UNIVERSITÄT
KOBLENZ · LANDAU
Fachbereich 4: Informatik

Modellbasierte Poseschätzung in monokularen Aufnahmen unter Verwendung geometrischer Modelle

Dissertation
von
Stefan Wirtz

Koblenz, 2017

Vom Promotionsausschuss des Fachbereichs 4: Informatik der Universität Koblenz-Landau zur Verleihung des akademischen Grades **Doktor der Naturwissenschaften (Dr. rer. nat.)** genehmigte Dissertation.

Datum der wissenschaftlichen Aussprache: 12.07.2017
Vorsitzende des Promotionsausschusses: Prof. Dr. Maria A. Wimmer

Vorsitzender der Promotionskommision: Prof. Dr. Klaus Diller
Erster Berichterstatter: Prof. Dr.-Ing. Dietrich Paulus
Zweiter Berichterstatter: Prof. Dr. Jürgen Ebert

Lebenslauf

Stefan Wirtz graduated from Otto-Hahn Gymnasium Monheim am Rhein in 2001 and obtained a diploma in Biomathematics (Dipl.-Math. (FH)) from the University of Applied Science RheinAhrCampus Remagen in 2008. Since 2012, he has been working for Motec GmbH, specialising in Advanced Driver Assistance Systems development. Previously, he was based at the Institute of Computational Visualistics at the University of Koblenz-Landau, where he was part of the Active Vision Working Group (Prof. Paulus). He was offered a role as PhD student on the "Software Techniques for Object Recognition (STOR) project", which was funded by the German Research Foundation (DFG). His scientific interests are in the fields of Image Processing, Pattern Recognition and in particular the handling of uncertain knowledge.

Stefan Wirtz

Vorwort

Die Grundlagen dieser Dissertation entstanden während meiner Zeit als wissenschaftlicher Mitarbeiter in der Arbeitsgruppe Aktives Sehen bei Herrn Prof. Dr.-Ing. Paulus. Dabei wurde die Arbeit gefördert durch die DFG im Projekt Software Techniques for Object Recognition (STOR). Herrn Prof. Dr.-Ing. Paulus danke ich für das entgegengebrachte Vertrauen, das diese Arbeit erst ermöglichte, und die vielen Gespräche und die daraus entstandenen konstruktiven Anmerkungen und Anregungen, die die Arbeit maßgeblich vorangebracht haben. Auch der ganzen Arbeitsgruppe gilt mein Dank für die vielen konstruktiven Diskussionen.

Der Arbeitsgruppe Ebert und vor allem Herrn Prof. Dr. Jürgen Ebert und Frau Kerstin Falkowski danke ich für ihr Engagement im gemeinsamen STOR Projekt. Sowohl die Einführung in das weite Feld der Softwaretechnik und der Blick auf die Computervisualistik aus Sicht der Softwaretechnik, als auch die intensiven Diskussionen und Prof. Dr. Jürgen Eberts konstruktive Anmerkungen zur Arbeit, haben großen Einfluss auf die Arbeit gehabt.

Im Laufe der Arbeit werden die zum Verständnis nötigen Dinge näher definiert. Dabei werden einige der Definitionen in späteren Kapiteln weiter konkretisiert, um so den Lesefluss zu vereinfachen.

Das in dieser Arbeit verwendete Komponentenkonzept und das Modellschema entstammen den Arbeiten von Frau Kerstin Falkowski. Viele wertvolle Diskussionen haben geholfen, Modelle zu erstellen, die den Spagat zwischen Softwaretechnik und Objekterkennung meistern.

Ich möchte meinen Eltern dafür danken, dass sie mir eine gute Ausbildung ermöglicht haben. Für das Korrekturlesen bedanke ich mich bei Rosel, Hans-Werner, Anne, Jasmin und Doro. Mein Dank für ihre Unterstützung und Hilfe gilt zudem meiner ganzen Familie und auch allen Freunden. Ein besonderer Dank gilt meiner Frau, meiner Tochter und meinem Sohn, ohne deren Unterstützung und Geduld es nicht möglich gewesen wäre, die Dissertation fertigzustellen.

Stefan Wirtz

Kurzfassung

Die Forschung im Bereich der modellbasierten Objekterkennung und Objektlokalisierung hat eine vielversprechende Zukunft, insbesondere die Gebäudeerkennung bietet vielfältige Anwendungsmöglichkeiten. Die Bestimmung der Position und der Orientierung des Beobachters relativ zu einem Gebäude ist ein zentraler Bestandteil der Gebäudeerkennung.

Kern dieser Arbeit ist es, ein System zur modellbasierten Poseschätzung zu entwickeln, das unabhängig von der Anwendungsdomäne agiert. Als Anwendungsdomäne wird die modellbasierte Poseschätzung bei Gebäudeaufnahmen gewählt. Vorbereitend für die Poseschätzung bei Gebäudeaufnahmen wird die modellbasierte Erkennung von Dominosteinen und Pokerkarten realisiert. Eine anwendungsunabhängige Kontrollstrategie interpretiert anwendungsspezifische Modelle, um diese im Bild sowohl zu lokalisieren als auch die Pose mit Hilfe dieser Modelle zu bestimmen. Es wird explizit repräsentiertes Modellwissen verwendet, sodass Modellbestandteilen Bildmerkmale zugeordnet werden können. Diese Korrespondenzen ermöglichen die Kamerapose aus einer monokularen Aufnahme zurückzugewinnen. Das Verfahren ist unabhängig vom Anwendungsfall und kann auch mit Modellen anderer rigider Objekte umgehen, falls diese der definierten Modellrepräsentation entsprechen. Die Bestimmung der Pose eines Modells aus einem einzigen Bild, das Störungen und Verdeckungen aufweisen kann, erfordert einen systematischen Vergleich des Modells mit Bilddaten. Quantitative und qualitative Evaluationen belegen die Genauigkeit der bestimmten Gebäudeposen.

In dieser Arbeit wird zudem ein halbautomatisches Verfahren zur Generierung eines Gebäudemodells vorgestellt. Das verwendete Gebäudemodell, das sowohl semantisches als auch geometrisches Wissen beinhaltet, den Aufgaben der Objekterkennung und Poseschätzung genügt und sich dennoch an den bestehenden Normen orientiert, ist Voraussetzung für das Poseschätzverfahren. Leitgedanke der Repräsentationsform des Modells ist, dass sie für Menschen interpretierbar bleibt. Es wurde ein halbautomatischer Ansatz gewählt, da die automatische Umsetzung dieses Verfahrens schwer die nötige Präzision erzielen kann. Das entwickelte Verfahren erreicht zum einen die nötige Präzision zur Poseschätzung und reduziert zum anderen die Nutzerinteraktionen auf ein Minimum. Eine qualitative Evaluation belegt die erzielte Präzision bei der Generierung des Gebäudemodells.

Abstract

The research in the area of model-based object recognition and object localization has a promising future. Building recognition in particular offers a variety of possibilities for application. Determining the observer's position and orientation is a central component of building recognition.

The overall aim of this work is the development of a model-based system for the purpose of pose estimation, which operates independently of the application domain. The benefit of the system is demonstrated by a practical case study of model-based pose estimation for buildings. In preparation for the pose estimation for buildings, the model-based approach is demonstrated using dominoes and poker cards. A control strategy interprets the specific models for the application domain in order to locate models in the image and further determine their pose. The advantage of this control strategy is the ability to generalize since it is independent of the application domain. Explicit knowledge of the model is used to assign the model components to image features. These correspondences allow obtaining the camera pose from a monocular image. The procedure is independent of the application domain and is also able to handle models of other rigid objects, provided that these models follow the predefined model representation style. The calculation of the pose of a model from one single picture, which can be affected by noise and obstructions, requires a systematic comparison of the model with the picture data. Quantitative and qualitative evaluations prove how accurate the results of building poses are.

A semiautomatic procedure is introduced for the generation of the building-model. It provides the necessary precision for pose estimation while also reducing the user interactions to a minimum. The building model ensures the task of object recognition and provides a proper basis for the pose estimation procedure. This model contains semantic as well as geometrical knowledge and fulfills existing norms. The basic premise of the representation-form of the model is that it remains interpretable by humans. A semiautomatic approach was chosen, because in today's automatic generation the necessary precision cannot be achieved yet. A qualitative evaluation proves the most precise approach to the processing of building-models.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Ein Mensch kann ein Gebäude, das er noch nie gesehen hat allein dadurch erkennen, dass er eine Beschreibung des Gebäudes kennt und prinzipiell weiß wie ein Gebäude aussieht. Mit diesem Wissen kann er sich zum Gebäude orientieren und seine Position grob abschätzen (Genauigkeit sinkt mit zunehmenden Abstand zum Gebäude). Algorithmen benötigen zur Positionsschätzung in der Regel Trainingsdaten beispielsweise in Form von Fotos oder Laserdaten und bieten die Möglichkeit eine sehr präzise Schätzung der eigenen Position zu berechnen. Im Bereich der *wissensbasierten Objekterkennung/Poseschätzung* wird bereits seit Jahren geforscht [WZ02, LN04, MZWG07, BVTP10, ZZD15] und dennoch stellt sie auch heutzutage noch eine Herausforderung dar.

Das Ziel dieser Arbeit ist es die Vorteile beider Welten zu verbinden (keine Trainingsdaten und trotzdem eine präzise Poseschätzung) und ein Verfahren zu entwickeln, das nur anhand einer Aufnahme und der geometrischen Beschreibung eines auf der Aufnahme befindlichen Objektes die Position, aus der die Aufnahme gemacht wurde, präzise berechnet. Die Herausforderung besteht darin, aus einer Vielzahl plausibler Positionen die korrekte Position zu identifizieren (siehe Abbildung 1.1).

Folgende wissenschaftlichen Fragestellungen werden in der Arbeit behandelt: Wie kann die Wissensakquisition im Kontext von Gebäuden möglichst effizient, komfortabel und trotzdem präzise erfolgen? Ist die Berechnung der Pose von Gebäuden mit einer anwendungsunabhängigen Kontrolle möglich und ist sie auch dann noch möglich, wenn mit unvollständigen und unpräzisen Daten gearbeitet werden muss? Auch mit der Frage nach einer geeigneten Datenbasis beschäftigt sich diese Arbeit.

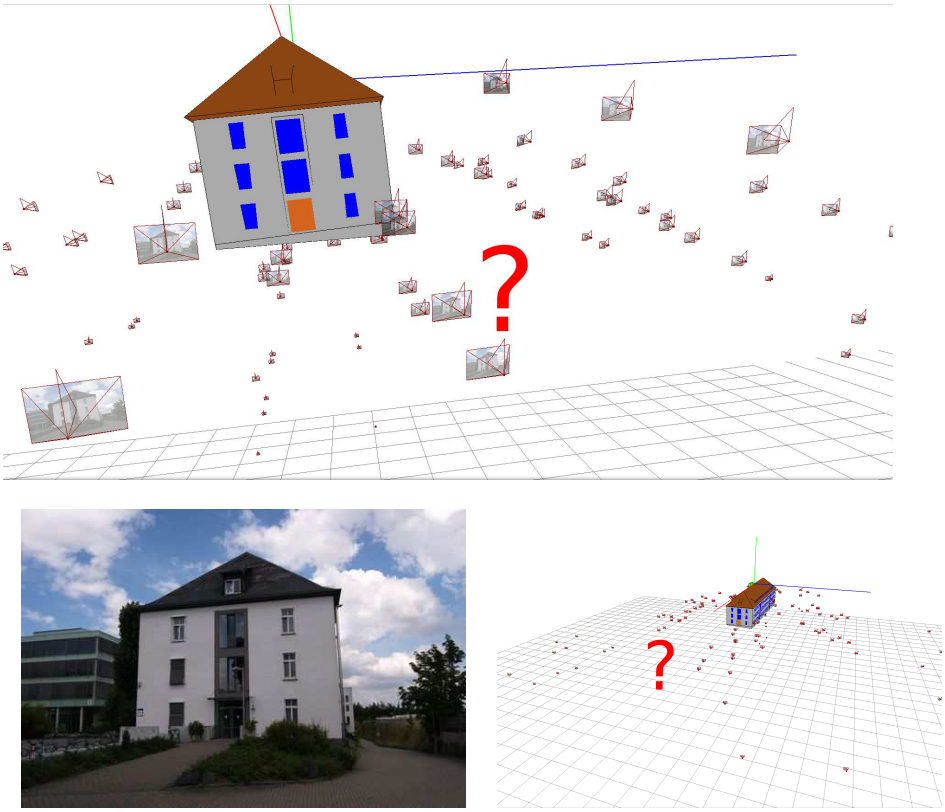


Abbildung 1.1: Ein Mensch sieht ein Gebäude und kann sich zum Gebäude orientieren. Dabei werden verschiedene Positionen in Erwägung gezogen. Die Aufgabe ist es herauszufinden welche die korrekte Position (Kamera) ist.

Ziel 1: Semiautomatische Modellgenerierung des Gebäudemodells [a]

Das Ziel ist es mit Hilfe einer einheitlichen Beschreibungssprache konkrete geometrische Beschreibungen (=geometrische Modelle (Definition folgt)) für konkrete Objekte zu generieren.

[a] (Konkretisierung folgt)

Ziel 2: Entwicklung einer anwendungsunabhängigen Kontrollstrategie zur Poseschätzung [a]

Das Ziel ist geometrische Modelle so zu interpretieren, dass daraus die Position, aus der eine Aufnahme gemacht wurde, berechnet werden kann.

[a] (Konkretisierung folgt)

Aus den Zielen 1 und 2 entstehen zwei Verfahren, zum einen zu Ziel 1 das Verfahren **sage-sb** (*Semiautomatic generation of semantic building models from image series*), das mit Hilfe einer einheitlichen Beschreibungssprache konkrete geometrische Beschreibungen für konkrete Objekte generiert. Zum anderen zu Ziel 2 das Verfahren **knoPoE** (*Knowledgebased Pose Estiamtion*), das geometrischen Modelle so interpretiert, dass daraus die Position, aus der eine Aufnahme gemacht wurde, berechnet werden kann.

1.1 Motivation

Bilder und digitale Medien beherrschen mehr und mehr das Leben in unserer Gesellschaft. Mobiltelefone sind in der Lage hochauflösende Fotos und Videosequenzen aufzunehmen und bieten leistungsstarke Plattformen für Anwendungen aller Art. Sie besitzen integrierte Navigationssysteme und via Internet per GPRS und LTE sind Umgebungsinformationen jederzeit performant zugänglich. Mit *Google Earth*[1] können die örtlichen Satellitenbilder betrachtet werden, *Google Street View*[2] ermöglicht Routenplanung mit realen Straßenbildern auf virtuellen Fußwegen durch Städte und Gemeinden. Mit *Google Sketch-Up*[3] wird den Nutzern die Möglichkeit gegeben, 3-D-Modelle von Gebäuden und Monumenten zu erstellen, in Google Earth zu registrieren und mit anderen Nutzern zu teilen.

Allein am Beispiel der Firma Google ist hier ein deutlicher Trend zu erkennen. Es steht eine große Menge frei zugänglicher Daten zur Verfügung, die nur darauf warten, ausgewertet zu werden. Die Deutung der Bilder verspricht eine Vielzahl von Anwendungsfeldern. Im urbanen Umfeld eröffnet sich die Möglichkeit der exakten Lokalisierung durch den Abgleich von Bildmerkmalen und 3-D-Modellen. Interaktive Anwendungen im Bereich der erweiterten Realität (engl. Augmented Reality) können entwickelt werden und die Navigation in Städten unterstützen. Die genannten Punkte zeigen, dass die Forschung im Bereich der modellbasierten Objekterkennung und Objektlokalisierung eine vielversprechende Zukunft, insbesondere im urbanen Bereich, bietet.

Die Bestimmung der Position und der Orientierung des Beobachters beziehungsweise der Kamera ist dabei von zentraler Bedeutung. In der Photogrammetrie werden häufig Marker in der Umgebung angebracht, aus deren Projektion ins Bild die Kameralage errechnet wird [AS09, ES09]. Weitere Möglichkeiten ohne Marker bieten die Techniken, die unter dem Begriff „Struktur aus Bewegung“ zusammengefasst werden [HZ03]. Die dritte Alternative wird in dieser Arbeit verfolgt: Explizit repräsentiertes Modellwissen wird verwendet, um eine Zuordnung von Modellbe-

[1] Google Earth: `http://earth.google.com/intl/de`
[2] Google Street View: `http://maps.google.de/help/maps/streetview`
[3] Google Sketch-Up: `http://sketchup.google.de`

standteilen und Bildmerkmalen durchzuführen. Aus diesen Korrespondenzen wird die Kamerapose aus einer monokularen Aufnahme ermittelt.

> **Definition 1.1** (Modell). *"Ein Modell ist ein zielgerichtetes Abbild eines Systems, das zum einen ähnliche Beobachtungen und Aussagen wie das System selbst ermöglicht und zum anderen diese Realität durch Abstraktion auf die jeweils problembezogen relevanten Aspekte vereinfacht." [Win00, Seite 104]*

Ein Ansatzpunkt dieser Arbeit ist die Repräsentation des Wissens in eine für den Menschen interpretierbare Form. Dies kann man vergleichen mit dem Versuch, einem anderen Menschen zu beschreiben wie ein konkretes Haus aussieht, während man vor einer Reihe Häuser steht. Vorausgesetzt, die Beschreibung ist stimmig, ist man nun in der Lage das Haus zu identifizieren und sich im Verhältnis zum Haus zu lokalisieren. Es ist alleine das Wissen vonnöten, wie ein Haus im Allgemeinen aussieht, inklusive der Beschreibung des Hauses im Speziellen. Eine solche Beschreibung könnte beispielsweise lauten: "Das gesuchte Haus hat zwei Etagen, in der Fassade der Vorderseite befinden sich drei Fenster und eine Tür und das Dach ist ein Satteldach.".

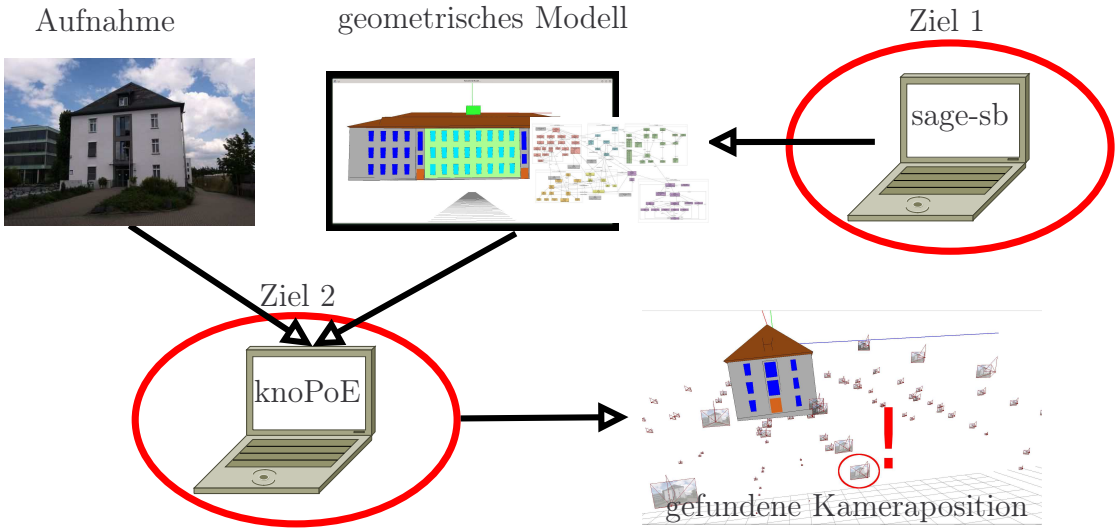


Abbildung 1.2: Übersicht vom Verfahren **knoPoE** zur Bestimmung der Position der Aufnahme. Dabei wird sowohl eine Aufnahme eines Gebäudes als auch ein geometrisches Modell übergeben. Als Ausgabe erhält man die Position der Eingangsaufnahme.

Das Verfahren **knoPoE** berechnet die Pose der Aufnahme und verwendet dazu allein das geometrische Modell des Objektes. Handelt es sich bei dem Objekt um ein Gebäude entstammt das geometrischen Modells dem Verfahren **sage-sb**. Abbildung 1.2 setzt die Verfahren **sage-sb** und **knoPoE** am Fallbeispiel von Gebäuden in Beziehung.

Begriffsbestimmung 1.1 (Pose)**.** *Die Pose beschreibt die Position und Blickrichtung einer Kamera in der Welt zu einem bestimmten Zeitpunkt. (angelehnt an [Dec12])*

Man unterscheidet zwischen *relativer* und *absoluter Pose*.

Begriffsbestimmung 1.2 (Relative und absolute Pose)**.** *Die relative Pose beschreibt die Pose zwischen zwei Kamerabildern und stellt die Voraussetzung für eine automatische bzw. semiautomatische Modellgenerierung aus monokularen Kamerabildpaaren dar. Die absolute Pose beschreibt eine Kamerapose relativ zu dem verwendeten Weltkoordinatensystem. (angelehnt an [Dec12])*

Diese Arbeit untersucht die *wissensbasierte Poseschätzung* in monokularen Aufnahmen und setzt diese am Beispiel von Gebäuden um. Die Begrifflichkeiten zur Pose werden im Kapitel 3.1 definiert und die verwendeten geometrische Modelle werden in Kapitel 3.3 und 3.4 beschrieben.

Das Verfahren **knoPoE** wurde zunächst für Dominosteine (ohne perspektivische Verzerrung) entwickelt, dann für Pokerkarten (mit perspektivischer Verzerrung) erweitert und im Abschluss an die Anforderungen von Gebäuden in der realen Welt adaptiert (siehe Abbildung 1.3). In Kapitel 2 werden die Fallstudien im Detail vorgestellt.

Abbildung 1.3: Entwicklungsabfolge von **knoPoE** durch verschiedene Fallstudien.

Der folgende Abschnitt klärt Grundbegriffe, die zum Verständnis der Arbeit benötigt werden und zum Teil auch schon (ohne Einführung) verwendet wurden.

1.2 Grundbegriffe der Zielstellungen

Allgemein anerkannt ist, dass *Wissen* über die Einsatzdomäne der Objekte benötigt wird, um Objekte klassifizieren zu können. Quint [Qui97, Seite 37] unterscheidet im Fall der Bildauswertung zwischen vier Arten von Wissen: physikalisches, geometrisches, topologisches und semantisches Wissen.

Das *physikalische Wissen* beschreibt hauptsächlich die Oberflächeneigenschaften, die zusammen mit Sensor und Beleuchtung die Pixelwerte im Bild bestimmen. *Geometrisches Wissen* beschreibt die zu analysierenden Objekte hinsichtlich ihrer

Form- und Größeneigenschaften. *Topologisches Wissen* beschreibt die Lage und Anordnung der Modellbestandteile im Raum und das *semantische Wissen* gibt Objekten eine Bedeutung. Dies geschieht indem Objekten ein Bezeichner zugewiesen wird, der in einer gemeinsamen Sprache eine vereinbarte Bedeutung hat.

Definition 1.2 (Wissen)**.** *"Im Kontext der Mustererkennung repräsentiert Wissen Bedingungen und Beziehungen zwischen Objekten und Ereignissen, welche in der realen Welt vorkommen können. Dabei ist ein sehr wichtiger Punkt, dass das Wissen unvollständig, unpräzise und unsicher sein kann." (aus dem Englischen [Nie90, Seite 271])*

Konventionelle Programme integrieren das Wissen über den Problembereich direkt in die Software. In dieser Arbeit kommt ein wissensbasiertes System zum Einsatz, das das Wissen über den betreffenden Problembereich (der *Wissensbasis*) von der Verarbeitung dieses Wissens (der *Wissensverarbeitung*) trennt. Jung [Jun02, Seite 13] untergliedert ein wissensbasiertes System noch etwas feingranularer in eine *Wissensbasis*, *Wissensakquisitionskomponente*, *Inferenzkomponente*, *Dialogkomponente* und eine *Erklärungskomponente*.

Definition 1.3 (Wissensbasis)**.** *"Die Wissensbasis enthält das Wissen, mit dem eine Problembearbeitung erfolgen kann. Das Wissen kann dazu mittels unterschiedlicher Strukturen repräsentiert werden. Ist das Wissen in Form einheitlicher Strukturen in der Wissensbasis abgelegt, wird sie als homogen, anderenfalls als heterogen bezeichnet." [Jun02, Seite 13]*

Definition 1.4 (Wissensakquisitionskomponente)**.** *"Die Wissensakquisitionskomponente erlaubt den Aufbau und die Modifizierung der Wissensbasis durch das Hinzufügen, Ändern oder Löschen von Wissensstrukturen entsprechend der für das System gewählten Darstellungsweise des Wissens. Die dazu bereitgestellten Werkzeuge können vom einfachen Texteditor zum Schreiben von Quelltext, bis zu kompliziert aufgebauten Eingabewerkzeugen reichen." [Jun02, Seite 13]*

"Die *Dialogkomponente* steuert den Dialog zwischen System und Benutzer. Ihre Hauptaufgabe besteht in der Aufarbeitung der systeminternen Darstellungen des Wissens für den Benutzer, sodass es verständlich erscheint und eine Kommunikation in adäquater Form erfolgen kann. ... Um die Transparenz von Eingabeaufforderungen für den Benutzer zu sichern, ist die Kopplung der Dialogkomponente mit einer Erklärungskomponente sinnvoll." [Jun02, Seite 13]

"Die Aufgaben der *Erklärungskomponente* beschränken sich nicht nur auf eine Darstellung von Hilfetexten zur Benutzerführung. Durch spezielle Protokollfunktionen ist sie in der Lage, Fragestellungen und Ergebnisse zu begründen. Benut-

zeranfragen werden z. B. transparenter, wenn erklärt wird, warum sie gestellt wurden. Ergebnisse sind zu verstehen, wenn aufgezeigt werden kann, wie sie gefolgert wurden. " [Jun02, Seite 13]

Definition 1.5 (Inferenzkomponente bzw. Kontrolle)**.** *"Das gespeicherte Wissen wird durch die Inferenzkomponente verarbeitet. Sie enthält bestimmte Strategien (Inferenzmechanismen), mit deren Hilfe eine Lösung für ein vom Benutzer behandeltes Problem generiert wird. Dabei werden Fakten, die während des Lösungsprozesses gefolgert werden, ebenfalls in der Wissensbasis - stationär oder temporär - abgelegt. Hartmann und Lehner [HL13] bezeichnen Inferenz als eine Verarbeitungsstrategie, die Fähigkeiten elementaren Denkens besitzt, aber selbst über kein Wissen verfügt." [Jun02, Seite 13] Die Inferenzkomponente wird im Folgenden Kontrolle genannt.*

In enger Kooperation mit dieser Arbeit entwickelte Falkowski [FE09a, FE11, WFP12] sowohl ein *Referenz-* und *Metamodell* als *Graphenkonzept* [ERW08] als auch ein spezielles Gebäudeschema basierend auf dem Referenzschema.

Definition 1.6 (Metamodell)**.** *"Ein Metamodell ist die konzeptionelle Beschreibung einer Modellierung, die sowohl die verwendeten Modellierungskonzepte als auch ihre Verwendung verdeutlicht." [Win00, Seite 116]*

Das Referenz- und Metamodell (im Folgenden *Referenzschema* genannt) repräsentiert Wissen aus den Bereichen der Computervisualistik, der Bildverarbeitung und Computergrafik und ermöglicht die algorithmische Verarbeitung von dazu konformen Modellen (siehe Kapitel 3.3). Auf Basis dieses Referenzschemas wurde ein *Gebäudeschema* (als *spezielles Modell*) mit Fokus auf die Poseschätzung entwickelt. Zudem wurden für die Fallstudien zur Dominostein- und Pokerkartenerkennung ebenfalls spezielle Objekt-Modell-Schemata entwickelt.

Definition 1.7 (Referenzmodell)**.** *"Ein Referenzmodell ist ein ausgewiesenes Modell, das charakteristische Eigenschaften einer Klasse gleichartiger Systeme beschreibt und als Bezugspunkt für spezielle Modelle dient." [Win00, Seite 106] Das Referenzmodell wird im Folgenden Referenzschema genannt.*

Definition 1.8 (Spezielle Modelle)**.** *"Spezielle Modelle für konkrete Modellierungsaufgaben erhält man durch Übernahme und ggfs. Anpassung des Referenzmodells." [Win00, Seite 9]*
Die speziellen Modelle werden in dieser Arbeit Objekt-Modell-Schemata genannt.

Begriffsbestimmung 1.3 (Geometrisches Modell)**.** *Unter einem geometrischen Modell wird in dieser Arbeit verstanden, dass es die 3-D- und 2-D-Geometrie eines Objektes beschreibt. Aus diesem Grund sind die Modelle von Dominosteinen, Pokerkarten und Gebäuden als geometrische Modelle anzusehen, da sie die geometrischen Informationen der Objekte enthalten.*

Ziel 1: Semiautomatische Modellgenerierung des Gebäudemodells - Konkretisierung
Das Ziel ist die Entwicklung und Umsetzung eines möglichst automatischen und präzisen Verfahren zur Modellgenerierung des Gebäudemodells aus dem Gebäudeschema. Dies entspricht der zuvor beschriebenen Wissensakquisitionskomponente.

Die hierbei entstehenden Gebäudemodelle dienen als Eingabe für die anwendungsunabhängige Kontrollstrategie (Ziel 2 (**knoPoE**)).

Es werden für die Poseschätzung verschiedene Gebäudemodelle generiert. Zur weitgehend automatischen Modellgenerierung von Gebäudemodellen, die später auch zur Interpretation von Bildern verwendet werden können, sind die folgenden Schritte erforderlich (siehe Abbildung 1.4): Bildaufnahme, Merkmalszuordnung, Poseschätzung, 3-D-Rekonstruktion, semantische Interpretation (Fassaden, Dach, Fenster, usw.) und Modellerzeugung.

Eine weitere Herausforderung dieser Arbeit ist es, ein System zu entwickeln, das unabhängig von der Anwendungsdomäne unter Verwendung von anwendungsspezifischen Modellen konform zum Referenzschema die Pose bestimmt. In dieser Arbeit kommt dabei das Gebäudeschema und entsprechende Gebäudemodelle zum Einsatz. Ein *Analysemodell* beinhaltet alle bei der Bildanalyse extrahierten Informationen und deren Beziehung zueinander.

Begriffsbestimmung 1.4 (Analysemodell)**.** *Das Analysemodell wird aus dem in dieser Arbeit verwendeten Referenzschema (siehe STOR-Referenzschema in Kapitel 3.3) erzeugt und speichert die bei der Bildanalyse extrahierten Informationen, Schlussfolgerungen anhand dieser Informationen und deren Beziehung zueinander.*

Die bei der Bildanalyse extrahierten und im Analysemodell gespeicherten Informationen werden mit den Elementen der Objektmodelle in Verbindung gesetzt, um so die Kamerapose zu ermitteln. Es ist naheliegend, dass bei der Bildanalyse mit unvollständigen, unpräzisen und unsicheren Daten umgegangen werden muss. Dies führt dazu, dass Zuordnungen zwischen Modellelementen und Elementen des Analysemodells nicht eindeutig sind. Konkurrierende Zuordnungen werden in unterschiedlichen *Hypothesen* gespeichert, welche wiederum in einem sogenannten

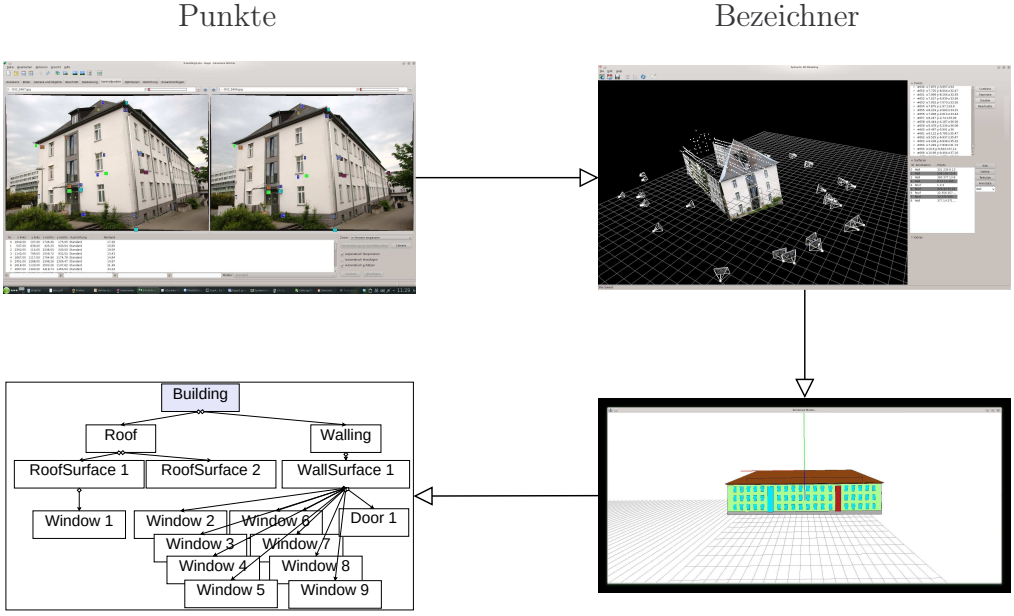


Abbildung 1.4: Ein Gebäudemodell wird erzeugt, indem Punktkorrespondenz ausgewählt, Flächen mit einem Bezeichner versehen und automatisch geometrische Beziehungen ermittelt werden.

Hypothesenraum verwaltet werden. Jede Zuordnung und jede Hypothese besitzt eine Bewertung in Form eines *Vertrauensmaßes*.

Definition 1.9 (Hypothese)**.** *Eine Hypothese, im Verständnis dieser Arbeit, beschreibt eine oder mehrere Zuordnungen zwischen Elementen des Analysemodells und eines speziellen Modells. Jede Hypothese besitzt eine Bewertung in Form eines Vertrauensmaßes.*

Definition 1.10 (Hypothesenraum)**.** *Der Hypothesenraum spannt einen Baum [GRRW10] mit Hypothesen auf und kann, da jede Hypothese ein Vertrauensmaß besitzt, schon während der Erstellung mit Graph-Suchalgorithmen durchsucht werden. Die Kontrolle nutzt den Hypothesenraum, um die Analyse zu steuern. Durch die Baumstruktur ist zu jeder Hypothese die Entstehungsgeschichte bekannt. Der Hypothesenraum und dessen Verarbeitung wird in Kapitel 6.4 beschrieben.*

Definition 1.11 (Baum)**.** *Ein Baum ist ein spezieller Graph und besitzt folgende Eigenschaften. Ein Knoten ist als Wurzel gekennzeichnet. Mit Ausnahme der Blätter gehen von jedem Knoten Kanten aus und Knoten mit dem-*

selben Elternknoten sind Geschwisterknoten. Blätter haben keine Kinder und von der Wurzel führt zu jedem Knoten genau ein Weg.

In dieser Arbeit wird die Dempster-Shafer-Theorie [BKI06] zur Modellierung der Vertrauensmaße verwendet, sodass die Möglichkeit besteht Unwissenheit explizit zu modellieren und Vertrauensmaße komfortabel zu kombinieren. Ein Vertrauensmaß ist nicht mit einer Wahrscheinlichkeit gleichzusetzen.

Definition 1.12 (Vertrauensmaß)**.** *Ein Vertrauensmaß beschreibt das Vertrauen in das Eintreffen eines Ereignisses. Im Folgenden wird das Ergebnis eines Vertrauensmaßes Belief genannt.*

Der Prototyp eines Systems mit menschenlesbaren Modellen und anwendungsunabhängigen Kontrollverfahren stellt *ERNEST* [NSSK90] dar. Es wurde bereits in den 90er Jahren entwickelt und identifiziert Objekte durch die Instanziierung von Konzepten[4] in *semantischen Netzwerken*, wobei es in verschiedensten Anwendungsdomänen beheimatet ist, wie beispielsweise Luftbild-Gebäudeerkennung [Qui97] und Spracherkennung [AFD+99].

Definition 1.13 (Semantisches Netz)**.** *"Ein semantisches Netzwerk ist ein Spezialfall eines gerichteten Graphen. Die markierten Knoten und Kanten des Graphen dienen der Strukturierung von Wissen und werden in unterschiedlichen Typen eingeteilt. Knoten bilden die grundlegende Wissenseinheit. Beziehungen zwischen den Wissenseinheiten werden durch Kanten dargestellt." [Sal95, Seite 122]*

Eine anwendungsunabhängige Kontrolle steuert den Ablauf der Modellanalyse. Mit anwendungsunabhängig ist gemeint, dass ein beliebiges rigides Objekt als geometrisches Modell zur Eingabe vorliegen kann. Das Modell wiederum beeinflusst über seine Bestandteile die Bildanalyse. Im Zuge der Analyse entstehen Hypothesen für mögliche Posen und Zuordnungen. Diese werden in einem Hypothesenraum verwaltet. Dies erfordert die Möglichkeit Hypothesen zu bewerten und zu verifizieren. Das Ergebnis der Analyse ist eine Pose inklusive ihrer Bewertung.

Ziel 2: Entwicklung einer anwendungsunabhängigen Kontrollstrategie zur Poseschätzung - Konkretisierung
Das Ziel ist die Entwicklung und Umsetzung einer anwendungsunabhängigen Kontrollstrategie zur Poseschätzung und Belegung ihrer Güte am Beispiel der Poseschätzung von Gebäuden. Dabei soll sowohl das Analyseergebnis, als

[4] Durch *Konzepte* werden in ERNEST Begriffe, Oberklassen oder Klassen von Ereignissen dargestellt.

auch die Zwischenergebnisse interpretierbar und nachvollziehbar sein. Das entspricht der zuvor beschriebenen Kontrolle und der Erklärungskomponente.

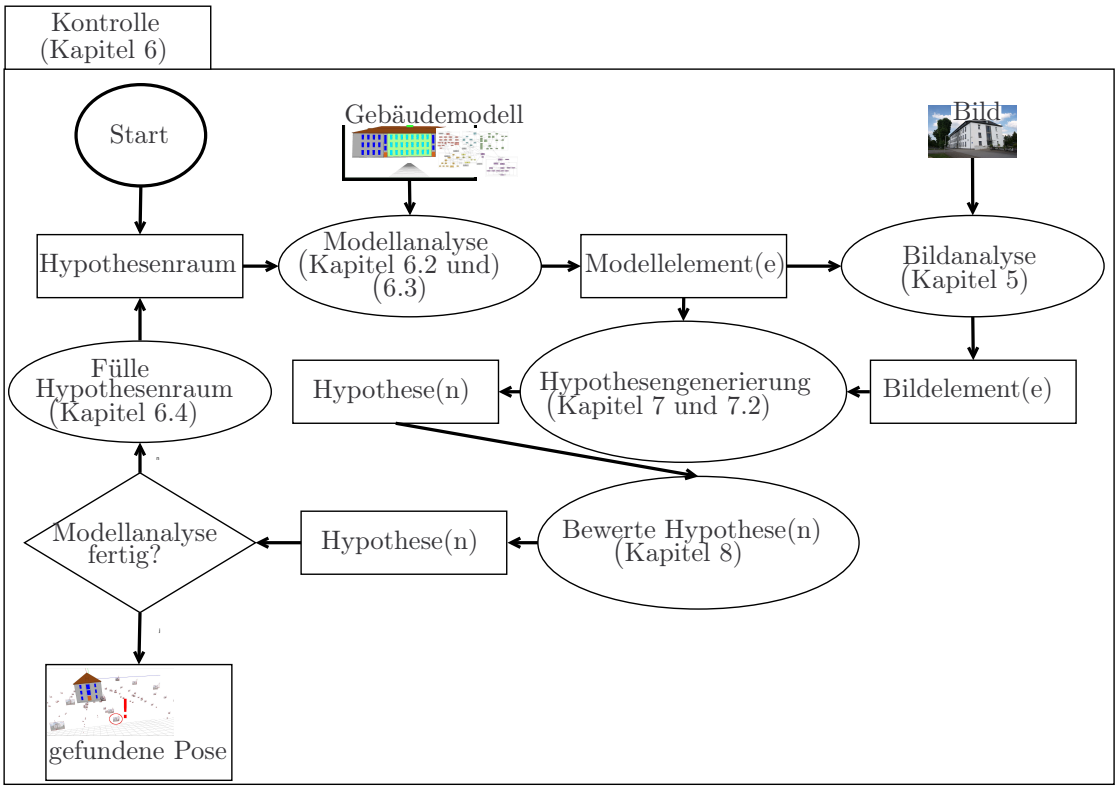


Abbildung 1.5: Übersicht über die wichtigsten Komponenten bei der Posebestimmung von **knoPoE** . Die Kontrolle steuert die Modellanalyse, dabei initiieren die Modelle über ihre Bestandteile die Bildanalyse. Die bei der Bildanalyse extrahierten Informationen werden mit den Elementen der Modelle in Verbindung gesetzt, um so die Kamerapose zu ermitteln. Konkurrierende Zuordnungen werden in unterschiedlichen Hypothesen gespeichert, welche im Hypothesenraum verwaltet werden. Jede Zuordnung und jede Hypothese besitzt eine Bewertung in Form eines Vertrauensmaßes.

Abbildung 1.5 beschreibt die Zusammenhänge und listet die wichtigsten Komponenten für das Verfahren **knoPoE** auf. Diese Komponenten finden sich auch in der Gliederung der Arbeit wieder. Die Kontrolle (Kapitel 6) steuert die Modellanalyse, dabei initiieren die Modelle (Kapitel 3) über ihre Bestandteile die Bildanalyse. Kapitel 5 beschreibt Verfahren, die bei der Bildanalyse zum Einsatz kommen, wobei die Ergebnisse der Bildanalyse, wie in Abbildung 1.6 (Veränderung durch graue Einfärbung kenntlich gemacht) angedeutet, wiederum die Modellanalyse beeinflussen. Die bei der Bildanalyse extrahierten und im Analysemodell gespeicherten Informationen werden mit den Elementen der Modelle in Verbindung gesetzt, um

so die Kamerapose (Kapitel 7.2) zu ermitteln. Es ist naheliegend, dass bei der Bildanalyse mit unvollständigen, unpräzisen und unsicheren Daten umgegangen werden muss. Dies führt dazu, dass Zuordnungen zwischen Modellelementen und Elementen des Analysemodells nicht eindeutig sind. Konkurrierende Zuordnungen werden in unterschiedlichen Hypothesen (Kapitel 7) gespeichert, welche im Hypothesenraum verwaltet werden. Jede Zuordnung und jede Hypothese besitzt eine Bewertung in Form eines Vertrauensmaßes (Kapitel 8). Die Kontrollstrategie analysiert, wie in Abbildung 1.5 beschrieben, Top-Down ein Gebäudemodell. Die Bildanalyse beeinflusst aber zusätzlich Bottom-Up durch Ergebnisse in der Bildanalyse den Ablauf der Analyse (siehe Abbildung 1.6).

