript wurde unter Ubuntu 9.04 und Ubuntu 9.10 getestet.

Kompilierung

Für die Kompilierung existiert im Ordner `30_prog/` ein Skript namens *Build.sh*. Diese ruft nacheinander die Befehle *qmake-qt4 -project*, *qmake-qt4 -makefile* und *make* auf. Zusätzlich werden die für PUMA benötigten Pfade der Projektdatei hinzugefügt. Unter einigen Distribution steht der Aufruf *qmake-qt4 -project* nicht zur Verfügung. In diesem Fall kann mit *qmake -v* überprüft werden, ob Qt in der Version 4 oder höher vorliegt und in den beiden entsprechenden Zeilen der Datei *Build.sh* das -qt4 beim Aufruf entfernt werden.

Doxygen

Für die Erstellung der Dokumentation erzeugt im Ordner `30_prog/` bei vorhandener Doxygen-Installation der Befehl *doxygen Documentation/doxygen.conf* die gewünschten Dateien.

Anhang B

Mathematische Symbole

M	Menge M
$\|M\|$	Mächtigkeit der Menge M
$\|m\|$	Mächtigkeit der Untermenge $m \subset M$
$\bigcup$	Universum $\bigcup$
$A \times B$	Cartesisches Produkt von A und B
$A \cap B$	Vereinigung von A und B
$A \cup B$	Disjunkte Vereinigung von A und B
$A \subseteq B$	A ist Teilmenge von B
$A \subset B$	A ist echte Teilmenge von B
$A \backslash B$	A ohne B
$x \in A$	Element x ist in der Menge A enthalten
$x \notin A$	Element x ist nicht in der Menge A enthalten
$\forall\, x$	Für alle x
$\exists\, x$	Es existiert ein x
$\nexists\, x$	Es existiert kein x
$\mathbb{R}$	Reelle Zahlen
$\mathbb{N}$	Natürliche Zahlen
v	Vektor v
v^{T}	Transposition des Vektors v
f	Eine Funktion f
$dom\ f$	Definitionsbereich einer Funktion f, Domain
$ran\ f$	Bildbereich einer Funktion f, Range
$A \rightarrowtail B$	Eine partielle Funktion

$a \Leftrightarrow b$	a ist äquivalent zu b
$a \parallel b$	a und b sind parallel zueinander
$a \perp b$	a und b sind orthogonal zueinander
$\overline{AB}$	Linie vom Startpunkt A zum Endpunkt B
$\overline{l}$	Sei $l = \overline{AB}$, dann ist $\overline{l}$ die Länge der Linie l mit $\overline{l} = \sqrt{(x_A - y_A)^2 + (y_A - y_B)^2}$ wobei x und y jeweils die x- beziehungsweise y-Koordinate des indexierten Punktes sind
$\overline{l} \approx \overline{m}$	Die Länge der Linie l entspricht ungefähr der Länge der Linie m
$\mathrm{O}(n)$	Komplexität, Komplexitätsklasse n