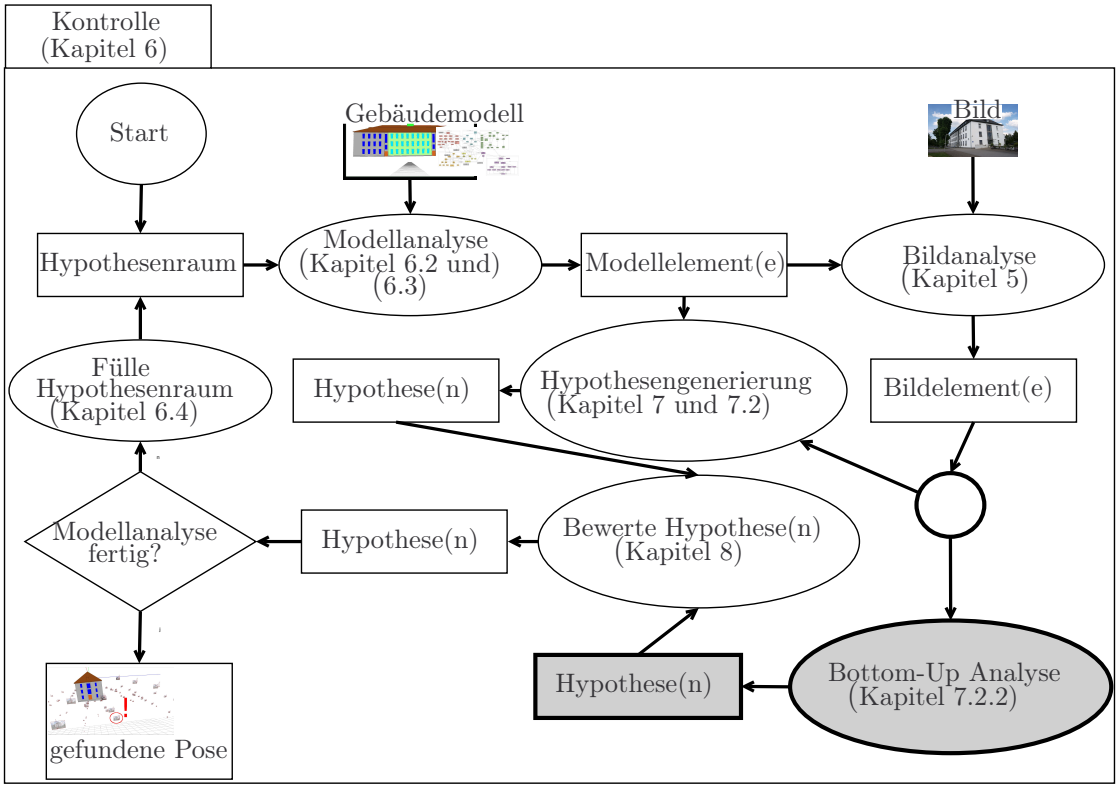


Abbildung 1.6: Kontrollstrategie analysiert sowohl Top-Down als auch Bottom-Up. Abbildung 1.5 wurde um die grau eingefärbten Elemente ergänzt.

1.3 Stand der Wissenschaft zur modellbasierten Objekterkennung und Poseschätzung

Der Stand der Wissenschaft unterteilt sich, entsprechend dem Vorgehen in der modellbasierten Poseschätzung, in die Bereiche Modellgenerierung, Fenster-, Fassaden-, Gebäudeerkennung und modellbasierte Posebestimmung.

1.3.1 Modellgenerierung

Verschiedene Forschungsgruppen haben in den letzten Jahren mit unterschiedlichem Fokus an der Rekonstruktion von Gebäuden und urbanen Szenen aus Bildern gearbeitet.

Debevec et al. [DTM96] stellen einen fotobasierten Ansatz zum Modellieren und Rendern von architektonischen Szenen vor. Der Modellierungsansatz kombiniert sowohl geometriebasierte als auch bildbasierte Modellierungstechniken. Mit dem geometriebasierten Ansatz wird ein Modell erzeugt, das die grundlegende Geometrie des Gebäudes wieder herstellt. Der bildbasierte Ansatz verwendet Struktur aus Bewegungstechniken, um die zuvor gewonnene grundlegende Geometrie in eine präzise Darstellung zu überführen. Ihr System benötigt Benutzerinteraktion und bietet typische geometrische Primitive, die sich zur Anwendung in architektonischen Szenen eignen. Diese Primitiven werden benutzt, um Oberflächenmodelle zu erstellen. Darüber hinaus wird das Modell texturiert, indem Bilder auf die Oberflächen abgebildet werden.

Lee et al. [LHN00] schlagen eine effiziente Methode zur 3-D-Modellierung von komplexen Gebäuden mittels Benutzerinteraktion vor. Es wird versucht die Modellierungszeit und Anzahl an Interaktionen zu minimieren, die benötigt werden, um das automatische System bei dieser Aufgabe zu unterstützen. In diesem Ansatz werden Luftbildaufnahmen verwendet, um so großflächige Modelle zu erzeugen. Dem Ansatz liegt ein automatisches System zugrunde. Dieses System generiert aus mehreren Luftbildaufnahmen auf Basis von Strecken, Schnittpunkten und deren geometrischer Beziehung zueinander rechteckige 3-D-Modelle. Im nächsten Schritt muss der Anwender per Hand sinnvolle Strecken, Schnittpunkte und Modellhypothesen mittels Interaktionen auswählen, um so Modelle mit komplexerer Geometrie zu erzeugen.

Außerdem präsentieren Lee et al. [LJN02] eine Methode zur automatischen Integration von Bodenaufnahmen in 3-D-Modelle, um auf diese Weise hoch aufgelöste Fassadentexturen für 3-D architektonische Modelle zu erhalten. Sie verwenden einen hybriden Merkmalsextraktionsansatz, der zum einen eine globale Geradengruppierung basierend auf der *Gauss-Sphäre* und zum anderen eine lokale Geradengruppierung basierend auf einer Regionssegmentierung nutzt.

Weiterhin stellen Lee et al. [LN04] eine Methode zur Rekonstruktion der 3-D-Fenster von Gebäuden aus einer einzelnen kalibrierten Bodenaufnahme vor. Sie extrahieren automatisch Fenster im rektifizierten Bild und nutzen dafür eine Profilprojektionsmethode, die die Regelmäßigkeit der vertikalen und horizontalen Fenster-Platzierungen nutzt und diese klassifiziert, um die extrahierten Fenster mittels ihrer Texturinformationen zu klassifizieren. Eine semantische Annotation findet hierbei nicht statt.

Dick et al. [DTC04] beschreiben ein Framework zur automatischen Strukturbestimmung und zur Identifizierung von Teilen der Architektur aus einer kleinen Menge an Bodenaufnahmen (2-6). Das Ergebnis ist ein 3-D-Oberflächenmodell in dem die Oberflächen gekennzeichnet sind (Fenster, Tür, Säule, Dach, Stufen, usw.).

Rusu et al. [RSM+08] präsentieren ein System zur Echtzeit-3-D-Kartierung, wobei als Daten Punktwolken, gewonnen aus Stereoaufnahmen, dienen. Die lokal akquirierten Ansichten der Punktwolken werden in einem konsistenten Frame registriert und in eine Polygonrepräsentation transformiert. Basierend auf den geometrischen Eigenschaften der einzelnen Polygone werden automatisch semantische Annotationen hinzugefügt (“Boden”, “Stockwerk”, “Vertikal”, “Stufen”, “Unbekannt”).

Lafarge et al. [LKBH13] beschreiben ein Verfahren, das aus Stereoaufnahmen automatisch ein 3-D-Modell bestehend aus Polygonnetzen und Primitiven erzeugt. Zur Segmentierung der Polygonnetze kommt ein Markov-Random-Field zum Einsatz. Ein *Markov-Random Field* ist ein statistisches Modell, welches ungerichtete Zusammenhänge in einem Feld beschreibt. Die resultierenden Segmentierungen werden per Jump-Diffusion (wenn möglich) in Primitive überführt. *Jump-Diffusion* ist ein stochastischer Prozess, der Sprünge und Diffusion berücksichtigt.

Larfarge und Alliez [LA13] stellen ein robustes Verfahren vor, das eine 3-D-Punktwolke so strukturiert, dass ein polygonales Modell entsteht. Dafür werden zunächst mittels *RANSAC* (**ran**dom **sa**mple **c**onsensus) Polygone aus der Punktwolke extrahiert. Die Polygone und Punkte werden automatisch kombiniert, sodass im resultierenden Modell Rauschen unterdrückt und trotzdem Strukturen erhalten bleiben. Dazu wird ein *Graph-Cut* auf der Delauny-Triangulation der vorher strukturierten Punkte durchgeführt. Die *Delauny-Triangulation* erstellt aus einer Punktwolke ein Dreiecksnetz.

Ein weiterer Ansatz zur Erzeugung von semantischen 3-D-Modellen wird von Nguatem et al. präsentiert. Auf Basis einer 3-D-Punktewolke einer Fassade präsentieren sie einen Ansatz, der in der Lage ist automatisch Fenster bzw. Türen aus der Fassade zu extrahieren [NDM14] und auch das Dach zu rekonstruieren [NDM13]. Dieses mit Semantik angereicherte Modell kann nach *CityGML* (siehe Kapitel 3.2.3) exportiert werden.

Simon et al. [STKP11] beschreiben ein Verfahren, das Gebäudemodelle auf Basis des Grundrisses und eines Farbbildes erstellt. Hierzu werden Gebäude im *Pariser Haussmann-Stil* als *Form-Grammatik* (*Shape-Grammar*) beschrieben. Die Pixel der Bilder werden durch ein Markov Random Field, einen Random Forest oder durch Gaussian-Mixed-Models automatisch mit Wahrscheinlichkeiten für semantische Klassen versehen. Ein *Random-Forest* ist ein Klassifikationsverfahren, das aus mehreren Entscheidungsbäumen besteht. Das*Gaussian-Mixed-Models* ist eine parametrische Dichtefunktion, welche als gewichtete Summe von Gauss-Dichten besteht. Per *Random-Walk* werden die Grammatiken so angepasst, dass die wahrscheinlichste Kombination an Regionen gefunden wird. Dieses Verfahren funktioniert relativ gut für Gebäude im Pariser Haussmann-Stil, ist aber nur schwer auf andere Gebäudearten anpassbar.

Venegas et al. [VAB10] stellen einen Ansatz vor, der automatisch 3-D-Gebäudemodelle aus kalibrierten Luftbildaufnahmen von *Manhattan-World*-Gebäuden erstellt. Dabei werden zunächst der Hintergrund und die Fenster entfernt. Eine speziell auf den Anwendungsfall Manhattan-World angepasste Grammatik wird verwendet, um die Fassaden und Etagen sowie deren Ausrichtung zu bestimmen. Die Gebäudemodelle werden texturiert. Der Ansatz ist vollautomatisch und sehr spezialisiert, wobei die Etagen nicht sicher identifiziert werden, was aber auch nicht Ziel des vorgestellten Ansatzes ist. Das 3-D-Gebäudemodell besitzt dadurch eine schöne Optik, aber wenig Semantik, sodass sich dieses Vorgehen für Modelle zur wissensbasierten Gebäudeerkennung kaum eignet.

1.3.2 Fenster-, Fassaden- und Gebäudeerkennung

Dieser Abschnitt gibt einen Überblick über Arbeiten zur Gebäudeerkennung und Gebäuderekonstruktion. Es wird in ansichts-, merkmals-, modellbasierte Ansätze und Ansätze, die neuronale Netze verwenden, unterteilt. Im Folgenden wird nicht zwischen Arbeiten zur Gebäudeerkennung und solchen zur Gebäuderekonstruktion unterschieden. Das hat den Grund, dass die Aufgaben thematisch sehr nah beieinander liegen und oftmals ähnliche Vorgehensweisen genutzt werden. So bedingt eine automatische Gebäuderekonstruktion in der Regel die Erkennung von Gebäudeelementen.

Ansichtsbasierte Erkennungsverfahren Eine ansichtsbasierte Erkennung basiert nicht auf Merkmalen, sondern auf Bildern der zu erkennenden Objekte.

In [LLS04, LLS06] wird ein allgemeiner Ansatz zur Erkennung von Objektklassen unter Verwendung eines *impliziten Formmodells* (engl. Implicit Shape Model) vorgestellt. Es handelt sich um ein Verfahren, das Erkennung und Segmentierung gleichzeitig vollführt, indem beide Prozesse in ein Bezugssystem basierend auf

Wahrscheinlichkeiten zusammengefügt werden. Das Verfahren kann in zwei Schritte unterteilt werden: Im ersten Schritt werden charakteristische Merkmale (Ansichten von Teilen des Objekts) für jede Klasse von Objekten gelernt. Auf diese Weise wird ein sogenanntes *Codebook of local Appearance* erstellt, das zum späteren Vergleichen dient. Im zweiten Schritt wird für diese lokalen Strukturen angegeben, wo sie im Objekt auftreten können. Nun werden Teile des Eingabebildes mit Einträgen im Codebuch verglichen und daraus Wahrscheinlichkeiten für mögliche erkannte Objekte und Positionen abgeleitet.

Aufbauend auf der Arbeit von Leibe und Schiel ist die Arbeit von Mayer und Reznik [RM07]. Die Methode arbeitet auf perspektivisch entzerrten Fassaden in einer Serie von Bildern. Mayer und Reznik wenden die impliziten Formmodelle ausschließlich auf Fenster an. Aus mehreren Ansichten eines Gebäudes modellieren sie die Fassade dreidimensional mit Hilfe der ansichtsbasierten Erkennung und beschreiben Fenster durch das implizite Formmodell. Die gefundenen Fenster, welche als Bestandteile einer Fassade als Reihe, Spalte oder als Raster organisiert sind, werden gruppiert und bilden so die Fassade. Da mit einer Serie von Bildern gearbeitet wird, können abschließend die Fassaden mittels *Plane Sweeping* dreidimensional rekonstruiert werden.

Eine Arbeit zum Extrahieren markanter Merkmale einer Fassade (wie beispielsweise Fenster, Türen oder Balkone) wird von Jahangiri und Petrou [JP08] vorgestellt. Voraussetzung für das Verfahren ist das Vorliegen perspektivischer entzerrter Fassaden. Es werden zuerst Blobs durch die Anwendung von Schwellwertverfahren, morphologischen Operatoren und einer Zusammenhangskomponentenanalyse extrahiert. Bei einem *Blob* handelt es sich um eine Gruppe verbundener Pixel, die eine gemeinsame Eigenschaft besitzen. Die so gefundenen Regionen werden auf Ähnlichkeit geprüft, um sich wiederholende Strukturen zu identifizieren. Auf diese Weise können auch weitere, anfangs nicht detektierte Regionen ermittelt oder bereits gefundene und fälschlicherweise verbundene wieder getrennt werden.

Musialski et al. [MRM+10] beschreiben ein Verfahren, welches ortho-rektifizierte, segmentierte Bilder der Fassaden in Kacheln sich wiederholender Muster segmentiert. Das Verfahren verwendet die *normalisierte Kreuzkorrelation* (engl. Normalized cross-correlation (NCC)), um mögliche *Verschiebungsdistanzen* zwischen sich wiederholenden Mustern zu finden. Eine Bildpyramide mit ausreichend grober Skalierung wird verwendet, um Genauigkeit gegen Geschwindigkeit abzuwiegen. Des Weiteren wird eine *Monte-Carlo-Strategie* basierend auf den *Canny-Strecken* des Bildes angewendet, um die Komplexität zu minimieren. Ein Histogramm der dominanten Verschiebungsdistanzen wird aus allen Paarungen mit der größten Ähnlichkeit der Muster über einem Schwellwert erzeugt. Die Histogramme der örtlichen Offsets ähnlicher Muster werden gaussgefiltert und mit *Mean-Shift* gruppiert. Ganzzahlige Vielfache der Verschiebungsdistanzen werden eliminiert. Die

enthaltenen Verschiebungsdistanzen werden erneut mit der NCC untersucht, um die korrekte Position der Trennungslinien zu bestimmen, sodass das Bild segmentiert werden kann.

Merkmalsbasierte Erkennungsverfahren Die im Folgenden vorgestellten Ansätze repräsentieren bzw. erkennen Gebäude oder Gebäudeteile durch markante Merkmale. Merkmale stellen dabei keine primitiven Bildmerkmale dar, sondern sollten möglichst einzigartig in einer begrenzten näheren Umgebung (lokale Merkmale) oder für das gesamte Bild (globale Merkmale) sein. Diese können beispielsweise spezifisch für ein Objekt ermittelt werden und dem späteren Abgleich für eine Erkennung in unklassifizierten Eingabebildern dienen.

Aus dem Bereich der 3D-Gebäude-Rekonstruktion stammt die Fensterdetektion von Lee & Nevatia [LN04]. Sie gehen davon aus, dass dazu grobe Drahtgittermodelle von Gebäuden aus Luftaufnahmen vorliegen. Diese werden, wie in [LHN00] beschrieben, halbautomatisch erstellt. Mittels dieser Modelle sowie vom Boden aufgenommener unkalibrierter Aufnahmen können Fassaden extrahiert und perspektivisch entzerrt werden. Zeilen- und spaltenweise Kantenstärken-Histogramme des Bildes ermöglichen eine Segmentierung in Fensterkandidaten, welche zusätzlich durch Anpassung an die im Bild vorhandenen Linien verfeinert werden. Bogenfenster können durch eine *Hough-Transformation* mit den der Rundung entsprechenden Parametern erkannt werden. Lee & Nevatia gruppieren die Fensterkandidaten anschließend mittels einem Fensterähnlichkeitsmaß, welches auf den Seitenverhältnissen und der Größe sowie der NCC der Kandidatenregionen basiert.

Haugeard et al. [HPFP09] entwickelten basierend auf der Kandidatensuche von Lee & Nevatia [LN04] ein alternatives konturbasiertes Ähnlichkeitsmaß. Sie setzen das Vorhandensein von perspektivisch entzerrten Fassaden voraus. Auch sie gehen von einer starken Regelmäßigkeit in der Geometrie von Fassaden aus und nutzen diese, um Fenster zu finden. Die Fenster werden durch Liniensegmente sowie deren relative Positionen und Orientierungen als Graph repräsentiert (Linien sind Knoten und örtliche Relationen sind Kanten) und können mit einer Fensterdatenbank zugeordnet werden. Eine heuristische, unvollständige Graphenzuordnung wird benutzt, um die Konturgraphen von Fenstern ohne die Betrachtung aller Pfade zu vergleichen. Das Ähnlichkeitsmaß erlaubt außerdem ein Fassaden- bzw. Fensterretrieval aus einer Datenbank.

In [MR05] stellen Mayer und Reznik ein Vorgehen vor, das aus einer Bildsequenz die Fassaden erkennt und dreidimensional modelliert. Dazu wird die Kamerakalibrierung und der *trifokale Tensor* aus zusammengehörenden Paaren von *Förstnerpunkten* [FG87] für jeweils zwei Aufnahmen berechnet. Mit Hilfe dieser Informationen können dreidimensionale Punkte aus den Bildern gewonnen und planare Flächen für Fassaden eingepasst werden. Die nun bekannten Fassadenflächen

und die bereits berechneten Projektionsmatrizen für die Aufnahmen ermöglichen ein Berechnen von *Homografien* zur Entzerrung der Fassaden.

Košecká et al. [KZ05] beschreiben ein Verfahren, welches recht ähnlich zur Fenstererkennung von Lee & Nevatia [LN04] ist. Die Fluchtpunkte und dazugehörigen Fluchtlinien werden auf kalibrierten Bildern mittels des *Expectation-Maximization-Algorithmus (EM)-Algorithmus* ermittelt und im Bild verifiziert. Mögliche rechteckige Strukturen ergeben sich aus den Schnittpunkten der Fluchtlinien, welche unterschiedlichen Fluchtpunkten zugeordnet wurden. Ein Gradientenhistogramm wird verwendet, um zu erkennen, ob alle Ränder des Rechteckes auf derselben Ebene liegen oder eine Gebäudekante/-ecke enthalten ist und deshalb verworfen werden sollte. Die Zuordnung der rechteckigen Strukturen erfolgt ebenfalls über Seitenverhältnisse und die NCC.

In [RN04] wird von Rajashekahar und Namboodiri eine Methode zur Gebäudeerkennung im Bereich *Imageretrieval* vorgestellt, die das *Doppelverhältnis* nutzt. Mittels *Harris-Eckendetektor* und anschließender Houghtransformation werden Merkmalspunkte gewonnen und für jedes kollineare Punktquadrupel das Doppelverhältnis bestimmt. Auf Basis dieser Werte wird ein Histogramm für das Bild erstellt, aus dem ein Merkmalsvektor gebildet wird. Dieser Merkmalsvektor wird zur Auffindung ähnlicher Bilder in einer Bilddatenbank verwendet.

Wenzel et al. [WDF08] stellen eine Methode vor, die wiederholend auftretende Strukturen erkennt und beschreibt. Dazu werden lokale Merkmale durch den *SIFT-Operator* gefunden und mit Hilfe seiner gespiegelten Darstellung dominante Symmetrien im Bild erkannt. Da kein spezielles Modell für Fenster genutzt wird, eignet sich die Methode auch zum Finden von künstlichen Strukturen im Allgemeinen.

Fröhlich et al. [FRD10] stellen ein Verfahren zur Klassifizierung von Fassaden basierend auf lokalen Farbmerkmalen vor. Der Algorithmus nutzt zur Erstellung der lokalen Farbmerkmale den Opponent-SIFT-Operator. Das Resultat besteht demnach aus Regionen von Pixeln, die einer Kategorie wie Fenster, Gebäude (Wand), Tür oder Himmel angehören. Klassifiziert wird mittels eines speziellen Waldes aus Entscheidungsbäumen (engl. Randomized Decision Forest). Im Gegensatz zu herkömmlichen Entscheidungsbäumen, die zu einem Wald zusammengefasst werden, wird das Lernen von Merkmalen zur Klassifikation durch bestimmte Randomisierungen effizienter und schneller. Abschließend wird die Wahrscheinlichkeit für die Zugehörigkeit eines Pixels zu einer Klasse durch eine Glättung der Wahrscheinlichkeitskarte mit einem Gaußfilter angepasst.

Fröhlich et al. [FRKD12] präsentieren eine Erweiterung ihres Verfahrens zur pixelweisen semantischen Segmentierung mittels Gaußprozesses [FRD10]. Mögliche Beschränkungen von Laufzeit und verwendetem Speicher des Gaussprozessklassifizieres werden durch eine Vorgruppierung der Daten unter Verwendung eines

Entscheidungsbaums verringert. Daher kann der Prozess in unterschiedliche Klassifizierungsaufgaben, deren Komplexität sich durch die Anzahl an Trainingssamples in den Baumblättern definiert, aufgeteilt werden. Es werden dabei verschiedene Wahrscheinlichkeiten für die möglichen Klassen eines Pixels berechnet.

Teboul et al. [TKS+11] nutzen auch Form-Grammatiken zum Beschreiben der Gebäude. Die Form-Grammatik wird dabei mittels *Verstärkenden Lernens* (*Reinforcement Learning*) analysiert. Eine Formanalyse muss zum einen gleichzeitig Geometrie und Topologie der Fassade optimieren und zum anderen die geschätzte Form auf Pixellevel anpassen. Dies wird durch einen Markov Entscheidungsprozess mittels *recursive binary split grammar* umgesetzt. Dabei erzielt das Verfahren ähnliche Ergebnisse wie der Stand der Technik, aber beschleunigt die Analyse signifikant im Vergleich zum Stand der Technik.

Mathias et al. [MMVG16] präsentieren einen Ansatz zur semantischen Segmentierung von Gebäudefassaden, die zur Rekonstruktion des Gebäudes genutzt werden können. Der Ansatz zur Fassadenanalyse besteht aus drei Phasen. In der ersten Phase wird das Fassadenbild in verschiedene Regionen segmentiert, dann werden in der zweiten Phase verschiedene Objekterkenner verwendet, um ein initiales *Labeling* der Regionen zu bekommen. Die dritte Phase verwendet "Metawissen" in Form von schwachen architektonischen Prinzipien, um die Plausibilität und Konsistenz der Label deutlich zu steigern, was an zwei Datensätzen belegt werden konnte.

Brust et al. [BSS+15] stellen *convolutional patch* Netze vor, die trainiert wurden, um zwischen verschiedenen Bildregionen zu unterscheiden und Label pixelweise zu verteilen. Das Verfahren liefert gute Ergebnisse im urbanen Bereich, insbesondere in der Straßenerkennung. Dabei wurden die räumlichen Eigenschaften der Bildregionen als zusätzliche Eingabe für das Netz verwendet, was das Trainieren/Lernen örtlicher Relationen ermöglicht.

Li und Shapiro finden in [LS02] einheitliche Liniengruppierungen in Bildern, um darin befindliche Gebäude zu erkennen. Farbe, Orientierung und räumliche Merkmale von Liniensegmenten werden genutzt, um diese zu gruppieren. Zuerst werden mittels eines Kantendetektors Liniensegmente aus dem Bild extrahiert. Für jedes segmentierte Liniensegment werden benachbarte Farben im Originalbild als Farbpaar genutzt, um Hauptfarbpaare zu finden und so nach Möglichkeit Gebäude voneinander zu unterscheiden. Nach dieser ersten Vorgruppierung werden die Liniensegmente je nach horizontaler oder vertikaler Ausrichtung einem Fluchtpunkt zugeordnet und weiter klassifiziert. Das Verfahren nutzt zur Gebäudeerkennung die Anzahl Schnittpunkte und Überlappungen innerhalb eines Clusters.

Trinh et al. stellen in [TKJ08] einen Algorithmus zur Erkennung von Gebäuden vor. Merkmale wie Fluchtpunkte, Flächen und Farbhistogramme der Wände werden dem Bild entnommen und gemeinsam mit den durch den SIFT-Operator

erhaltenen lokalen Merkmalen in einer Datenbank gespeichert. Ein neues Bild wird hinsichtlich seiner größten gefundenen Fläche (Fassade) mit der Datenbank verglichen und die beste Pose ermittelt. Durch diese Auswahl soll der Aufwand für die Erkennung verringert und somit das Verfahren beschleunigt werden. Die Datenbank wird mit Aufnahmen des Gebäudes in verschiedenen Posen, für die jeweils sämtliche Merkmale berechnet werden, trainiert. Es werden die Korrespondenzen der lokalen Merkmale des Eingabebildes und der Bilder aus der Datenbank berechnet und mittels Doppelverhältnis verifiziert. Eine *Singulärwertzerlegung* wird abschließend angewendet, um die Größe der Datenbank möglichst klein zu halten.

Ali et al. [ASJ+07] verwendeten die *OpenCV-Implementation*[5] des Objekterkennungsverfahrens von Viola & Jones [VJ01], um Fenster zu erkennen. Es werden Haar-Wavelet-ähnliche Features verwendet, welche dank der speziellen Bildrepräsentation des '*Integal Image*' leicht zu berechnen sind. Es fließen weiterhin die Fläche und der Winkel der Rotation der jeweiligen Rechtecke mit in das Merkmal ein. Das Vorgehen teilt sich in zwei Phasen, eine Trainings- und eine Erkennungsphase. In der Trainingsphase werden Merkmale durch *Boosting* in einem Entscheidungsbaum trainiert. Während der Erkennungsphase wird ein Suchfenster über das Eingabebild geschoben, für das die Merkmale berechnet werden. Die berechneten Merkmale werden mit den trainierten Klassifizierern bewertet. Als Ergebnis liefert das Verfahren *Regions of Interest* (ROIs) vermeintlicher Fenster parallel zu den Bildachsen. Diese Fenstererkennung benötigt zwar keine perspektivisch entzerrten Gebäudeaufnahmen, die Ergebnisse verschlechtern sich laut der Autoren allerdings mit dem Grad der Verzerrung.

Erkennungsverfahren mittels neuronaler Netze Neuronale Netze haben in den letzten Jahren wieder deutlich an Aufmerksamkeit gewonnen. Nicht zuletzt deshalb, weil sie zahlreiche Wettbewerbe im Bereich *pattern-recognition* gewonnen haben. Schmidthuber gibt in [Sch14] einen guten Überblick über die Erfolge der neuronalen Netze und insbesondere im Bereich *Deep Learning*. So konnte ein *Deep Neuronal Network* im *IJCNN 2011 trafficsign recognition contest in San Jose (CA)* nicht nur gewinnen, sondern war etwa doppelt so gut wie Menschen, dreimal so gut wie jedes *Artificial Network* und sechs mal so gut wie jede Methode, die ohne neuronale Netze arbeitet. Ein *Deep Neuronal Network* ist eine spezielle Form eines *künstlichen neuronalen Netzes* (ein Netz aus künstlichen Neuronen). Das *Deep Neuronal Network* besitzt eine Eingangs- und Ausgangsschicht und besitzt mindestens zwei versteckte Neuronenschichten. Da es sich bei neuronalen Netzen um Blackbox-Systeme handelt, die zudem Trainingsdaten in ausreichender Menge benötigen, wurde dieser Ansatz in dieser Arbeit nicht weiter verfolgt. Dennoch wird hier ein kurzer Überblick über das Thema gegeben.

[5] `http://docs.opencv.org/modules/objdetect/doc/cascade_classification.html`

Goodfellow et al. [GWFM+13] präsentieren eine neue Aktivierungsfunktion *Maxout*, die gut für das Training mittels *Dropout*, einer Methode um große Datenmengen und Modelle mit gemeinsamen Parametern effizient zu trainieren, geeignet ist. Die *Maxout* Methode testeten sie mit gleichem Erfolg wie *State of the art* Verfahren an fünf Benchmark-Tests.

Hinton et al. [HSK+12] präsentieren einen Ansatz, um eine Überanpassung an Trainingsdaten zu vermeiden, falls nur eine kleine Menge an Trainingsdaten vorhanden ist. Dafür wird beim Training eine zufällig ausgewählte Teilmenge der Merkmalserkenner nicht berücksichtigt. Hinton et al. erzielten mit diesem Ansatz sehr gute Ergebnisse bei Benchmark-Tests.

Krizhevsky et al. [KSH12] haben ein neuronales Netz mit 650000 Neuronen und 60 Millionen Parametern geschaffen, das 1,2 Millionen Bilder in 1000 verschiedene Klassen einteilen kann. Dabei nutzten sie eine GPU-Implementation und konnten die Ergebnisse von Hinton et al. [HSK+12] bei Benchmark-Tests noch übertreffen.

Modellbasierte Erkennungsverfahren In der modellbasierten Objekterkennung werden zurzeit statistische, ansichtsbasierte oder geometrische Modellierungen gewählt. Statistische oder ansichtsbasierte Modelle arbeiten hierbei weitgehend ohne explizites geometrisches Modellschema [BMP02, CSW01, CSS06, SC96, SLS06, KWBE02]. Im geometrischen Modell können unterschiedliche Merkmale wie *Punkte* [DD92], *Segmente* [RP98], spezifische Merkmale von *Segmenten*, *Geraden* [KH94], *Konturen* [MBCM99], *Merkmale aus dem Frequenzraum* [GXLP05] oder weitgehend *abstrakte Merkmale* [Low04] annotiert werden. Diese lokalen Merkmale können u. a. durch statistische Klassifikationstechniken (z. B. *Principal Component Analysis* (PCA) [GW01], *Nearest Neighbour Search* [TK09] oder *Randomized Trees* [LLF05]) zugeordnet werden.

Da eine Vielzahl modellbasierter Ansätze existiert, wird sich in diesem Abschnitt auf den urbanen Bereich konzentriert. Modellbasierten Ansätzen liegt ein Modell samt Semantik des Gebäudes zugrunde. Detektierte Teile des Gebäudes werden demzufolge zueinander in Beziehung gesetzt. Kernidee der modellbasierten Objekterkennung ist es, Bilddaten mit der Modellrepräsentation der Objekte abzugleichen.

Müller et al. stellen in [MZWG07] eine automatische bildbasierte Modellierung von Fassaden vor. Die Vorverarbeitung besteht aus der Rektifizierung von Fassaden und ihrer hierarchischen Unterteilung. Anschließend werden die einzelnen Fassadenteile Modellen von architektonischen Elementen zugeordnet und auf diese Weise aus einem Bild das texturierte 3-D-Modell generiert sowie die semantische Struktur in einem shape tree abgelegt. Dieser shape tree kann automatisch in eine *Computer Generated Architecture (CGA)-Grammatik* überführt werden. Bei der

CGA-Grammatik handelt es sich um eine formale Grammatik. Diese erzeugt aus einfachen geometrischen Objekten komplexe 3-D-Objekte.

Werner und Zisserman [WZ02] rekonstruieren automatisch texturierte 3-D-Modelle von Gebäuden aus mehreren Nahaufnahmen. Die Vorverarbeitung besteht aus der Erstellung grober planarer Modelle für die Hauptflächen der Gebäude. Dazu werden aus drei Bildern desselben Hauses 3-D-Punkte und Linien erstellt unter Zuhilfenahme von trifokaler Geometrie und der Berechnung von Fluchtpunkten. Aus dem Wissen über die Kameras, die Fluchtpunkte und aus den 3-D-Punkten und Linien werden zuerst planare Flächen bestimmt, die die Fassaden darstellen. Es werden dann die Regionen des Grobmodells identifiziert, die nicht gut zum Bild passen und versucht diese zu verbessern. Dazu werden rechteckige Formen für Türen und Fenster und keilförmige Formen für Dachfenster und Gauben in das Grobmodell eingepasst.

Burochin et al. [BTP09] stellen ein Konzept zur Unterteilung von Fassaden in zwei Flächenmodelle vor, welches in [BVTP10] um ein *periodisches Modell* erweitert wird. Hierbei handelt es sich um einen modellbasierten Ansatz, bei dem eine hierarchische Repräsentation der Gebäudefassade aufgebaut wird. Ein, auf Straßenhöhe aufgenommenes, Bild wird mit zwei Fluchtpunkten perspektivisch entzerrt. Die rektifizierte Fassade wird anschließend rekursiv horizontal und vertikal aufgeteilt bis die Radiometrie einer Region zu einem von drei Grundmodellen passt. Da direkt nebeneinanderstehende Gebäude zum gleichen horizontalen Fluchtpunkt zeigen und gemeinsam mit diesem entzerrt werden, werden erst einmal mittels vertikaler Kanten verschiedene Fassaden getrennt. Innerhalb der Fassaden wird abwechselnd entlang der Hauptrichtungen in *einfache zylindrische* oder *planare Modelle* aufgespalten. Konnten Wiederholungen festgestellt werden, werden diese in einem periodischen Modell erfasst. Ein planares Modell besitzt innerhalb der rechteckigen Fläche eine homogene Farbverteilung, welche höchstens durch Rauschen gestört ist. Zylindrische Modelle hingegen weisen beispielsweise bei Fensterläden ein Streifenmuster und keine Einfarbigkeit auf. Das Erstellen der Hypothesen zur Aufspaltung der Fassaden durch Gradientenakkumulation ist angelehnt an die Arbeit von Lee und Nevatia [LN04]. Die Hypothesen werden durch Berechnung einer *Spaltungsenergie* optimiert, welche ebenfalls auf Gradienten und Kanten aufbaut.

1.3.3 Modellbasierte Posebestimmung

Es existieren modellbasierte Ansätze zur Poseschätzung aus unterschiedlichen Bereichen. In diesem Abschnitt wird darauf geachtet, dass sie nur mit einer monokularen Aufnahme arbeiten.

Im Bereich der modellbasierten 3-D-Posebestimmung von Menschen unter Verwendung nur einer Aufnahme wird zurzeit viel geforscht [SSRA+12, SSQTMN13, WWL+14]. Da dieser Bereich eine sehr kleine Schnittmenge mit der Poseschät-

zung von rigiden Objekten wie Gebäuden hat, wird dieser Bereich nicht weiter beleuchtet.

Bei rigiden Objekten geht es, vereinfacht ausgedrückt, um das Auffinden von 3-D↔2-D-Korrespondenzen. Bei einer ausreichenden Anzahl und günstigen Verteilung der Korrespondenzen im Bild kann die Pose der Aufnahme bestimmt werden. Ein solches Verfahren zur Berechnung der Pose aus 3-D↔2-D-Korrespondenzen stellen DeMenthon und Davis [DD92] vor. Es werden mindestens vier nicht koplanare 3-D↔2-D-Korrespondenzen benötigt. Die initial bestimmte Pose wird iterativ verbessert.

Im Bereich der Robotik ist es zur Manipulation von Objekten wichtig deren 3-D-Position zu kennen. Dafür kommen unter anderem auch Verfahren zum Einsatz, die nur eine Aufnahme des zu manipulierenden Objektes benötigen. Collet et al. [CRBSF09] beschreiben ein Verfahren, das zunächst Objekte lernt und danach in der Lage ist die gelernten Objekte im Bild zu lokalisieren und basierend darauf die Pose zu bestimmen. In der Trainingsphase werden Aufnahmen aus verschiedenen Ansichten des Objektes gemacht, SIFT-Merkmale extrahiert und mittels gefundener Korrespondenzen die 3-D-Struktur des Objektes rekonstruiert. Zur Lokalisierung und Posebestimmung der Objekte werden auch SIFT-Merkmale verwendet. Das *Clusteringverfahren Mean Shift* ermöglicht dann mehrere Objekte in einem Bild zu detektieren. Anhand der ermittelten Korrespondenzen wird dann die Pose berechnet.

Darüber hinaus stellen Zhu et al. [ZZD15] ein Verfahren vor, das sowohl die 3-D-Struktur eines Objektes als auch beschreibende Bestandteile des Objektes lernt und mit der 3-D-Struktur verbindet. Im Gegensatz zu den zurzeit gängigen Ansätzen wird gleichzeitig das Erscheinungsbild und die Geometrie des 2-D-Objektes aus dem Bild mit einem 3-D-Modell verglichen und die Poseschätzung auf dieser Basis optimiert.

Zia et al. [ZSS14a, ZSS14b, ZSS13, ZSSS13] präsentieren einen Ansatz, der die Posen von Autos aus einem Bild bestimmt. Dafür verwenden sie ein 3-D-Drahtgittermodell, das in Teile zerlegt wird, die mit Teilbildern und den entsprechenden Posen verbunden werden. Zur Stabilisierung des Verfahrens extrahieren sie eine gemeinsame Ebene aller Objekte und arbeiten in einem 3-D-Koordinatensystem. Sie extrahieren zunächst 2-D-Boundingboxen für Autos aus denen eine grobe 3-D-Boundingbox geschätzt wird. Diese grobe 3-D-Boundingbox verfeinern sie mit Hilfe des Modells iterativ, sodass sie als Ergebnis eine akkurate Pose und eine detaillierte Segmentierung erhalten.

1.3.4 Einordnung des eigenen Verfahrens in die Literatur

Modellgenerierung Es gibt unterschiedlichste Verfahren zur Modellgenerierung, wobei jedes seine speziellen Vor- und Nachteile besitzt.

Ziel 1: Semiautomatische Modellgenerierung des Gebäudemodells - Konkretisierung

Das Ziel ist die Entwicklung und Umsetzung eines möglichst automatischen und präzisen Verfahren zur Modellgenerierung des Gebäudemodells aus dem Gebäudeschema. Dies entspricht der zuvor beschriebenen Wissensakquisitionskomponente.

*Die hierbei entstehenden Gebäudemodelle sollen als Eingabe für die anwendungsunabhängigen Kontrollstrategie (Ziel 2 (**knoPoE**)) dienen.*

Es wird folgender Weg eingeschlagen: 2-D-Bildserien werden in ein 3-D-Modell verwandelt, indem die Bilder entzerrt und die jeweiligen Posen mit Struktur aus Bewegungstechniken bestimmt werden. Das entstandene 3-D-Modell wird zum Abschluss händisch um Polygone mit vordefinierten Bezeichnern versehen. D. h. den Polygonen werden definierte Bedeutungen, wie beispielsweise "Wandfläche" oder "Fenster" zugewiesen und im Gebäudemodell gespeichert.

Die Verfahren [DTC04, DTM96, LN04] sind halbautomatisch, bildbasiert und erzeugen keine semantische Annotation, sodass sie für die wissensbasierte Poseschätzung nicht geeignet sind. Die folgenden Verfahren benötigen Luftbildaufnahmen und sind entweder halbautomatisch [LHN00, LJN02] oder automatisch [VAB10]. Es standen dieser Arbeit allerdings keine Luftbildaufnahmen zur Verfügung. Das Verfahren [STKP11] funktioniert relativ gut für Gebäude im Pariser Haussmann-Stil, ist aber nur schwer auf andere Gebäudearten anpassbar. Daher ist dieses Verfahren in dieser Form nicht für die in dieser Arbeit vorgesehene Gebäudeerkennung verwendbar. Die beiden Ansätze [LKBH13] und [RSM$^+$08] sind sehr interessant, bedürfen aber Stereoaufnahmen, die uns in dieser Arbeit nicht zur Verfügung stehen. Der Ansatz aus [LA13] benötigt ein System zur Erzeugung einer umfangreichen 3-D-Punktwolke und stellt keine Semantik zur Verfügung, sodass er nicht für diese Arbeit geeignet ist. In Kombination mit den Verfahren [NDM14] und [NDM13], die aus 3-D-Punktwolken Fenster und Dächer extrahieren, wäre dies trotzdem interessant. Allerdings liegt in dieser Arbeit der Fokus nicht auf der Verarbeitung von 3-D-Punktwolken.

Erkennungsverfahren mittels neuronaler Netze Da es sich bei neuronalen Netzen um Blackbox-Systeme handelt, die zudem Trainingsdaten in ausreichender Menge benötigen, wurde dieser Ansatz nicht weiter verfolgt. Die große Stärke neuronaler Netze, soweit bekannt, liegt in der Verarbeitung von 2-D Informationen, sodass ein direkte Anwendbarkeit auf den 3-D Bereich dieser Arbeit nicht gegeben ist.

Erkennungsverfahren Es werden an dieser Stelle die Erkennungsverfahren zur Extraktion von Fassaden, Fenstern und Türen aufgelistet, da dies die markantesten Merkmale zur Identifikation eines Gebäudes sind.