Anhang C

Struktogramme

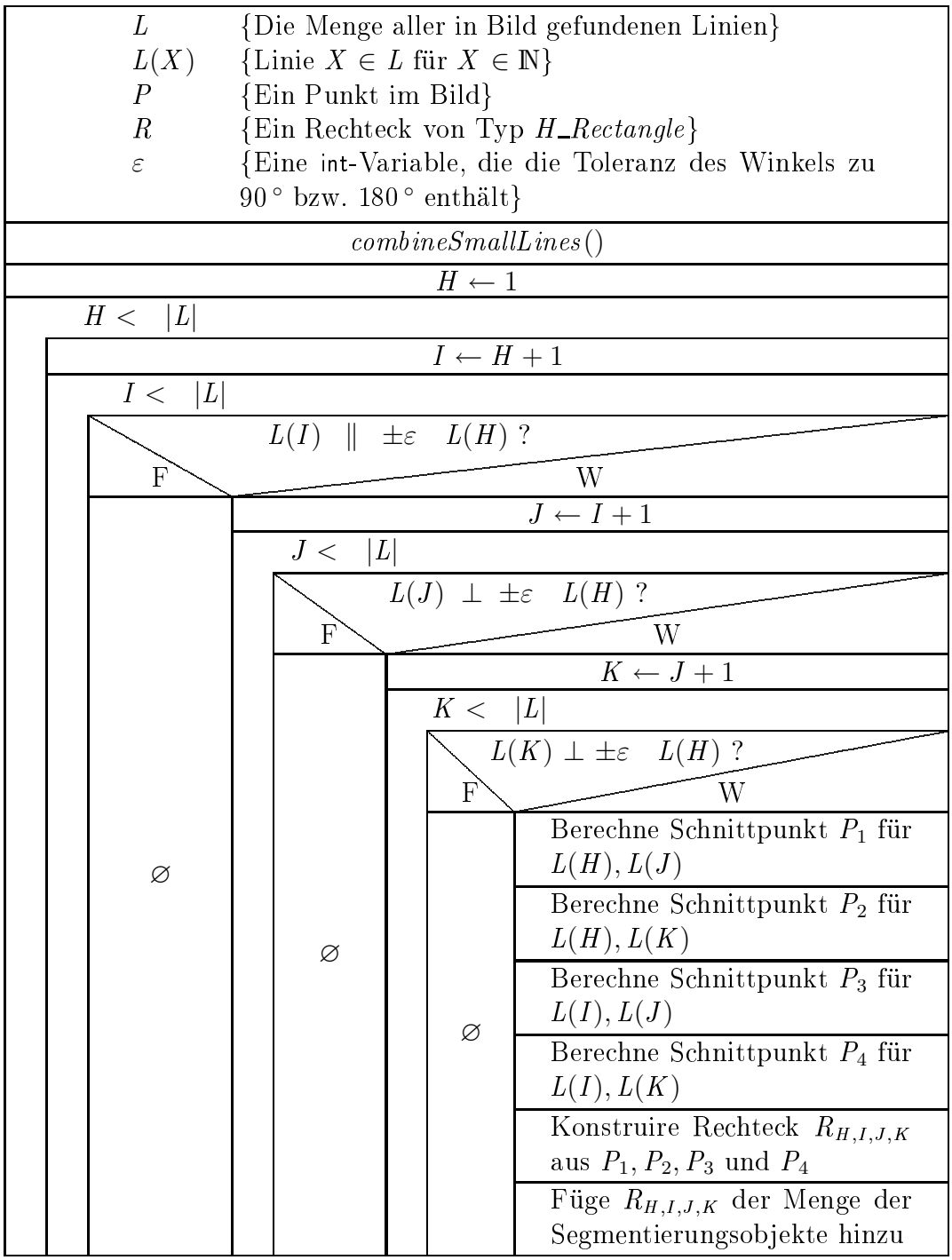


Abbildung C.1: Nassi-Shneiderman-Diagramm des Rechteckfinders.

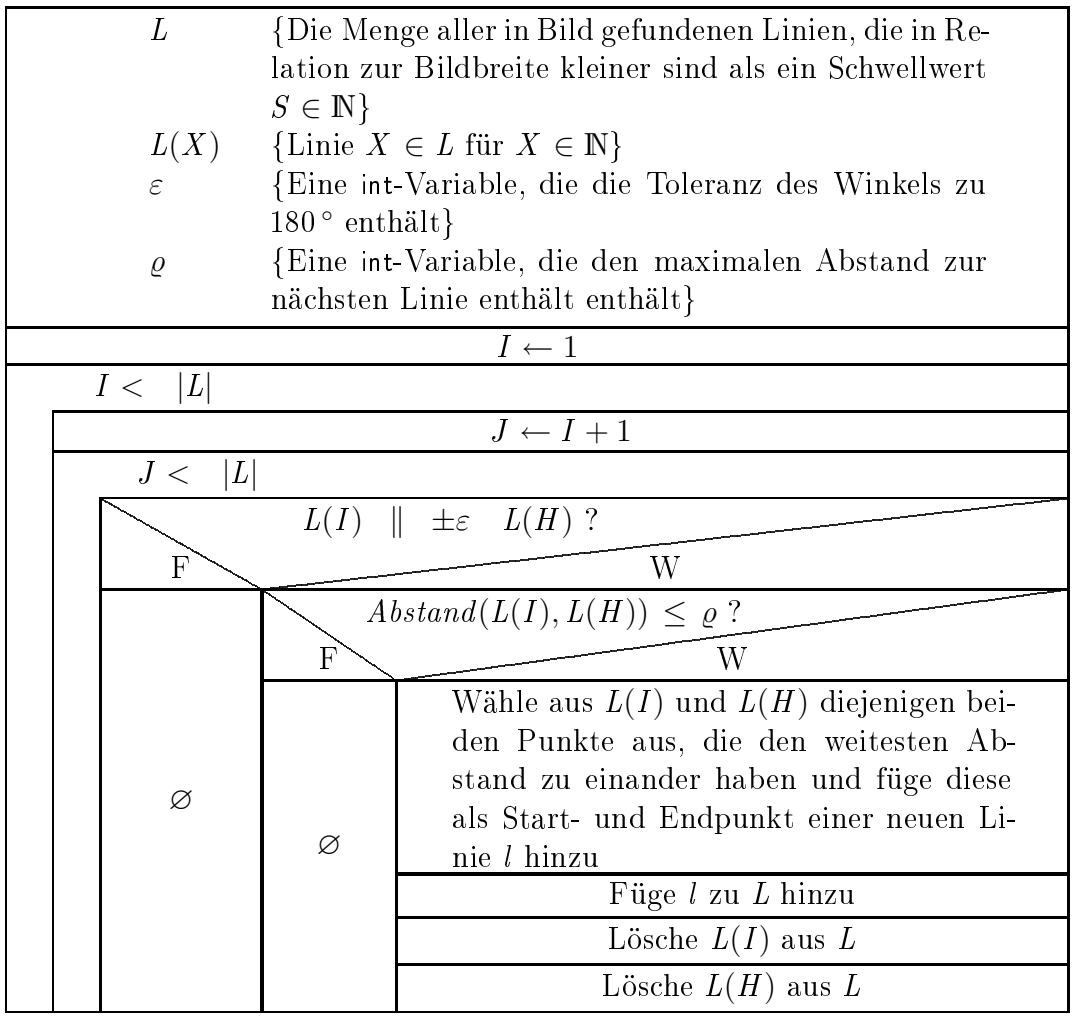


Abbildung C.2: Nassi-Shneiderman-Diagramm der combineSmallLines-Funktion.

Anhang D

Grafiken

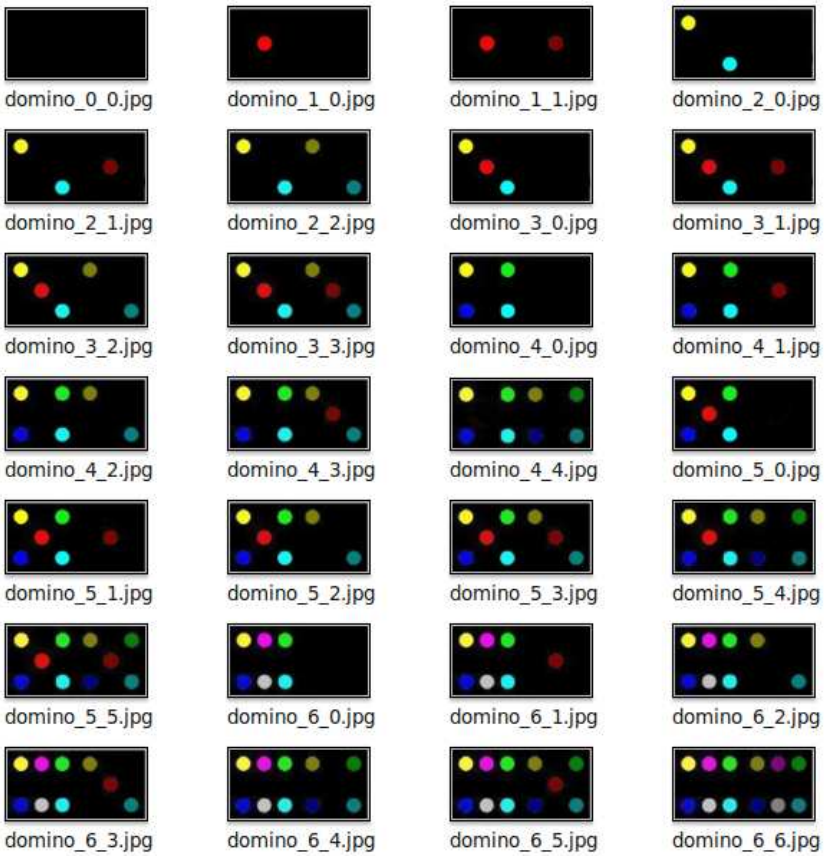


Abbildung D.1: Grafische Übersicht der verwendeten Farben für die Zuordnung der Pips zu den Kreisen im Bild.