Ziel 3: Merkmalsextraktion - Fenster und Türen

Das Ziel ist die möglichst einfache Extraktion von Fenstern und Türen aus Bildern als Vorbereitung und Unterstützung des Ziels 2 "Entwicklung einer anwendungsunabhängigen Kontrollstrategie".

Das Verfahren [MR05] extrahiert und entzerrt Fassaden aus Gebäudeaufnahmen. Dieses Verfahren ist interessant und die in der Arbeit verwendete Fassadenextraktion ähnelt dem verwendeten Verfahren, wobei es bewusst zu keiner Rektifizierung der Fassade in dieser Arbeit kommt. Viele Verfahren [RM07, JP08, MRM+10, HPFP09, TKS+11, MMVG16] benötigen vorher rektifizierte Fassadenaufnahmen, welche in dieser Arbeit nicht zur Verfügung stehen. Die folgenden Verfahren benötigen alle ein Lernverfahren vorweg, wobei die gelernten "Modelle" nicht dem Modellbegriff dieser Arbeit entsprechen und keine übereinstimmende Semantik besitzen. Die Verfahren [LLS04, LLS06] segmentieren und erkennen gleichzeitig; die Verfahren [FRKD12, FRD10] nutzen Randomized Forests; [BSS+15] nutzt convolutional patch Netze und [TKJ08] nutzt eine SIFT-Datenbank. Ein Verfahren, das sehr gute Ergebnisse verspricht, ist [ASJ+07]. Das Verfahren ist relativ einfach zu implementieren, allerdings konnten die präsentierten Ergebnisse nicht reproduziert werden.[6]

Aus den zuvor genannten Gründen wurden eigene Verfahren zur Fassadenextraktion und Erkennung von Fenstern und Türen implementiert.

Wichtigster Gesichtspunkt bei der Implementation war, dass diese Verfahren möglichst unabhängig voneinander sind und jedes für sich einen Anhaltspunkt auf das Objekt und die Pose liefert. Die modellbasierte Erkennung bzw. Poseschätzung ist allerdings so implementiert (siehe Kapitel 3.6), dass sich die Verfahren jederzeit und mit wenig Aufwand austauschen lassen.

Modellbasierte Verfahren Es existieren einige Verfahren zur modellbasierten Objekterkennung bzw. Poseschätzung, wobei sich bei diesen Ansätzen die verwendeten Modellrepräsentationen stark unterscheiden können.

[6] Eine Kontaktaufnahme mit den Autoren ergab, dass nur der Erstautor weiß, wie das Verfahren funktioniert und wie die Ergebnisse erzielt wurden. Die Kontaktaufnahme mit diesem scheiterte an fehlenden Kontaktdaten, selbst die Mitautoren hatten keine Kontaktdaten. Daher konnten die Zweifel an den in der Veröffentlichung beschriebenen Ergebnissen nicht ausgeräumt werden.

Ziel 2: Entwicklung einer anwendungsunabhängigen Kontrollstrategie zur Poseschätzung - Konkretisierung
Das Ziel ist die Entwicklung und Umsetzung einer anwendungsunabhängigen Kontrollstrategie zur Poseschätzung und Belegung ihrer Güte am Beispiel der Poseschätzung von Gebäuden. Dabei soll sowohl das Analyseergebnis, als auch die Zwischenergebnisse interpretierbar und nachvollziehbar sein.

In dieser Arbeit werden die verwendeten Objektmodelle durch Graphen repräsentiert, welche durch ein anwendungsunabhängiges Kontrollverfahren interpretiert werden.

Das modellbasierte Erkennungsverfahren [MZWG07] nutzt shape-trees und CGA Grammatiken zur Modellrepräsentation, wohingegen die Ansätze [WZ02, BTP09, BVTP10] eine geometrische Modellrepräsentation verwenden. In dieser Arbeit soll ein graphbasierter Ansatz, ähnlich den semantischen Netzen, zum Einsatz kommen, um so eine natürlichsprachliche Beschreibung der Objekte als Modellrepräsentation zu gewährleisten. Die Ansätze [ZK05, WFFP08] sind im Bereich *Image-retrieval* anzusiedeln und sind zwar in der Lage, die Pose zu bestimmen, benötigen aber eine Datenbank mit Bildern und zugehörigen Posen. In dieser Arbeit sollte keine Bilddatenbank mit vorberechneten Posen zum Einsatz kommen. Die Berechnung der Posen erfolgt ausschließlich über das vorhandene Modellwissen. Viele Verfahren [CRBSF09, ZDY+14, ZZD15, ZSS14a] benötigen Textur- und Merkmalsinformationen. Auf solche Informationen wurde in dieser Arbeit verzichtet, da sie dem Ansatz der Erkennung durch natürlichsprachliche Beschreibung widersprechen.

1.4 Eigener wissenschaftlicher Beitrag

Das Ziel der Arbeit ist die modellbasierte Poseschätzung mit einem anwendungsunabhängigen Kontrollverfahren. Der Fokus dieser Arbeit liegt auf der modellbasierten Poseschätzung von Gebäuden. Dabei soll das Modell semantische Informationen beinhalten und graphbasiert sein, um somit von Menschen interpretierbar zu sein. Das Verfahren an sich soll unabhängig vom Anwendungsfall sein und auch mit Modellen anderer rigider Objekte umgehen können (vorausgesetzt, dass diese der definierten Modellrepräsentation entsprechen).

Die wissenschaftlichen Beiträge dieser Arbeit sind: (i) Die Entwicklung und Umsetzung eines Dominostein-, Pokerkarten- und Gebäudeschemas, das den Aufgaben der Objekterkennung und Poseschätzung genügt und sich dennoch an den bestehenden Normen orientiert (siehe Kapitel 3); (ii) die halbautomatische Modellgenerierung eines präzisen Gebäudemodells (siehe Kapitel 4); (iii) Verfahren zur

Erkennung von Vierecken, Ellipsen, Kartenfarben (Pik, Kreuz, Herz, Karo) und den Buchstaben "J,D,K,A" (siehe Kapitel 7) sowie zwei Verfahren zur Extraktion von Fassaden und Fenstern/Türen (siehe Kapitel 5); (iv) ein anwendungsunabhängiges Verfahren unter Verwendung eines geometrischen Modells zur Bestimmung der Pose (siehe Kapitel 6, 7 und 8).

Zusammenfassend wurden zwei Verfahren geschaffen.

Das **erste Verfahren** (*Semiautomatic generation of semantic building models from image series* (**sage-sb**)) wurde entwickelt, um das Gebäudemodell komfortabel zu generieren und die Modelle in verschiedenen Ausgabeformaten zur Verfügung stellen zu können. Bei dem **zweiten Verfahren** (*Knowledgebased Pose Estimation (**knoPoE**)*) wird eine anwendungsunabhängige Kontrollstrategie zur Extraktion der Pose bei Gebäudeaufnahmen genutzt. Vorbereitend für **knoPoE** wird die Erkennung von Dominosteinen und Pokerkarten mittels der auch in **knoPoE** genutzten anwendungsunabhängigen Kontrollstrategie präsentiert. Als Wissensbasis des zweiten Verfahrens dienen die Gebäudemodelle, die mit dem ersten Verfahren generiert wurden.

Es wurden Verfahren zur Merkmalsextraktion und der Interpretation dieser Merkmale als Basis für das zweite Verfahren entwickelt.

Das in Kapitel 4 vorgestellte System wurde in [WDP11] vorgestellt und unter der MIT Lizenz[7] veröffentlicht[8]. Die in den Kapiteln 6, 7 und 8 beschriebene Strategie und die anwendungsunabhängige Kontrolle wurde erstmals auf der DAGM [WHP10] vorgestellt und durch den auf der PRIA vorgestellten Ansatz verfeinert [WFP12, WP10, WP11].

1.5 Aufbau der Arbeit

Das folgende Kapitel spezifiziert die schon zum Teil beschriebene Fallstudie zur Posebestimmung von Gebäudeaufnahmen und zwei Fallstudien, die die Grundlage für die Gebäudefallstudie darstellen. Kapitel 3 beschreibt die grundlegende Geometrie von Kameras sowie das gewählte Modell und ordnet dieses Modell in bestehende Modellformen ein. In Kapitel 4 wird die Modellgenerierung von 3-D-Gebäudemodellen (**sage-sb**) auf Basis des zuvor beschriebenen Modells erläutert.

Die weiteren Kapitel beschreiben das Verfahren zur Schätzung der Pose bei Gebäudeaufnahmen (**knoPoE**), das die Modelle aus dem vorherigen Kapitel verwendet (siehe Abbildung 1.5 auf Seite 21). Die modellbasierte Poseschätzung von Gebäuden benötigt Verfahren zur Lokalisierung und Erkennung von Fenstern und

[7] `http://opensource.org/licenses/MIT`
[8] `http://sourceforge.net/projects/sagesb/`

Fassaden. Diese werden in Kapitel 5 beschrieben. Die verwendete Kontrollstrategie wird in Kapitel 6 beschrieben. Kapitel 7 beleuchtet den Prozess der Hypothesengenerierung. In Kapitel 8 werden Vertrauensmaße im Allgemeinen erläutert und die verwendeten Maße zur Bewertung der erzeugten Hypothesen beschrieben. Kapitel 9 beschreibt die Implementation der Modelle sowie des Komponentenkonzeptes und geht in einem Beispiel auf die Verknüpfung des Modells mit den Komponenten ein. Die erzielten Ergebnisse der Fallstudien beschreibt Kapitel 10. Ein abschließendes Fazit wird in Kapitel 11 gezogen.

Im Anhang A werden die verwendeten Datensätze zur Evaluation des Verfahrens präsentiert. Ein nicht zur Anwendung gekommenes, aber sehr interessantes Verfahren zur Schätzung der Brennweite einer Kamera wird in Anhang B beschrieben. In Anhang C wird das in dieser Arbeit verwendete Verfahren von Fiore zur Berechnung der Pose bei bekannter Zuordnung von 2-D-Punkten zu 3-D-Punkten beschrieben. Die Ungarische Methode kommt in dieser Arbeit zum Einsatz und wird daher in Anhang D näher erläutert. Zur Bewertung von Hypothesen (siehe Kapitel 8) wurden Abstandsmaße von Strecken evaluiert und weiterentwickelt. Die detaillierte Beschreibung davon befindet sich in Anhang E.

Kapitel 2

Fallstudien

Abbildung 2.1: Entwicklungsabfolge von **knoPoE** durch verschiedene Fallstudien.

Ziel 2: Entwicklung einer anwendungsunabhängigen Kontrollstrategie zur Poseschätzung - Wiederholung
Das Ziel ist die Entwicklung und Umsetzung einer anwendungsunabhängigen Kontrollstrategie zur Poseschätzung und Belegung ihrer Güte am Beispiel der Poseschätzung von Gebäuden. Dabei soll sowohl das Analyseergebnis, als auch die Zwischenergebnisse interpretierbar und nachvollziehbar sein.

In dieser Arbeit werden die verwendeten Objektmodelle durch Graphen repräsentiert, welche durch ein anwendungsunabhängiges Kontrollverfahren (**knoPoE** genannt) interpretiert werden.

Das Verfahren **knoPoE** wurde zunächst für die Erkennung von Dominosteinen (ohne perspektivische Verzerrung) entwickelt, dann für Pokerkarten (mit perspektivischer Verzerrung) ausgebaut und im Abschluss um die Anforderungen an die Poseschätzung von Gebäuden in der realen Welt erweitert (siehe Abbildung 2.1).

Im Folgenden werden die Fallstudien zur Dominostein- und Pokerkartenerkennung sowie die Fallstudie zur modellbasierten Poseschätzung von Gebäuden beschrieben. Die Fallstudien bauen aufeinander auf, um so das Ziel der anwendungsunabhängigen modellbasierten Poseschätzung von 3-D-Objekten (in dieser Arbeit

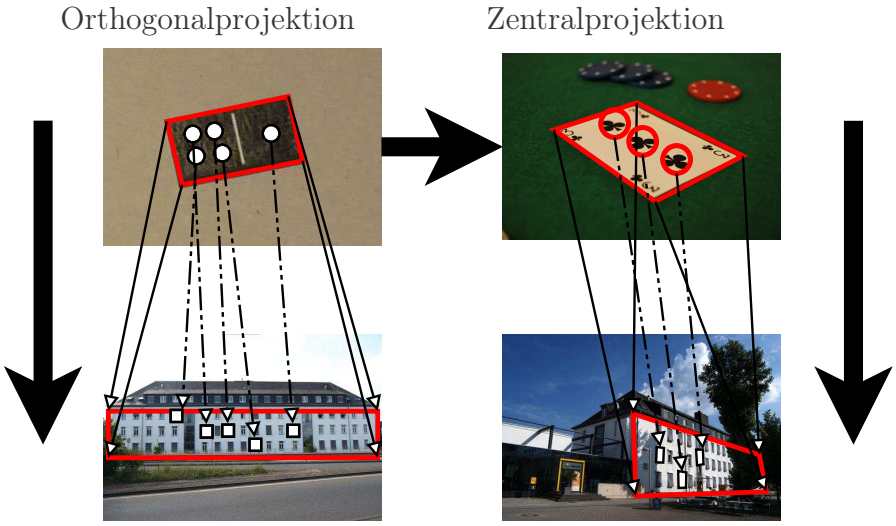


Abbildung 2.2: Zusammenhang der Fallstudien. Die Bilder entstammen den Datensätzen (siehe Anhang A) zur Evaluation der Fallstudien. Die Fallstudien Dominostein- und Pokerkartenerkennung wurden bewusst gewählt, da der Aufbau der Spielsteine und -karten mit enthaltenen Punkten und Symbolen dem einer Fassade mit Fenstern und Türen ähnelt.

von Gebäuden) iterativ zu erreichen. Daher wurde mit der Fallstudie zur Dominosteinerkennung zunächst ein Erkennungsproblem in 2-D mit kaum perspektivischer Verzerrung gewählt. Diese Fallstudie wurde dann mit der Pokerkartenerkennung auf ein Erkennungsproblem in 2-D mit perspektivischer Verzerrung erweitert. Die Erkenntnisse davon wurden genutzt, um die anwendungsunabhängige modellbasierte Poseschätzung von 3-D-Objekten am Beispiel von Gebäuden in einer Fallstudie zu belegen.

Die Fallstudien Dominostein- und Pokerkartenerkennung wurden bewusst gewählt, da der Aufbau der Spielsteine und -karten mit enthaltenen Punkten und Symbolen dem einer Fassade mit Fenstern und Türen ähnelt (siehe Abbildung 2.2). So ist die Fallstudie “Dominostein” für die modellbasierte Posebestimmung bei Gebäuden wichtig, da die Klassifikation von Dominosteinen über die Anordnung der Punkte des Dominosteins vergleichbar ist mit der Klassifikation einer Gebäudefassade, wenn zur Klassifikation die Anordnung der Fenster und Türen verwendet wird. Die Fallstudie “Pokerkartenerkennung” stellt eine Erweiterung der Domino-Fallstudie dar. In dieser Fallstudie muss zusätzlich mit perspektivischer Verzerrung umgegangen werden. Dies steigert die Komplexität, da nun raumparallele Strecken nicht abbildungsparallel dargestellt werden, sondern sich im Fluchtpunkt treffen (ein Rechteck wird in der 2-D-Projektion zum Viereck, siehe Abbildung 2.2). Die Suche nach einem Modellrechteck wird ersetzt durch eine Suche nach einem Viereck mit anschließender Homographieberechnung zur Entzerrung des Vierecks. Dies

bedeutet einen großen Schritt hin zur Posebestimmung von Gebäuden in einer monokularen Aufnahme. Die Fallstudie ist als Vorbereitung für die modellbasierte Posebestimmung bei Gebäuden wichtig, da die Erkennung von Karten über die Anordnung der Kartenfarben vergleichbar ist mit der Erkennung einer Gebäudefassade, wenn als Merkmal die Anordnung der Fenster und Türen verwendet wird (siehe Abbildung 2.2).

Die Schätzung der Pose im anwendungsunabhängigen Kontrollverfahren erfolgt über die Zuordnung von Segmenten, extrahiert aus dem *Bild*, zu Elementen des entsprechenden Modells.

Definition 2.1 (Bild)**.** *"Ein Bild I (engl. image) ist eine Abbildung von einem Ortsbereich Loc in einen Wertebereich Val:*

$$I : Loc \rightarrow Val \tag{2.1}$$

Da in dieser Arbeit nur 2-D Bilder verwendet werden, gilt für den Ortsbereich Loc:

$$I : [0 \ldots h[\times [0 \ldots w[\rightarrow Val \text{ mit } h,w \in \mathbb{N} \tag{2.2}$$

Eine mögliche Repräsentation für ein 2-D-Bild ist eine zweidimensionale Matrix I. Dabei gibt h die Anzahl der Zeilen und damit die Höhe des Bildes und w die Anzahl der Spalten und damit die Breite des Bildes an.

Der Wertebereich Val (engl. range) eines Bildes I ist eine Menge, die üblicherweise durch ein kartesisches Produkt $C_1 \times \cdots \times C_n$ gegeben ist, sodass gilt:

$$I : Loc \rightarrow (C_1 \times \cdots \times C_n) \text{ mit } n \in \mathbb{N} \tag{2.3}$$

Dabei gibt n die Anzahl der Kanäle an. Da in dieser Arbeit nur Farbbilder verwendet werden, gilt $n = 3$. Zusammenfassend gilt:

$$I : [0 \ldots h[\times [0 \ldots w[\rightarrow (C_1 \times C_2 \times C_3) \tag{2.4}$$

..." [Pri15, Seite 63ff.]

Begriffsbestimmung 2.1 (Kameramodell)**.** *Eine Ausführliche Beschreibung findet sich in Kapitel 3.1.*

In dieser Arbeit wird von einer perspektivischen Projektion mittels Lochkameramodell für die Modellrekonstruktion ausgegangen. Die in diesem Kapitel verwendeten Definitionen und mathematischen Beschreibungen sind angelehnt an die Beschreibungen von Harley und Zisserman [HZ04] und Decker [Dec12]. Es wird im Folgenden die perspektivische Abbildung von einem 3-D-Modellpunkt in einen 2-D-Bildpunkt beschrieben. Dabei werden verschiedene Koordinatensysteme benötigt (siehe Abbildung 3.1):

- *das Modellkoordinatensystem; mit beliebigem Ursprung in der 3-D-Welt. Jedes Objekt hat sein eigenes unabhängiges Koordinatensystem.*
- *das Kamerakoordinatensystem; im 3-D-Raum mit dem Kamerazentrum als Ursprung.*
- *das Bildkoordinatensystem; der Hauptpunkt H bildet den Ursprung dieses 2-D-Koordinatensystems. Der Schnittpunkt von der optischen Kameraachse mit der Bildebene definiert den Hauptpunkt.*
- *das Pixelkoordinatensystem; die linke obere Ecke der Bildebene definiert den Ursprung dieses 2-D-Koordinatensystems.*

Es werden Korrespondenzen zwischen Modell- und Bildelementen gesucht. Diese Korrespondenzen werden bewertet, so dass es möglich ist über die einzelnen Zuordnungen eine Aussage zu treffen, wie gut das Modell insgesamt zu einem durch die Zuordnungen definierten Bildbereich passt. In allen Fallstudien erfolgt die Bewertungen übereinstimmend mit der Dempster-Shafer-Theorie [Sha76] (siehe Kapitel 8).

Für die beschriebenen Fallstudien seien SEGMENT und MODELGEO die Universen aller möglichen *Segmentierungsobjekte* bzw. aller möglichen geometrischen Modellelemente.

Definition 2.2 (Segmentierungsobjekt)**.** *"Das Ergebnis einer Segmentierung ist ein so genanntes Segmentierungsobjekt mit seinen Attributen. Ein Segmentierungsobjekt ist eine initiale symbolische Beschreibung des Bildes. Typische und wichtige Segmentierungsobjekte sind Linien, Linienkreuzungen, Regionen oder Volumen. Typische Attribute dieser Segmentierungsobjekte sind ihr Ort in Bild- oder in Weltkoordinaten, ihr mittlerer Grauwert, Farbwert, Textturmerkmale, Bewegung, Entfernung, Oberflächentyp oder -normale, Form sowie eine Zuverlässigkeit für die Erkennung und für die Genauigkeit der ermittelten Attribute." [Pau01, Seite 140]*

Ein Segmentierungsobjekt wird beschrieben als Menge S $\subset$ SEGMENT, die die Segmentierungsergebnisse einer Bildanalyse enthält. Segmentierungsergebnisse eines Bildes werden in einem Segmentierungsobjekt gebündelt. Beispielsweise können 2-D-Kreise und -Rechtecke enthalten sein (siehe Ungleichung 2.8). Das Segmentierungsobjekt dieser Arbeit wird als spezielles Modell umgesetzt und speichert die extrahierten Modellinformation konform zum STOR-Referenzschema in Knoten eines Graphen und bringt sie über Kanten in Beziehung.

Definition 2.3 (Segmentierungsobjekt - Konkretisierung)**.** *Ein Segmentierungsobjekt entspricht im Verständnis dieser Arbeit dem Analysemodell. Im Zuge der Bildanalyse werden Informationen aus den Bildern extrahiert*

und in einem gesonderten Analysemodell gespeichert (siehe Kapitel 3.4.3). Das Analysemodell wurde aus dem in dieser Arbeit verwendeten STOR-Referenzschemas generiert (siehe STOR-Referenzschema in Kapitel 3.3). Es werden nicht nur die verarbeiteten Bilder, Bildregionen, anfallenden Merkmale, geometrischen Objekte, Bereiche von Interesse (ROIs) etc. gespeichert, sondern auch deren Beziehung zueinander. Das Analysemodell enthält also die aus dem Bild extrahierten und geschlussfolgerten Konzepte.

Die Relation CONTAINS beschreibt für ein Segmentierungsobjekt S, ob sich ein Segmentierungsobjekt $S_j \in S$ innerhalb (im Bild) des Segmentierungsobjektes $S_i \in S$ befindet:

$$\text{CONTAINS} \subseteq S_i \times S_j. \tag{2.5}$$

Zudem beschreibt die Relation für ein Modellelement $mg_j \in$ MODELGEO, ob es Bestandteil eines anderen Modellelementes $mg_i \in$ MODELGEO ist:

$$\text{CONTAINS} \subseteq mg_i \times mg_j. \tag{2.6}$$

Für die Analyse sind zwei Arten von Modellen besonders wichtig. Zum einen das Objekt-Modell-Schema und die daraus generierten Modelle und zum anderen das Analysemodell, das die aus dem Bild extrahierten und geschlussfolgerten Konzepte enthält.

Definition 2.4 (Modell - Konkretisierung)**.** *Ein Modell der Menge M bildet die geometrischen und semantischen Eigenschaften eines Objektes ab. Jedes Modell wird repräsentiert durch eine traversierbare Graphenstruktur. Die Knoten des Modellgraphen beschreiben die semantischen und geometrischen Bestandteile des Objektes. Die Kanten des Modellgraphen setzen die Knoten zueinander in Beziehung.*
Der Teilgraph der Semantikknoten bildet einen aufspannenden Baum über die Aggregationskanten. Einem Semantikknoten können Geometrieknoten zugeordnet werden, welche die Geometrie des Objektes im zwei- oder im dreidimensionalen Raum definieren.

Die Modelle werden aus den Objekt-Modell-Schemata des STOR-Referenzschema erzeugt und verwenden Teilmengen der Bild-, Geometrie- und Semantikpakete des STOR-Referenzschema (siehe Kapitel 3.4).

Definition 2.5 (STOR-Referenzschema)**.** *Das STOR-Referenzschema ist ein graphbasiertes Metamodell, das die (automatische) algorithmische Verarbeitung ermöglicht. Es ist in fünf Basispakete aufgeteilt, bestehend aus verschiedenen Aspekten der Bildverarbeitung (BV), der Computervision (CV) und der Computergrafik (CG). Ein Semantik-Paket, welches mit Unterpaketen für spezielle Anwendungsgebiete ergänzt werden kann, vervollständigt die Basis-*

pakete. Modelle, konform zum STOR-Referenzschema, können für alle Arten von Algorithmen, die Daten erzeugen oder die enthaltenen Daten manipulieren und transformieren, genutzt werden [FE09a]. Das STOR-Referenzschema ist unabhängig vom konkreten Anwendungsgebiet und es müssen nicht alle Pakete in einer Anwendung genutzt werden. (siehe Kapitel 3.4)

Definition 2.6 (Modell - Konkretisierung (Fortsetzung))**.** *Bei dem verwendeten Modell handelt es sich um einen gerichteten Graphen $G = (V, E)$ mit einer Menge Knoten V und einer Menge Kanten $E \subseteq V \times V$. Die Menge E ist in dieser Arbeit auf die folgenden drei Kantentypen beschränkt: VerbundenMit (linksTo), Bestandteil (consistsOf) und RepräsentiertDurch-(isRepresentedBy) und die Menge V ist in dieser Arbeit auf die folgenden fünf Klassen beschränkt: Geometry, Appearance, Feature, Image und Semantic. Abstrakte Kanten und Knoten werden in den Objekt-Modell-Schemata spezialisiert. Dabei können sowohl Kanten als auch Knoten Attribute enthalten.*

*Aus den in den Fallstudien entstandenen Objekt-Modell-Schemata werden entsprechend Modelle generiert(siehe Kapitel 3.4). Eine komfortable Methode zur Generierung von Gebäudemodellen wird mit dem Verfahren **sage-sb** in Kapitel 4 beschrieben.*

Das Objekt-Modell-Schema dieser Arbeit für die Fallstudie "Dominosteinerkennung" zeigt Abbildung 2.3. In Kapitel 3.4.1 auf Seite 81 wird dieses Objekt-Modell-Schema vorgestellt. Das Objekt-Modell-Schema für die Fallstudie "Pokerkartenerkennung" wird ebenfalls in Kapitel 3.4.1 und das Objekt-Modell-Schema für die Gebäudefallstudie wird in Kapitel 3.4.2 beschrieben.

Die Verbindung zwischen einem Objektmodell und dem Analysemodell erfolgt über die Speicherung von Zuordnungen zwischen deren Elementen in einer Hypothese. In den Beschreibungen der Fallstudien wird klar zwischen Modell- und Segmentierungselementen getrennt. Segmentierungsergebnisse von Bildern sind stets 2-D, wohingegen Modellelemente durchaus drei Dimensionen besitzen können.

2.1 Dominosteinerkennung

Ziel der Fallstudie "Dominosteinerkennung" ist die Klassifikation und Lokalisierung von Dominosteinen in Farbaufnahmen. Diese Fallstudie legt den Grundstein für das anwendungsunabhängige Kontrollverfahren, da hier die Zuordnung von Bild- zu Modellelementen an einem vereinfachten Beispiel umgesetzt und die Machbarkeit demonstriert wird. Die Fallstudie ist zudem als Vorbereitung für die modellbasierte Posebestimmung bei Gebäuden wichtig, da die Klassifikation von Dominosteinen über die Anordnung der Punkte des Dominosteins vergleichbar ist mit der Klassi-

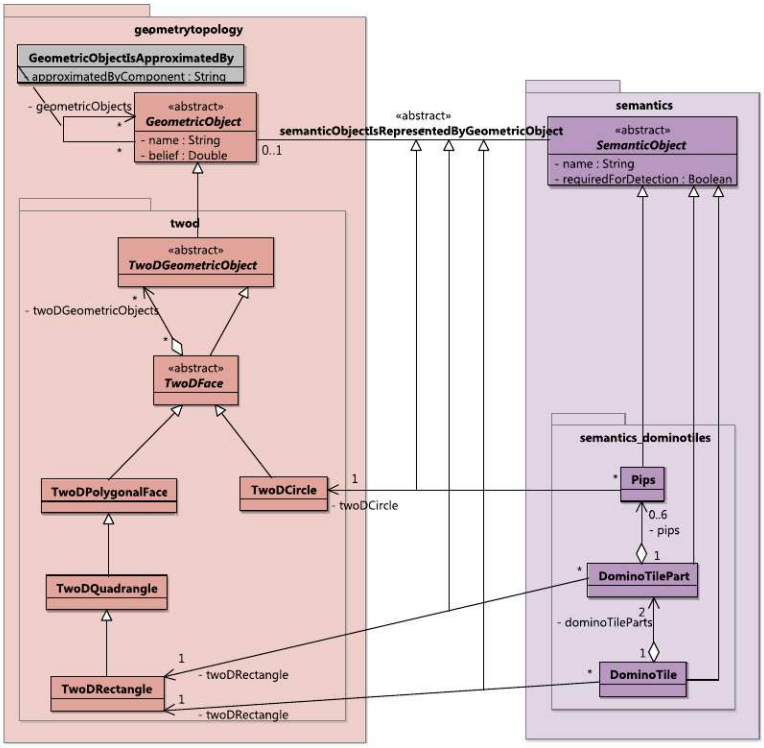


Abbildung 2.3: Elemente des Schemas für Dominosteine. (Quelle: Gemeinsame Veröffentlichung mit Frau Falkowski [WFP12])

fikation einer Gebäudefassade, wenn zur Klassifikation die Anordnung der Fenster und Türen verwendet wird (siehe Abbildung 2.4).

2.1.1 Eingabe

Für diese Fallstudie werden verschiedene Sätze von Dominosteinen benutzt, wobei jeder Satz aus 28 Steinen besteht, jede Dominosteinhälfte eine Punktzahl von 0 bis 6 aufweist und kein Dominostein zweimal vorkommt.

In der Fallstudie wird zudem davon ausgegangen, dass sich die Kamera bei den Aufnahmen orthogonal mit einer möglichen Abweichung von bis zu 10° zu den Dominosteinen befindet (siehe Abbildung 2.5) und dass es weder zu Verdeckungen noch zu Verzerrungen kommt. Die Aufnahmen wurden sowohl unter verschiedenen Kunstlichteinstellungen als auch unter Tageslicht gemacht. Auf einem Bild können mehrere Dominosteine zu sehen sein. Eine Teilmenge der verwendeten Aufnahmen wird im Anhang A in Abbildung A.1 präsentiert.

Orthogonalprojektion

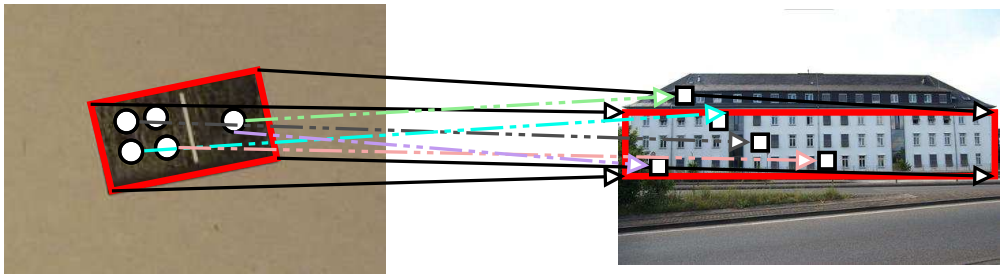


Abbildung 2.4: Erkennung von Dominosteinen als Vorbereitung für die Posebestimmung bei Gebäuden. Die Fallstudie ist als Vorbereitung für die modellbasierte Posebestimmung bei Gebäuden wichtig, da die Klassifikation von Dominosteinen über die Anordnung der Punkte des Dominosteins vergleichbar ist mit der Klassifikation einer Gebäudefassade, wenn zur Klassifikation die Anordnung der Fenster und Türen verwendet wird.

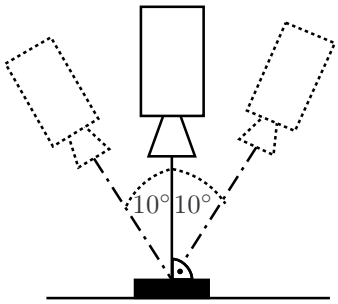


Abbildung 2.5: Kamerawinkel bei der Dominofallstudie.

Als Eingabe wird die Menge aller im Bild I gefundenen **Segmentierungsobjekte S, bestehend aus Rechtecken $\mathbf{R_S}$ und Kreisen $\mathbf{C_S}$**, vorausgesetzt (siehe Analysemodell in Abbildung 2.7(c)), für die gilt:

$$S \subset \text{SEGMENT}, S = R_S \dot{\cup} C_S \text{ mit } R_S \subset \text{RECT}_S \text{ und } C_S \subset \text{CIRCLE}_S. \quad (2.7)$$

Die aus dem Bild extrahierten Kreise C_S und Rechtecke R_S bilden jeweils eine Teilmenge der Menge aller segmentierbarer Rechtecke RECT_S und Kreise CIRCLE_S:

$$\begin{aligned} \text{RECT}_S &\subset \text{SEGMENT}, \\ \text{CIRCLE}_S &\subset \text{SEGMENT}, \\ \text{CIRCLE}_S \cap \text{RECT}_S &= \emptyset. \end{aligned} \quad (2.8)$$

Äquivalent zum Segmentierungsobjekt werden auch die Mengen aller Modellkreise $\mathrm{CIRCLE_M}$ und Modellrechtecke $\mathrm{RECT_M}$ beschrieben:

$$\begin{aligned} \mathrm{RECT_M} &\subset \mathrm{MODELGEO}, \\ \mathrm{CIRCLE_M} &\subset \mathrm{MODELGEO}, \\ \mathrm{CIRCLE_M} \cap \mathrm{RECT_M} &= \emptyset. \end{aligned} \tag{2.9}$$

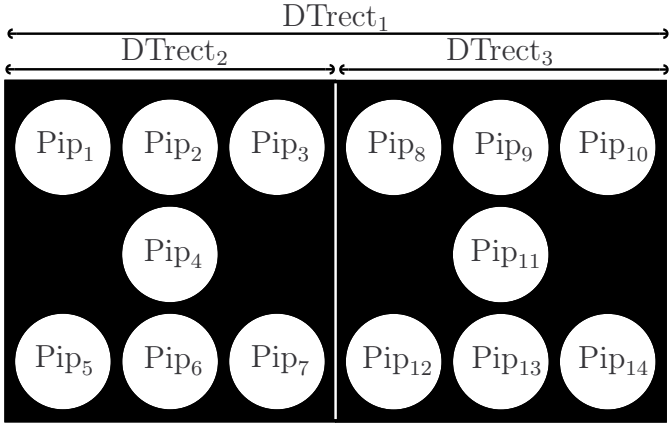


Abbildung 2.6: Verteilung der Punkte PIP auf dem Dominostein $\mathrm{DTrect_1}$.

Ein Dominostein wird beschrieben durch ein Rechteck $\mathrm{DTrect_1}$, welches zwei Hälften $\mathrm{DTrect_2}$ und $\mathrm{DTrect_3}$ besitzt, für die gilt $\mathrm{DTrect}_i \in \mathrm{RECT_M}$ und $\mathrm{DTrect_1}$ CONTAINS $(\mathrm{DTrect_2} \wedge \mathrm{DTrect_3})$. Die Menge $\mathrm{LAYO_{md}}$ möglicher Layouts definiert die Verteilung der Punkte auf den Dominosteinen mit

$$\mathrm{LAYO_{md}} \subset 2^{\mathrm{PIP}}, \quad \mathrm{PIP} \subset \mathrm{CIRCLE_M}, \quad |\mathrm{PIP}| = 14, \quad |\mathrm{LAYO_{md}}| = 28. \tag{2.10}$$

Dabei beschreibt die Menge PIP die Punkte (Kreise) auf dem Dominostein inklusive ihrer Position auf dem Stein. Es gibt auf jeder Hälfte ($\mathrm{DTrect_2}$ und $\mathrm{DTrect_3}$) des Dominosteins ($\mathrm{DTrect_1}$) 7 mögliche Positionen und insgesamt gilt somit $|\mathrm{PIP}| = 14$. Abbildung 2.6 beschreibt die Verteilung der Punkte $\mathrm{Pip}_i \in \mathrm{PIP}$ auf dem Dominostein $\mathrm{DTrect_1}$. Ein konkretes geometrisches Modell $\mathrm{md}_j \in \mathrm{MD} \subset \mathrm{M}$ wird somit beschrieben mit der Menge seiner Dominosteinrechtecke $\mathrm{DT}_j = \{\mathrm{DTrect_1}, \mathrm{DTrect_2}, \mathrm{DTrect_3}\}$ und der Verteilung der Punkte auf diesen Rechtecken als Menge der Punkte $\mathrm{LO}_{\mathrm{md},j} \in \mathrm{LAYO_{md}}$ und $\mathrm{LO}_{\mathrm{md},j} \subset \mathrm{PIP}$:

$$\mathrm{md}_j = \mathrm{DT}_j \times \mathrm{LO}_{\mathrm{md},j} \tag{2.11}$$

Zur Repräsentation von Dominosteinen wird, wie oben beschrieben, ein Objekt-Modell-Schema für Dominosteine vom Referenzschema spezialisiert und für 28

Dominosteine in 28 Modelle überführt. In Kapitel 3.4.1 wird das Objekt-Modell-Schema vorgestellt. Abbildung 3.6 zeigt das Objekt-Modell-Schema und Abbildung 3.7 präsentiert das Modell eines Dominosteins. Die 28 Modelle, repräsentiert als **Menge an Dominosteinmodellen MD $\subset$ M mit** $|\mathbf{MD}| = 28$, dienen als Eingabe für die Fallstudie.

2.1.2 Ausgabe

Als Ausgabe erhält man **die Menge MD$'$ $\subset$ MD von Dominosteinmodellen.** Die Menge MD$'$ setzt sich zusammen aus den Modellen, deren Belief in die Zuordnung von Kreisen und Rechtecken des Modells zu segmentierten Kreisen und Rechtecken größer ist als der vorher definierte Schwellwert θ_1 =0,75:

$$\mathrm{MD}' = \{\mathrm{md}_i \in \mathrm{MD} | \mathrm{Bel}_{\mathrm{MD}}(2^{\mathrm{DT}_i}, 2^{\mathrm{LO}_{\mathrm{md},i}}, 2^{\mathrm{R_S}}, 2^{\mathrm{C_S}}) \geq \theta_1\}, \text{ mit } \theta_1 \in [0,1], \quad (2.12)$$

Die Gesamtbewertung der Zuordnung vom Segmentierungsobjekt S zum Modell $\mathrm{md}_i \in$ MD wird beschrieben mit einer totalen Funktion[1]:

$$\mathrm{Bel}_{\mathrm{MD}} : \quad (2^{\mathrm{DT}_i} \times 2^{\mathrm{LO}_{\mathrm{md},i}}) \times (2^{\mathrm{R_S}} \times 2^{\mathrm{C_S}}) \quad \rightarrow [0,1] \subset \mathbb{R}. \quad (2.13)$$

Es entsteht auf diese Weise eine sortierte Menge an Dominosteinmodellen. Die Dominosteinmodelle, die die Bedingung in Definition 2.12 erfüllen, sind mit dem durch Gleichung 2.13 ausgedrückten Belief im Bild zu finden. Wenn man explizit nur nach einem Dominostein suchen würde, wäre der Dominostein im Bild zu finden der optimal zugeordnet werden kann, also der der Gleichung 2.12 erfüllt und den größten Belief bei der Gesamtbewertung erzielt (Definition 2.13).

Jedes gefundene Dominosteinmodell (muss Definition 2.12 erfüllen) wird separat mit seinen Zuordnungen und seiner Bewertung $\mathrm{Bel}_{\mathrm{MD}}$ in einer Hypothese gespeichert. Das heißt für alle gefundenen Dominosteinmodelle, dass jeder entsprechenden Hypothese neben dem Dominosteinmodell auch eine partielle Zuordnung der segmentierten Rechtecke (Gleichung 2.14) und Kreise (Gleichung 2.15) vorliegt. Zuordnungen von mehreren segmentierten Elementen des Bildes zu einem Modellelement oder die Zuordnung von mehren Modellelementen zu einem segmentierten Bildelement sind nicht erlaubt. Es kann sowohl vorkommen, dass nicht alle segmentierten Bildelemente als auch nicht alle Modellelemente zugeordnet werden können. Für die zugeordneten Kreise und Rechtecke gilt, dass die segmentierten Kreise im segmentierten Rechteck, das den gesamten Dominostein beschreibt, enthalten sein müssen und dass die segmentierten Rechtecke, die die Dominosteinhälften beschreiben, im segmentierten Rechteck, das den gesamten Dominostein beschreibt,

[1]Eine Relation ist genau dann eine totale Funktion, wenn sie linkstotal und rechtseindeutig ist. Die totale Funktion wird symbolisiert mit "$\rightarrow$"

enthalten sein müssen. Die Lokalisation des Dominosteins im Bild erfolgt über die in der Hypothese gespeicherten Zuordnungen der Modell- und Bildelemente. Die Zuordnungen für ein konkretes Modell $\mathrm{md}_i \in \mathrm{MD}$ werden als rechtseindeutige und gleichzeitig linkseindeutige Relation[2] beschrieben:

$$\gamma_\mathrm{R} = \mathrm{R_S} \rightsquigarrow \mathrm{DT}_i, \tag{2.14}$$

$$\gamma_\mathrm{C} = \mathrm{C_S} \rightsquigarrow \mathrm{LO}_{\mathrm{md},i}. \tag{2.15}$$

Konnte bei einem Dominostein $\mathrm{md} \in \mathrm{MD}$ eine Zuordnung vom Modellrechteck DTrect_1 zu einem Rechteck $r_1 \in \mathrm{R_S}$ gefunden werden, müssen die segmentierten Kreise Bestandteil dieses Rechtecks sein:

$$\forall c_j \in \mathrm{C_S} | c_j \in \mathrm{dom}(\gamma_\mathrm{C}) : \; r_1 \,\mathrm{CONTAINS}\, c_j. \tag{2.16}$$

Konnte zudem den Dominosteinhälften DTrect_2 und DTrect_3 Rechtecke zugeordnet werden, bezeichnen wir diese als $r_2, r_3 \in \mathrm{R_S}$ und es gilt entsprechend:

$$\forall r_j \in \mathrm{R_S} | r_j \in \mathrm{dom}(\gamma_\mathrm{R}) : r_1 \,\mathrm{CONTAINS}\, r_j \text{ mit } j = \{2, 3\}. \tag{2.17}$$

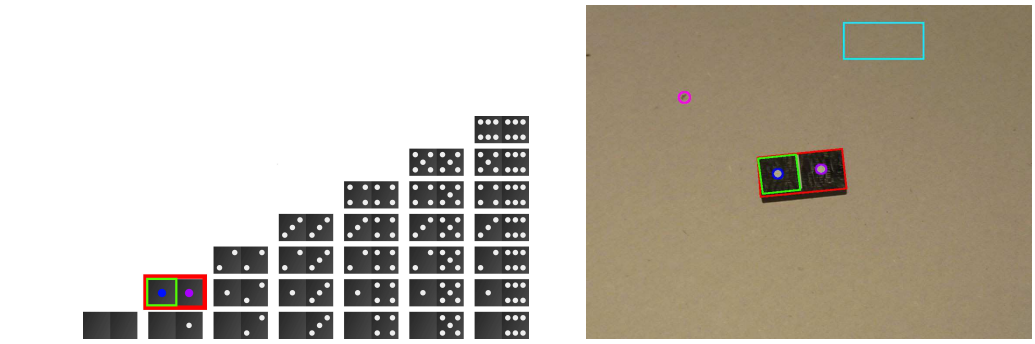
Eine Hypothese steht für einen möglicherweise im Bild enthaltenen Dominostein. Die Bewertung gibt an, wie sicher sich das Verfahren ist, dass dieser Stein wirklich im Bild zu finden ist. Die Graphenstruktur der Dominosteinmodelle verändert sich während der Analyse nicht. In der späteren Umsetzung wird das Analysemodell während der Analyse gefüllt, so dass das Analysemodell nicht schon vorher befüllt ist und sich die Graphstruktur abhängig von der Bildanalyse aufbaut.