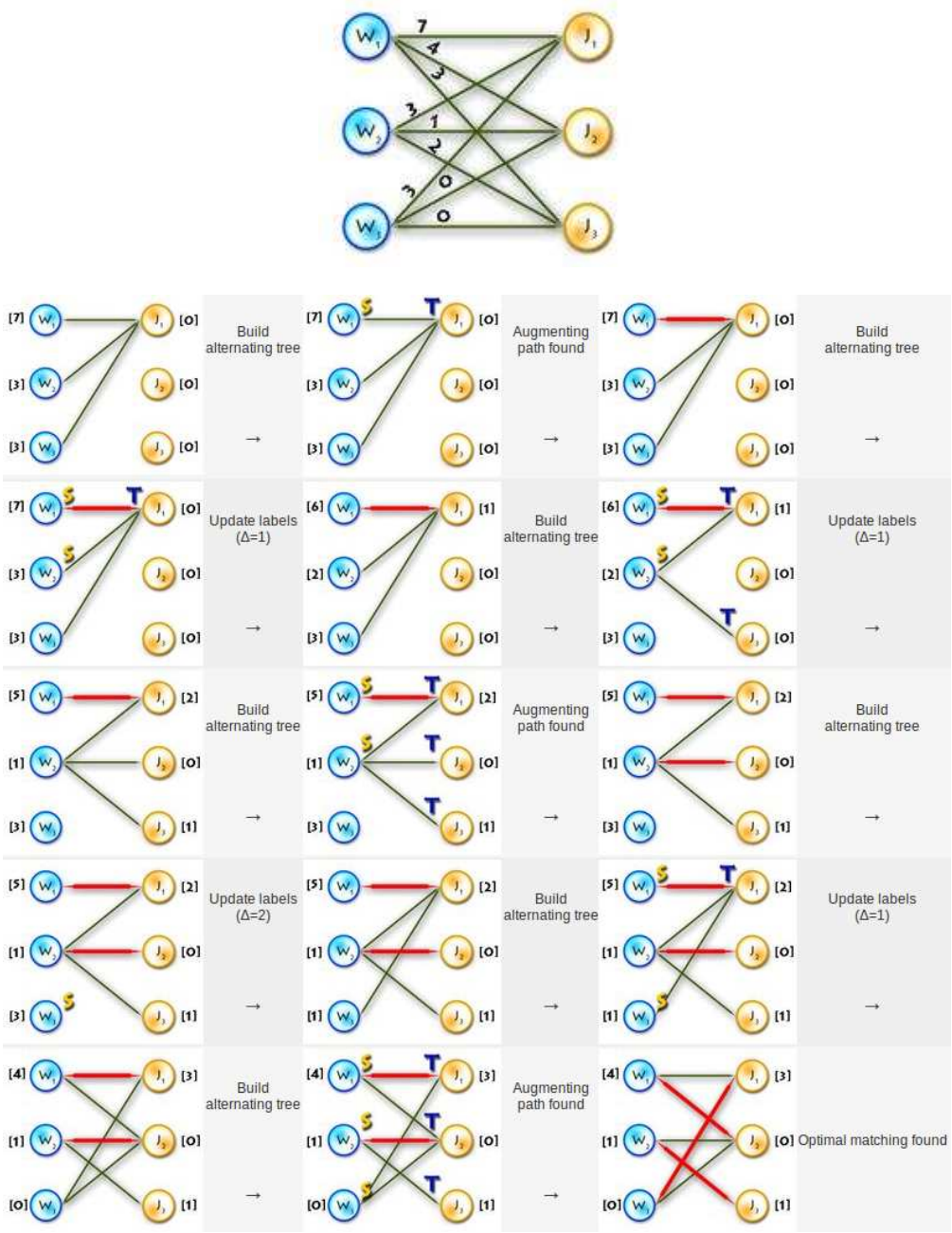


Abbildung D.2: Anwendung der Ungarischen Methode an einem Beispiel. Die Abbildung wurde [Top09] entnommen und demonstriert den Ablauf der Ungarischen Methoden an einem bipartiten Graphen mit den Teilmengen W und J.

Anhang E

Ergebnistabellen

Datensatz DS_0 Der Datensatz DS_0 umfasst 517 Bilder von einzelnen Dominosteinen auf größtenteils homogenem Untergründen. Bei den Aufnahmen handelt es sich um Bilder, bei denen die Kamera einen ungefähr senkrechten Winkel zum Objekt hat.

Datensatz DS_1 Der Datensatz DS_1 umfasst 314 Bilder von einzelnen Dominosteinen auf unterschiedlichen Untergründen. Die Kamera weicht teilweise um bis zu 20 Grad von einem senkrechten Betrachtungswinkel der Objekte ab.

ER Die Erkennungsrate *ER* gibt an, in wie vielen untersuchten Bildern ein Dominostein gefunden wurde. Dabei zählen sowohl die richtig erkannten, als auch die falsch erkannten Dominosteine.

TQ Die Trefferquote *TQ* (engl. recall) gibt an, wie viele von den erkannten Dominosteinen dem richtigen Modell zugeordnet wurden.

RE Das Kriterium *RE* beinhaltet die Anzahl der richtig erkannten Dominosteine.

HV Das Kriterium *HV* beschreibt, falls vorhanden, die häufigste Verwechslung eines Dominosteins. Als Schwellwert wurden 20% gewählt.

Verwendete Parameter Für die Experimente wurden die Standardwerte $theta = 0.50$, $epsilon = 7$ und $min\_length = 11$ verwendet. theta beeinflusst die Toleranz des Kreisfinder, epsilon die des Rechteckfinders und min_length stellt einen Schwellwert für das Split & Merge Modul dar.

Dominostein	ER	TQ	RE	HV
Domino 0-0	93.75%	93.75%	100.00%	-
Domino 1-0	94.44%	94.44%	100.00%	-
Domino 1-1	100.00%	100.00%	100.00%	-
Domino 2-0	100.00%	94.74%	94.74%	-
Domino 2-1	95.83%	91.67%	95.65%	-
Domino 2-2	100.00%	100.00%	100.00%	-
Domino 3-0	100.00%	84.62%	84.62%	-
Domino 3-1	100.00%	95.65%	95.65%	-
Domino 3-2	100.00%	100.00%	100.00%	-
Domino 3-3	92.31%	88.47%	95.83%	-
Domino 4-0	100.00%	81.82%	81.82%	Domino 3-0
Domino 4-1	100.00%	82.61%	82.61%	Domino 3-1
Domino 4-2	100.00%	94.44%	94.44%	-
Domino 4-3	100.00%	100.00%	100.00%	-
Domino 4-4	100.00%	83.33%	83.33%	-
Domino 5-0	87.50%	81.25%	92.86%	-
Domino 5-1	100.00%	94.74%	94.74%	-
Domino 5-2	80.00%	50.00%	62.50%	-
Domino 5-3	92.86%	78.57%	84.62%	-
Domino 5-4	95.00%	90.00%	94.74%	-
Domino 5-5	100.00%	92.31%	92.31%	-
Domino 6-0	100.00%	66.67%	66.67%	Domino 5-1
Domino 6-1	100.00%	88.89%	88.89%	-
Domino 6-2	95.00%	80.00%	84.21%	-
Domino 6-3	88.89%	88.89%	100.00%	-
Domino 6-4	100.00%	93.33%	93.33%	-
Domino 6-5	100.00%	88.89%	88.89%	-
Domino 6-6	90.00%	90.00%	100.00%	-
Gesamtergebnis	96.63%	88.18%	91.16%	

Tabelle E.1: Detailansicht der Ergebnisse der ersten Abstandsfunktion angewendet auf den Datensatz DS_0.

Dominostein	ER	TQ	RE	HV
Domino 0-0	87.50%	87.50%	100.00%	-
Domino 1-0	90.91%	90.91%	100.00%	-
Domino 1-1	100.00%	100.00%	100.00%	-
Domino 2-0	100.00%	90.00%	90.00%	-
Domino 2-1	90.00%	80.00%	88.89%	-
Domino 2-2	100.00%	100.00%	100.00%	-
Domino 3-0	100.00%	84.62%	84.62%	-
Domino 3-1	100.00%	88.89%	88.89%	-
Domino 3-2	100.00%	100.00%	100.00%	-
Domino 3-3	81.82%	72.73%	88.89%	-
Domino 4-0	100.00%	81.82%	81.82%	Domino 3-0
Domino 4-1	100.00%	71.43%	71.43%	Domino 3-1
Domino 4-2	100.00%	85.71%	85.71%	-
Domino 4-3	100.00%	100.00%	100.00%	-
Domino 4-4	100.00%	72.73%	72.73%	-
Domino 5-0	83.33%	75.00%	90.00%	-
Domino 5-1	100.00%	90.00%	90.00%	-
Domino 5-2	80.00%	50.00%	62.50%	-
Domino 5-3	90.91%	72.73%	80.00%	-
Domino 5-4	90.91%	81.82%	90.00%	-
Domino 5-5	100.00%	90.00%	90.00%	-
Domino 6-0	100.00%	60.00%	60.00%	Domino 5-1
Domino 6-1	100.00%	88.89%	88.89%	-
Domino 6-2	88.89%	55.56%	62.50%	-
Domino 6-3	88.89%	88.89%	100.00%	-
Domino 6-4	100.00%	87.50%	87.50%	-
Domino 6-5	100.00%	88.89%	88.89%	-
Domino 6-6	90.00%	90.00%	100.00%	-
Gesamtergebnis	95.11%	83.06%	87.26%	