2.1.3 Beispiel

Eingabe des Verfahrens sind 28 Dominosteinmodelle für die 28 Dominosteine eines Sets (Abbildung 2.7(a)) und die Aufnahme eines Dominosteins (Abbildung 2.7(b)).

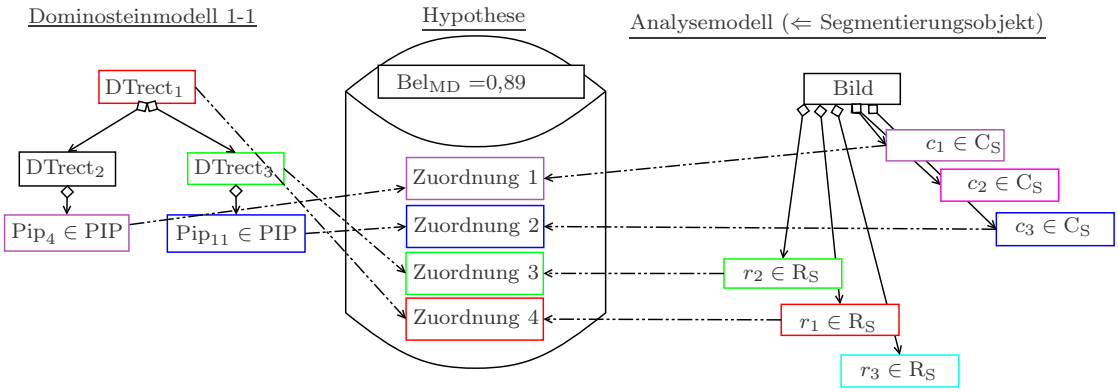
Es wird das Segmentierungsobjekte S, bestehend aus Rechtecken und Kreisen, aus dem Bild extrahiert, wobei es sowohl zu falschen Extraktionen (siehe türkises Rechteck, pinker Kreis in Abbildung 2.7(b)) als auch Elemente nicht extrahiert werden konnten (zweite Dominosteinhälfte wurde nicht erkannt). Die extrahierten Bildelemente werden im Analysemodell gespeichert. In Abbildung 2.7(c) entsprechen die Farben der Knoten und Zuordnungen den Farben der extrahierten Bildelemente in Abbildung 2.7(b). Mittels Zuordnungsfunktion 2.14 werden die segmentierten Rechtecke den Modellrechtecken partiell zugeordnet und dann mittels

[2] Das heißt für die Relation, dass jedem Element aus der Ausgangsmenge höchstens ein Element aus der Zielmenge zugeordnet wird und gleichzeitig wird jedem Element aus der Zielmenge höchstens einem Element aus der Ausgangsmenge zugeordnet. Diese Relation wird in der Arbeit symbolisiert mit "$\rightsquigarrow$".



(a) Satz Dominosteine

(b) Eingabebild mit segmentierten Bildelementen



(c) Beispiel für die Zuordnung

Abbildung 2.7: Beispiel für die Zuordnung von Segmentierungsobjekten und Modellelementen. Die aus dem Bild extrahierten Rechtecke $r_i \in \mathrm{R_S}$ und extrahierten Kreise $c_j \in \mathrm{C_S}$ werden im Analysemodell gespeichert. Durch eine konsistente Einfärbung der Bild und Modellelemente in (a), (b) und (c) wird deren Extraktion und Zuordnung visualisiert.

Zuordnungsfunktion 2.15, unter Erfüllung von Bedingung 2.16 (die segmentierten Kreise müssen sich im Bild innerhalb der segmentierten Rechtecke, welche den Modellrechtecken zugeordnet wurden, befinden), die segmentierten Kreisen den Modellkreisen partiell zugeordnet. In diesem Beispiel wurden jedem Modellkreis $\mathrm{Pip_4}$ und $\mathrm{Pip_{11}}$ ein segmentierter Kreis zugeordnet. Die Verwendung von Bedingung 2.16 führt dazu, dass der in Abbildung 2.7(b) pink markierte Kreis nicht berücksichtigt wird, da er nicht in einem Rechteck enthalten ist. Die Zuordnungen werden für alle Dominomodelle (siehe Dominosteinmodell 1-1 in Abbildung 2.7(c)) ermittelt und dann mittels Bewertungsfunktion 2.13 bewertet. Es werden nur Modelle berücksichtigt, deren Belief, dass sie im Bild zu finden sind, den definierten Schwellwert von 0,75 überschreiten (siehe Definition 2.12), sodass, wie in Abbildung 2.7(c) an-

gedeutet, der korrekte Dominostein mit einem Belief von 0,89 identifiziert werden kann.

Für jedes Dominosteinmodell entsteht eine Hypothese, die die Zuordnungen verwaltet und eine Bewertung der Zuordnung nach Gleichung 2.13 zur Verfügung stellt (Abbildung 2.7(c)). Die Zuordnungen ermöglichen die Lokalisation des Dominosteins im Bild.

2.2 Pokerkartenerkennung

Ziel der Fallstudie "Pokerkartenerkennung" ist die Klassifikation und Lokalisation von Pokerkarten. Diese Fallstudie stellt eine Erweiterung der Domino-Fallstudie dar.

Zentralprojektion

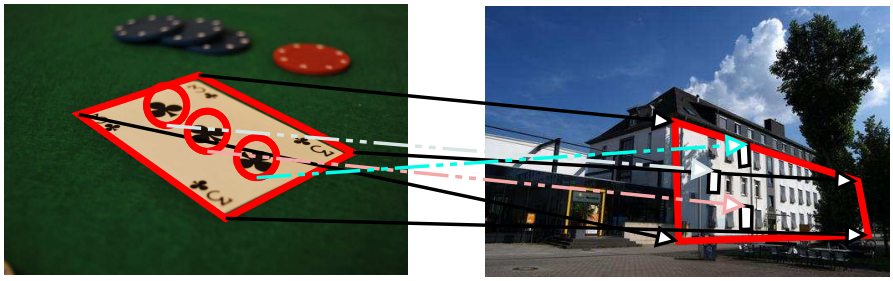


Abbildung 2.8: Erkennung von Pokerkarten.

In dieser Fallstudie muss zusätzlich mit perspektivischer Verzerrung umgegangen werden. Dies steigert die Komplexität, da nun raumparallele Strecken nicht abbildungsparallel dargestellt werden, sondern sich im Fluchtpunkt treffen (ein Rechteck wird in der 2-D Projektion zum Viereck, siehe Abbildung 2.8). Die Suche nach einem Modellrechteck wird ersetzt durch eine Suche nach einem Viereck mit anschließender Homographieberechnung zur Entzerrung des Vierecks. Dies bedeutet einen großen Schritt hin zur Posebestimmung von Gebäuden in einer monokularen Aufnahme. Die Fallstudie ist als Vorbereitung für die modellbasierte Posebestimmung bei Gebäuden wichtig, da die Erkennung von Karten über die Anordnung der Kartenfarben vergleichbar ist mit der Erkennung einer Gebäudefassade, wenn als Merkmal die Anordnung der Fenster und Türen verwendet wird (siehe Abbildung 2.8).

Begriffsbestimmung 2.2 (Perspektivische Projektion)**.** *Die perspektivische Projektion entspricht der natürlichen Wahrnehmung des Menschen. Es treffen sich die Sichtstrahlen hierbei im Projektionszentrum. Die realen Seitenverhält-*

nisse und Winkel ändern sich und die Parallelität von Geraden ist in der Regel nach der Projektion nicht mehr gegeben. (angelehnt an [HZ03])

Bei Pokerkarten muss man zwischen zwei Arten von Karten unterscheiden:
- *Zahlenkarten*: 2 . . . 10 und
- *Bildkarten*: König, Dame, Bube und Pik-Ass.

Begriffsbestimmung 2.3 (Pokerkarten)**.** *Pokerkarten*[a] *entsprechen normalerweise der Rangfolge und Bezeichnung der einzelnen Kartenwerte des französischen Blatts und bestehen damit aus 52 Karten von vier verschiedenen Farben (Kreuz, Herz, Pik, Karo) und dreizehn Werten (2 bis 10, Bube, Dame, König, Ass), tragen jedoch die Bezeichnungen des angloamerikanischen Blatts (J für Jack, anstelle von B für Bube und Q für Queen anstelle von D für Dame).*

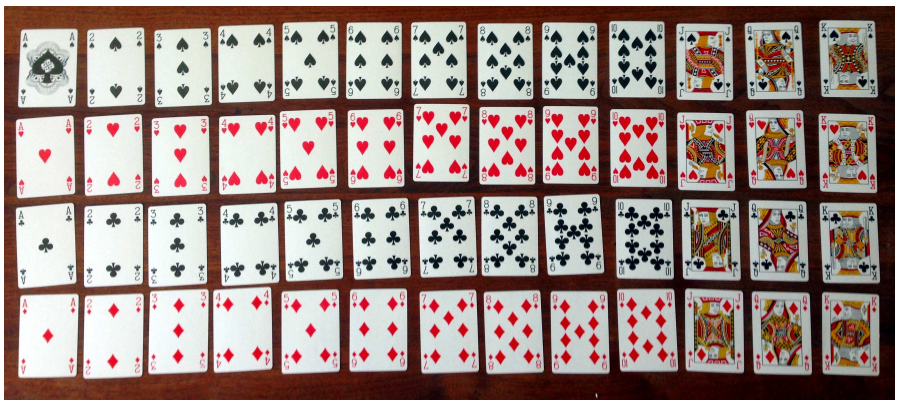


Abbildung 2.9: Beispiel eines Satzes angloamerikanischer Pokerkarten.

[a] `https://de.wikipedia.org/wiki/Pokerkarten`

Bei den Bildkarten bietet sich eine Erkennung über die Beschriftung (A, K, Q, J) an. Das könnte man ebenfalls bei den Zahlenkarten realisieren, dies würde allerdings keinerlei Erfahrungsgewinn für die modellbasierte Posebestimmung bei Gebäuden ermöglichen. Im Folgenden wird nur die Erkennung der Zahlenkarten, die aus dargelegten Gründen große Relevanz für die modellbasierte Posebestimmung bei Gebäuden besitzt, detaillierter beschrieben.

2.2.1 Eingabe

Für diese Fallstudie werden verschiedene Kartensätze (je 52 Karten) verwendet und wie auch in der Dominofallstudie sowohl unter Tageslicht als auch unter Kunstlicht Aufnahmen erstellt. Keine Karte kommt zweimal vor. Es wird in dieser Fallstudie von drei Verzerrungsstufen, beschrieben über den Winkel zwischen Kamera und

Spielkarte, ausgegangen: starke Verzerrung: ca. 30°, wenig Verzerrung: ca. 45°, keine Verzerrung: ca. 90°(siehe Abbildung 2.10). Auf einem Bild können mehrere Dominosteine zu sehen sein. Zudem wird davon ausgegangen, dass die Karten vollständig auf der Aufnahme zu sehen sind. Eine Teilmenge des verwendeten Datensatzes wird im Anhang A in Abbildung A.2 präsentiert.

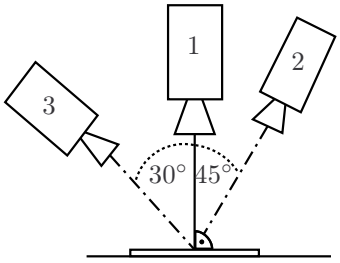


Abbildung 2.10: Kamerawinkel bei der Pokerkartenfallstudie.

Als Eingabe dient die Menge aller im Bild im gefundenen **Segmentierungsobjekte S, bestehend aus Vierecken Q_S und Kartenfarben CO_S** (siehe Analysemodell in Abbildung 2.13), für die gilt:

$$S \subset \text{SEGMENT}, S = Q_S \dot{\cup} CO_S \text{ mit } Q_S \subset \text{QUAD}_S \text{ und } CO_S \subset \text{COLOR}_S. \quad (2.18)$$

Ein Kartenrechteck wird durch die perspektivische Verzerrung in der Regel im Bild zu einem Viereck. Die Kartenfarben CO_S (Kreuz, Pik, Herz, Karo) werden beschrieben durch die Menge aller segmentierbaren Vierecke QUAD_S und Kartenfarben COLOR_S für die gilt:

$$\begin{aligned} \text{QUAD}_S &\subset \text{SEGMENT}, \\ \text{COLOR}_S &\subset \text{SEGMENT}, \\ \text{COLOR}_S &\cap \text{QUAD}_S = \emptyset. \end{aligned} \quad (2.19)$$

Äquivalent zum Segmentierungsobjekt werden auch die Mengen aller Modellrechtecke RECT_M und Modellkartenfarben COLOR_M beschrieben:

$$\begin{aligned} \text{RECT}_M &\subset \text{MODELGEO}, \\ \text{COLOR}_M &\subset \text{MODELGEO}, \\ \text{COLOR}_M &\cap \text{RECT}_M = \emptyset. \end{aligned} \quad (2.20)$$

Eine Pokerkarte wird beschrieben durch ein Rechteck $\text{CARD} \in \text{RECT}_M$, welches Kartenfarben CO_M eines angloamerikanischen Pokerkarten-Decks enthält. Die Menge der Modellkartenfarben $CO_M \subseteq \text{COLOR}_M$ beschreibt sowohl die Kartenfarben als auch die Position der Kartenfarben auf der Karte. Dabei existieren 15

Positionen für die Kartenfarben auf der Karte für 4 verschiedene Kartenfarben (Kreuz ♣, Pik ♠, Herz ♡, Karo ♢). Abbildung 2.11 beschreibt die Verteilung der Kartenfarben $\diamondsuit_i, \spadesuit_i, \clubsuit_i, \heartsuit_i \in \mathrm{CO_M}$ auf der Pokerkarte CARD. Auf einer Karte ist immer nur ein Typ Kartenfarbe zu finden. Die Menge $\mathrm{LAYO_{mp}}$ der möglichen Layouts der Kartenfarben beschreibt die Verteilung der Kartenfarben $\mathrm{CO_M}$ für die 52 Karten:

$$\mathrm{LAYO_{mp}} \subset 2^{\mathrm{CO_M}}, \quad |\mathrm{CO_M}| = 4 * 15 = 60, \quad |\mathrm{LAYO_{mp}}| = 52. \tag{2.21}$$

Die Verteilung der Kartenfarben auf der Karte (dem Modellrechteck CARD) wird als Menge der Kartenfarben $\mathrm{LO_{mp,i}} \in \mathrm{LAYO_{mp}}$ und $\mathrm{LO_{mp,i}} \subset \mathrm{CO_M}$ beschrieben. Ein konkretes geometrisches Modell $\mathrm{mp}_i \in \mathrm{MP} \subset \mathrm{M}$ wird somit beschrieben mit dem Kartenrechteck CARD_i und der Menge $\mathrm{LO_{mp,i}}$:

$$\mathrm{mp}_i = \mathrm{CARD}_i \times \mathrm{LO_{mp,i}} \tag{2.22}$$

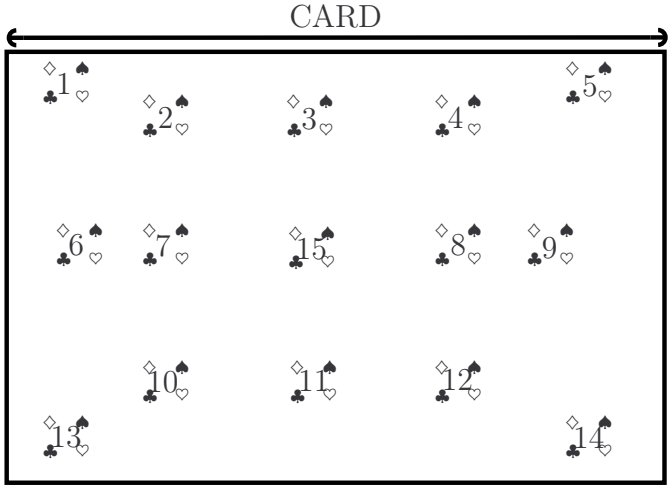


Abbildung 2.11: Verteilung der Kartenfarben$\diamondsuit_i, \spadesuit_i, \clubsuit_i, \heartsuit_i \in \mathrm{CO_M}$ auf der Pokerkarte CARD. Es existieren 15 Positionen für die Kartenfarben auf der Karte für 4 verschiedene Kartenfarben (Kreuz ♣, Pik ♠, Herz ♡, Karo ♢). Auf einer Karte ist immer nur ein Typ Kartenfarbe zu finden.

Zur Repräsentation von Pokerkarten wird, wie oben beschrieben, ein Objekt-Modell-Schema für Pokerkarten vom Referenzschema spezialisiert und für die 52 Pokerkarten in 52 Pokerkartenmodelle überführt. In Kapitel 3.4.1 wird das Objekt-Modell-Schema vorgestellt. Abbildung 3.8 zeigt das Objekt-Modell-Schema und Abbildung 3.9 präsentiert das Modell einer Pokerkarte. Die 52 Modelle, repräsentiert als **Menge an Pokerkartenmodellen MP $\subset$ M mit |MP| = 52**, dienen als Eingabe für die Fallstudie.

2.2.2 Ausgabe

Als Ausgabe erhält man **die Menge MP′ ⊂ MP von Pokerkartenmodellen.** Die Menge MP′ setzt sich zusammen aus den Modellen, deren Belief in die Zuordnung von Kartenfarben und dem Rechteck des Modells zu segmentierten Kartenfarben und Rechtecken größer ist als der vorher definierte Schwellwert θ_2 =0,75:

$$\begin{aligned} \mathrm{MP}' = &\{\mathrm{mp}_i \in \mathrm{MP} | \mathrm{Bel}_{\mathrm{MP}}(2^{\{\mathrm{CARD}_i\}}, 2^{\mathrm{LO}_{\mathrm{mp},i}}, 2^{\mathrm{Q_S}}, 2^{\mathrm{CO_S}}) \geq \theta_2\}, \\ &\text{mit } \theta_2 \in [0,1]. \end{aligned} \tag{2.23}$$

Die Gesamtbewertung der Zuordnung von Segmentierungsobjekt S zum Modell ($\mathrm{mp}_i \in \mathrm{MP}$) wird beschrieben mit einer totalen Funktion:

$$\mathrm{Bel}_{\mathrm{MP}} : \quad (2^{\{\mathrm{CARD}_i\}} \times 2^{\mathrm{LO}_{\mathrm{mp},i}}) \times (2^{\mathrm{Q_S}} \times 2^{\mathrm{CO_S}}) \quad \rightarrow [0,1] \subset \mathbb{R}. \tag{2.24}$$

Es entsteht auf diese Weise eine sortierte Menge an Pokerkartenmodellen. Die Modelle, die die Bedingung in Definition 2.23 erfüllen, sind mit dem durch Gleichung 2.24 ausgedrückten Belief im Bild zu finden. Wenn man explizit nur nach einer Pokerkarte suchen würde, wäre die Pokerkarte im Bild zu finden die optimal zugeordnet werden kann, also die Karte die Definition 2.23 erfüllt und den größten Belief bei der Gesamtbewertung erzielt (Gleichung 2.24).

Jedes gefundene Pokerkartenmodell (muss Definition 2.23 erfüllen) wird separat mit seinen Zuordnungen und seiner Bewertung $\mathrm{Bel}_{\mathrm{MP}}$ in einer Hypothese gespeichert. Das heißt für alle gefundenen Pokerkartenmodelle, dass jeder Hypothese neben dem entsprechenden Modell auch eine partielle Zuordnung der segmentierten Vierecke (Gleichung 2.25) und Kartenfarben (Gleichung 2.26) vorliegt, wobei die Kartenfarben im Viereck, das die Karte beschreibt, enthalten sein müssen. Zuordnungen von mehreren segmentierten Elementen des Bildes zu einem Modellelement oder die Zuordnung von mehren Modellelementen zu einem segmentierten Bildelement sind nicht erlaubt. Es kann vorkommen, dass nicht allen im Bild gefundenen Kartenfarben eine Kartenfarbe des Modells und umgekehrt nicht allen Kartenfarben des Modells eine Kartenfarbe im Bild zugeordnet werden kann. Die Zuordnung eines im Bild gefundenen Vierecks zum Modellrechteck CARD ist allerdings erforderlich. Die Zuordnungen für ein konkretes Modell $\mathrm{mp}_i \in \mathrm{MP}$ werden als rechtseindeutige und gleichzeitig linkseindeutige Relation beschrieben:

$$\gamma_{\mathrm{Q}} = \quad (q_j \in \mathrm{Q_S}) \quad \rightsquigarrow \mathrm{CARD}_i, \tag{2.25}$$

$$\gamma_{\mathrm{CO}} = \quad \mathrm{CO_S} \quad \rightsquigarrow \mathrm{LO}_{\mathrm{mp},i}. \tag{2.26}$$

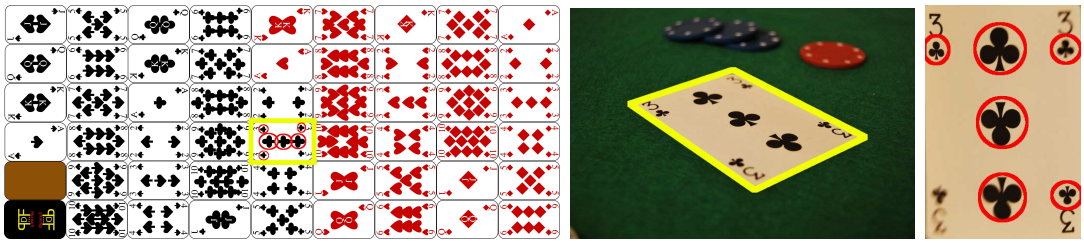
Konnte bei einer Pokerkarten $\mathrm{mp} \in \mathrm{MP}$ eine Zuordnung von einem Modellrechteck CARD zu einem segmentierten Viereck $q_j \in \mathrm{Q_S}$ gefunden werden, müssen die segmentierten Kartenfarben Bestandteil dieses Vierecks sein:

$$\forall co_k \in \mathrm{CO} | co_k \in \mathrm{dom}(\gamma_{\mathrm{CO}}) : \; q_j \,\mathrm{CONTAINS}\, co_k. \tag{2.27}$$

Jede Hypothese steht für möglicherweise im Bild enthaltene Pokerkarten. Die Bewertung der Hypothese gibt an, wie sicher das Verfahren ist, dass diese Karte wirklich im Bild zu finden ist. Die Graphenstruktur des Pokerkartenmodells verändert sich während der Analyse nicht. In der späteren Umsetzung wird das Analysemodell während der Analyse gefüllt, so dass das Analysemodell nicht schon vorher befüllt ist und sich die Graphstruktur abhängig von der Bildanalyse aufbaut.

2.2.3 Beispiel

Als Eingabe dienen die 52 Pokerkartenmodelle für die 52 Karten eines Pokerkartendecks (Abbildung 2.12(a)) und die Aufnahme einer Pokerkarte (Abbildung 2.12(b)).



(a) (b) (c)

Abbildung 2.12: Beispiel für ein Segmentierungsergebnis und die Homographieberechnung für eine potentiell gefundene Karte. (a) zeigt ein Pokerkartendeck, (b) zeigt ein Eingabebild mit Viereckssegmentierung und (c) zeigt ein entzerrtes Viereck mit Segmentierung der Kartenfarben.

Es werden das Viereck und die Kartenfarben aus dem Bild extrahiert (siehe gelbes Viereck und rot umkreiste Kartenfarben in Abbildung 2.12(b) und 2.12(c)). Die extrahierten Bildelemente werden im Analysemodell gespeichert (siehe Abbildung 2.13, in der die Farben der Knoten und Zuordnungen den Farben der extrahierten Bildelemente in Abbildung 2.12 entsprechen).

Mittels Zuordnungsfunktion 2.25 wird das segmentierte Viereck $q_1 \in \mathrm{Q_S}$ dem Modellrechteck CARD zugeordnet und dann werden mittels Zuordnungsfunktion 2.26, unter Erfüllung von Bedingung 2.27 (Kartenfarben müssen im zugeordneten Viereck enthalten sein), den segmentierten Kartenfarben $\mathrm{CO_S}$ die Modellkartenfarben $\mathrm{CO_M}$ partiell zugeordnet. Die Zuordnungen werden für alle Pokerkartenmodelle (siehe Kreuz 3 in Abbildung 2.13) ermittelt und dann mittels Bewertungsfunktion 2.24 bewertet. Es werden nur Modelle berücksichtigt, deren Belief, dass sie im Bild zu finden sind, den definierten Schwellwert von 0,75 überschreiten (siehe Definition 2.23). So kann, wie in Abbildung 2.13 angedeutet, die korrekte Pokerkarte mit einem Belief von 0,78 identifiziert werden.

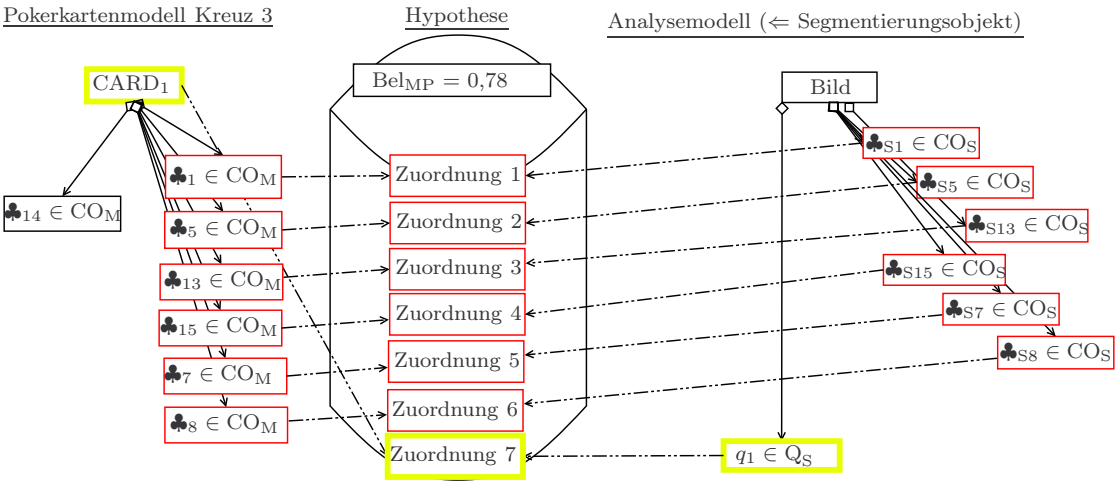


Abbildung 2.13: Beispiel für die Zuordnung von Segmentierungsobjekten und Modellelementen (basierend auf dem Modell und der Segmentierung aus Abbildung 2.12). Das aus dem Bild extrahierte Viereck $q_1 \in \mathrm{Q_S}$ und die extrahierten Kartenfarben $\clubsuit_{Si} \in \mathrm{CO_S}$ werden im Analysemodell gespeichert. Durch eine konsistente Einfärbung der Bild- und Modellelemente in dieser Abbildung und Abbildung 2.12 wird deren Extraktion und Zuordnung visualisiert.

Für jedes Pokerkartenmodell entsteht eine Hypothese, die die Zuordnungen verwaltet und eine Bewertung der Zuordnung nach Gleichung 2.24 zur Verfügung stellt (Abbildung 2.13). Die Zuordnungen ermöglichen die Lokalisation der Pokerkarte im Bild.

2.3 Poseschätzung von Gebäuden

Das Ziel dieser Fallstudie ist die modellbasierte Poseschätzung anhand einer Farbaufnahme eines Gebäudes (siehe Abbildung 2.2). Die Herausforderung bei der Bestimmung der Pose nur mittels eines geometrischen 3-D Modells und einer Aufnahme ist, dass man eine Zuordnung von 3-D Bestandteilen des Modells mit 2-D Bestandteilen des Bildes benötigt. Dies ist nicht trivial, da sich durch die perspektivische Projektion und Verdeckungen ein 3-D Objekt in Realität in beliebige 2-D Formen auf einem Bild verwandeln kann. Um die Herausforderung der Poseschätzung unter diesen Bedingungen meistern zu können wird neben dem reinen Wissen über die Geometrie, das Wissen benötigt welche Bedeutung diese Geometrie besitzt. Beispielsweise ist es schwierig die Strecken der Dachkanten im Bild zu finden. Wenn man aber beispielsweise weiß, dass ein Dach in der Regel an den Himmel grenzt, kann man auf diesem Wege geeignete Kandidaten für Dachkanten extrahieren.

Definition 2.7 (Pose - Konkretisierung)**.** *Die euklidische Transformation im $\mathbb{R}^3$ überführt einen Punkt aus Modellkoordinaten in einen Punkt in Kamerakoordinaten durch*

$$\boldsymbol{p}^c = \boldsymbol{t} + \boldsymbol{R}\boldsymbol{p}^m. \tag{2.28}$$

Der Punkt $\boldsymbol{p}^m$ wird mittels Rotationsmatrix $\boldsymbol{R}$ rotiert und per linearer Transformation $\boldsymbol{t}$ verschoben. Es gilt: $\boldsymbol{R} \in SO(3)$, wobei es sich bei SO(3) um eine spezielle orthogonale Gruppe nach Lie [LH96] handelt, die alle Drehungen um eine durch den Koordinatenursprung verlaufende Achse im dreidimensionalen Raum beschreibt. Rotation und Translation beschreiben zusammen die Pose der Kamera, wobei für die Pose υ gilt: $\upsilon \in SE(3)$. Die Pose bildet eine spezielle euklidische Gruppe, welche als SE(3) [Sel04] bezeichnet wird. $\boldsymbol{R}_{cam} = \boldsymbol{R}^T$ und $\boldsymbol{t}_{cam} = -\boldsymbol{R}^T\boldsymbol{t}^T$ beschreiben die Pose der Kamera. (vgl. Kapitel 3.1)

Die Poseschätzung mittels eines Objektmodells und einer Gebäudeaufnahme stellt ein Suchproblem dar, da es eine Vielzahl an Hypothesen gibt, aus denen es gilt die korrekte Hypothese zu identifizieren.

Begriffsbestimmung 2.4 (Posehypothese)**.** *Der Begriff Posehypothese ist als Spezialisierung zum vorher definierten Begriff der Hypothese zu verstehen.*

Zunächst wird eine Menge möglicher Posen bzgl. der Gebäudeaufnahme berechnet und im Analysemodell gespeichert. Aus den Posen, im Folgenden Posehypothesen genannt, werden neue Hypothesen erzeugt, da mittels semantischem Rendering (siehe Kapitel 3.5) Modellelemente mit anderen Attributen zur Zuordnung mit dem Analysemodell verfügbar sind.

2.3.1 Eingabe

Ein Bild ist im Folgenden als Fotografie eines Gebäudes zu verstehen. Das Gebäude ist bezogen auf seine Dimensionen komplett zu sehen, also in kompletter Höhe und kompletter Breite und/oder Länge. Dabei ist mit Verdeckungen durch Vegetation und andere Objekte zu rechnen. Die Aufnahmeposition des Bildes liegt maximal 2 m über dem Erdboden. Aufnahmen erfolgen unter verschiedensten Witterungsbedingungen, sodass es zu Schattenwurf oder auch bewölktem Himmel kommt. **Die Menge plausibler Posehypothesen Υ dient als Eingabe.** Ein geeignetes Verfahren zur Erzeugung der Menge plausibler Posehypothesen Υ gilt es zu finden. Eine Pose $\upsilon \in \Upsilon$ gilt dann als plausibel, wenn sie die beschriebenen Anforderungen erfüllt:

1. Kamera befindet sich zur Aufnahme maximal 2 Meter über dem Boden

2. Das Gebäude ist aus dieser Pose komplett, bezogen auf seine Maße ohne Berücksichtigung von Verdeckungen durch andere Objekte, zu sehen.

Das Verfahren, das in dieser Arbeit dazu verwendet wird, wird in Kapitel 7.2.1 vorgestellt. Die Menge Υ wird im Analysemodell verwaltet.

Definition 2.8 (Fläche - 3-D). *Eine Fläche $Surf_{3\text{-}D}$ ist definiert als geschlossenes flaches Polygon mit mindestens drei nicht kollinearen Punkten:*

$$Surf_{3\text{-}D} = (Po_1, Po_2, \ldots, Po_n) \quad Po_i \in \mathbb{R}^3, \quad 1 \leq i \leq n, \quad n > 3 \quad \text{und } Po_n = Po_1. \tag{2.29}$$

Die Punkte $(Po_1, Po_2, \ldots, Po_n)$ heißen Eckpunkte des Polygons und die Strecken $\overline{Po_i Po_{i+1}}$ $(i = 1, \ldots, n-1)$ sowie $\overline{Po_n Po_1}$ beschreiben die Seiten des Polygons. Das Polygon muss nicht konvex sein.

Die Definition wurde abgeleitet aus den Definitionen in [OQT05, Krä03] und stellt wenig Anforderungen an das Polygon, um bei der Modellierung bewusst wenig Grenzen zu setzen.

Bei der Segmentierung wird sich auf die Ermittlung relevanter Bildmerkmale in Form von Liniensegmenten, Geraden und Flächen konzentriert. Dazu wird die Menge aller möglichen segmentierbarer Strecken LINE_S und die Menge aller möglichen segmentierbarer Flächen SURFACE_S definiert. Die Menge der segmentierbaren Strecken und Flächen (geschlossenes Polygon, äquivalent zur 3-D-Fläche) ist zweidimensional. Zusätzlich werden Fenster und Türen als *Middle-Level-Merkmal* (siehe Kapitel 5) extrahiert.

Die Menge OP_I beschreibt die Menge an Türen und Fenstern eines Gebäudes, extrahiert aus einem Bild. **Als Eingabe dienen die Menge OP_I aller im Bild gefundenen Türen und Fenster und die Menge S aller im Bild gefundenen Segmentierungsobjekte, bestehend aus Strecken L_S und Flächen Surf_S**, (Siehe Analysemodell in Abbildung 2.14):

$$\begin{gathered}\text{S} \subset \text{SEGMENT}, \text{S} = \text{L}_\text{S} \dot{\cup} \text{Surf}_\text{S} \text{ mit} \\ \text{L}_\text{S} \subset \text{LINE}_\text{S} \text{ und } \text{Surf}_\text{S} \subset \text{SURFACE}_\text{S}.\end{gathered} \tag{2.30}$$

Als Eingabe dient ein Gebäudemodell mb $\in$ MB aus der Menge möglicher Gebäudemodelle MB $\subset$ M. Die Geometrieelemente eines Gebäudemodells mb setzt sich aus der Menge von Modellstrecken $\text{L}_{\text{M}_{3\text{-D}}}$ und der Menge der Gebäudeflächen $\text{BS}_{\text{M}_{3\text{-D}}}$ zusammen:

$$\text{L}_{\text{M}_{3\text{-D}}} \subset \text{LINE}_\text{M}, \text{ und } \text{BS}_{\text{M}_{3\text{-D}}} \subset \text{SURFACE}_\text{M}. \tag{2.31}$$

Die im Modell enthaltenen Strecken und Flächen werden beschrieben durch die Menge aller Modellstrecken $LINE_\text{M}$ und die Menge aller Modellflächen SURFACE_M.

Zudem beschreibt die Menge $\mathrm{OP_M}$ die Fenster und Türen des Modells. Jedes Fenster $\mathrm{op}_i \in \mathrm{OP_M}$ muss in genau einer Gebäudefläche $\mathrm{bs}_j \in \mathrm{BS}$ enthalten sein: $\mathrm{bs}_j\,\mathrm{CONTAINS}\,\mathrm{op}_i$. Ein konkretes geometrisches Modell $\mathrm{mb} \in \mathrm{MB}$ wird somit beschrieben mit der Menge Gebäudeflächen $\mathrm{BS_{M_{3\text{-}D}}}$ und der Menge an Fenstern/Türen $\mathrm{OP_M}$:

$$\mathrm{mb} = \mathrm{BS_{M_{3\text{-}D}}} \times \mathrm{OP_M}. \tag{2.32}$$

Die geometrische Beschreibung des Modells $\mathrm{mb} \in \mathrm{MB}$ ist abhängig von der dem Modell zugeordneten Posehypothese $\upsilon \in \Upsilon$, da, wie schon beschrieben, die Sichtbarkeit und Darstellung der Modellelemente von der gegeben Pose abhängt. Ein Modell mb_υ mit Posehypothese $\upsilon \in \Upsilon$ wird daher beschrieben als:

$$\mathrm{mb}_\upsilon = \mathrm{BS_{M,\upsilon}} \times \mathrm{OP_{M,\upsilon}}. \tag{2.33}$$

2.3.2 Ausgabe

Man erhält als Ausgabe die Posehypothese, welche Ungleichung 2.34 erfüllt und somit den größten Belief bei der Gesamtbewertung (Gleichung 2.36) erzielt. Für die gesuchte Posehypothese $\upsilon_j \in \Upsilon$ gilt, dass:

$$\begin{aligned}\forall_{\upsilon_i \in \Upsilon} \mathrm{Bel_{MB}}(2^{\mathrm{BS_{M,\upsilon_i}}}, 2^{\mathrm{OP_{M,\upsilon_i}}}, 2^{\mathrm{Surf_S}}, 2^{\mathrm{OP_I}}) < \\ \mathrm{Bel_{MB}}(2^{\mathrm{BS_{M,\upsilon_j}}}, 2^{\mathrm{OP_{M,\upsilon_j}}}, 2^{\mathrm{Surf_S}}, 2^{\mathrm{OP_I}}),\ \text{mit } i \neq j\end{aligned} \tag{2.34}$$

Die Posehypothese υ_j muss zusätzlich noch einen Belief besitzen der größer als der vorher definierte Schwellwert $\theta_3 = 0.6$ ist:

$$\mathrm{Bel_{MB}}(2^{\mathrm{BS_{M,\upsilon_j}}}, 2^{\mathrm{OP_{M,\upsilon_j}}}, 2^{\mathrm{Surf_S}}, 2^{\mathrm{OP_I}}) \geq \theta_3 \text{ mit } \theta_3 \in [0,1]. \tag{2.35}$$

Die Gesamtbewertung der Zuordnung von Bild zu Modell wird beschrieben mit einer totalen Funktion:

$$\mathrm{Bel_{MB}} : (2^{\mathrm{BS_{M,\upsilon}}} \times 2^{\mathrm{OP_{M,\upsilon}}}) \times (2^{\mathrm{Surf_S}} \times 2^{\mathrm{OP_I}}) \rightarrow [0,1] \subset \mathbb{R}. \tag{2.36}$$

Die gefundenen Posehypothesen Υ werden separat mit ihren Zuordnungen und ihrer Bewertung $\mathrm{Bel_{MP}}$ in jeweils einer Hypothese gespeichert. Das heißt für alle gefundenen Posehypothesen, dass jeder Hypothese neben dem Gebäudemodell auch eine partielle Zuordnung der segmentierten Flächen und Fenster/Türen vorliegt. Zuordnungen von mehreren segmentierten Elementen des Bildes zu einem Modellelement oder die Zuordnung von mehren Modellelementen zu einem segmentierten Bildelement sind nicht erlaubt. Es kann vorkommen, dass nicht allen im Bild gefundenen Flächen bzw. Fenster/Türen eine Gebäudefläche bzw. ein Fenster/eine Tür des Modells und umgekehrt nicht allen Gebäudeflächen bzw. Fenstern/Türen des

Modells eine Fläche bzw. ein Fenster/eine Tür im Bild zugeordnet werden kann. Für alle Posehypothesen Υ erhält man die entsprechenden partiellen Zuordnungen, sodass für ein Modell mb_υ mit Posehypothese $\upsilon \in \Upsilon$ die rechtseindeutigen und gleichzeitig linkseindeutigen Relationen für die Zuordnung von Flächen und Fenstern/Türen gelten:

$$\gamma_{\text{Surf}} = \text{Surf}_\text{S} \rightsquigarrow \text{BS}_{\text{M},\upsilon}. \tag{2.37}$$

$$\gamma_{\text{OP}} = \text{OP}_\text{I} \rightsquigarrow \text{OP}_{\text{M},\upsilon}. \tag{2.38}$$

Konnte bei einer Posehypothese $\upsilon \in \Upsilon$ Zuordnungen von Gebäudeflächen $\text{BS}' \subseteq \text{BS}_{\text{M},\upsilon}$ zu extrahierten Bildflächen $\text{Surf}'_\text{S} \subseteq \text{Surf}_\text{S}$ gefunden werden, muss bei der Zuordnung der Fenster beachtet werden, dass sich die segmentierten Fenster in den zugehörigen extrahierten Bildflächen befinden.

Die Bewertung der Hypothese gibt an, wie sicher das Verfahren ist, dass diese Posehypothese wirklich zum Gebäude im Bild passt. Die Graphenstruktur des Gebäudemodells verändert sich während der Analyse, da jeweils zu einer Pose $\upsilon_i \in \Upsilon$ die entsprechende 2-D-Geometrie dem Modell hinzugefügt wird. Eine Poseschätzung ermöglicht die Projektion des Modells in das *Pixelkoordinatensystem* der zur Gebäudeaufnahme verwendeten Kamera (siehe semantisches Rendering in Kapitel 3.5). In der späteren Umsetzung wird das Analysemodell während der Analyse gefüllt, so dass das Analysemodell nicht schon vorher befüllt ist und sich die Graphstruktur abhängig von der Bildanalyse aufbaut.