Tabelle E.2: Detailansicht der Ergebnisse der ersten Abstandsfunktion angewendet auf den Datensatz DS_1.

Dominostein	ER	TQ	RE	HV
Domino 0-0	93.75%	93.75%	100.00%	-
Domino 1-0	94.44%	94.44%	100.00%	-
Domino 1-1	100.00%	100.00%	100.00%	-
Domino 2-0	100.00%	94.74%	94.74%	-
Domino 2-1	95.83%	91.67%	95.65%	-
Domino 2-2	100.00%	100.00%	100.00%	-
Domino 3-0	100.00%	84.62%	84.62%	-
Domino 3-1	100.00%	95.65%	95.65%	-
Domino 3-2	100.00%	100.00%	100.00%	-
Domino 3-3	92.31%	88.47%	95.83%	-
Domino 4-0	100.00%	81.82%	81.82%	Domino 3-0
Domino 4-1	100.00%	86.96%	86.96%	Domino 3-1
Domino 4-2	100.00%	94.44%	94.44%	-
Domino 4-3	100.00%	100.00%	100.00%	-
Domino 4-4	100.00%	83.33%	83.33%	-
Domino 5-0	87.50%	81.25%	92.86%	-
Domino 5-1	100.00%	94.74%	94.74%	-
Domino 5-2	80.00%	50.00%	62.50%	-
Domino 5-3	92.86%	78.57%	84.62%	-
Domino 5-4	95.00%	95.00%	100.00%	-
Domino 5-5	100.00%	92.31%	92.31%	-
Domino 6-0	100.00%	94.44%	94.44%	-
Domino 6-1	100.00%	88.89%	88.89%	-
Domino 6-2	95.00%	85.00%	89.47%	-
Domino 6-3	88.89%	88.89%	100.00%	-
Domino 6-4	100.00%	93.33%	93.33%	-
Domino 6-5	100.00%	88.89%	88.89%	-
Domino 6-6	90.00%	90.00%	100.00%	-
Gesamtergebnis	96.63%	89.69%	92.68%	

Tabelle E.3: Detailansicht der Ergebnisse der zweiten Abstandsfunktion angewendet auf den Datensatz DS_0.

Dominostein	ER	TQ	RE	HV
Domino 0-0	87.50%	87.50%	100.00%	-
Domino 1-0	90.91%	90.91%	100.00%	-
Domino 1-1	100.00%	100.00%	100.00%	-
Domino 2-0	100.00%	90.00%	90.00%	-
Domino 2-1	90.00%	80.00%	88.89%	-
Domino 2-2	100.00%	100.00%	100.00%	-
Domino 3-0	100.00%	84.62%	84.62%	-
Domino 3-1	100.00%	88.89%	88.89%	-
Domino 3-2	100.00%	100.00%	100.00%	-
Domino 3-3	81.82%	72.73%	88.89%	-
Domino 4-0	100.00%	81.82%	81.82%	Domino 3-0
Domino 4-1	100.00%	78.57%	78.57%	Domino 3-1
Domino 4-2	100.00%	85.71%	85.71%	-
Domino 4-3	100.00%	100.00%	100.00%	-
Domino 4-4	100.00%	72.73%	72.73%	-
Domino 5-0	83.33%	75.00%	90.00%	-
Domino 5-1	100.00%	90.00%	90.00%	-
Domino 5-2	80.00%	50.00%	62.50%	-
Domino 5-3	90.91%	72.73%	80.00%	-
Domino 5-4	90.91%	90.91%	100.00%	-
Domino 5-5	100.00%	90.00%	90.00%	-
Domino 6-0	100.00%	90.00%	90.00%	-
Domino 6-1	100.00%	88.89%	88.89%	-
Domino 6-2	88.89%	66.67%	75.00%	-
Domino 6-3	88.89%	88.89%	100.00%	-
Domino 6-4	100.00%	87.50%	87.50%	-
Domino 6-5	100.00%	88.89%	88.89%	-
Domino 6-6	90.00%	90.00%	100.00%	-
Gesamtergebnis	95.11%	85.11%	89.39%	