2.3.3 Beispiel

Als Eingabe dient ein Gebäudemodell für das entsprechende Gebäude (Abbildung 2.14) und die Aufnahme des Gebäudes (Abbildung 2.15). Es werden die Kanten/Flächen und Fenster/Türen aus dem Bild extrahiert (Abbildung 2.15(a) und 2.15(b)). Alle extrahierten Bildelemente werden im Analysemodell gespeichert. In Abbildung 2.14(b) werden die Fenster im Analysemodell konform zu den extrahierten Fenstern rot umrandet visualisiert. Die Extraktion der Flächen ist nur schwer zu visualisieren, daher wird zur Einfachheit darauf verzichtet. Im Analysemodell wird zudem die Menge an plausiblen Posen Υ gespeichert (repräsentiert als Pose $\upsilon_1, \upsilon_2, \ldots, \upsilon_n$ in Abbildung 2.14(b)).

Auf Basis der Zuordnung der Flächen (Gleichung 2.37) und der Zuordnung der Fenster und Türen (Gleichung 2.38) kann für alle Posehypothesen Υ des Gebäudemodells $\text{mb} \in \text{MB}$ (siehe Abbildung 2.14(a)) eine Bewertung mittels Bewertungsfunktion 2.24 erfolgen, sodass, wie in Abbildung 2.15(c) und Abbildung 2.14(b) angedeutet, die korrekte Pose $\upsilon \in \Upsilon$ mit einem Belief von $\text{Bel}_\text{MB} = 0{,}93$ identifiziert werden kann. In Abbildung 2.14) ist der Belief in die Pose $\upsilon \in \Upsilon$ größer als der

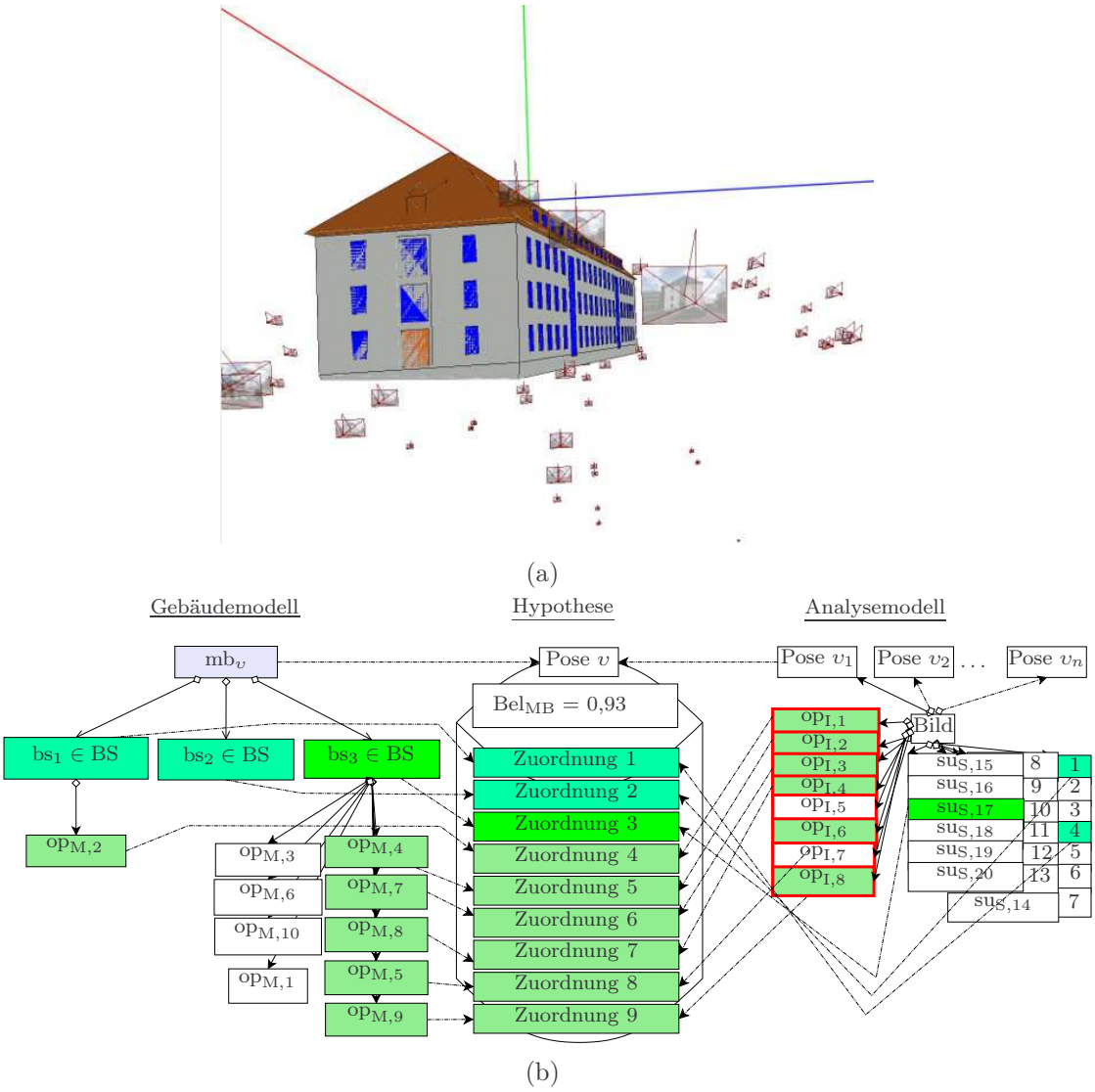


Abbildung 2.14: (a) Beispiel für die Berechnung von Posehypothesen. Die eingezeichneten Kameras stellen die Posehypothesen dar. (b) Beispiel für die Zuordnung von Segmentierungsobjekten und Modellelementen. Es gilt $\mathrm{op}_{\mathrm{M},i} \in \mathrm{OP}_\mathrm{M}$, $\mathrm{op}_{\mathrm{I},j} \in \mathrm{OP}_\mathrm{I}$ und $\mathrm{su}_{\mathrm{S},k} \in \mathrm{Surf}_\mathrm{S}$. Im Analysemodell befinden sich die Flächen und Fenster/Türen extrahiert aus dem Bild (Abbildung 2.15 visualisiert eine mögliche Extraktion). Die Elemente des Gebäudemodells mb_υ werden für die Pose $\upsilon \in \Upsilon$ dem Analysemodell zugeordnet. Die Zuordnungen werden in der Hypothese vermerkt.

definierte Schwellwert von 0,6 und erfüllt somit Ungleichung 2.35. Die Pose $\upsilon \in \Upsilon$ entspricht der im Analysemodell gespeicherten Pose $\upsilon_1 \in \Upsilon$. Für jede Posehypothese entsteht eine Hypothese, die die Zuordnungen verwaltet und eine Bewertung

(a) Kantenbild (b) Fenster/Türen (c) Gerenderte Pose

Abbildung 2.15: Beispiel für die Identifikation einer plausiblen Pose mit Hilfe von aus dem Bild extrahierter Strecken (Kanten), Fenster und Türen. Basierend auf den Zuordnungen von Strecken und Fenster/Türen zwischen Bild und Modell (siehe Abbildung 2.14) wird bewertet, wie gut eine Posehypothese zu im Bild segmentierten Elementen passt. In (c) wird das Modell mb_v aus der Pose v in das Ausgangsbild gerendert. Man kann sehen, dass die Pose gut zum Modell und zum Ausgangsbild passt.

der Zuordnung nach Gleichung 2.36 zur Verfügung stellt (Abbildung 2.14). In Abbildung 2.15(b) sieht man das sowohl nur eine Teilmenge aller Fenster erkannt wurden, als auch Fenster erkannt wurden, die keine sind. Dies führt dazu, dass nicht allen Modellfenstern ein Bildfenster zugeordnet werden kann und umgekehrt (siehe Abbildung 2.14(b)). Es wird nicht zwischen Fenstern und Türen im Bild unterschieden. Deshalb werden diese im Analysemodell als $\mathrm{op}_{\mathrm{I},j}$ gespeichert.

Kapitel 3

Modelle

Modelle bilden das zentrale Element dieser Arbeit und werden daher näher betrachtet. Diese bilden die Basis für die modellbasierte Poseschätzung (siehe Abbildung 1.2 auf Seite 14 und Abbildung 1.6 auf Seite 22.

Die in diesem Kapitel verwendeten Modelle und deren Beschreibung sind in enger Zusammenarbeit in einem gemeinsamen Projekt[1] mit der der Arbeitsgruppe Ebert[2] des Instituts für Softwaretechnik und Frau Kerstin Falkowski entstanden.

Definition 3.1 (Modellmerkmale)**.** *Ein Modell ist nach Herbert Stachowiak durch mindestens drei Merkmale gekennzeichnet [Sta73, Seite 221 ff.]:*

(i) dem Abbildungsmerkmal: "Modelle sind stets Modelle von etwas, nämlich Abbildungen, Repräsentationen natürlicher oder künstlicher Originale, die selbst wieder Modelle sein können.",

(ii) dem Verkürzungsmerkmal: "Modelle erfassen im Allgemeinen nicht alle Attribute des durch sie repräsentierten Originals, sondern nur solche, die den jeweiligen Modellerschaffern und/oder Modellbenutzern relevant scheinen." und

(iii) dem pragmatischen Merkmal: "Modelle sind ihren Originalen nicht per se eindeutig zugeordnet. Sie erfüllen ihre Ersetzungsfunktion:

(a) für bestimmte - erkennende und/oder handelnde, modellbenutzende - Subjekte,

(b) innerhalb bestimmter Zeitintervalle und

(c) unter Einschränkung auf bestimmte gedankliche oder tatsächliche Operationen."

In dieser Arbeit kommen verschiedene Modelle, wie ein *Kameramodell* (siehe Kapitel 3.1), ein *Analysemodell* und *Objektmodelle* (siehe Abschnitt 3.3 und 3.4),

[1] `https://www.uni-koblenz-landau.de/de/koblenz/fb4/icv/agpaulus/agas-forschung/agas-projects/stor`

[2] `https://www.uni-koblenz-landau.de/de/koblenz/fb4/ist/rgebert`

zum Einsatz. Das Kameramodell formalisiert die Kameraeigenschaften, indem es die Abbildungsparameter für die Kamera selbst und die Beziehung zu anderen Koordinatensystem beschreibt. Die Objektmodelle und das Analysemodell hingegen finden explizit Einzug in den Prozess der modellbasierten Poseschätzung. Abschnitt 3.2 beschreibt unter welchen Einflüssen die Objektmodelle entstanden sind. Ein Verfahren, die Modelle um die aus Pose v gerenderte 2-D-Geometrie zu erweitern, beschreibt Abschnitt 3.5. Die Objektmodelle besitzen neben deklarativem Wissen auch prozedurales Wissen, das in Abschnitt 3.6 beschrieben wird.

3.1 Kameramodell

Das Kameramodell formalisiert die Kameraeigenschaften, indem es die Abbildungsparameter für die Kamera selbst und die Beziehung zu anderen Koordinatensystem beschreibt, und bildet somit die grundlegenden Annahmen bzgl. der Kamerageometrie und Koordinatensysteme für alle Verfahren dieser Arbeit ab.

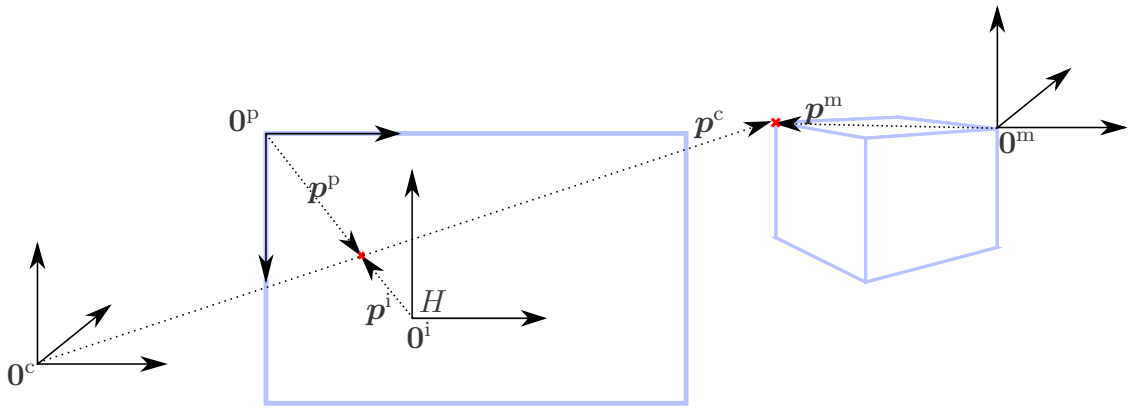


Abbildung 3.1: Es wird der Zusammenhang von Pixel-, Bild-, Kamera- und Modellkoordinatensystem dargestellt. Dabei wird von einer perspektivischen Projektion mittels Lochkameramodell ausgegangen. Die jeweiligen Kamerazentren sind beschrieben durch $\mathbf{0}^{\mathrm{p}}$, $\mathbf{0}^{\mathrm{i}}$, $\mathbf{0}^{\mathrm{c}}$ und $\mathbf{0}^{\mathrm{m}}$. Die entsprechenden Ortsvektoren bzw. Punkte sind gekennzeichnet durch $\boldsymbol{p}^{\mathrm{p}}$, $\boldsymbol{p}^{\mathrm{i}}$, $\boldsymbol{p}^{\mathrm{c}}$ und $\boldsymbol{p}^{\mathrm{m}}$.

In dieser Arbeit wird von einer *perspektivischen Projektion* mittels *Lochkameramodell* für die Modellrekonstruktion ausgegangen. Die in diesem Kapitel verwendeten Definitionen und mathematischen Beschreibungen sind angelehnt an die Beschreibungen von Harley und Zisserman [HZ04] und Decker [Dec12]. Es wird im Folgenden die perspektivische Abbildung von einem 3-D-Modellpunkt in einen 2-D-Bildpunkt beschrieben. Dabei werden verschiedene Koordinatensysteme benötigt (siehe Abbildung 3.1):

- das *Modellkoordinatensystem*; mit beliebigem Ursprung in der 3-D-Welt. Jedes Objekt hat sein eigenes unabhängiges Koordinatensystem.

- das *Kamerakoordinatensystem*; im 3-D-Raum mit dem Kamerazentrum als Ursprung.

- das *Bildkoordinatensystem*; der Hauptpunkt H bildet den Ursprung dieses 2-D-Koordinatensystems. Der Schnittpunkt von der optischen Kameraachse mit der Bildebene definiert den Hauptpunkt.

- das *Pixelkoordinatensystem*; die linke obere Ecke der Bildebene definiert den Ursprung dieses 2-D-Koordinatensystems.

Die euklidische Transformation im $\mathbb{R}^3$ überführt einen Punkt aus *Modellkoordinaten* in einen Punkt in *Kamerakoordinaten* durch

$$\boldsymbol{p}^{\mathrm{c}} = \boldsymbol{t} + \boldsymbol{R}\boldsymbol{p}^{\mathrm{m}}. \tag{3.1}$$

Der Punkt $\boldsymbol{p}^{\mathrm{m}}$ wird mittels *Rotationsmatrix* $\boldsymbol{R}$ rotiert und per linearer Transformation $\boldsymbol{t}$ verschoben. Es gilt: $\boldsymbol{R} \in \mathrm{SO}(3)$, wobei es sich bei SO(3) um eine spezielle orthogonale Gruppe nach Lie [LH96] handelt, die alle Drehungen um eine durch den Koordinatenursprung verlaufende Achse im Dreidimensionalen beschreibt. Rotation und Translation beschreiben zusammen die Pose der Kamera, wobei für die Pose υ gilt: $\upsilon \in \mathrm{SE}(3)$. Die Pose bildet eine spezielle euklidische Gruppe, welche als SE(3) bezeichnet wird [Sel04]. $\boldsymbol{R}_{cam} = \boldsymbol{R}^T$ und $\boldsymbol{t}_{cam} = -\boldsymbol{R}^T\boldsymbol{t}^T$ beschreiben die Pose der Kamera. Mit Hilfe von homogenen Koordinaten ist es möglich, diese Transformation als Matrix-Operation zu beschreiben. *Homogene Koordinaten* werden gekennzeichnet mit "$\sim$".

$$\tilde{\boldsymbol{p}}^{\mathrm{c}} = \begin{pmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0} & 1 \end{pmatrix} \tilde{\boldsymbol{p}}^{\mathrm{m}}. \tag{3.2}$$

Da von einer perspektivischen Projektion ausgegangen wird, kann ein Punkt $\tilde{\boldsymbol{p}}^{\mathrm{m}}$ in Modellkoordinaten direkt in einen Punkt $\tilde{\boldsymbol{p}}^{\mathrm{i}}$ in *Bildkoordinaten* überführt werden.

$$\tilde{\boldsymbol{p}}^{\mathrm{i}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0} & 1 \end{pmatrix} \tilde{\boldsymbol{p}}^{\mathrm{m}}. \tag{3.3}$$

Gleichung 3.3 ist nur bis auf einen skalaren Faktor gültig. Die Projektionsmatrix $\boldsymbol{P}$ ist daher definiert als:

$$\boldsymbol{P} = (\boldsymbol{R}\boldsymbol{t}) \tag{3.4}$$

$$\tilde{\boldsymbol{p}}^{\mathrm{i}} = \boldsymbol{P}\tilde{\boldsymbol{p}}^{\mathrm{m}}. \tag{3.5}$$

Unter Verwendung der *Kameramatrix* $\boldsymbol{K}$ kann ein Punkt von Bildkoordinaten in *Pixelkoordinaten* überführt werden.

$$\tilde{\boldsymbol{p}}^{\mathrm{p}} = \boldsymbol{K}\boldsymbol{p}^{\mathrm{i}}. \tag{3.6}$$

Die Kameramatrix enthält einen Teil der intrinsischen Kameraparameter. Bei den *intrinsischen Kameraparametern* unterscheidet man die Parameter, die die Kamera beschreiben und in der Kameramatrix $\boldsymbol{K}$ zusammengefasst sind, und die *Verzerrungsparameter*. Die Kameramatrix

$$\boldsymbol{K} = \begin{pmatrix} f_x & s & H_x \\ 0 & f_y & H_y \\ 0 & 0 & 1 \end{pmatrix} \tag{3.7}$$

beinhaltet den Abstand vom Zentrum der Linse zur Bildebene (*Brennweite* f) kombiniert mit der *Höhe* d_y und *Breite* d_x *der Pixel* ($f_x = \frac{f}{\mathrm{d}_x}$ und $f_y = \frac{f}{\mathrm{d}_y}$), den *Hauptpunkt* $H = (H_x, H_y)$ und den *Skew(Schräge)-Winkel* s für den Fall, dass die Achsen des Bildkoordinatensystems nicht orthogonal sind.

In der Regel handelt es sich bei der Linsenverzerrung um eine *radiale Verzerrung*, Bildpunkte werden vom Fokus weg oder in Richtung Fokus radial verzerrt. Die radiale Verzerrung eines idealen Bildpunktes $\boldsymbol{p}^{\mathrm{i}} = (\boldsymbol{x}, \boldsymbol{y})^{\mathrm{i}}$ in einen verzerrten Bildpunkt $\breve{\boldsymbol{p}}^{\mathrm{i}} = (\breve{\boldsymbol{x}}, \breve{\boldsymbol{y}})^{\mathrm{i}}$ wird beschrieben durch:

$$\breve{x} = x\frac{1 + \kappa_1(x^2+y^2) + \kappa_2(x^2+y^2)^2 + \kappa_3(x^2+y^2)^3}{1 + \kappa_4(x^2+y^2) + \kappa_5(x^2+y^2)^2 + \kappa_6(x^2+y^2)^3} \tag{3.8}$$

$$\breve{y} = y\frac{1 + \kappa_1(x^2+y^2) + \kappa_2(x^2+y^2)^2 + \kappa_3(x^2+y^2)^3}{1 + \kappa_4(x^2+y^2) + \kappa_5(x^2+y^2)^2 + \kappa_6(x^2+y^2)^3}. \tag{3.9}$$

Die *tangentiale Verzerrung* spielt im Vergleich zur radialen Verzerrung eine kleinere Rolle. Diese hilft die Genauigkeit der Entzerrung noch etwas zu erhöhen. Hierbei werden Bildpunkte anhand eines tangentialen Vektors verzerrt:

$$\breve{x} = 2\rho_1 xy + \rho_2(x^2 + y^2 + 2x^2) \tag{3.10}$$

$$\breve{y} = \rho_1(x^2 + y^2 + 2y^2) + 2\rho_2 xy. \tag{3.11}$$

Ziel der *Kamerakalibrierung* ist es, die Abbildungseigenschaften der Kamera zu bestimmen. Dabei werden zum einen die intrinsischen Kameraparameter, also die innere Orientierung, und zum anderen die *extrinsischen Kameraparameter*,

also die äußere Orientierung aus den bekannten Bild- und 3-D-Koordinaten der Objektpunkte berechnet. In dieser Arbeit werden nur die intrinsischen Kameraparameter benötigt. Von Zhang [Zha00] und Bouguet [Bou08] werden mögliche Algorithmen zur Bestimmung der Parameter beschrieben. Diese nutzen zur Berechnung der Parameter Punktkorrespondenzen zwischen 3-D-Objektpunkten und deren korrespondierenden 2-D-Projektionen. Dazu wird zumeist ein Objekt mit bekannter Geometrie und leicht zu detektierenden Merkmalen verwendet, wie beispielsweise ein Schachbrett (siehe Abbildung 3.2).

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

Abbildung 3.2: Bildfolge eines Schachbrettes zur Bestimmung der intrinsischen Kameraparameter einer Kamera.

3.2 Modellkonzeption

Der folgende Abschnitt beschreibt unter welchen Einflüssen die verwendeten Modelle entstanden sind. Die Objektmodelle und das Analysemodell finden explizit Einzug in den Prozess der modellbasierten Poseschätzung.

Die Repräsentation von Wissen richtet sich nach der Anwendungsdomäne und der Art der Informationen. So kommen in der Bildverarbeitung zumeist *semantische Netze bzw. Ontologien* zur Interpretation und Analyse von Wissen zum Einsatz (siehe Abschnitt 3.2.1).

Ein Modell enthält Wissen über die zu beschreibenden Objekte. Allerdings ist in der Bildauswertung der Verwendungszweck entscheidend für die Modellierung, da nur Bestandteile modelliert werden müssen, die auch benötigt werden. Dies wären beispielsweise bei der Erkennung von Gebäuden auf Fotos die Beschreibungen der Fassade, des Dachs und der Fenster. Nicht hilfreich wären zum Beispiel Informationen über den Grundriss oder die Statik des Gebäudes.

Für die Bildauswertung werden vor allem die Eigenschaften und Merkmale modelliert, die helfen, die gesuchten Objekte im Bild wiederzufinden. Dazu muss das Wissen in ein explizites Modell verwandelt werden, sodass das Wissen beschreibend vorliegt. Beispielsweise kann dann ein Haus durch ein Dach, mehrere Fassaden, Fenster und Türen beschrieben werden.

Implizites Wissen, das nicht direkt vorliegt, sondern sich beispielsweise in einer Menge Beispiele versteckt, wird nicht als Modellwissen, wie es in dieser Arbeit

Verwendung finden soll, betrachtet. Kann man beispielsweise auf fünf Gebäude zeigen, weiß man prinzipiell, was ein Gebäude ist. Dies beweist aber nicht, dass man explizit die Regeln benennen kann, nach denen man das Gebäude als solches erkannt hat.

Unter Modellrepräsentation wird die Darstellung eines Modells in einer Sprache verstanden. Diese Sprache kann natürliche Sprache, eine graphische Darstellung, ein mathematischer Formalismus oder die formale Sprache eines Rechners sein.

Alle Sprachen haben gemeinsam, dass sie eine eindeutige Semantik und Syntax besitzen. Die *Syntax* bestimmt die formalen Regeln über die zulässigen Elemente und über zulässige Möglichkeiten ihrer Verwendung. Die *Semantik* beschreibt dann die Bedeutung der Elemente der Sprache.

Begriffsbestimmung 3.1 (Syntax vs. Semantik)**.** *"Unter Syntax versteht man im Allgemeinen eine Menge von Regeln zur Strukturierung von Zeichen und Zeichenketten. In der Informatik, . . . , steht Syntax für eine Menge von Regeln, um Programme oder Dokumente mit bestimmten Eigenschaften, z. B. gültige XML-Dokumente, zu erzeugen.*
Semantik steht allgemein für "Bedeutung von Wörtern, Phrasen oder Symbolen". In der Informatik, und insbesondere im Bereich Semantic Web, versteht man unter Semantik die Bedeutung von Wörtern bzw. Zeichen(-ketten) und ihre Beziehung untereinander.

Allgemein steht . . . Syntax für die normative Struktur von Daten, welche erst durch Semantik eine Bedeutung erhält." [HKRS08, Seite 13]

In der Bildanalyse wird zwischen explizit und implizit repräsentiertem Modellwissen unterschieden. Es gibt Verfahren, die auf *explizites* Wissen zur Bildanalyse setzen, das sich beispielsweise in einer Ontologie widerspiegelt, und es gibt Verfahren, die *implizites* Wissen verwenden, indem sie beispielsweise über Datensätze Merkmale trainieren und damit hervorragende Ergebnisse erzielen, ohne die Erkennung in Regeln fassen zu können. (Absatz ist extrahiert aus [Pol67, Neu99])

Begriffsbestimmung 3.2 (Implizites und explizites Wissen)**.** *Vereinfacht ausgedrückt handelt es sich bei explizitem Wissen um Wissen, das mündlich und schriftlich ausgedrückt und zwischen Einzelpersonen ausgetauscht werden kann. Dagegen zeichnet sich implizites Wissen dadurch aus, dass es nicht auf eine solche Weise verfügbar ist. Man kann sagen, dass es den unsichtbaren Teil des Wissens darstellt. Jemand "weiß, wie es geht", aber sein Wissen steckt implizit in seinem Können, ihm fehlen die Worte, um dieses Können zu beschreiben oder es anderen verbal zu vermitteln. Ein Beispiel dafür ist die Fähigkeit, auf dem Fahrrad das Gleichgewicht zu halten. (Absatz ist extrahiert aus [Pol67, Neu99])*

Definition 3.2 (Deklaratives Wissen)**.** *Deklarative Darstellungen von Wissensinhalten sind Beschreibungen von Sachverhalten, die keine Angaben über Konstruktion und Gebrauch von Wissen enthalten.*

Ein Beispiel für deklaratives Wissen ist: Das Produkt aus 2 und 3 ist 6.

Definition 3.3 (Prozedurales Wissen)**.** *Prozedurale Wissensdarstellungen beschreiben Verfahren zur Konstruktion, Verknüpfung und Anwendung von Wissen.*

Ein Beispiel für prozedurales Wissen ist die Beschreibung eines Verfahren zur Berechnung des Produkts aus 2 und 3.

3.2.1 Ontologie-Beschreibungssprachen[3]

Durch die Vision des *Semantic Web* [HKRS08] hat sich die Modellierung, Repräsentation und Verarbeitung von Wissen über spezifische Domänen in der Welt mit Hilfe von *Ontologien* [SS09] durchgesetzt. Der Begriff Ontologie lässt sich dabei bis auf semantische Netze, wie beispielsweise die *Konzeptgraphen* (engl. Conceptual Graphs)[4] von Sowa [MMS93, Sow92], zurückführen, in denen Wissen jedoch lediglich repräsentiert wird. Moderne *Ontologiesprachen* sind syntaktisch so aufgebaut, dass sie sich auf eine *Beschreibungslogik* (engl. Description Logic (DL)) [BCM+03] abbilden lassen, die eine entscheidbare Untermenge der *Prädikatenlogik* erster Ordnung darstellt und von Reasonern automatisch verarbeitet werden kann [HST99].

Bestehende inferierbare Ontologiesprachen können nach der Mächtigkeit der korrespondierenden Beschreibungslogik unterschieden werden. Einen guten Überblick über die Entstehung und Entwicklung einiger dieser Sprachen sowie ihrer Vorgänger gibt [HMW03]. Die vom *World Wide Web Consortiums (W3C)*[5] standardisierte *Web Ontology Language (OWL)* [DMB+06] ist die zur Zeit am weitesten verbreitete inferierbare Ontologie-Beschreibungssprache. Sie ist historisch der direkte Nachfolger von *DAML+OIL* und basiert technisch auf dem *Resource Description Framework Schema (RDFS)* [W3C04b, W3C04a].

Reasoner wandeln Ontologien in Wissensbasen um, die aus einer *Terminological Box (TBox)* mit Konzeptwissen sowie einer *Assertional Box (ABox)* mit Instanzwissen bestehen, und können daraufhin mit Standard-Inferenz-Diensten, wie beispielsweise *Konsistenzcheck*, *Konzepterfüllung*, *Klasseninklusion*, *Klassifikation*,

[3] Dieser Abschnitt orientiert sich am Stand der Forschung im Antrag: "Komponentenorientiertes Konzept zur Nutzung von Modellen und Wissen bei der Objektwiedererkennung in Bildern und Bildfolgen" von Prof. Dr. Jürgen Ebert und Prof. Dr. Dietrich Paulus

[4] `http://www.jfsowa.com/cg`

[5] `http://www.w3.org`

neues implizites Wissen aus vorhandenem explizitem Wissen ableiten. Verschiedene Reasoner unterscheiden sich in der Ausdrucksstärke der umgesetzten Beschreibungslogik, den aufgrund dessen unterstützten Ontologie-Beschreibungssprachen sowie den für die Berechnungen verwendeten Algorithmen. Ein häufig Anwendung findendes Standardverfahren ist das *Tableaux-Kalkül* [SSS91]. Einen guten Überblick über die Entstehung und Entwicklung von Reasoner gibt [BCM+03].

Zur Repräsentation des Wissens eignen sich auch Graphen als *leicht-gewichtige Ontologie*. Ein ähnlicher Ansatz wird auch in dieser Arbeit verfolgt. Walter et al. [WSR10] konnten zeigen, dass sich die wesentlichen Eigenschaften von OWL auch auf den im Folgenden beschriebenen Graphenansatz der *TGraphen* übertragen lassen.

Definition 3.4 (TGraph)**.** *TGraphen sind ein leistungsstarkes Graphkonzept für die explizite Darstellung und den effizienten Umgang mit Wissen in Form von Entitäten mit Attributen und deren Beziehungen zueinander.*

TGraph Knoten und Kanten sind typisiert und attribuiert. Eine Mehrfachvererbung dieser Konzepte ist erlaubt. Die Kanten sind gerichtet und Knoten, Kanten und Inzidenzen sind geordnet. Durch die Kantenanordnung und durch die Verwendung von Kanten als Objekte erster Ordnung haben sie eine höhere Modellierungsmächtigkeit als konventionelle konzeptionelle Graphen.

Eine effiziente Implementierung ist gegeben durch die GraLab C++- bzw. JgraLab[a] Java-Bibliothek [DEL94], die eine einfache Handhabung der TGraphen ermöglicht. TGraphen entsprechen Metamodellen, welche im TGraph-Sprachgebrauch Schemas genannt werden und auch in dieser Arbeit so genannt werden. (entnommen aus [EWD+96, Seite 2 und 3])

[a] `http://jgralab.uni-koblenz.de`

Schemas werden in *grUML* [DEF+98] spezifiziert und sind dadurch formal definiert.

Begriffsbestimmung 3.3 (*grUML*)**.** *grUML bildet eine Teilmenge der UML-Klassendiagramme und besitzt eine formale TGraph-Semantik. So werden beispielsweise nur die Elemente verwendet, die auch in einem Graph dargestellt werden können. Klassen entsprechen Knotentypen, Assoziationen entsprechen Kantentypen, Spezialisierungen und Generalisierungen führen zu Hierarchien von Typen und Attribute verfeinern die Informationen eines Knoten- oder Kantentyps. Multiplizitäten entsprechen Einschränkungen im Knotengrad. (entnommen aus [ERW08, Seiten 9 und 10])*

Definition 3.5 (Multiplizität in UML)**.** *Die Multiplizität in der UML Modellierung beschreibt ein Intervall. Ein Modellelement mit Multiplizität darf*

eine bestimmte Anzahl Werte oder Objekte besitzen. Die Multiplizität wird als Intervall mit unterer und oberer Schranke angegeben.

TGraphen als Modelle wurden und werden in verschiedenen Bereichen eingesetzt, wie beispielsweise in der Softwarepflege der Volksfürsorge oder bei der BDI zur Programmanalyse. Ebert et al. [ERW08] geben einen guten Überblick über TGraphen und deren Verwendung.

3.2.2 Semantische Netze

Semantische Netze lassen sich anwenden, um beispielsweise die geometrischen Eigenschaften und die Bedeutung physikalischer Objekte zu definieren. Allgemein lässt sich Wissen in ihnen repräsentieren. Die Basisidee ist, dass Begriffe in Knoten (Konzepten) und Beziehungen in Kanten repräsentiert werden (siehe Abbildung 3.3).

Definition 3.6 (Semantisches Netz - Konkretisierung)**.** *Im Allgemeinen ist ein semantisches Netz ein (gerichteter) Graph $G = (V, E, I)$, der nur eine eingeschränkte (kleine) Menge von Knoten- und Kantentypen besitzt. Er besteht aus einer endlichen, nicht leeren Menge V von Knoten, einer endlichen Menge E von Kanten und einer* Inzidenzabbildung *$I : E \to V \times V$. Semantische Netze unterscheiden sich vor allem in zwei Punkten: das Vorhandensein bzw. Fehlen einer formalen Definition und der Menge an Knoten und Verbindungstypen.*

Semantische Netzwerke basieren auf der mathematischen Struktur eines Graphen. Sie bestehen aus einem Netz von durch Kanten verbundenen Knoten. Knoten repräsentieren Dinge oder *Konzepte* und (gerichtete) Kanten repräsentieren *Relationen*. Semantische Netze erlauben mathematische Operationen auf begrifflichen Einheiten. Sie ermöglichen also beispielsweise automatische Schlüsse durch die Verfolgung der Verbindungen zwischen den Knoten.

Semantische Netze [Qui68] wurden für die Bild- und Sprachanalyse eingeführt [NFPF96] und erfolgreich verwendet [SN97, KSN92, Qui95, Kum92]. Der Formalismus ERNEST [NSSK90] verbindet eine Repräsentation von Konzepten in einem Graph, in dem wenige Kantentypen erlaubt sind, mit einer allgemeinen Kontrolle auf Basis des A* - Algorithmus [Nil80]. Alternativ dazu wurde die Kontrolle durch ein kombinatorisches Optimierungsproblem gelöst [FNN98].

Die allgemeinen Kontrollverfahren in *ERNEST* wurden möglich, da die Struktur der semantischen Netze im Vergleich beispielsweise zu Ontologien sehr stark eingeschränkt wurde. Sechs mögliche Kantentypen werden in ERNEST erlaubt, von denen die *Bestandteilsbeziehung*, die *Konkretisierung* und die *Spezialisierung* die wichtigsten sind. Konzepte werden in Ebenen zusammengefasst. Zwischen Kon-

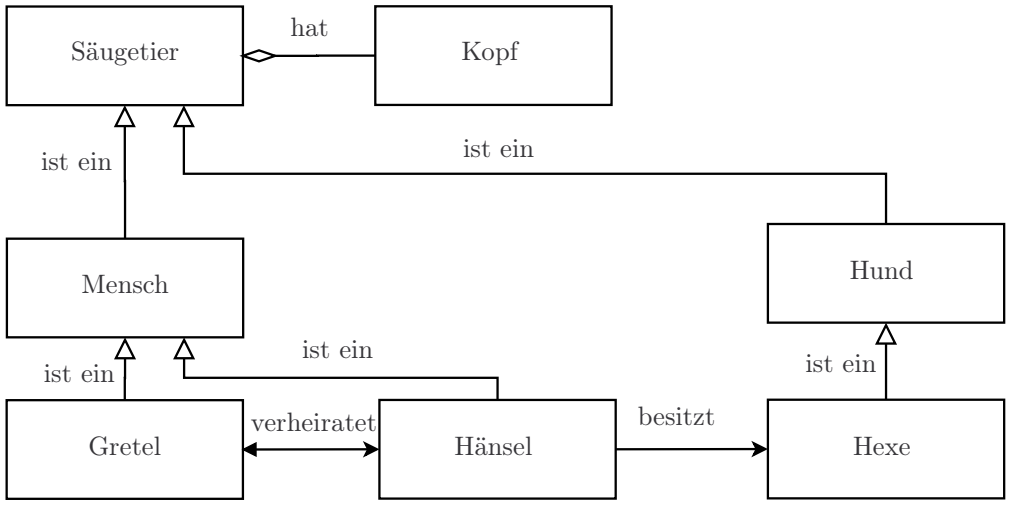


Abbildung 3.3: Beispiel eines semantischen Netzes.

zepten einer Ebene sind Bestandteilsbeziehungen und Generalisierungen möglich. Konzepte verschiedener Ebenen werden durch Konkretisierungen verbunden. Die Korrektheit der Struktur des semantischen Netzes ist syntaktisch überprüfbar. Jedes Konzept erhält eine *Bewertungsfunktion*, mit der sich der Grad der Übereinstimmung einer Instanz mit Sensordaten überprüfen lässt.

Eine perfekte Grundlage, um semantische Netze auch heutzutage noch zu nutzen, bieten TGraphen mit ihrer expliziten Darstellung, effizienten Implementierung und klaren Definition. Aus den erwähnten Gründen kommt die TGraph-Technologie in dieser Arbeit zum Einsatz.

3.2.3 Modellierung im urbanen Umfeld[6]

Aus dem Bereich der *Geoinformatik (GI)* sind Sprachen entstanden, die eine Modellierung von (urbanen) Objekten in 2-D und/oder 3-D ermöglichen. Im Gegensatz zu den Sprachen aus den anderen Bereichen enthalten die GI-Sprachen häufig schon Konzepte zur Modellierung semantischer Objektbestandteile und deren Zusammenhänge. Dafür sind sie aber in der Regel XML-basiert und eignen sich dadurch eher als Austauschformat als für eine direkte (effiziente) Verarbeitung. Wichtige Modellierungsansätze in der GI sind *GML*, *CityGML* und *KML/COLLADA*.

[6]Dieser Abschnitt orientiert sich am Stand der Forschung im Folgeantrag: "Komponentenorientiertes Konzept zur Nutzung von Modellen und Wissen bei der Objektwiedererkennung in Bildern und Bildfolgen" von Prof. Dr. Jürgen Ebert

Die *Geography Markup Language (GML)*[7] ist eine XML-basierte Sprache und dient zur Modellierung, zum Austausch und zur Speicherung von geographischen Informationen [CDL+01]. GML wurde entwickelt als Spezifikation des *Open GIS Konsortiums*, um Interoperabilität zu fördern und den Austausch von Daten zwischen verschiedenen Systemen zu ermöglichen. Ursprünglich basierte das GML-Modell auf dem *Resource Description Framework (RDF)* des World Wide Web Consortiums (W3C). GML erlaubt die Übermittlung von Objekten mit Attributen, Relationen und Geometrien im Bereich der Geodaten unter Einbeziehung von nicht-konventionellen Daten, wie Sensordaten.

Die *City Geography Markup Language (CityGML)*[8] ist ebenfalls eine XML-basierte Sprache und dient zur Erstellung, Verarbeitung und zum Austausch von dreidimensionalen Stadt- und Landschaftsmodellen. Sie wird seit 2002 von der *Special Interest Group 3-D (SIG 3-D)*[9] mit Unterstützung des European Spatial Data Research (Euro SDR)[10] entwickelt und ist seit April 2012 ein Standard in der Version 2.0.0 des *OpenGeospatial Consortium (OGC)*[11]. Der Standard besteht aus einer Spezifikation [GKCN08] sowie einer Sammlung von XML-Schemadateien, die jeweils die Strukturen eines Moduls aus der Spezifikation definieren. Die CityGML-Spezifikation unterscheidet zwischen Semantik, Geometrie, Topologie und Erscheinungsbild von urbanen Objekten. Die Modellierungssprache unterscheidet fünf *„Level of Detail“ (LOD)* [Con14] (siehe auch Tabelle 3.1):

- LOD 0: (Regionalmodell) 2,5-D-Geländemodell mit Luftbildtextur
- LOD 1: (Klötzchenmodell) Gebäudeblock (Grundfläche hochgezogen)
- LOD 2: (vereinfachtes Modell) 3-D-Modell der Außenhülle und Dachstrukturen und einfachen Texturen
- LOD 3: (Architekturmodell) 3-D-Modell der Außenhülle mit Textur
- LOD 4: (Innenraummodell) 3-D-Modell des Gebäudes mit Etagen, Innenräumen etc. und Texturen

Bekannte Werkzeuge zur Visualisierung von CityGML Modellen sind der *IfcExplorer* (unterstützt zusätzlich Modellerstellung und -verarbeitung)[12], der LandXplorer CityGML Viewer[13] und der *AristotelesViewer-GML* 3-D[14].

[7] `http://www.opengeospatial.org/standards/gml`
[8] `http://www.citygml.org`, `http://www.citygmlwiki.org`
[9] `http://www.ikg.uni-bonn.de/sig3d`
[10] `http://www.eurosdr.net`
[11] `http://www.opengeospatial.org/standards/citygml`
[12] `http://www.iai.fzk.de/www-extern/index.php?id=1570`
[13] `http://www.3dgeo.de/citygml.aspx`
[14] `http://www.ikg.uni-bonn.de/aristoteles`, `http://www.ikg.uni-bonn.de/forschung/aristoteles`

	LOD0	LOD1	LOD2	LOD3	LOD4
Modellskalierung	Landschaft	Stadt	Stadtteile	Architektonische Modelle (exterior)	Architektonische Modelle (interior)
Genauigkeit	niedrigste	niedrig	mittel	hoch	sehr hoch
Absolute 3-D-Punktgenauigkeit (Position/Höhe)	weniger als LOD1	5/5m	2/2m	0.5/0.5m	0.2/0.2m
Gebäudeinstallationen	nein	nein	ja	repräsentative äußere Merkmale	wirkliche Objektform
Dachstruktur/Repräsentation	ja	flach	unterschiedliche Dachformen	wirkliche Objektform	wirkliche Objektform

Tabelle 3.1: LOD 0-4 von CityGML mit der vorgeschlagenen Genauigkeitsanforderungen. [Con14]

Die *Keyhole Markup Language (KML)* ist eine XML-basierte Sprache zur Visualisierung von geographischen Daten in digitalen Globen (wie beispielsweise Google Earth). Sie wurde ursprünglich von Google Inc.[15] entwickelt und ist seit April 2008 ein OGC-Standard[16]. Der Standard besteht ebenfalls aus einer Spezifikation sowie einer Sammlung von XML-Schemadateien. KML-Dateien können im Internet in digitale Globen integriert werden, die die Daten direkt interpretieren und visualisieren können. Die (geometrischen) Modellierungsmöglichkeiten in KML orientieren sich ebenfalls an GML, sind allerdings sehr begrenzt. Daher werden komplexere Zusammenhänge häufig in Modelle anderer CG-Formate ausgelagert, wie COLLADA [17] oder auch Extensible 3-D (X3D)[18].

[15] `http://code.google.com/intl/de-DE/apis/kml`
[16] `http://www.opengeospatial.org/standards/kml`
[17] `http://collada.org`
[18] `http://www.web3d.org/x3d`

COLLADA[19] ist ein XML-basiertes Dateiformat zum Austausch von 3-D-Modellen zwischen Werkzeugen und/oder Anwendungen, bei dem ein Objekt gleichzeitig mehrere Repräsentationen haben kann. Ein bekanntes Werkzeug zur Visualisierung von KML/COLLADA-Modellen ist *Google Earth*[20] und ein bekanntes Werkzeug zur Erstellung von KML/COLLADA-Modellen ist *Google Sketchup*[21]. COLLADA-Modelle können von vielen (computergraphischen) Modellierungswerkzeugen importiert, verarbeitet und exportiert werden (beispielsweise *Maya*[22], 3ds Max[23] von Autodesk oder die Open Source Software *Blender*[24]).

3.3 STOR-Modell

Im urbanen Umfeld existiert eine Vielzahl von Daten und Werkzeugen, sodass es wichtig ist, diese Daten möglichst komfortabel und ohne Verlust von Informationen auszutauschen, speichern und verarbeiten zu können (siehe Abschnitt 3.2.3). Diese Arbeit beschäftigt sich sowohl mit der Modellierung als auch mit der Verarbeitung von Wissen. Aus diesem Grund werden die etablierten Ansätze dieser Bereiche kombiniert, um kompatibel zu bleiben und die Vorteile beider Domänen zu nutzen.

Das von Frau Falkowski entwickelte STOR (Software Techniques for Object Recognition)-Referenzschema [FE11] (siehe Definition 2.5) bildet die Referenz aller daraus spezialisierten Objekt-Modell-Schemata.

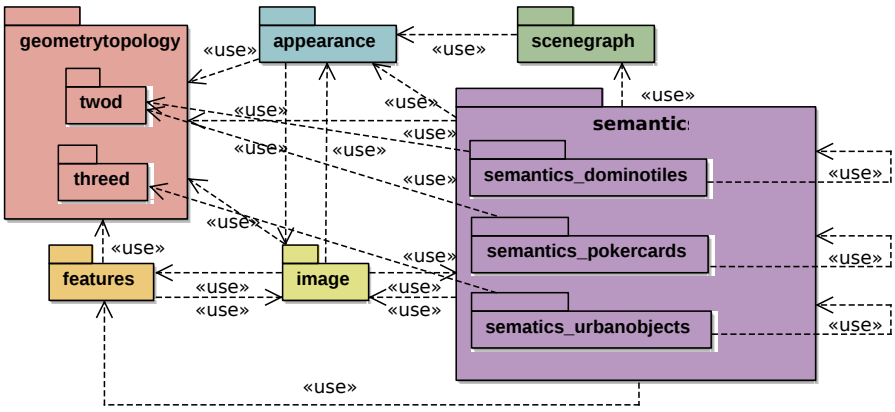


Abbildung 3.4: STOR-Referenzschema (Pakete). (Quelle: Gemeinsame Veröffentlichung mit Frau Falkowski [WFP12])

[19] http://www.khronos.org/collada
[20] http://earth.google.com
[21] http://sketchup.google.com
[22] http://www.autodesk.com/maya
[23] http://www.autodesk.com/3dsmax
[24] http://www.blender.org

Abbildung 3.4 gibt einen groben Überblick über die Schemapakete und deren Beziehungen. Das *image package* enthält Bilder und typische Bildteile, wie Pixel, Kanten und Regionen. Eine Region wird beispielsweise als abstraktes Element mit dem Namen *Region* im Paket repräsentiert. Es wird ausdrücklich zwischen einem Bild als Teil des Modells und einem Bild als physisches Objekt unterschieden. Das *feature package* umfasst typische BV-Elemente, die Ergebnisse der verschiedenen Feature-Erkennungs- und Extraktionsalgorithmen sind, wie beispielsweise Ecken und wichtige Punkte mit entsprechenden Deskriptoren. Das *geometrytopology package* enthält typische Geometrie- und Topologie-Elemente aus der CG, es beschreibt also die Struktur eines Objekts und die Positionen der einzelnen Bestandteile in einem Koordinatensystem. Das Paket enthält ein abstraktes Element mit dem Namen *Geometry*. Alle geometrischen Elemente müssen von diesem geometrischem Objekt abgeleitet sein. Dabei handelt es sich bei den geometrischen Entitäten um Punkte, Strecken, verschiedene Arten von Flächen und spezifische Sammlungen (beispielsweise Dreieckslisten). Topologische Elemente bilden Beziehungen zwischen geometrischen Entitäten und beschreiben so Substrukturen höherer Ordnung. Es gibt Unterpakete für 2-D- und 3-D-Geometrie, die Spezialisierungen des allgemeinen geometrischen Objektes sind. Das *appearance package* umfasst Objekteigenschaften wie Farben oder Texturen. *Appearance* Entitäten besitzen gewöhnlich ein korrespondierendes geometrisches Element. Ein Objekt kann für verschiedene Situationen mehr als ein Erscheinungsbild besitzen. Das *scenegraph package* enthält typische in der CG genutzte Szenegraphentitäten. Ein Szenengraph bildet eine Art Baum, der globale Objektpositionen über Transformationen aus verschiedenen lokalen Koordinatensystemen beschreibt. Das *semantic package* enthält ein abstraktes Element mit dem Namen *SemanticObject*, das die Brücke zwischen semantischen Elementen und allen anderen Paketen schlägt. Für eine konkrete anwendungsspezifische Semantik können Unterpakete in das Schema integriert werden. Alle semantischen Elemente in jedem semantischen Unterpaket müssen von diesem semantischen Objekt abgeleitet werden.

Das *STOR-Referenzschema* bzw. ein spezielles Modell (beschrieben im nächsten Abschnitt) entspricht dem TBox Konzept einer Ontologie-Beschreibungssprache, da es ebenso das Wissen über die Konzepte der Domäne enthält. Genauso wie das Abox Konzept, die ABox enthält das Wissen über das konkrete Modell einer Domäne, können und werden die Objekt-Modell-Schemata für die spätere Analyse in konkrete Modelle überführt. Das *STOR-Referenzschema* entspricht exakt der Definition eines semantischen Netzes und es ist in der Lage CityGML komplett abzubilden [FE11] und effizient zu verarbeiten. Zusätzlich wurden bei der Modellierung der Konzepte und Relationen darauf geachtet die Modelle für die Objekterkennung und Poseschätzung zu optimieren.

Das Konzept *GraphMarker* ermöglicht das Markieren ("kolorieren") von Knoten und Kanten im Modell. Es können zusätzlich zur Information "markiert" oder "nicht markiert" auch weitere Attribute mittels GraphMarker an Modellelemente angehängt werden. Der GraphMarker dient in der Regel zur temporären Markierung ausgewählter Elemente im Verarbeitungsprozess.

Die im Folgenden beschriebenen spezialisierten Objekt-Modell-Schemata des Referenzschemas enthalten neben deklarativem Wissen auch prozedurales Wissen, sodass Objekte mit Aktivitäten versehen werden können (siehe Abschnitt 3.6). Das prozedurale Wissen wird im STOR-Komponentenkonzept [FE09b, Fal10] verwaltet. Die Verbindung von Modellen mit dem Komponentenkonzept erfolgt, indem den Knoten als Methoden entsprechende Komponenten hinzugefügt werden (siehe Kapitel 9).

3.4 Objektmodelle

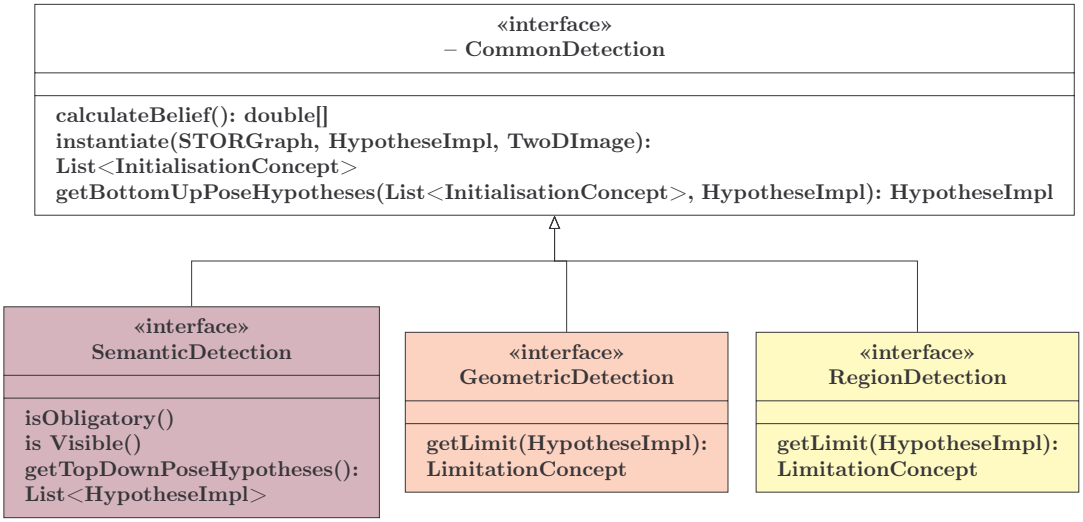


Abbildung 3.5: Bei den Objekt-Modell-Schemata muss jedes Element des Typs *SemanticObjects* des Referenzschemas das Interface *SemanticDetection* implementieren. Jedes Element des Typs *Geometry* implementiert das Interface *GeometricDetection* und jede *Region* implementiert *RegionDetection*. Elemente vom Typ *SemanticObject*, *Geometry* oder *Region* implementieren alle *CommonDetection*.

Für die Fallstudien und die dafür entwickelten *Objekt-Modell-Schemata* wurden Teilmengen der Bild-, Geometrie- und Semantikpakete verwendet (siehe Abschnitt 3.3). Diese Objekt-Modell-Schemata wurden für die Aufgabe der modellbasierten Poseschätzung und Objekterkennung um spezielle *representedBy*- und *consistsOf*-Kanten erweitert.

Es wurden Interfaces für verpflichtende Methodenimplementierungen entwickelt (siehe Abbildung 3.5). Die Interfaces nutzen die Konzepte *InitialisationConcept* und *LimitationConcept*, welche die Zuordnungen von Modellelementen zu Elementen aus dem Analysemodell (siehe Abschnitt 3.4.3) verwalten und beschränken.

Begriffsbestimmung 3.4 (InitialisationConcept)**.** *Ein InitialisationConcept beschreibt die konkrete Zuordnung eines Modellelementes zu einem Bildelement und besitzt als Attribute das Modellelement, das segmentierte Bildelement aus dem Analysemodell und eine Bewertung der Zuordnung. Die Bewertung von einzelnen Elementen oder Zuordnungen wird Belief genannt. Dabei kann es vorkommen, dass einem Modellelement kein Bildelement zugeordnet werden kann. In diesem Fall wird ein sogenanntes "Dummyelement" dem Modellelement zugeordnet, sodass die Ablaufkontrolle das Fehlen mitbewerten kann.*

Begriffsbestimmung 3.5 (LimitationConcept)**.** *Ein LimitationConcept beschreibt Beschränkungen des Suchraums im Bild, sodass Objekte, die in einem bereits gefundenen Objekt enthalten sind, nur noch in der entsprechenden Regions of Interest (ROI) gesucht werden. Dabei wird mit einer Polygonrepräsentation gearbeitet.*

Bei den Objekt-Modell-Schemata muss jedes Element des Typs *SemanticObjects* (siehe Abbildung 3.6) des Referenzschemas das Interface *SemanticDetection* implementieren. Jedes Element des Typs *Geometry* (siehe Abbildung 3.6) implementiert das Interface *GeometricDetection* und jede *Region* (siehe Abbildung 3.8) implementiert *RegionDetection*. Elemente vom Typ *SemanticObject*, *Geometry* oder *Region* implementieren alle *CommonDetection*. Dies hat als Konsequenz, dass nun jedes Objekt des zuvor genannten Typs Methoden besitzt, um sich selbst im Bild zu lokalisieren bzw. zu initialisieren, zu bewerten und auf Basis bereits erfolgter Zuordnungen neue *Posehypothesen* (siehe Kapitel 7.2.2) zu erzeugen. Die Methode *instantiate(STORGraph, HypotheseImpl, TwoDImage)* des Interface *CommonDetection* mit den Argumenten *STORGraph* für das Analysemodell, *HypotheseImpl* für die Hypothese und *TwoDImage* für das Eingabebild ist für die Lokalisierung/Initialisierung zuständig und gibt die getätigten Zuordnungen als *List<InitialisationConcept>* zurück. Die Methode *getBottomUpPoseHypotheses* erzeugt Bottom-Up Posehypothesen und benötigt als Argumente die durch die Instanziierung entstandenen Zuordnungen als *List<InitialisationConcept>* und die Hypothese als *HypotheseImpl* und gibt eine neue Hypothese als Ausgabe zurück. Die Methoden des Interfaces *SemanticDetection* verwalten das Wissen über Sichtbarkeit und ob ein Element obligatorisch für die Erkennung ist. Zusätzlich steuert es "Top-Down" die Erzeugung von Posehypothesen. Die Interfaces *GeometricDe-*

tection und *RegionDetection* verwalten die aus der Segmentierung entstehenden Begrenzungen des Suchraums. Es ist für die Ablaufkontrolle nicht erforderlich, dass alle Methoden und Attribute definiert werden. Die Konzepte und deren Umsetzung werden in Kapitel 9 näher beschrieben. Eine Beschreibung der Ablaufkontrolle erfolgt in Kapitel 6.

Der Kantentyp *representedBy* beschreibt, dass ein Objekt durch ein anderes repräsentiert wird. In der Regel wird ein semantisches Objekt durch ein geometrisches Objekt oder eine Region repräsentiert. Der Kantentyp *consistsOf* setzt die Enthaltenseinrelationen als Aggregation um und beschreibt auf diese Weise, ob und wie ein Objekt in einem anderen Objekt enthalten ist. Dies ermöglicht eine hierarchische Analyse des Modells. Diese Kantentypen sind abstrakt und werden für konkrete Modelle spezialisiert.

Die TGraphen ermöglichen eine modellgetriebene Softwareentwicklung, weshalb sie perfekt zu dem Ansatz dieser Arbeit passen. Die mittels UML-Werkzeug entwickelten grUML-Schema-Diagramme werden zunächst als XMI-Datei (XML Metadata Interchange[25]) gespeichert. Dieses Modell wird dann in das sogenannte TG-Format (eine TGraph-Schema-Datei) mittels XML-Transformation überführt. Der bereits erwähnte JGraLab Codegenerator überführt das Schema dann in eine Objektorientierte API für Graphen, die komfortablen Zugriff auf die Modelle gestattet [ERW08].

3.4.1 Dominostein- und Pokerkartenmodell

Die Abbildungen 3.6 und 3.8 präsentieren die Elemente der Schemas einschließlich der Semantik Unterpakete *semantics_ poker cards* und *semantics_ dominotiles* und orientieren sich an den Fallstudienbeschreibungen für die Dominostein- und Pokerkartenerkennung aus den Kapiteln 2.1 und 2.2. Diese Entitäten umfassen Teilmengen der Pakete *Bild*, *Geometrie* und *Semantik* für urbane Objekte. Entsprechend den Forderungen der Ungleichung 2.9 und 2.20 aus den Kapiteln 2.1 und 2.2 spezialisieren alle konkreten geometrischen Elemente des Modells das Element *GeometricObject*.

Abbildung 3.6 zeigt die Elemente des Schemas, die genutzt werden, um Dominosteine zu beschreiben. Ein Dominostein (*DominoTile*, DTrect_1) besteht aus zwei Hälften (*DominoTilePart, DTrect*$_2$ *und DTrect*$_3$) und beide Konzepte werden repräsentiert von einem 2-D-Rechteck (*TwoDRectangle*). Zugleich besitzt eine Dominosteinhälfte (*DominoTilePart*) eine Augenzahl (*Pips*, $\mathrm{LO}_{\mathrm{md}}$) von null bis sechs, die von 2-D-Kreisen (*TwoDCircles*) repräsentiert werden. Die semantischen Konzepte eines Dominosteinmodells sind Spezialisierungen von *SemanticObject*. Das 2-D-Rechteck (*TwoDRectangle*) und der 2-D-Kreis (*TwoDCircle*) sind Spezialisierun-

[25] `http://www.omg.org/spec/XMI/`

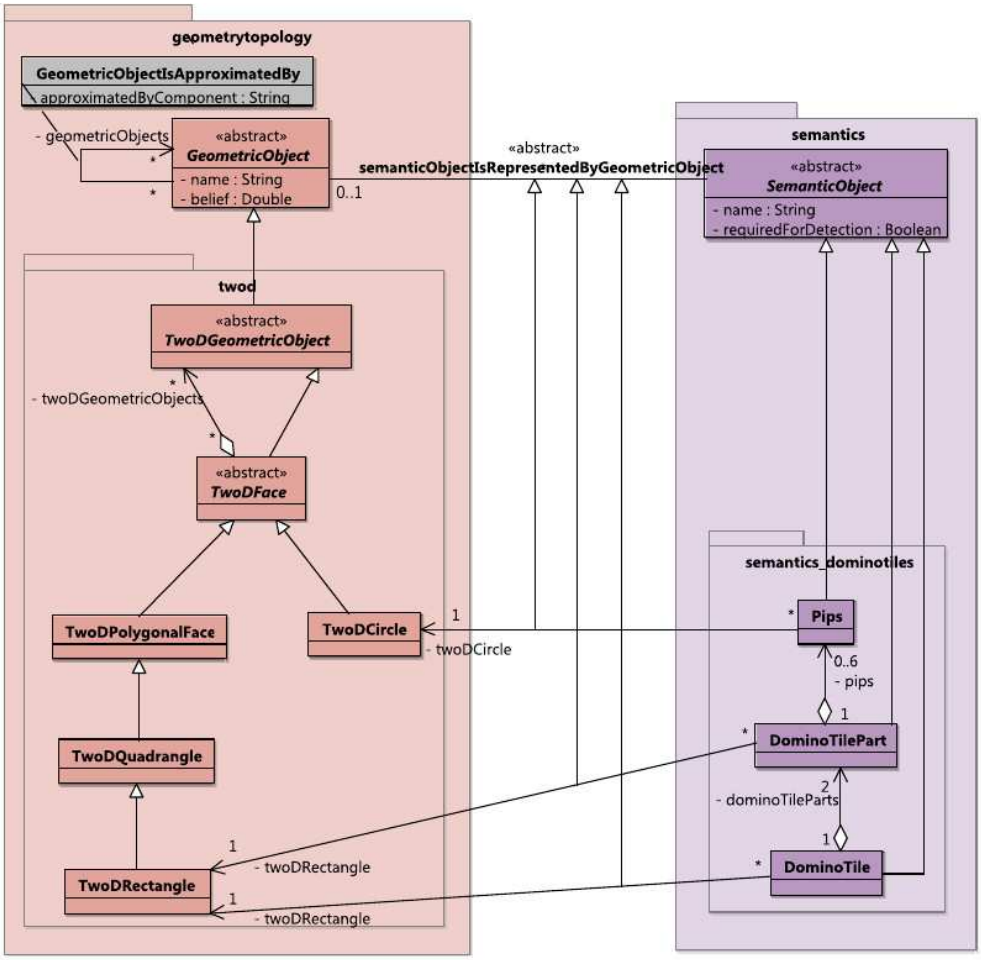


Abbildung 3.6: Elemente des Schemas für Dominosteine. (Quelle: Gemeinsame Veröffentlichung mit Frau Falkowski [WFP12])

gen von *TwoDGeometricObject*. Abbildung 3.7 zeigt ein Modell eines Dominosteins mit 3 Augen (*Pips*) auf der einen Seite und 2 Augen auf der anderen Seite. Die Augen (*Pips*) werden repräsentiert durch Kreise (*TwoDCircle*) und verbunden mit der Kante *PipsIsRepresentedByTwoDCircle*. Der Dominostein (*DominoTile*) und seine Hälften (*DominoTilePart*) werden durch Rechtecke (TwoDRectangle) repräsentiert und verbunden mit den Kanten *DominoTileIsRepresentedByTwoDRectangle* bzw. *DominoTilePartIsRepresentedByTwoDRectangle*.

Abbildung 3.8 zeigt die Elemente des Schemas, die genutzt werden, um Pokerkarten zu beschreiben. Eine Karte (*Card*, CARD) kann aus den Kartenfarben (*SuitForms*, $\mathrm{LO_{mp}}$) (Karo, Herz, Pik, Kreuz), den Buchstaben (*CharacterForms*) (A,J,Q,K) und/oder Zahlen (*NumberForms*) $(0 \ldots 9)$ bestehen. Die Karte (*Card*) selbst ist eine Generalisierung von einer Zahlenkarte (*FigureCard*) und Figurenkar-

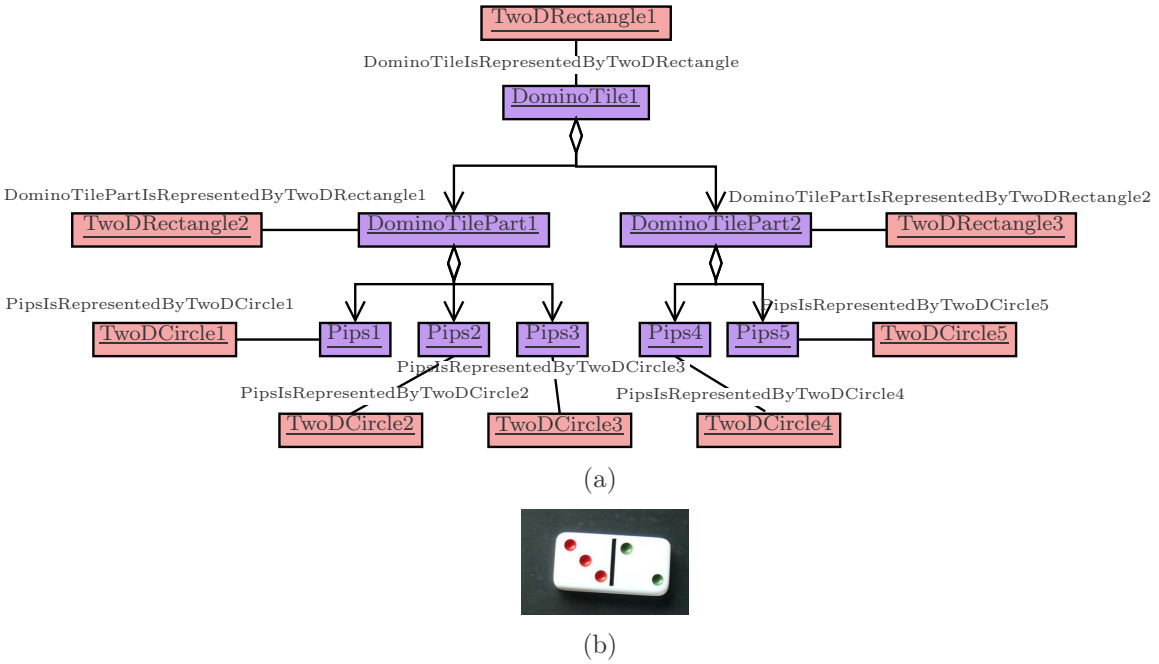


Abbildung 3.7: Beispiel für ein Modell des Dominosteinschemas.

te (*PictureCard*), sodass die Multiplizitäten von Kartenfarben (*SuitForm*) unterschieden werden können. Eine Karte wird repräsentiert von einem 2-D-Rechteck (*TwoDRectangle*) und den Formen *SuitForm*, *CharacterForm* und *NumberForm*. Formen werden repräsentiert durch Regionen (*Region*) und charakterisiert durch Zentrale- und Hu-Momente [Hu62]. Die semantischen Konzepte eines Pokerkartenmodells sind Spezialisierungen von *SemanticObject*.

Abbildung 3.9 zeigt ein Modell, einer Karo-2, des Pokerkartenschemas. Die Zahlenkarte (*FigureCard*) besitzt sechs Elemente *SuitForm* mit dem Attribut für Karo (*Diamond*). Dabei werden sowohl die Kartenfarben aus dem Zentrum als auch die vom Rand modelliert. Eine Karte besitzt zudem vier Elemente vom Typ *NumberForm* mit Attribut *Two* für die Zahlen auf der Karte. *Die SuitForms* und die *NumberForm* werden repräsentiert durch Regionen (*Region*) und die entsprechenden Kanten heißen dann *SuitFormIsRepresentedByRegion* und *NumberFormIsRepresentedByRegion*. Jede Region wird durch *Momente* charakterisiert. Die Karte wird repräsentiert durch das Rechteck *TwoDRectangle*. Da Dominosteine und Pokerkarten über ihre 2-D-Vorderseiten klassifizierbar sind, werden sie in 2-D beschrieben, obwohl es sich um 3-D-Objekte handelt.

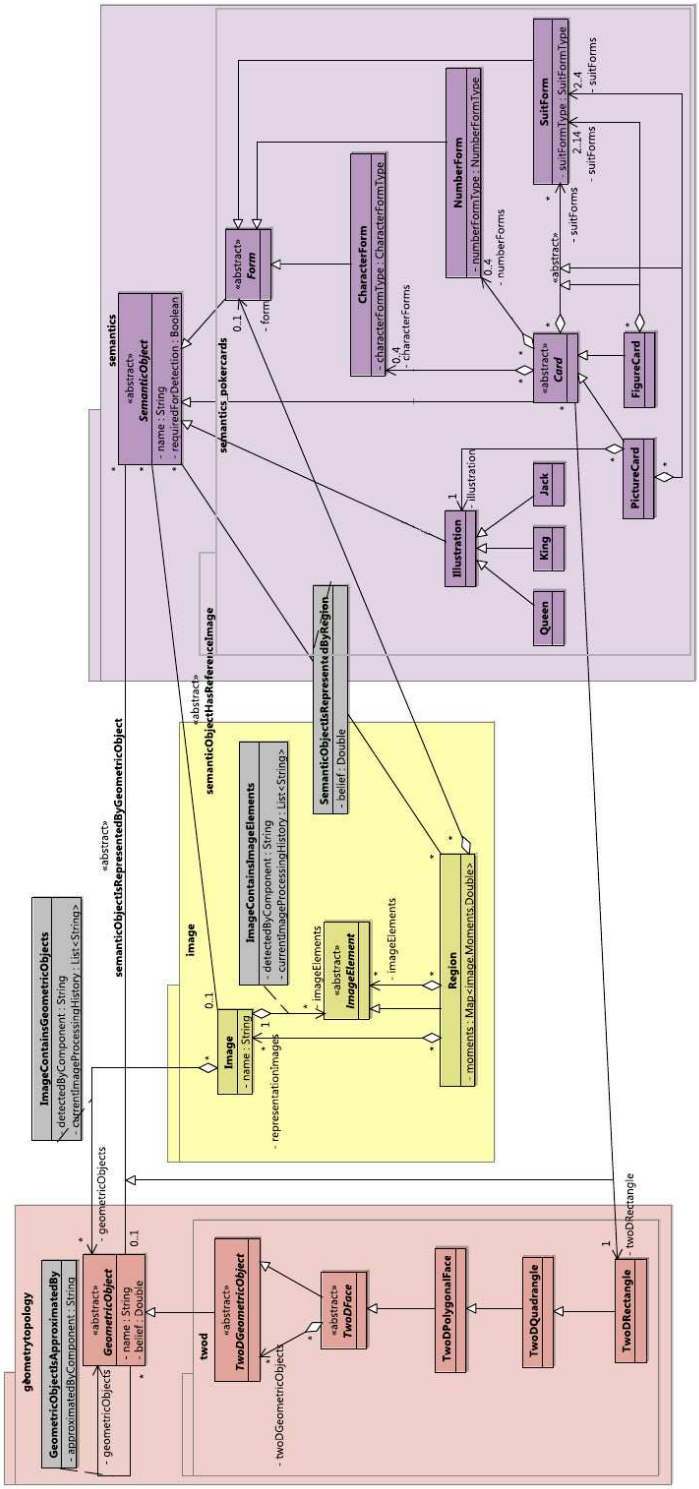
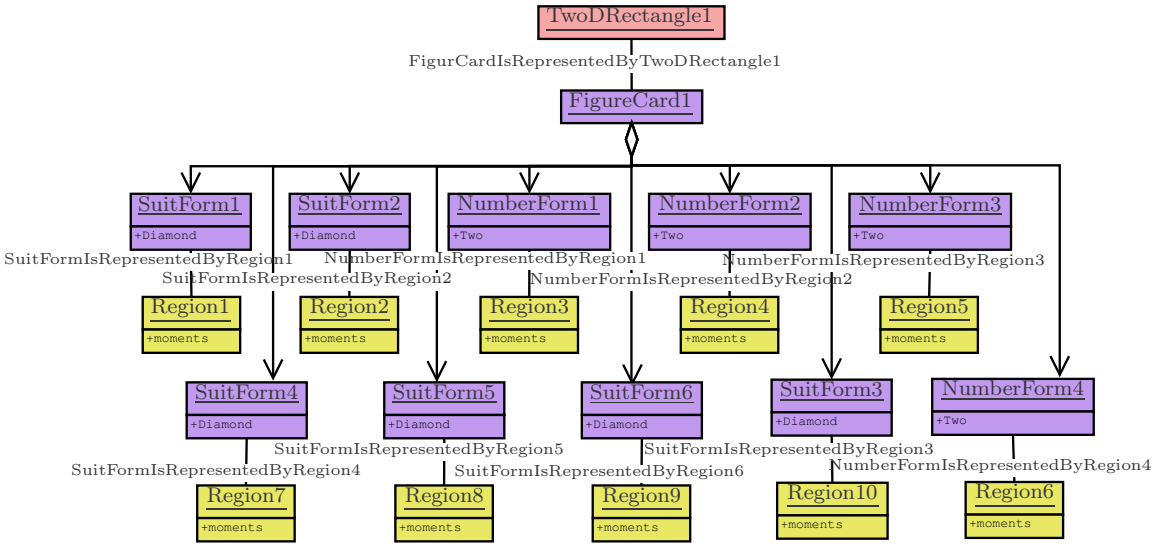
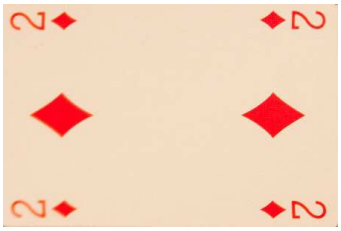


Abbildung 3.8: Elemente des Schemas für Pokerkarten. (Quelle: Gemeinsame Veröffentlichung mit Frau Falkowski [WFP12])



(a)



(b)

Abbildung 3.9: Beispiel für ein Modell des Pokerkartenschemas.

3.4.2 Gebäudemodell

Das Gebäude-Modell-Schema wurde von Frau Falkowski entwickelt, um verschiedene Austauschformate (OGRE[26] /COLLADA[27]) für städtische Objekte zu vereinen und in jedes dieser Formate wieder überführen zu können [FE09a]. Es basiert auf dem Semantik-Paket des STOR-Referenzschema und ist angelehnt an CityGML (siehe Kapitel 3.2.3). Zudem wurde bei der Modellierung auch die Verwendung des Modells zur Poseschätzung berücksichtigt. Abbildung 3.10 zeigt die Entitäten des speziellen Gebäudeschemas, die für diesen Ansatz verwendet werden. Diese Entitäten umfassen Teilmengen der Pakete *Bild*, *Geometrie* und *Semantik* für urbane Objekte. Die semantischen Objekte sind im Unterpaket *semantics_building* enthalten. Das graphbasierte Modell beschreibt die Semantik eines Gebäudes mit *consistsOf*-Kanten, sodass ersichtlich ist, ob beispielsweise ein Fenster in einer Fas-

[26] `http://www.ogre3d.org/`
[27] `https://www.khronos.org/collada/`

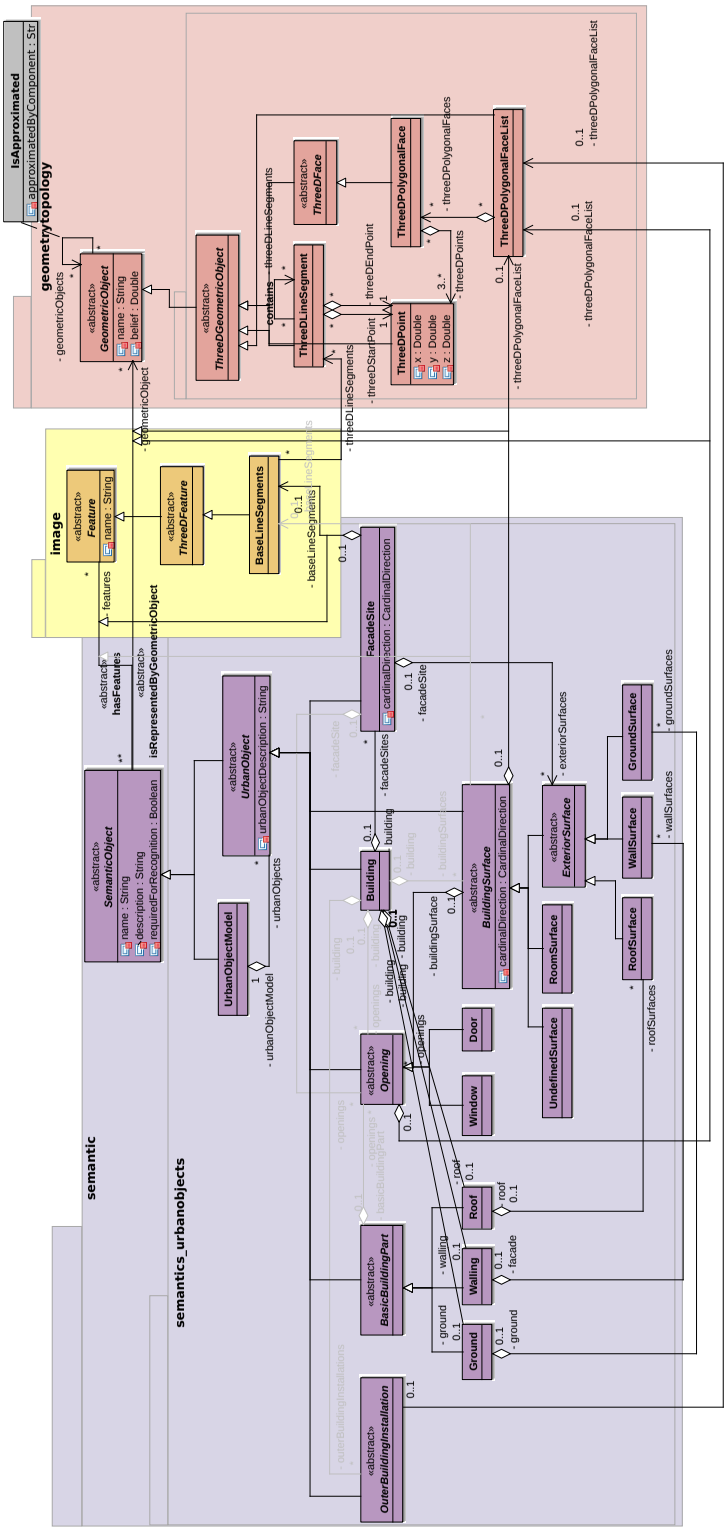
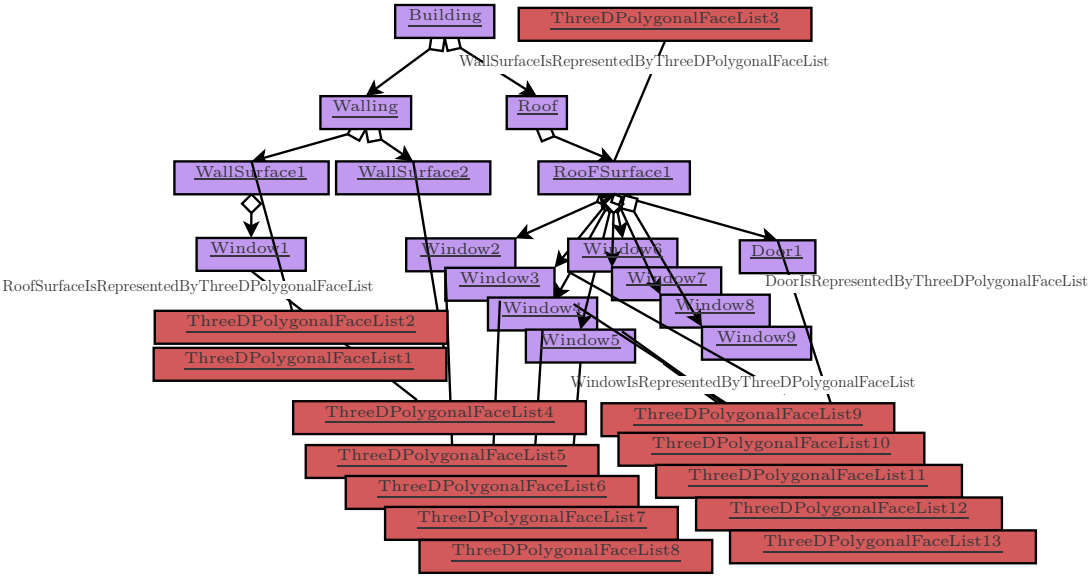


Abbildung 3.10: Elemente des Schemas für Gebäude.

sade enthalten ist. Die semantischen Objekte, wie Fassade, Dach, Tür und Fenster, werden geometrisch durch Strecken und Polygone repräsentiert. Diese Geometrie ist mit *isRepresentedBy*-Kanten an die jeweiligen Semantiken gekoppelt.



(a)

(b)

Abbildung 3.11: Beispiel für die Teilmenge eines Gebäudemodells. Hier werden nur die Semantikelemente und ihre direkten Verbindungen zur Geometrie, der in Abbildung 3.11(b) zu sehenden Fassade, gezeigt.

Ein Gebäude hat beliebig viele Gebäudeseiten (*FacadeSite*,BS) und besteht aus maximal drei Gebäudeteilen (*BasicBuildingPart*): dem Dach (*Roof*), dem Wandsystem (*Walling*) und dem Boden/Keller (*Ground*), die wiederum aus beliebig vielen Flächen (*GroundSurface*, *Wallsurface* und *RoofSurface*) bestehen können. Die Bestandsteilbeziehungen werden mit Spezialisierungen der *consistsOf*-Kante modelliert. Ein Gebäude und alle seine Teile sind urbane Objekte und damit semantische Objekte. Das Schema bietet die Möglichkeit, ein semantisches Objekt durch

ein geometrisches Objekt darzustellen (abstrakte Kante *isRepresentedByGeometricObject*). Diese Kante wird für Bestandteile des Gebäudemodells spezialisiert. Gebäudeflächen (*BuildingSurfaces*) sowie Öffnungen (*Opening*, $\mathrm{OP_M}$) werden von einer 3-D-Polygon-Liste (*ThreeDPolygonalSurfaceList*) repräsentiert werden.

Mittels *semantischem Rendering* (siehe Kapitel 3.5) kann ein 3-D-Modell um die 2-D-Projektion in Abhängigkeit einer gegebenen Kameraposition ergänzt werden. Es wird die 3-D-Geometrie in 2-D projiziert und die sichtbaren Modellelemente mit entsprechender 2-D-Geometrie versehen. Das Resultat ist ein 3-D-Modell, welches um eine 2-D-Modellprojektion des Ausgangsmodells angereichert wurde.

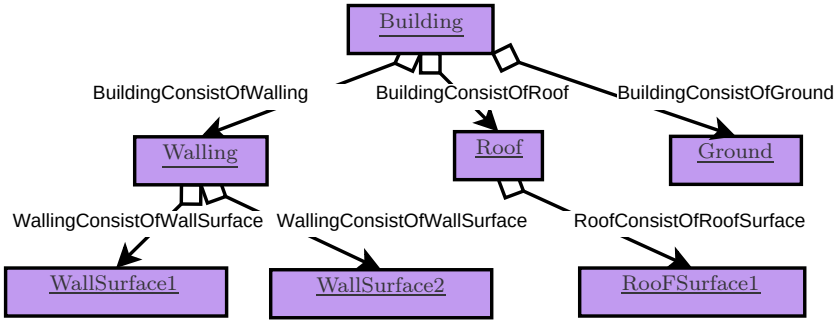


Abbildung 3.12: Beispiel für die Teilmenge eines Gebäudemodells. Hier werden nur die ersten Bestandsteilrelationen gezeigt.

Ein Beispiel für ein Gebäudemodell gibt Abbildung 3.11. Es werden nur die semantischen Objekte und deren direkte geometrische Beziehung von der in Abbildung 3.11(b) sichtbaren Fassade präsentiert, da sonst die Lesbarkeit nicht mehr möglich gewesen wäre. Das Gebäude (*Building*) besteht aus dem Wandsystem (*Walling*) und dem Dach (*Roof*). Das Dach besteht aus zwei Dachflächen und das Wandsystem aus einer Wandfläche. Ein Auszug der Bestandsteilbeziehungen mit den spezialisierten Kanten *BuildingConsistsOfRoof*, *BuildingConsistsOfWalling*, *BuildingConsistsOfGround*, *WallingConsistsOfWallSurface* und *RoofConsistsOfRoofSurface* wird in Abbildung 3.12 gezeigt.

Die Konzepte *RoofSurface*, *WallSurface*, *Window* und *Door* werden repräsentiert durch jeweils eine Liste von 3-D-Polygonen (*ThreeDPolygonalFaceList*) (siehe Abbildung 3.12). Entsprechende konkrete Kanten sind *RoofSurfaceIsRepresentedByThreeDPolygonalFaceList*, *WallSurfaceIsRepresentedByThreeDPolygonalFaceList*, *WindowIsRepresentedByThreeDPolygonalFaceList* und *DoorIsRepresentedByThreeDPolygonalFaceList*.