Tabelle E.4: Detailansicht der Ergebnisse der zweiten Abstandsfunktion angewendet auf den Datensatz DS_1.

Dominostein	ER	TQ	RE	HV
Domino 0-0	93.75%	93.75%	100.00%	-
Domino 1-0	94.44%	94.44%	100.00%	-
Domino 1-1	100.00%	100.00%	100.00%	-
Domino 2-0	100.00%	94.74%	94.74%	-
Domino 2-1	95.83%	91.67%	95.65%	-
Domino 2-2	100.00%	100.00%	100.00%	-
Domino 3-0	100.00%	84.62%	84.62%	-
Domino 3-1	100.00%	95.65%	95.65%	-
Domino 3-2	100.00%	100.00%	100.00%	-
Domino 3-3	92.31%	88.47%	95.83%	-
Domino 4-0	100.00%	81.82%	81.82%	Domino 3-0
Domino 4-1	100.00%	86.96%	86.96%	Domino 3-1
Domino 4-2	100.00%	100.00%	100.00%	-
Domino 4-3	100.00%	100.00%	100.00%	-
Domino 4-4	100.00%	88.89%	88.89%	-
Domino 5-0	87.50%	81.25%	92.86%	-
Domino 5-1	100.00%	94.74%	94.74%	-
Domino 5-2	80.00%	50.00%	62.50%	-
Domino 5-3	92.86%	78.57%	84.62%	-
Domino 5-4	95.00%	95.00%	100.00%	-
Domino 5-5	100.00%	92.31%	92.31%	-
Domino 6-0	100.00%	94.44%	94.44%	-
Domino 6-1	100.00%	88.89%	88.89%	-
Domino 6-2	95.00%	85.00%	89.47%	-
Domino 6-3	88.89%	88.89%	100.00%	-
Domino 6-4	100.00%	93.33%	93.33%	-
Domino 6-5	100.00%	88.89%	88.89%	-
Domino 6-6	90.00%	90.00%	100.00%	-
Gesamtergebnis	96.63%	90.08%	93.08%	

Tabelle E.5: Detailansicht der Ergebnisse der dritten Abstandsfunktion angewendet auf den Datensatz DS_0.

Dominostein	ER	TQ	RE	HV
Domino 0-0	87.50%	87.50%	100.00%	-
Domino 1-0	90.91%	90.91%	100.00%	-
Domino 1-1	100.00%	100.00%	100.00%	-
Domino 2-0	100.00%	90.00%	90.00%	-
Domino 2-1	90.00%	80.00%	88.89%	-
Domino 2-2	100.00%	100.00%	100.00%	-
Domino 3-0	100.00%	84.62%	84.62%	-
Domino 3-1	100.00%	88.89%	88.89%	-
Domino 3-2	100.00%	100.00%	100.00%	-
Domino 3-3	81.82%	72.73%	88.89%	-
Domino 4-0	100.00%	81.82%	81.82%	Domino 3-0
Domino 4-1	100.00%	78.57%	78.57%	Domino 3-1
Domino 4-2	100.00%	100.00%	100.00%	-
Domino 4-3	100.00%	100.00%	100.00%	-
Domino 4-4	100.00%	81.82%	81.82%	-
Domino 5-0	83.33%	75.00%	90.00%	-
Domino 5-1	100.00%	90.00%	90.00%	-
Domino 5-2	80.00%	50.00%	62.50%	-
Domino 5-3	90.91%	72.73%	80.00%	-
Domino 5-4	90.91%	90.91%	100.00%	-
Domino 5-5	100.00%	90.00%	90.00%	-
Domino 6-0	100.00%	90.00%	90.00%	-
Domino 6-1	100.00%	88.89%	88.89%	-
Domino 6-2	88.89%	66.67%	75.00%	-
Domino 6-3	88.89%	88.89%	100.00%	-
Domino 6-4	100.00%	87.50%	87.50%	-
Domino 6-5	100.00%	88.89%	88.89%	-
Domino 6-6	90.00%	90.00%	100.00%	-
Gesamtergebnis	95.11%	85.94%	90.22%	

Tabelle E.6: Detailansicht der Ergebnisse der dritten Abstandsfunktion angewendet auf den Datensatz DS_1.