3.4.3 Analysemodell

Eine Begriffsbestimmung des Analysemodells findet sich auf Seite 18 in Begriffsbestimmung 1.4. Im Zuge der Bildanalyse werden Informationen aus den Bildern extrahiert und in einem gesonderten Analysemodell gespeichert. Dieses Modell basiert ebenso auf dem STOR-Referenzmodell wie die Objekt-Modell-Schemata. Es werden nicht nur die verarbeiteten Bilder, Bildregionen, anfallenden Merkmale, geometrischen Objekte, *Regions of Interest* (ROIs) etc. gespeichert, sondern auch deren Beziehung zueinander, sodass dieses Wissen nur einmal generiert werden muss. Wiederum werden alle Informationen, die aus den extrahierten Bilddaten gewonnen wurden, ins Analysemodell hinzugefügt. Dies können beispielsweise auf Basis von Kanten und Regionen erkannte Fenster sein.

Für die Analyse sind zwei Arten von Modellen besonders wichtig. Zum einen das Objekt-Modell-Schema und die daraus generierten Modelle und zum anderen das Analysemodell, das die aus dem Bild extrahierten und geschlussfolgerten Konzepte enthält. Die Verbindung zwischen beiden Modelltypen erfolgt über die *Hypothese*.

Definition 3.7 (Hypothese - Konkretisierung)**.** *Eine Hypothese, im Verständnis dieser Arbeit, beschreibt eine oder mehrere Zuordnungen zwischen Elementen des Analysemodells und Elementen eines Modells. Jede Hypothese besitzt eine Bewertung in Form eines Vertrauensmaßes.*

Aus technischer Sicht wird eine Hypothese definiert durch: ein Modell (inkl. erzeugter 2-D-Geometrie), eine Pose, eine Sammlung von Zuordnungen mittels InitialisationConcept*, eine Sammlung von ROIs (Regions of Interest) für die Bildanalyse mittels* LimitationConcept *und einer Bewertung der Hypothese. Zudem ist jeder Hypothese die Vorgängerhypothese bekannt, sodass der Analyseprozess vollständig nachvollzogen werden kann.*

Dabei ist der Modellteil einer Hypothese keineswegs so starr, wie es womöglich scheint. Die Extraktion von neuen Posen bzgl. eines Modells hat zur Folge, dass für jede extrahierte Pose ein neues Modell mit entsprechender erweiterter 2-D-Geometrie entsteht, was für die Analyse wiederum einer neuen Hypothese entspricht.

3.5 Semantisches Rendering

Ziel ist es die sichtbaren Elemente eines Modells zu erkennen, die 2-D-Geometrien zu berechnen und gleichzeitig die Relationen zwischen den Modellelementen zu erhalten.

Definition 3.8 (Semantisches Rendering)**.** *Unter* semantischem Rendering[ab] *wird der Vorgang verstanden, der aus einem* semantischen Modell *und einer gegebenen Kamerapose eine Liste der semantischen und prägnanten Merkmale zurückliefert, die aus der Kameraperspektive (unter Berücksichtigung von Verdeckungen und Öffnungswinkel der Kamera) zu sehen sind. Dies ist explizit etwas anderes als ein visuell gerendertes Bild.*

[a] Angelehnt an: Glossar des DFG-Projekt Pose-Tracking mittels markanter Merkmale
[b] `www.uni-koblenz-landau.de/de/koblenz/fb4/icv/agpriese/research/PosE`

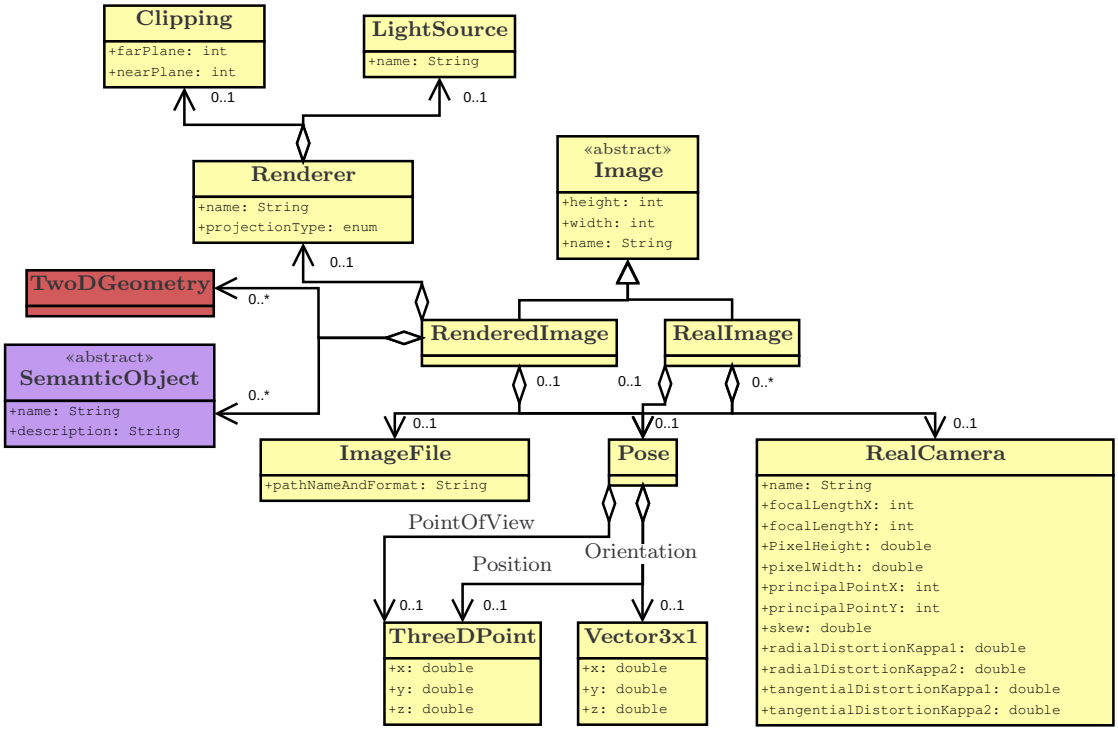


Abbildung 3.13: Die Elemente des Schemas, die zur Speicherung der Pose nötig sind. Ein echtes Bild (*RealImage*, I) kann unter anderem Informationen über die intrinsischen Kameraparameter der Kamera (*RealCamera*), den Speicherort des Bildes (*ImageFile*) und über die Pose (*Pose*, $v \in \Upsilon$) verfügen.

Aufgabe des Renderings ist es, die Sichtbarkeit von Objekten aus einer bestimmten Ansicht zu ermitteln, diese Objekte entsprechend der Materialeigenschaften ihrer Oberflächen realitätsnah darzustellen, und mittels Berechnung der Lichtverteilung innerhalb der Szene die direkte bzw. indirekte Beleuchtung und den entsprechenden Schattenwurf zu visualisieren. Für die verwendete Verifikation von Posehypothesen (siehe Kapitel 8) ist einzig die Sichtbarkeit und die entstehende

2-D-Geometrie für eine Posehypothese unter Verwendung eines 3-D-Modells von Bedeutung.

(a) (b)

Abbildung 3.14: Beispiel für ein Zwischenergebnis des semantischen Rendering. In einem Vorverarbeitungsschritt werden die geometrischen Objekte des Modells extrahiert und mit einem RGB-Wert codiert, sodass sie jederzeit rückverfolgbar sind. Im Anschluss an das Rendering einer Posehypothese werden die Bildelemente entsprechend ihres RGB-Wertes extrahiert und als 2-D-Polygonliste in das Modell übertragen.

Abbildung 3.13 präsentiert die Elemente des Schemas, die nötig sind, um die Pose im Modell zu speichern. Ein echtes Bild (*RealImage*, I) kann unter anderem Informationen über die intrinsischen Kameraparameter der Kamera (*RealCamera*), den Speicherort des Bildes (*ImageFile*) und über die Pose (*Pose*, $\upsilon \in \Upsilon$) verfügen. Die Pose wird über die Orientierung, Position und den Blickpunkt repräsentiert. Ein gerendertes Bild (*RenderedImage*) verfügt neben der Pose und dem Speicherort auch über Informationen wie es gerendert wurde (*Renderer*) und kann die gerenderte Geometrie (*TwoDGeometry*) und semantische Objekte (*SemanticObject*) enthalten.

Durch seine Plattform- und Hardwareunabhängigkeit sowie vorhandener Opensource Referenzimplementierungen eignet sich *OpenGL* sehr gut als Basis für das semantische Rendering, sodass die OpenGL Rendering Pipeline hier Verwendung findet [NFH07]. Abbildung 3.15 beschreibt den Ablauf des semantischen Renderings. In einem Vorverarbeitungsschritt werden die geometrischen Objekte des Modells extrahiert und mit einem RGB-Wert codiert, sodass sie jederzeit rückverfolgbar sind. Im Anschluss an das Rendering einer Posehypothese werden die Bildelemente entsprechend ihres RGB-Wertes extrahiert, als 2-D-Polygonliste in das Modell übertragen und mit den bestehenden Modellelementen mittels *LinksTo*-Kante verknüpft. Bezüglich der gerenderten Posehypothese ist die Information der Sichtbarkeit der Modellelemente im Modell verfügbar. Es werden zur Hypothesenbewertung nur die sichtbaren Modellelemente im Bild verwendet. Abbildung 3.15 und Abbildung 3.14 zeigen, dass die Gebäudeflächen und die Fenster und Türen seperat betrachtet werden, um so eine möglichst optimale Genauigkeit zu erzielen.

Abbildung 3.16 zeigt, wie das semantische Rendering in ein Modell integriert wird. Zunächst wird die bekannte Pose dem Objektmodell hinzugefügt, indem die

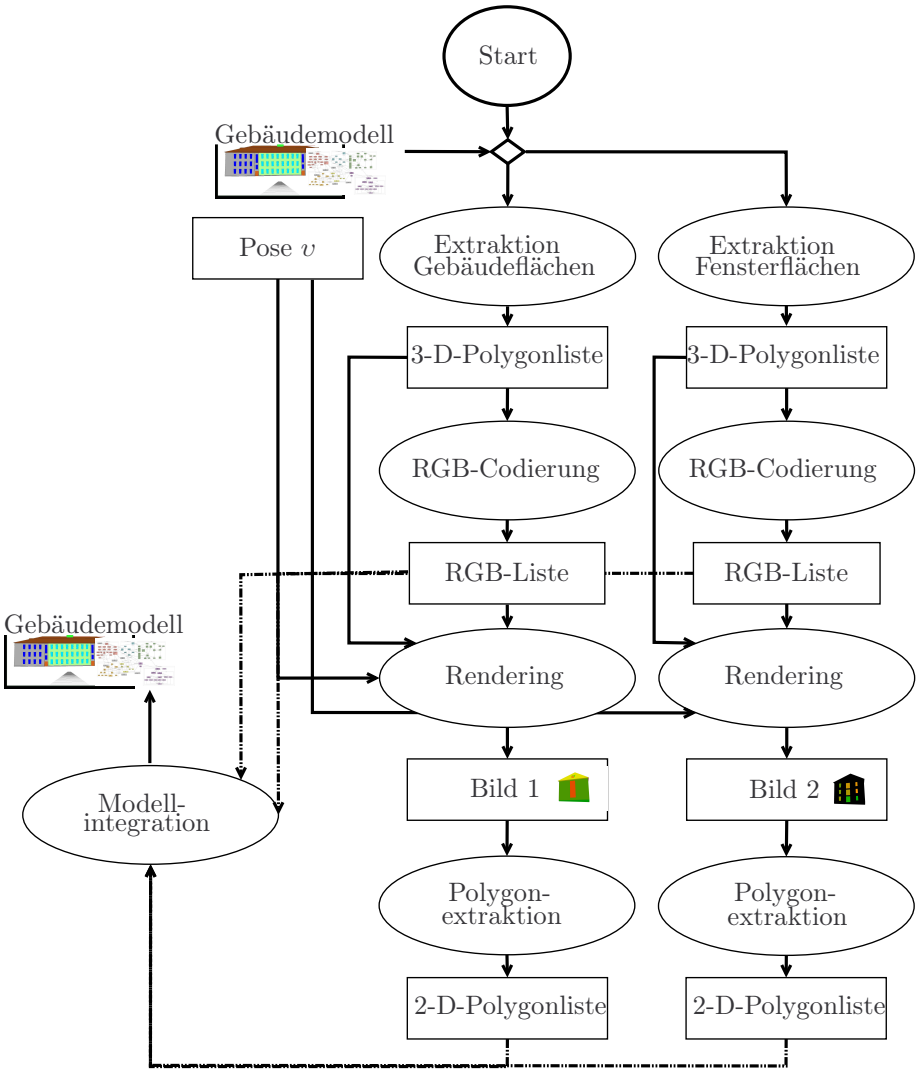


Abbildung 3.15: Ablauf des semantischen Renderings.

Pose mit Orientierung (*Orientation*), Blickpunkt (*PointOfView*) und Standpunkt (*Position*) hinzugefügt wird. Wichtig für das Rendering sind auch die intrinsischen Parameter der Kamera, die in *RealCamera* modelliert werden. Alle diese Konzepte werden dem entsprechenden Bild (*RealImage*) in Kombination mit einem Speicherort, beispielsweise "DSC0001.JPG", vom Typ *ImageFile* zugeordnet. Wird nun mittels dieser Pose ein Bild gerendert (eigentlich werden zwei Bilder gerendert, aber zur Vereinfachung wird hier nur eins modelliert, das schränkt aber nicht die Gültigkeit des Beispiels ein), wird ein entsprechendes Bild vom Typ *RenderedImage* mit einem Speicherort, in diesem Fall "rendered0001", ins Modell integriert und mit der Pose verbunden. In dem Beispiel werden drei Gebäudeflächen ge-

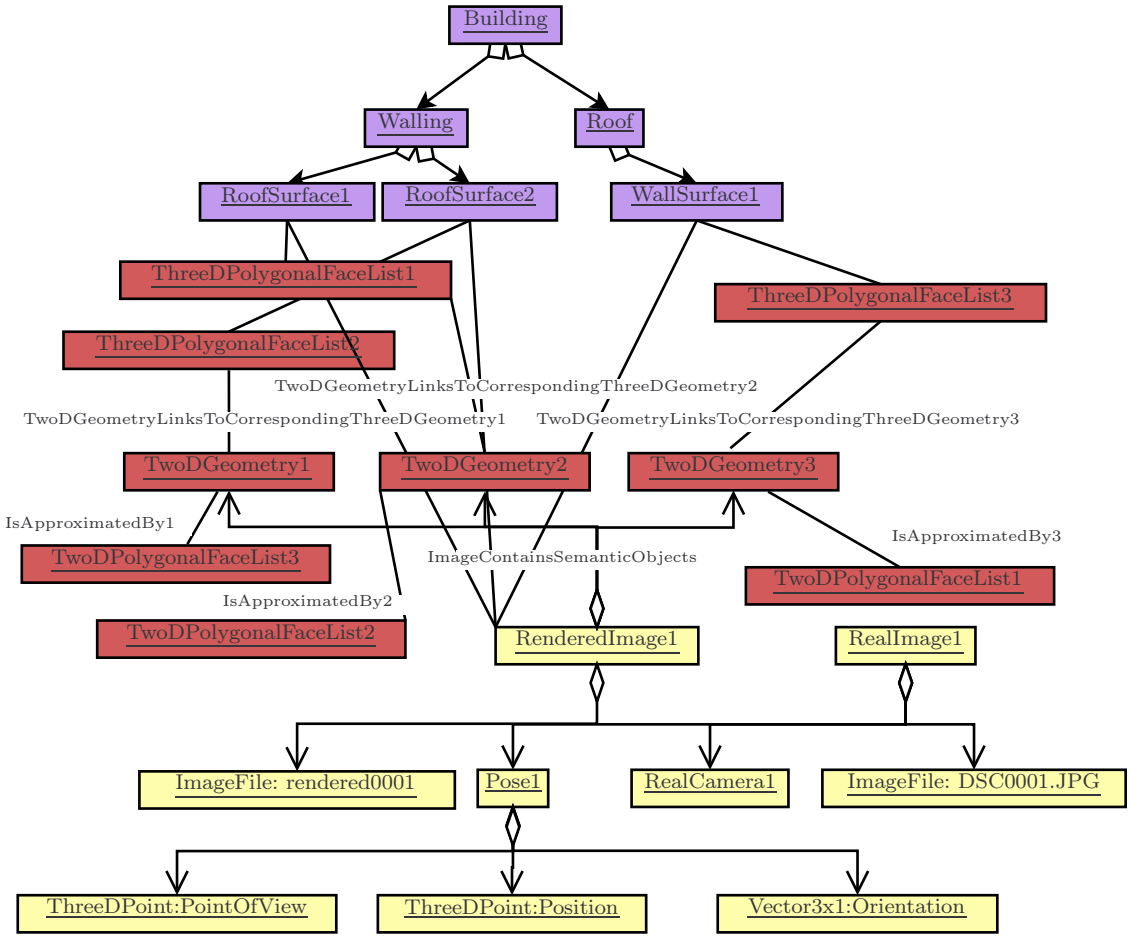


Abbildung 3.16: Modell einer beispielhaften Posezuordnung. Es wird die bekannte Pose im Modell hinzugefügt, indem die Pose mit Orientierung (*Orientation*), Blickpunkt (*PointOfView*) und Standpunkt (*Position*) hinzugefügt wird. Wichtig für das Rendering sind auch die intrinsischen Parameter der Kamera, die in *RealCamera* modelliert werden.

rendert und als Liste von 2-D-Polygonen (*TwoDPolygonalFaceList*) gespeichert. Diese Polygonelisten werden jeweils mittels dem Konzept *TwoDGeometry* über die Kante *isApproximatedBy* mit dem gerendert Bild (*RenderedImage*) verbunden. Das Konzept *TwoDGeometry* verbindet sich mit dem Kantentyp *TwoDGeometryLinksToCorrespondingThreeDGeometry* mit der entsprechenden 3-D Geometrie des Objektmodells, in diesem Fall mit einer Liste 3-D-Polygone (*ThreeDPolygonalFaceList*). Über die Verbindung zwischen gerenderter 2-D-Geometrie und 3-D-Modellgeometrie kann man vom Rendering auf die zugehörigen semantischen Modellbestandteile schließen und umgekehrt. Zusätzlichen werden, zum Zwecke einer möglichst effizienten Verarbeitung, die enthaltenen semantischen Objekte mittels Kantentyp *ImageContainsSemanticObjects* angebunden an *RenderedImage*, so

dass einem gerenderten Bild bekannt ist welche semantischen Modellelemente für die zugehörige Pose sichtbar sind (siehe Kapitel 9).

3.6 Prozedurales Wissen im STOR-Komponentenkonzept

Das prozedurale Wissen wird konform zu den Interfaces in Abbildung 3.5 zu den generierten Schemata hinzugefügt und in Komponenten entsprechend dem STOR-Komponentenkonzept gekapselt.

Definition 3.9 (STOR-Komponenten)**.** *STOR-Komponenten sind eine besondere Form von Strategieobjekten mit einer weitgehenden Selbstauskunftsfähigkeit. Komponenten werden als Komposita gesehen, die atomar oder aus Teilkomponenten zusammengesetzt sein können. Durch die Verwendung von Reflexion ist eine sehr hohe Flexibilität in Bezug auf Manipulierbarkeit und Analysierbarkeit der Komponenten möglich, was die Unterstützung von Experimenten fördert. (extrahiert aus [Fal10])*

Definition 3.10 (STOR-Komponentenkonzept)**.** *Das Konzept umfasst sowohl einfachere Services für die Bild- und Modellverarbeitung als auch komplexe Services, die einen kompletten Ablauf für die Wiedererkennung eines Objekts steuern. Der Fokus liegt dabei vor allem auf den Services für die Modellverarbeitung, die auf dem Referenzschema sowie dazu konformen Modellen für die Objektwiedererkennung in Bildern basieren und den Services für Objektwiedererkennung. Darauf aufbauend implementieren die existierenden STOR-Komponenten die Services, die für eine Objektwiedererkennung in Bildern notwendig sind. (extrahiert aus [FE09b])*

Das von Falkowski [Fal10] entwickelte *Komponentenkonzept* verwaltet das verfügbare prozedurale Wissen der Modelle, orientiert sich an den Konventionen der *Java-Beans* und baut auf gängigen Entwurfsmustern auf. Eine STOR-Komponente ist ein wiederverwendbares und ausführbares Software-Element, das einen oder mehrere Services bereitstellen kann. Ein Service wird ausführlich natürlichsprachlich beschrieben und durch ein Java-Interface spezifiziert. Die Java-Interfaces werden durch spezielle Annotationen ergänzt, die es einer Komponente ermöglichen ihre syntaktischen Eigenschaften und zum Teil auch ihre Semantik zur Laufzeit selbst zu beschreiben. Ein Service wird portbasiert beschrieben, d. h. für jede benötigte Entität (Datum oder Service) gibt es einen Eingabeport in Form einer `set()`-Methode, für jedes bereitgestellte Datum einen Ausgabeport in Form einer `get()`-Methode. Darüber hinaus besitzt jede Komponente eine `execute()`-Methode, die

die Komponente anstoßen kann. Die portbasierte Beschreibung ermöglicht die Trennung von Daten- und Kontrollfluss zur Assemblierungs- und Ausführungszeit.

Die Komposition von Komponenten kann auf zwei verschiedene Arten erfolgen. Zum einen kann eine Komponente ein Datum an eine andere Komponente liefern. Zum anderen kann eine Komponente selbst als Eingabe für eine andere Komponente dienen, wenn sie einen benötigten Service bereitstellt.

STOR-Services können aus verschiedenen Sichten betrachtet und dementsprechend in unterschiedliche Kategorien eingeordnet werden. Im Kontext dieser Arbeit gibt es zwei wichtige Datenarten: Bilder und Modelle. Für beide Datenarten gibt es zwei Hauptaktivitäten, das Management und die Verarbeitung. Komponenten für das Management von Daten sind eher technisch orientiert und bieten deren Import, Export, Konvertierung oder Betrachtung. Sie ermöglichen die Verarbeitung der Daten, verändern deren Bedeutung selbst jedoch nicht. Komponenten für die Verarbeitung, d. h. die Detektion und Extraktion von Informationen in/aus Bildern, fügen ihre Ergebnisse stets in ein Modell ein (siehe Abschnitt 3.4.3) und liefern dann eine Sammlung der neu erstellten/veränderten Modell-Entitäten zurück, über die indirekt auf das gesamte Modell zugegriffen werden kann.

Alle in dieser Arbeit entwickelten Verfahren zur Extraktion von Informationen aus dem Bild, beispielsweise Dachkanten- oder Fenstererkenner, Verfahren zur Manipulation des Modells, beispielsweise die Verschiebung im Modellkoordinatensystem, oder Poseschätzverfahren wurden nach den Regeln des Komponentenkonzept entwickelt und integriert.

3.7 Diskussion

Zur Modellierung des Gebäudeschemas werden Ansätze aus der Modellierung im urbanen Umfeld, im speziellen CityGML, und Ansätze zur Verarbeitung von Wissen, hier semantische Netze, miteinander kombiniert. Es entstanden in dieser Arbeit Objekt-Modell-Schemata für die Anwendungsfälle Dominosteine, Pokerkarten und Gebäude. Darüber hinaus wurde das Wissen aus der Computervision, Bildverarbeitung und der Computergrafik in das Referenzschema integriert. Vor allem aber wurden alle Modelle mit Fokus auf die Anforderungen des Erkennungsprozesses konzipiert.

Alle Modelle tragen neben deklarativem Wissen auch prozedurales Wissen in sich. Dieses prozedurale Wissen ist in dem leistungsstarken Komponentenkonzept verankert und die entsprechenden Modellelemente wissen selbst, welche Komponenten sie zu nutzen haben.

Für den Erkennungsprozess werden Modelle (Objektmodelle) benötigt, die basierend auf den Objekt-Modell-Schemata spezifische Objekte beschreiben, wie beispielsweise ein konkretes Gebäude. Da gerade für Gebäude der manuelle Modellie-

rungsaufwand immens wäre, wurde ein eigenes Modellierungswerkzeug (**sage-sb**) zur Modellgenerierung entwickelt (siehe Kapitel 4).

Kapitel 4

Modellgenerierung von Gebäuden

Eine Herausforderung dieser Arbeit ist es, ein System zu entwickeln, das unabhängig von der Anwendungsdomäne unter Verwendung von anwendungsspezifischen Modellen konform zum Referenzschema die Pose bestimmt. In dieser Arbeit kommt dabei das Gebäudeschema und entsprechende Modelle zum Einsatz.

Das Verfahren *Semiautomatic generation of semantic building models from image series* (**sage-sb**) wurde entwickelt, um komfortabel Gebäudemodelle zu generieren und die Modelle in verschiedenen Ausgabeformaten zur Verfügung stellen zu können. Es werden für die Poseschätzung verschiedene Gebäudemodelle generiert. Ein Gebäudemodell dient als Eingabe für **knoPoE** (Abbildung 4.1).

Es existieren im Allgemeinen drei Arten von Systemen, um Gebäudemodelle zu generieren (siehe auch Kapitel 1.3.1). Zum einen gibt es komplett *automatische Systeme*, die aber in der Regel Einschränkungen bezüglich der Art und Form der zu modellierenden Gebäude setzen müssen. Sie erzielen gewöhnlich nicht die Genauigkeit, die für Erkennungsprozesse nötig ist. Zum anderen existieren komplett *manuelle Systeme*, die präzise und detailreiche Modelle erzeugen, allerdings eine zeitaufwändige Erstellung benötigen. Außerdem existieren halbautomatische Ansätze. Da die Erstellung von präzisen Modellen mit Annotationen automatisch noch nicht möglich ist, wurde eine *halbautomatische Erzeugung* von Modellen gewählt. Im gewählten Verfahren werden City Geography Markup Language (CityGML) Modelle erzeugt (siehe Kapitel 3.2.3), die dann in das Gebäudemodell, konform zum in Kapitel 3.4.2 beschriebenen Schema, gewandelt werden, welche sich als Modelle für die anwendungsunabhängige Kontrollstrategie zur Extraktion der Pose bei Gebäudeaufnahmen (**knoPoE**) eignen (siehe Kapitel 6).

In dieser Arbeit wird von einer perspektivischen Projektion mittels Lochkameramodell für die Modellrekonstruktion ausgegangen (siehe Kapitel 3.1). Die in diesem Kapitel verwendeten Definitionen und mathematischen Beschreibungen sind angelehnt an die Beschreibungen von Harley und Zisserman [HZ04] und Decker

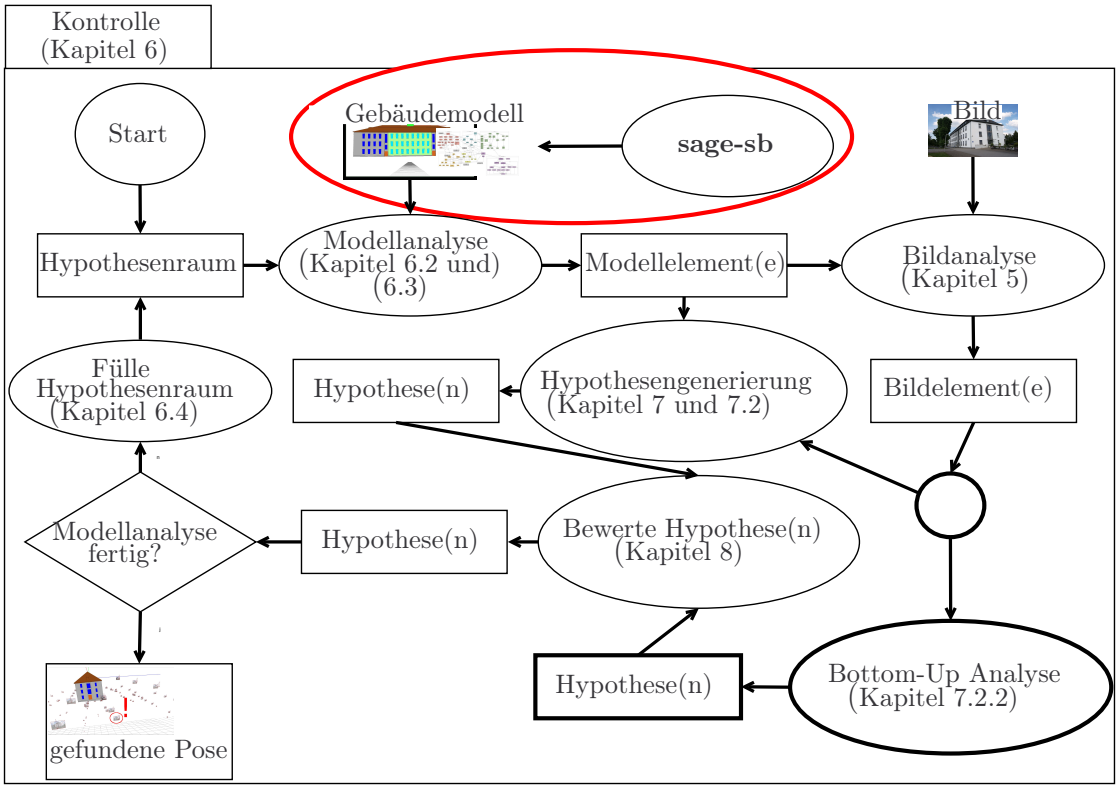


Abbildung 4.1: Das Verfahren **sage-sb** wurde entwickelt, um das Gebäudemodell komfortabel zu generieren (siehe rote Ellipse). Ein Gebäudemodell dient als Eingabe für **knoPoE**.

[Dec12]. Das vorgestellte Verfahren **sage-sb** ist in einem eigenständigen Programm realisiert [WDP11] und unter der MIT Lizenz[1] veröffentlicht[2].

4.1 Modellierung aus Bildfolgen

Das Verfahren ist in der Lage 3-D-Modelle (=geometrische Modelle), ähnlich zu *CAD-Modellen*, zu erzeugen und mit sinnvollen Annotationen zu versehen. Abbildung 4.2 gibt einen Überblick über das Verfahren, das fünf Phasen durchläuft:

(i) die Bildaufnahme und Entzerrung (siehe Abschnitt 4.2);
(ii) die Auswahl von bedeutsamen Eckpunkten, die in mindestens zwei Bildern sichtbar sind (siehe Abschnitt 4.3);
(iii) die Erzeugung der 3-D-Struktur und der Kameraposen unter Verwendung von "Struktur aus Bewegungstechniken" (siehe Abschnitt 4.4);

[1] `http://opensource.org/licenses/MIT`
[2] `http://sourceforge.net/projects/sagesb/`

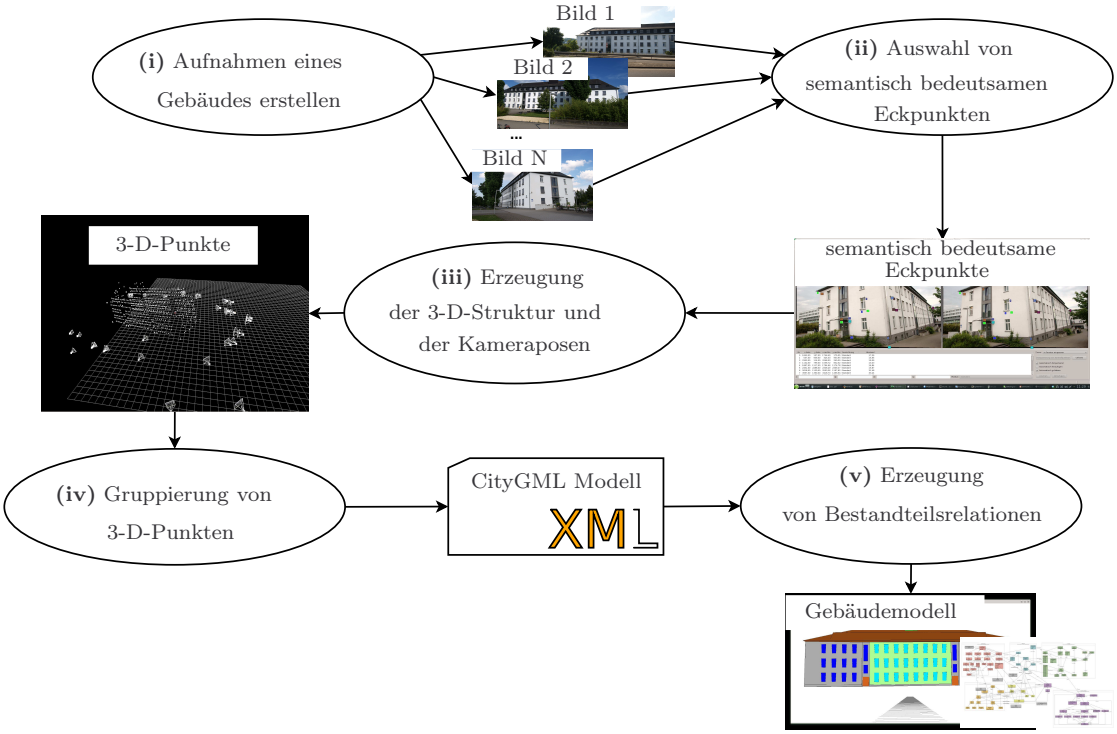


Abbildung 4.2: Datenflussdiagramm des Vorgehens zur Erzeugung von Gebäudemodellen.

(iv) die Gruppierung von 3-D-Punkten, die eine Grund-, Fassaden-, Dach-, oder Tür- bzw. Fensterfläche bilden (siehe Abschnitt 4.5)

(v) und Generierung des in dieser Arbeit verwendeten Gebäudemodells inklusive der Bestimmung von Enthaltenseinsbeziehung zur sofortigen Erzeugung (siehe Abschnitt 4.6) von Bestandteilsrelationen der annotierten 3-D-Polygone.

Der lizenzpflichtige *Autodesk ImageModeler*[3] geht einen sehr ähnlichen Weg wie der im Folgenden beschriebene Ansatz. Der ImageModeler wandelt 2-D-Bildserien in ein 3-D-Modell um und benötigt die gleichen Schritte wie das in dieser Arbeit verwendete Verfahren: *Entzerrung*, *Modellierung* und *Texturierung*. Dabei bietet er Funktionen zur Modellierung spezieller geometrischer Objekte wie beispielsweise Säulen. Trotz des größeren Funktionsumfanges ist der Autodesk ImageModeler nicht für unseren Anwendungsfall geeignet, da weder Bestandteilsrelationen noch Semantik abgebildet werden können.

[3] `http://usa.autodesk.com/adsk/servlet/pc/index?id=11390028&siteID=123112`

4.2 Bildaufnahme und Entzerrung

Zuerst müssen sich überlappende Aufnahmen des Gebäudes, das in 3-D rekonstruiert werden soll, gemacht werden. Bevor diese Aufnahmen zur Modellerstellung genutzt werden können, müssen diese entzerrt werden (siehe (i) in Abbildung 4.2). Dafür werden die intrinsischen Kameraparameter bestimmt (siehe Kapitel 3.1). Unter Verwendung der radialen und tangentialen Verzerrungskoeffizienten werden die Verzerrungen resultierend aus der Kameralinse beseitigt.

(a) Original (b) Entzerrt (c) Differenz

Abbildung 4.3: (a) zeigt das Originalbild, (b) zeigt das entzerrte Bild und (c) macht als Differenzbild der beiden Bilder deutlich, dass eine Entzerrung die Qualität verbessert.

4.3 Auswahl von Eckpunkten

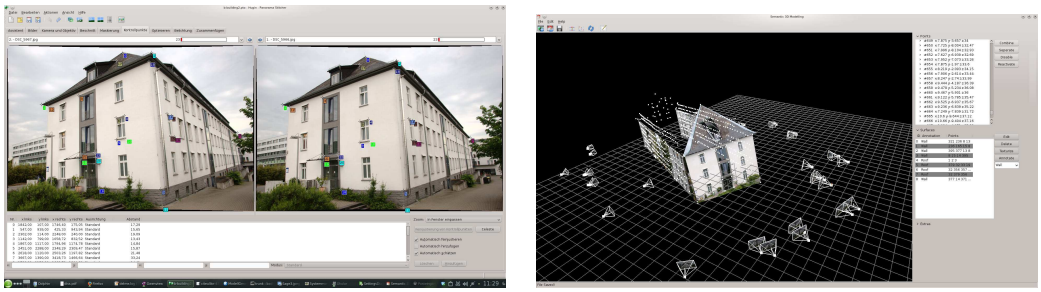
Die Grundlage für diesen Ansatz bildet *Hugin*[4], ein Werkzeug zur Erzeugung von Panoramas aus überlappenden Aufnahmen, da es komfortable Visualisierungs- und Editierungsfunktionen zur Erzeugung von Korrespondenzen bietet. Nach der Auswahl eines Punktes im Bild hilft der enthaltene subpixelgenaue *SIFT* [Low04] basierte Mechanismus den entsprechenden Punkt im anderen Bild auszuwählen, was die Qualität der zu generierenden Modelle verbessert und die Annotationsdauer verkürzt.

In Hugin werden in den entzerrten Bildern die Punktkorrespondenzen annotiert, die wichtig für die Geometrie und Beschreibung des Gebäudes sind (siehe Abbildung 4.4(a) und (ii) in Abbildung 4.2).

4.4 Erzeugung der 3-D Struktur

Die Erzeugung der 3-D-Punkte erfolgt iterativ und lässt sich in vier Schritte unterteilen (siehe Abbildung 4.5, (iii) in Abbildung 4.2 und Pseudocode 4.1):

[4] `http://hugin.sourceforge.net/`



(a) Punktkorrespondenzen (b) Annotation

Abbildung 4.4: (a) Punktkorrespondenzen in Hugin mit automatischer Positionsverbesserung, die Subpixelgenauigkeit erzielt und (b) Annotation von Polygonen in 3-D mit dem hier vorgestellten System. Es werden die Kameraposen mit dem der Kamera zugehörigen Bild visualisiert.

(i) Im ersten Schritt wird zunächst aus 2-D-Punktkorrespondenzen mittels Acht-Punkte-Algorithmus eine initiale Pose berechnet [Har97].
(ii) Aus dieser werden 3-D-Punkte aus 2-D-Punktkorrespondenzen trianguliert [Har97].
(iii) Nun werden bildweise 2-D-3-D-Punktkorrespondenzen bestimmt und mittels Fiore-Algorithmus für jedes Bild/jede Kameraposition die entsprechende Pose berechnet [Fio01] (Anhang C). Aus diesen werden wieder die 3-D-Punkte mittels der 2-D-Punktkorrespondenzen trianguliert.
(iv) In einem Optimierungsschritt werden die 3-D-Punkte und die entsprechenden Posen so angepasst, dass der Rückprojektionsfehler minimal wird [LA09].

Im Folgenden werden diese Verfahren im Detail beschrieben.

Der *Acht-Punkte-Algorithmus* liefert die *Fundamentalmatrix* $\boldsymbol{F}$ und da die Punkte manuell ausgewählt wurden, muss keine Ausreißerbehandlung erfolgen.

Begriffsbestimmung 4.1 (Pseudocode)**.** *Der in dieser Arbeit verwendete Pseudocode beschreibt stets Prozeduren. Prozeduren besitzen stets einen Namen, in Großbuchstaben geschrieben, und Argumente, in Klammern beschrieben. Neu eingeführte Variablen werden mit Typ und Namen beschrieben. Der Typ einer Liste von Elementen wird zwischen den Zeichen* $<\ldots>$ *deklariert. Eine Zuordnung wird beschrieben mit* $x \leftarrow y$*, so dass* y x *zugeordnet wird. Der Aufruf einer externen Prozedur wird beschrieben wie eine Prozedur selbst. Dabei kann das Ergebnis einer Prozedur, wie oben beschrieben, mit* $x \leftarrow$*PROZEDUR zugeordnet werden. Ein Kommentar ist gekennzeichnet durch das Zeichen* $\triangleright$*.* ***for*** $\ldots$ ***do*** *beschreibt eine Schleife und* ***return*** *den Rückgabewert der Prozedur. Mit "Füge* x *zu* y *hinzu" wird beschrieben, dass dem Container* y *das Element* x *hinzugefügt wird. Entsprechend wird mit*

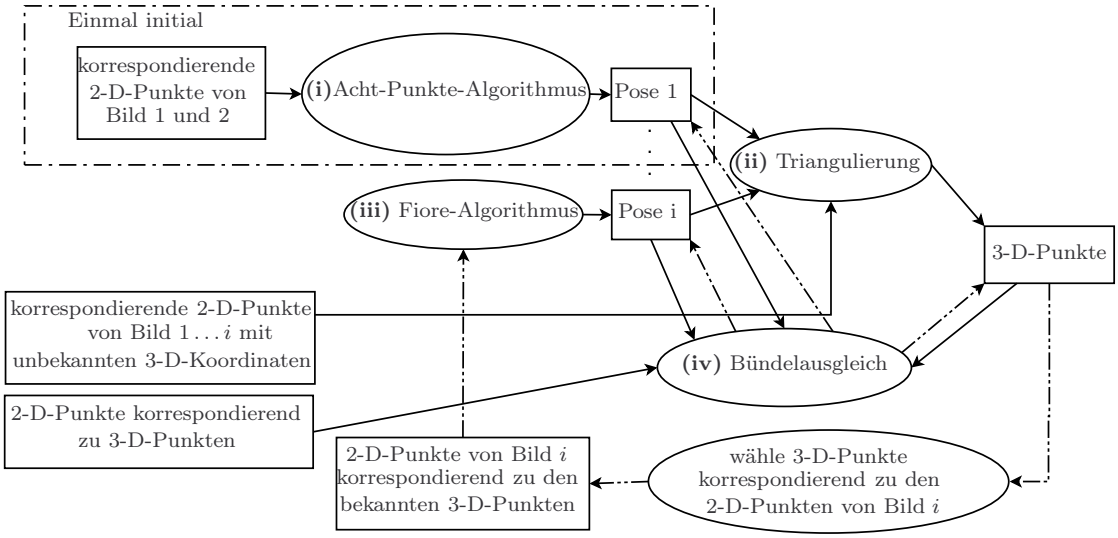


Abbildung 4.5: Datenflussdiagramm zur Erzeugung der 3-D-Punkte.

"Entnehme x aus y" beschrieben, dass Element dem Container y entnommen wird.