Dominostein	ER	TQ	RE	HV
Domino 0-0	93.75%	93.75%	100.00%	-
Domino 1-0	94.44%	94.44%	100.00%	-
Domino 1-1	100.00%	100.00%	100.00%	-
Domino 2-0	100.00%	94.74%	94.74%	-
Domino 2-1	95.83%	91.67%	95.65%	-
Domino 2-2	100.00%	100.00%	100.00%	-
Domino 3-0	100.00%	84.62%	84.62%	-
Domino 3-1	100.00%	95.65%	95.65%	-
Domino 3-2	100.00%	100.00%	100.00%	-
Domino 3-3	92.31%	88.47%	95.83%	-
Domino 4-0	100.00%	81.82%	81.82%	Domino 3-0
Domino 4-1	100.00%	86.96%	86.96%	Domino 3-1
Domino 4-2	100.00%	100.00%	100.00%	-
Domino 4-3	100.00%	100.00%	100.00%	-
Domino 4-4	100.00%	88.89%	88.89%	-
Domino 5-0	87.50%	81.25%	92.86%	-
Domino 5-1	100.00%	94.74%	94.74%	-
Domino 5-2	80.00%	50.00%	62.50%	-
Domino 5-3	92.86%	78.57%	84.62%	-
Domino 5-4	95.00%	95.00%	100.00%	-
Domino 5-5	100.00%	92.31%	92.31%	-
Domino 6-0	100.00%	94.44%	94.44%	-
Domino 6-1	100.00%	88.89%	88.89%	-
Domino 6-2	95.00%	85.00%	89.47%	-
Domino 6-3	88.89%	88.89%	100.00%	-
Domino 6-4	100.00%	86.67%	86.67%	-
Domino 6-5	100.00%	88.89%	88.89%	-
Domino 6-6	90.00%	90.00%	100.00%	-
Gesamtergebnis	96.63%	89.84%	92.84%	

Tabelle E.7: Detailansicht der Ergebnisse der vierten Abstandsfunktion angewendet auf den Datensatz DS_0.

Dominostein	ER	TQ	RE	HV
Domino 0-0	87.50%	87.50%	100.00%	-
Domino 1-0	90.91%	90.91%	100.00%	-
Domino 1-1	100.00%	100.00%	100.00%	-
Domino 2-0	100.00%	90.00%	90.00%	-
Domino 2-1	90.00%	80.00%	88.89%	-
Domino 2-2	100.00%	100.00%	100.00%	-
Domino 3-0	100.00%	84.62%	84.62%	-
Domino 3-1	100.00%	88.89%	88.89%	-
Domino 3-2	100.00%	100.00%	100.00%	-
Domino 3-3	81.82%	72.73%	88.89%	-
Domino 4-0	100.00%	81.82%	81.82%	Domino 3-0
Domino 4-1	100.00%	78.57%	78.57%	Domino 3-1
Domino 4-2	100.00%	100.00%	100.00%	-
Domino 4-3	100.00%	100.00%	100.00%	-
Domino 4-4	100.00%	81.82%	81.82%	-
Domino 5-0	83.33%	75.00%	90.00%	-
Domino 5-1	100.00%	90.00%	90.00%	-
Domino 5-2	80.00%	50.00%	62.50%	-
Domino 5-3	90.91%	72.73%	80.00%	-
Domino 5-4	90.91%	90.91%	100.00%	-
Domino 5-5	100.00%	90.00%	90.00%	-
Domino 6-0	100.00%	90.00%	90.00%	-
Domino 6-1	100.00%	88.89%	88.89%	-
Domino 6-2	88.89%	66.67%	75.00%	-
Domino 6-3	88.89%	88.89%	100.00%	-
Domino 6-4	100.00%	75.00%	75.00%	-
Domino 6-5	100.00%	88.89%	88.89%	-
Domino 6-6	90.00%	90.00%	100.00%	-
Gesamtergebnis	95.11%	85.49%	89.78%	

Tabelle E.8: Detailansicht der Ergebnisse der vierten Abstandsfunktion angewendet auf den Datensatz DS_1.