Pseudocode 4.1 Erzeugung der 3-D-Punkte

1: **procedure** GENERIERUNG3DPUNKTE(*Punktkorrespondenzen Bild* $I_1 \dots I_n$)
2: Pose $\upsilon_1 \leftarrow$ ACHT-PUNKTE-ALGORITHMUS(Korrespondenzen zwischen Bild I_1 und I_2) ▷ **(i)**
3: (3-D-Punkte) Liste$< \boldsymbol{p}^m > l \leftarrow$ Triangulierung **(ii)** mit Pose υ_1 und Korrespondenzen zwischen Bild I_1 und I_2
4: **for** Bild $I_i = I_2 : I_{n-1}$ **do**
5: Pose $\upsilon_i \leftarrow$ FIORE(Korrespondenzen zwischen I_i, I_{i+1} und Liste$<\boldsymbol{p}^m>l$) ▷ **(iii)**
6: **for** Bild $I_j = I_1 : I_{i-1}$ **do**
7: Liste$< \boldsymbol{p}^m > l \leftarrow$ TRIANGULIERUNG(Pose υ_i und Korrespondenzen zwischen Bild I_i und I_j) ▷ **(ii)**
8: BÜNDELAUSGLEICH(Liste$< \boldsymbol{p}^m > l$, Posen $\upsilon_1 \dots \upsilon_i$) ▷ **(iv)**
9: **return** Liste$< \boldsymbol{p}^m > l$

Unter Verwendung der zuvor für die verwendete Kamera bestimmte Kameramatrix $\boldsymbol{K}$ kann die *Essential Matrix* $\boldsymbol{E}$ aus der Fundamentalmatrix gewonnen werden:

$$\boldsymbol{E} = \boldsymbol{K}^T \boldsymbol{F} \boldsymbol{K}. \tag{4.1}$$

Die *Essential Matrix* wird dann in eine Rotation und eine Translation bis auf einen Skalierungsfaktor zerlegt:

$$\boldsymbol{E} = \lambda_{\boldsymbol{E}} \cdot [\boldsymbol{t}]_{\times} \boldsymbol{R}. \tag{4.2}$$

Die Rotationsmatrix $\boldsymbol{R}$ und der Translationsvektor $\boldsymbol{t}$ beschreiben die relative Orientierung und Position der zweiten Kamera zur ersten Kamera, wobei vorausgesetzt wird, dass für die Projektionsmatrix der ersten Kamera $\boldsymbol{P}_1 = \begin{pmatrix} \boldsymbol{I} & \boldsymbol{0} \end{pmatrix}$ gilt.

Die Faktorisierung von $\boldsymbol{E}$ ist nicht eindeutig, was zu einem Translationsvektor mit unbekanntem Vorzeichen und vier möglichen Rotationsmatrizen führt. Zwei Rotationsmatrizen können verworfen werden, da ihr Vorzeichen zu $det(\boldsymbol{R}) = -1$ führen würde. Für die vier verbleibenden möglichen Lösungen von $\boldsymbol{R}$ und $\boldsymbol{t}$ wird die Kombination gesucht, bei der ein einzelner triangulierter Punkt (siehe Gleichung (4.3)) vor der Kamera liegt [HZ03].

Die resultierende Projektionsmatrix der zweiten Kamera ist: $\boldsymbol{P}_2 = \begin{pmatrix} \boldsymbol{R} & \boldsymbol{t} \end{pmatrix}$. Mit den beiden bestimmten Projektionsmatrizen der Kameras können die 3-D-Koordinaten eines Modellpunktes $\tilde{\boldsymbol{p}}^{\mathrm{m}}$ von seinen Projektionen $\tilde{\boldsymbol{p}}^{\mathrm{i}}$ und $\tilde{\boldsymbol{q}}^{\mathrm{i}}$ mittels Triangulation mit minimalem quadratischen Rückprojektionsfehler bestimmt werden:

$$\tilde{\boldsymbol{p}}^{\mathrm{m}} = \underset{\tilde{\boldsymbol{p}}^{\mathrm{m}}}{\operatorname{argmin}} \left\| \tilde{\boldsymbol{\omega}_{\boldsymbol{1}}}^{\mathrm{i}} - \tilde{\boldsymbol{p}}^{\mathrm{i}} \right\|^2 + \left\| \tilde{\boldsymbol{\omega}_{\boldsymbol{2}}}^{\mathrm{i}} - \tilde{\boldsymbol{q}}^{\mathrm{i}} \right\|^2. \tag{4.3}$$

mit $\tilde{\boldsymbol{\omega}_{\boldsymbol{1}}}^{\mathrm{i}} = \boldsymbol{P}_1 \tilde{\boldsymbol{p}}^{\mathrm{m}}$ und $\tilde{\boldsymbol{\omega}_{\boldsymbol{2}}}^{\mathrm{i}} = \boldsymbol{P}_2 \tilde{\boldsymbol{p}}^{\mathrm{m}}$. Diese Gleichung kann bezüglich $\tilde{\boldsymbol{p}}^{\mathrm{m}}$ linearisiert und in einem einzigen Schritt mittels *Singulär Wert Zerlegung (SVD)* gelöst werden.

Das initiale Modell wird iterativ um ein Bild pro Schritt erweitert. Dabei wird die Kamerapose des neuen Bildes unter Verwendung der Korrespondenzen zwischen den 2-D-Punktmerkmalen im neuen Bild und den schon bekannten 3-D-Modellpunkten mittels des Ansatzes von *Fiore* [Fio01] bestimmt (siehe (iii) in Abbildung 4.5). Details zum Verfahren von Fiore finden sich in Anhang C.

Mit der aus dem Verfahren von Fiore berechneten Kamerapose können die neuen Modellpunkte aus den Korrespondenzen zwischen dem aktuellen Bild und aller anderen Bilder mit bereits bestimmter Kamerapose berechnet werden. In jeder Iteration wird der *Rückprojektionsfehler* minimiert, indem die Position der Modellpunkte im 3-D-Raum und die Position und Orientierung der Kameras mittels *Bündelausgleichs (BA)* [LA09] optimiert wird (siehe (iv) in Abbildung 4.5).

Der Bündelausgleich kann auf ein nichtlineares kleinste Quadrate-Problem reduziert werden, das typischerweise mit dem *Levenberg-Marquardt (LM)*-Algorithmus

[Kur04] gelöst wird. Der Bündelausgleich minimiert den Rückprojektionsfehler über alle Punkte und Kameraparameter:

$$\text{argmin}_{\boldsymbol{a}_j,\boldsymbol{b}_i} \sum_{i=1}^{n}\sum_{j=1}^{m} \boldsymbol{\nu}_{ij} \|Q(\boldsymbol{a}_j, \boldsymbol{b}_i) - \boldsymbol{x}_{ij}\| \tag{4.4}$$

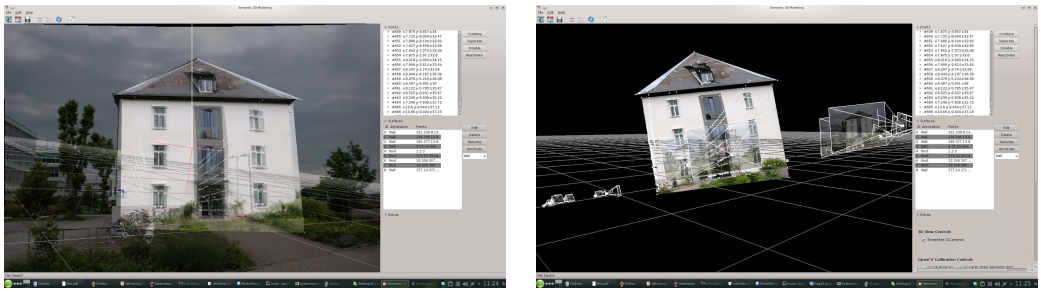
mit n als Anzahl 3-D-Punkte, m als Anzahl der Kameras, der Projektion $\boldsymbol{x}_{ij}$ des i-ten Punktes in das j-te Bild, dem Vektor $\boldsymbol{a}_j$ der Parameter der Kamera j, dem Vektor $\boldsymbol{b}_i$ der Parameter des Punktes i. Hierbei berechnet $Q(\boldsymbol{a}_j, \boldsymbol{b}_i)$ die Projektion des Punktes i in das Bild j und $\boldsymbol{\nu}_{ij}$ ist immer dann 1, wenn der Punkt i im Bild j sichtbar ist, und ansonsten ist $\boldsymbol{\nu}_{ij}$ 0. Da die Matrizen in großen Teilen nicht besetzt sind, wird ein speziell für dünn besetzte Matrizen entwickelter LM-Algorithmus verwendet, der die Berechnungszeit deutlich verringert (siehe [LA09]).

Das Ergebnis des Verfahrens ist eine Menge von 3-D-Punkten, die die geometrische Struktur des Gebäudes beschreibt.

4.5 Gruppierung

Um ein 3-D-Modell, angereichert mit Annotationen, zu erzeugen, werden die 3-D-Punkte beliebiger Polygone ausgewählt und mit Annotationen wie Boden, Fassade, Dach, Tür oder Fenster gekennzeichnet (siehe (iv) in Abbildung 4.2). Es werden alle Informationen verknüpft und gespeichert, sodass zu einer Annotation das Polygon, die 3-D-Punkte, die 2-D-Punktkorrespondenzen, die entsprechenden Bilder und das entsprechende Label verfügbar sind (siehe Abbildung 4.7). Die Annotationslabel entsprechen der Semantik des CityGML-Standards und des STOR-Referenzschemas. Es ist allerdings auch möglich, eigene Label zu definieren, die dann evtl. im Standard bzw. im Referenzschema (siehe Kapitel 2.5) nicht enthalten sind. Annotationen, die keine Entsprechung in CityGML haben, werden in CityGML dann als unspezifiziertes Stadtobjekt behandelt und später im STOR-Modell konform zum Schema als generisches Objekt geführt. Für jede annotiertes Polygon wählt das System automatisch eine Aufnahme, schneidet die Fläche auf Basis der 2-D-Punktkorrespondenzen aus, berechnet die Homographie und bildet die Textur auf der Oberfläche ab (siehe Abbildung 4.6(b) und 4.7).

Nach der Annotationsphase sind wir in der Lage, die Annotationen, die 3-D-Geometrie und die entsprechenden Texturinformationen im Austauschformat CityGML zu exportieren. Es besteht die Option, dieses System mit einem Updatethread direkt an Hugin zu koppeln, sodass es zu jedem Zeitpunkt und bei jedem Arbeitsschritt möglich ist, Änderungen vorzunehmen, ohne Informationen zu verlieren.



(a) (b)

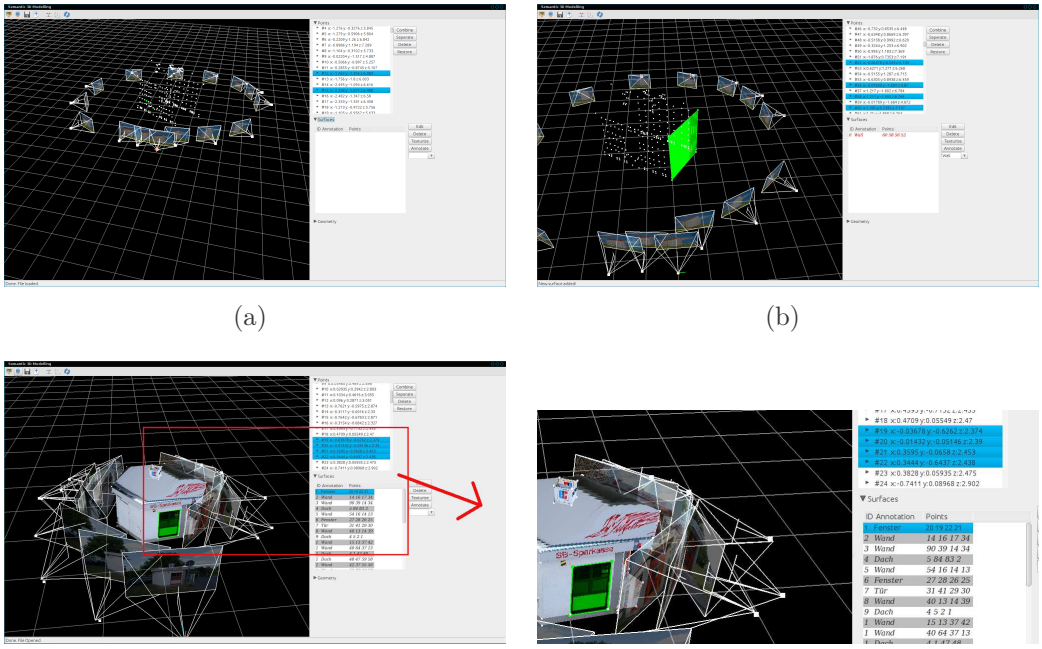
Abbildung 4.6: Es ist möglich die Kameras einzeln auszuwählen (a) und aus der Position der Kamera durch das entsprechende Bild zu schauen, um so die Annotation zu vereinfachen und die Korrektheit des Modells zu prüfen. Die bereits annotierten Polygone werden automatisch mit einer Textur aus einer dazu passenden Aufnahme versehen (b).

4.6 Generierung des TGraphen-Gebäudemodells

Das exportierte CityGML-Modell wird dann im letzten Schritt (siehe (v) in Abbildung 4.2) in ein TGraph-Modell konvertiert. Unter Verwendung der Ansätze von Falkowski et al. [FED+09] [FE11] können die Modelle um Informationen über die Beziehungen zwischen Geometrie und annotierten Bezeichnern angereichert werden. Hiermit ist es möglich, das Modell nach COLLAborative Design Activity(COLLADA)[5], Object-Oriented Graphics Rendering Engine (Ogre)[6] und in das urbane STOR-Modell (siehe Abschnitt 3.4.2) zu exportieren. Diese Ansätze sind vollautomatisch, müssen aber für jedes Modell neu parametrisiert werden bis sie für den speziellen Fall konsistente Modelle liefern. Dies liegt unter anderem daran, dass die Skalierung für jedes Modell unterschiedlich ist und kein metrisches System vorliegt.

Um den Aufwand bei der Parametrisierung und somit insgesamt Aufwand im Vergleich zum Verfahren von Falkowski zu sparen, wurde ein halbautomatisches Verfahren entwickelt, das im Folgenden beschrieben wird. Das Modell wird mittels XML-Parser in ein primitives STOR-Gebäudemodell, d. h. in ein Modell ohne jede Bestandteilsrelationen, konvertiert. Es enthält alle vorher modellierten (annotierten) Bezeichner und geometrischen Konzepte und ist noch ein reiner Datenspeicher. Auf Basis einer bekannten Strecke im Modell wird das Modell in ein metrisches System überführt. Der Benutzer kann zwischen einem Wand-, Dach-, Fenster- oder Türpolygon wählen, dort eine Strecke markieren und dann die entsprechende Länge in Metern angeben.

[5] `http://www.khronos.org/collada/`
[6] `http://www.ogre3d.org/`



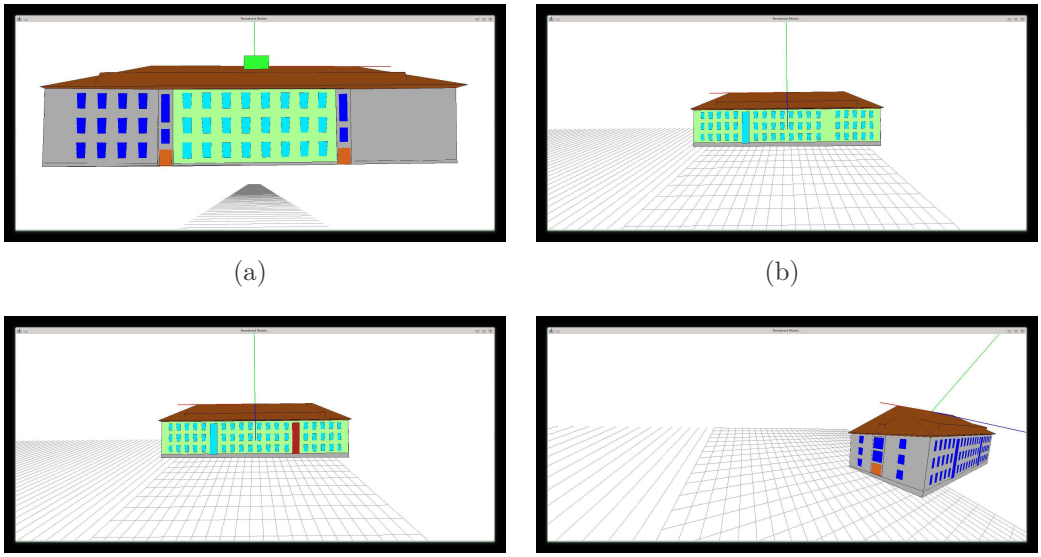
(a) (b)

(c) (d)

Abbildung 4.7: Punktkorrespondenzen werden in Hugin mit automatischer subpixelgenauer Positionsverfeinerung erstellt, dann geladen und in 3-D-Punkte gewandelt (a), diese 3-D-Punkte werden zu Flächen gruppiert und entsprechend annotiert (b). Es werden alle Informationen verknüpft und gespeichert, sodass zu einer Annotation die 3-D-Punkte, die 2-D-Punktkorrespondenzen, die entsprechenden Bilder und das entsprechende Label verfügbar sind (c) (d). Die Annotationen entsprechen der Semantik des CityGML-Standards und des STOR-Referenzschemas.

Nun werden die Bestandteilsrelationen erzeugt; dafür werden die entsprechenden Relationen nach und nach visualisiert und dem Benutzer die Möglichkeit gegeben ändernd einzugreifen. Die visualisierten Vorschläge sind in den meisten Fällen korrekt (Abbildung 4.8(a)) und es ist keine Änderung nötig; falls doch, wird jeweils dem Benutzer die Option zum Hinzufügen oder Löschen angeboten (Abbildung 4.8(b) & 4.8(c)). Das System schlägt je nach gewählter Option Kandidaten für das Hinzufügen oder Löschen nach und nach vor. Die Vorschläge sind geordnet. Als Kriterium für Ordnung und die automatische Auswahl dient der minimale Abstand $\mathrm{d}(\boldsymbol{p}_i^{\mathrm{m}}, \mathbf{P})$ der Punkte $\boldsymbol{p}_i^{\mathrm{m}}$ des Polygons, das hinzugefügt bzw. gelöscht werden soll, zum Polygon $\mathbf{P}$, in das eingefügt bzw. aus dem gelöscht werden soll. Jedes Objekt mit $\mathrm{d}(\boldsymbol{p}_i^{\mathrm{m}}, \mathbf{P}) < 15$ cm wird als Kandidat automatisch hinzugefügt.

Da bei den so entstehenden Bestandteilsrelationen für jedes Bestandteilselement eine etwas andere Ebenengleichung vorliegt, können diese angeglichen werden. Der Benutzer kann jeweils wählen zwischen der Übernahme der Ebene des



(a) (b)

(c) (d)

Abbildung 4.8: In der Regel werden alle Elemente, die es zu inkludieren gilt, direkt erkannt (a). Fehlt dennoch ein Element wie in (b), kann der Benutzer weitere Elemente anfordern und diese dann entsprechend hinzufügen lassen oder auch nicht (c). Nach der Fertigstellung erhält man ein Modell mit akkurater Geometrie (d).

Elternelementes, linearer Regression aller Elemente, einer RANSAC-Ebenenschätzung oder keiner Änderung. Das Ergebnis ist dann ein TGraph-Gebäudemodell und somit ein geometrisch akkurates Modell (Abbildung 4.8(d)), dessen Bestandteile hierarchisch angeordnet sind und das konform zum Gebäudeschema ist.

4.7 Evaluation

Es wurde zur Evaluation ein *H0 Modellhaus* im Maßstab 1:87, bei dem alle Flächen ausgemessen wurden, verwendet. Aus den Messungen wurde als Referenz manuell ein 3-D-Modell des Modellhauses erstellt (siehe Abbildung 4.9). Dieses wird im Folgenden als Referenzmodell bezeichnet. Alternativ hätte auch die Möglichkeit bestanden, ein fiktives Haus zu modellieren und die Fotos zu rendern. Diese Möglichkeit wurde aber verworfen, da man so eventuelle Störeinflüsse nicht mit modellieren würde.

Zur Evaluation wurden 14 Aufnahmen von dem Modellhaus gemacht und insgesamt 388 Punktekorrespondenzen in Hugin selektiert. Abbildung 4.10 zeigt eine Teilmenge der Aufnahmen, die zur Erzeugung des 3-D-Modells verwendet wurden. Es wäre möglich, weniger Aufnahmen zu verwenden, um das Modell zu erzeugen.

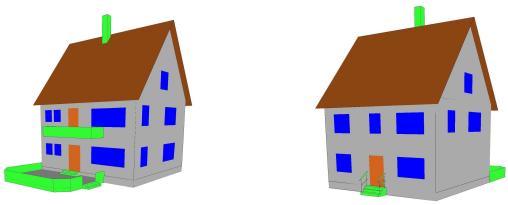


Abbildung 4.9: 3-D-Modell eines H0-Modellhauses im Maßstab 1:87 (siehe Abbildung 4.10). Die Farben stellen die unterschiedlichen Objekte wie Fenster, Tür usw. dar und sind nicht als Texturierung gedacht.

Verwendet man allerdings weniger als drei Bilder pro Gebäudeseite, wird die Auswahl von Punktkorrespondenzen schwierig.

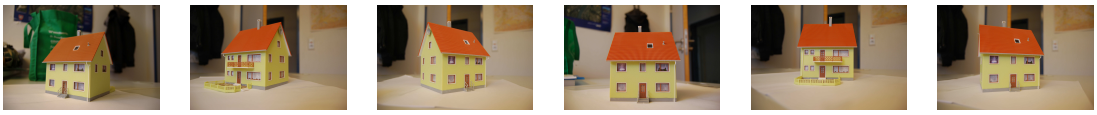


Abbildung 4.10: Bildreihe des H0-Modellhauses.

Zur Evaluation wird das erzeugte 3-D-Modell mit dem Referenzmodell verglichen. Dafür wird das Modellkoordinatensystem des erzeugten Modells in das Modellkoordinatensystem des Referenzmodells rotiert, verschoben und skaliert. Der Punkt-zu-Punkt-Fehler e ist dann:

$$e = \underset{\boldsymbol{R},\lambda,\boldsymbol{t}}{\operatorname{argmin}} \sum_{i}^{N} \|\boldsymbol{q}^{\mathrm{m}} - \lambda \boldsymbol{R}(\boldsymbol{p}^{\mathrm{m}} - \boldsymbol{t})\|. \tag{4.5}$$

Auf diese Weise bezeichnet e den Punkt-zu-Punkt-Fehler in cm, $\boldsymbol{q}^{\mathrm{m}}$ entspricht den 3-D-Punkten des Referenzmodells in Modellkoordinaten und $\boldsymbol{p}^{\mathrm{m}}$ entspricht den erzeugten 3-D-Punkten in Modellkoordinaten.

Der durchschnittliche Fehler $\overline{e} = \frac{1}{N}e$ pro Punkt beträgt 0.1269 cm. Somit liegt der Fehler in einem akzeptablen Bereich, da sich die Genauigkeit der Ausmessungen des Referenzmodells im Millimeterbereich bewegen und Abweichungen im Submillimeterbereich möglich sind.

Die Evaluation der Kameraposen erfolgte rein quantitativ. Dabei wurde händisch geprüft, ob die Kamerapose korrekt ist, und festgestellt, dass alle Kameraposen für das Modellhaus korrekt bestimmt wurden. Dies verwundert nicht, da es im Regelfall zu keinen Ausreißern bei den Punktkorrespondenzen kommt. Allerdings ist zu beachten, dass die Auswahl der Korrespondenzen für die ersten beiden Bilder entscheidend für die erste Kamerapose ist. Da es bei den ersten beiden Bildern

zu keiner Optimierung kommt, muss die Auswahl so exakt wie möglich geschehen. Je mehr Bilder und je mehr Korrespondenzen hinzukommen, umso höher wird die Genauigkeit der 3-D-Punkte und Kameraposen.

Eigentlich wurde das Verfahren zur Erzeugung von Gebäudemodellen entwickelt, dennoch ist es möglich, es auch auf anderen Gebieten zu verwenden, wie beispielsweise der *3-D-Innenraumrekonstruktion* (siehe Abbildung 4.11). Eine vorstellbare Anwendung wäre die 3-D-Visualisierung der Raumaufteilung mit Annotationen für Wände, Steckdosen, Lichtschalter, Fenster und Türen zur Planung der Einrichtung.

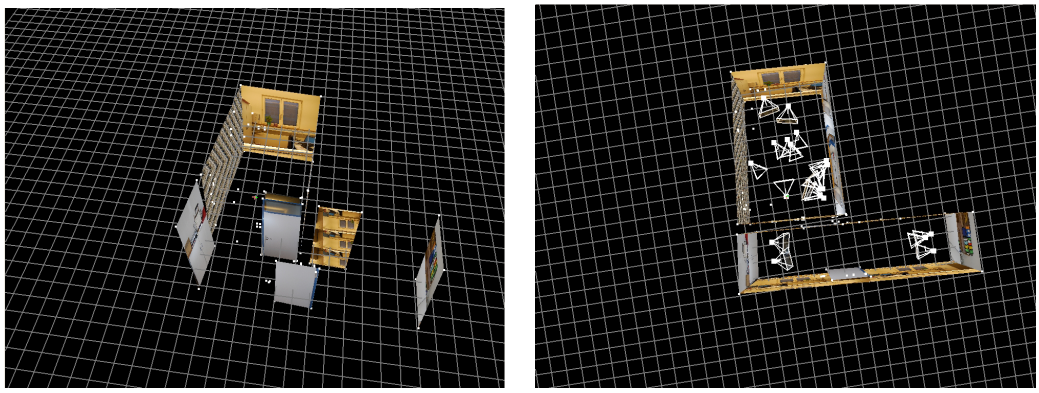


Abbildung 4.11: Raumplan eines Büros mit Vorflur.

Die automatische Auswahl der Bilder zur Texturierung funktioniert im Bereich der Gebäudemodellierung sehr gut, da meist große Teile des Gebäudes auf den Aufnahmen sichtbar sind und weil Gebäude zumeist nicht besonders verschachtelt sind. Allerdings ist in Bereichen wie der Innenraummodellierung eine komplexere Texturauswahl nötig, da oft nur kleine Teile eines Raums auf den Bildern zu sehen sind.

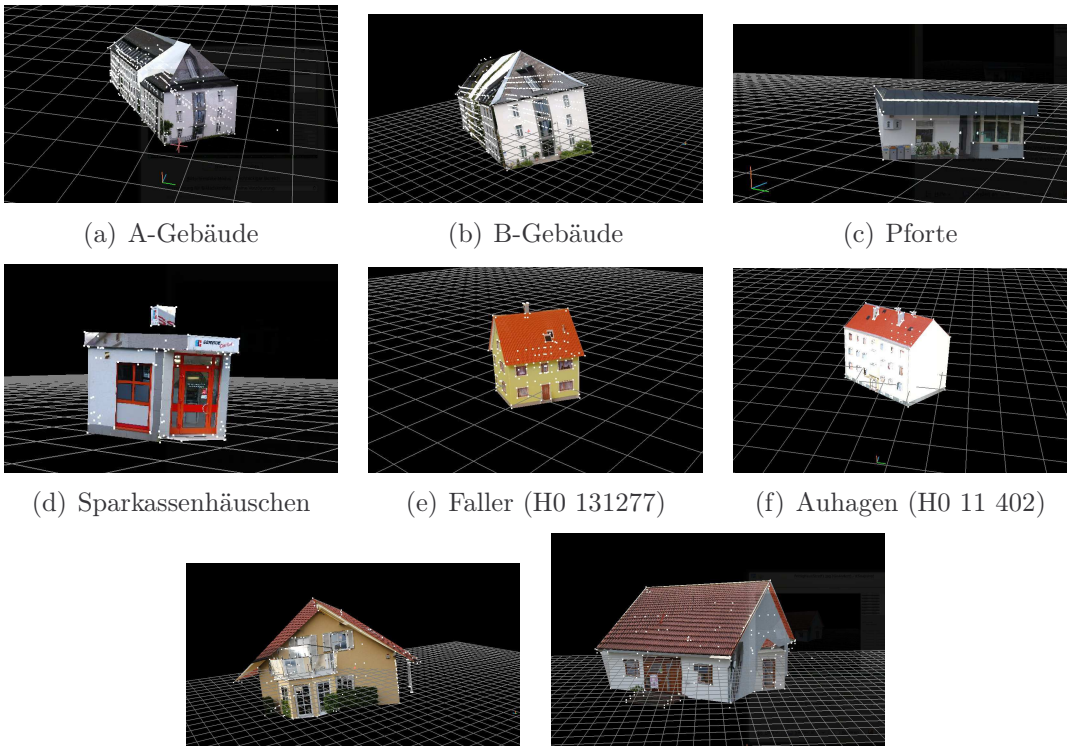
Es existiert eine Vielzahl von Verfahren, die sich schon mit diesem Thema beschäftigen. Da das Feld der Innenraummodellierung nicht Teil der Arbeit ist, wird hier nur ein kurzer Einblick gegeben. Oeasau et al. [OLA13] beschreiben ein Verfahren, das aus 3-D-Punktewolken die Wände, Böden und Decken extrahiert, woraus geschlossene Modelle von mehretagigen Gebäuden und aus komplexen Szenen gewonnen werden können. Dazu werden aber 3-D-Punktewolken benötigt, die meist aus Laserscannern erzeugt werden, welche sehr exakte Daten in großem Umfang liefern, allerdings nicht günstig in der Anschaffung sind. Ein interessanter Ansatz aus dem gleichem Gebiet kommt von Xiong und Huber [XH10], die nicht nur polygonale Modelle erzeugen, sondern diese zusätzlich mit Semantik anreichern.

Da Laserscanner, gerade im privaten Umfeld, nicht leicht verfügbar sind, könnte man diesen kamerabasierten Ansatz alternativ einsetzen, ohne große Ausgaben zu

befürchten. Allerdings ist die Genauigkeit eines hochwertigen Laserscanners besser als die Genauigkeit dieses Verfahrens.

4.8 Diskussion

Das vorgestellte System erstellt aus einer Serie von Aufnahmen 3-D-Gebäudemodelle. Diese Gebäudemodelle besitzen eine akkuraten Geometrie angereichert um Polygone, welche durch die Annotation eines Bezeichners (eine begrenzte Zahle Bezeichner sind definiert), eine Bedeutung besitzen. Zudem werden die annotierten Bezeichner durch explizite Bestandteilsrelationen in Verbindung gesetzt. Dabei werden bekannte "Struktur aus Bewegungstechniken" verwendet. Außerdem ist es möglich, jede Art von Objekt, bei dem bedeutsame Punkte ausgewählt werden können, zu modellieren. Der Aufwand der Erstellung eines 3-D-Modells und die Interaktion mit dem System konnte auf ein Minimum reduziert werden. Abbildung 4.12 gibt einen Überblick über einige der bisher erstellten Gebäudemodelle.



(a) A-Gebäude (b) B-Gebäude (c) Pforte

(d) Sparkassenhäuschen (e) Faller (H0 131277) (f) Auhagen (H0 11 402)

(g) Fertighaus Streif (h) Fertighaus Massa

Abbildung 4.12: Übersicht der bisher erstellten Modelle.

Der durchschnittliche Punkt-zu-Punkt-Fehler liegt bei 1,3 mm für die getesteten Modelle; das entspricht 11 cm in der Realität bei einem Maßstab von 1:87. Je mehr Bilder und Punktkorrespondenzen verwendet werden, desto besser wird die Qualität der berechneten Punkte und Posen. Wird dabei das gesamte Gebäude modelliert kommt, es idealerweise zu einem "Schleifenschluss", sodass das Optimierungsverfahren die Qualität des modellierten Gebäudes noch mal steigern kann.

Dieses Verfahren besitzt eine ergonomische Benutzeroberfläche, die eine einfache Mensch-Maschine-Interaktion ermöglicht. Dabei unterstützt die Thread-gekoppelte Anbindung an das Panoramatool Hugin die Entwicklung durch einfache und schnelle Änderungsmöglichkeiten auf beiden Seiten. Darüber hinaus ist es auch möglich das System zur Innenraummodellierung zu verwenden.

Für eine automatische Generierung von Gebäudemodellen werden Erkennungsverfahren zur automatischen Lokalisation der Bezeichner im Bild benötigt. Auch die modellbasierte Poseschätzung benötigt solche Verfahren. Daher werden im nächsten Kapitel die in dieser Arbeit verwendeten Verfahren beschrieben.

Kapitel 5

Middle-Level-Merkmale

Definition 5.1 (Middle-Level-Merkmal)**.** *"Ein Middle-Level-Merkmal*[ab] *ist ein detektierbares Objekt, das aus* Low-Level-Features *zusammengesetzt ist. Ein Beispiel für ein Middle-Level-Feature wäre "Fenster", welches mit Hilfe der Low-Level-Features "Farbe des Fensterrahmens" und "Geraden die das Fensterkreuz bilden" detektiert wird."*

[a] Quelle: Glossar des Partner-DFG-Projekt Pose-Tracking mittels markanter Merkmale
[b] `www.uni-koblenz-landau.de/de/koblenz/fb4/icv/agpriese/research/PosE`

Definition 5.2 (Low-Level-Merkmal)**.** *"Ein Low-Level-Merkmal*[ab] *ist ein elementares, atomares Merkmal eines Objektes. Beispiele für Low-Level-Features sind: Kanten, Farbe und Verhältnis Fläche zu Umfang."*

[a] Quelle: Glossar des Partner-DFG-Projekt Pose-Tracking mittels markanter Merkmale
[b] `www.uni-koblenz-landau.de/de/koblenz/fb4/icv/agpriese/research/PosE`

Die Kontrolle (Kapitel 6) steuert die Modellanalyse, dabei initiieren die Modelle (Kapitel 3) über ihre Bestandteile die Bildanalyse. Diese Kapitel beschreibt Verfahren, die bei der Bildanalyse zum Einsatz kommen, wobei die Ergebnisse der Bildanalyse, wie in Abbildung 5.1 angedeutet, wiederum die Modellanalyse beeinflussen. Die bei der Bildanalyse extrahierten und im Analysemodell gespeicherten Informationen werden mit den Elementen der Modelle in Verbindung gesetzt, um so die Kamerapose (Kapitel 7.2) zu ermitteln. Ein *Analysemodell* beinhaltet alle bei der Bildanalyse extrahierten Informationen und deren Beziehung zueinander. Es ist naheliegend, dass bei der Bildanalyse mit unvollständigen, unpräzisen und unsicheren Daten umgegangen werden muss. In diesem Kapitel werden die Middle-Level-Merkmale beschrieben, die für das Verfahren der Poseschätzung **knoPoE** genutzt und teilweise speziell entwickelt wurden.

Bei *Middle-Level-Merkmalen* handelt es sich um Komposita mehrerer atomarer Komponenten (vgl. Kapitel 2.3). Diese Merkmale kapseln das Wissen der zur

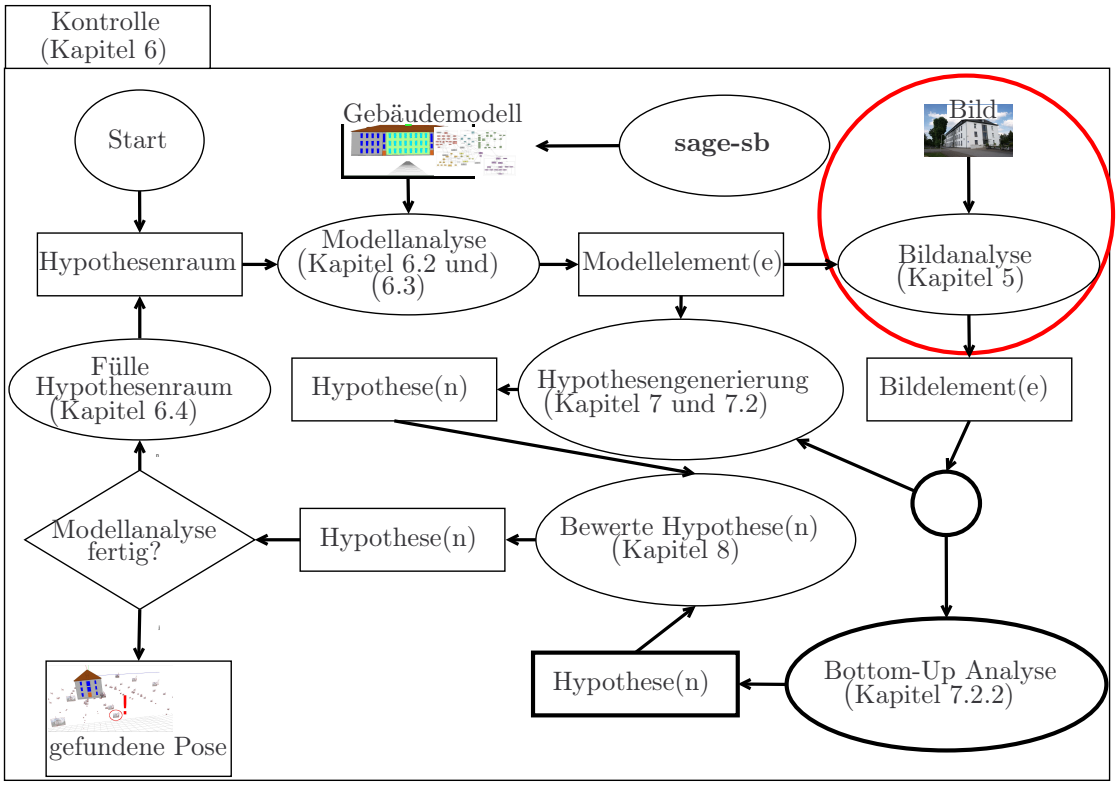


Abbildung 5.1: Die Kontrolle (Kapitel 6) steuert die Modellanalyse, dabei initiieren die Modelle (Kapitel 3) über ihre Bestandteile die Bildanalyse (siehe rote Ellipse). Ein Analysemodell beinhaltet alle bei der Bildanalyse extrahierten Informationen und deren Beziehung zueinander. Die Ergebnisse der Bildanalyse beeinflussen wiederum die Modellanalyse.

Extraktion nötigen Schritte in einer Komponente. Jedes dieser Verfahren beinhaltet (implizites) Wissen über die zur Extraktion nötigen Bildverarbeitungsschritte. Keines dieser Verfahren ist für sich selbst genommen in der Lage, die Aufgabe der modellbasierten Poseschätzung zu bewältigen. Zum derzeitigen Stand der Technik existieren viele Verfahren, die solche spezialisierten Aufgaben adäquat lösen können. Im Rahmen dieser Arbeit sind dennoch eigene Implementationen entstanden, die sich dank des Komponentenkonzeptes (Kapitel 3.6) und der Interface-Architektur (Kapitel 3.4) leicht durch andere Verfahren ersetzen lassen. Für diese Arbeit war die Möglichkeit zur Anwendbarkeit des Verfahrens in verschiedenen Anwendungsbereichen von höherer Bedeutung als die Auswahl bzw. Entwicklung des bestmöglichen Verfahrens.

Es existiert eine Vielzahl von Verfahren, die sich mit Fenster-, Fassaden- und Gebäudeerkennung beschäftigen. Ein Überblick gibt Kapitel 1.3.2.

5.1 Fluchtpunktbestimmung

Das Datenflussdiagramm in Abbildung 5.2 gibt einen Überblick über das von Schmitt entwickelte Verfahren [SP09b]. Es werden zwei bis drei Fluchtpunkte gesucht, sodass zwei oder drei Gruppen von detektierbaren Geraden gesucht werden, die in einen gemeinsamen Punkt münden. In architektonischen Umgebungen finden sich zumeist zwei bis drei Fluchtpunkte, ein vertikaler und im Horizontalen ein rechter und/oder linker Fluchtpunkt. Es wird davon ausgegangen, dass Bilder in urbanen Umgebungen in den meisten Fällen in Bodennähe aufgenommen werden (siehe Beschreibung der Fallstudie in Kapitel 2). Der linke Fluchtpunkt muss links und der rechte Fluchtpunkt rechts der Bildmitte liegen. Ein Fluchtpunkt muss nahe der Bildmitte liegen, falls ein weiterer horizontaler Fluchtpunkt existiert und dieser sich weit außerhalb des Bildes befindet. Daher wird der Suchraum für diesen Anwendungsfall auf zwei bis drei Fluchtpunkte (vgl. Kapitel 2.3) beschränkt. Die komplette Heuristik lässt sich in [SP09b] nachlesen.

Als Ausgangsbasis dienen mit dem *Canny-Algorithmus* gefundene Kanten, welche mittels eines Wildheitsdetektors [PPDS09, Seite 10-11] ausgedünnt werden. Dieser *Wildheitsdetektor* erkennt wilde Strukturen im Bild, welche auf nicht künstliche Objekte wie Baumkronen, Sträucher etc. hindeuten. Anschließend werden auf dem bearbeiteten Kantenbild die Geraden durch *Houghtransformation* detektiert.

Um Kandidaten für Fluchtpunkte aus den Geradenschnittpunkten zu bestimmen, sind nur Geraden relevant, welche eine ähnliche Ausrichtung haben und somit zu einem gemeinsamen Fluchtpunkt zeigen. Geraden mit ähnlicher Ausrichtung werden über ihren Winkel (Winkel der Hesseschen Geradengleichung) bestimmt. Dazu wird ein automatisches Clustering mittels *K-Harmonic-Means(KHM) Algorithmus* [ZHD01] verwendet. Es werden dabei zwei Durchläufe des KHM Algorithmus durchgeführt, erst mit zwei, dann mit drei Clustern. Der Durchgang, der geringere Intra-Cluster-Varianzen aufweist, wird als Ergebnis gewählt. Nach abgeschlossenem Clustering werden die Geradenschnittpunkte für jedes Paar von Geraden innerhalb einer Gruppe in einer Liste gespeichert.

Bei parallelen Geraden wird ein virtueller Schnittpunkt erzeugt, der entweder auf vertikal oder horizontal ausgerichteten Geraden beruht. Dieser virtuelle Schnittpunkt befindet sich außerhalb des Bildes mit einem auf Basis der verwendeten Geraden berechneten Abstand.

Das von Schmitt entwickelte *Automatic Grouping of Semantics (AGS)* Clustering liefert mögliche Fluchtpunkte, indem es Schnittpunkthäufungen findet. Über weitere geometrische Einschränkungen werden nun die endgültigen Fluchtpunkte aus der Kandidatenmenge bestimmt. So werden beispielsweise Fluchtpunktkandidaten, welche im oberen Drittel des Bildes liegen, verworfen. Fluchtpunkte solcher Art entstehen bei Vogelperspektiven oder bei Aufnahmen vom Dach eines hohen Gebäudes.

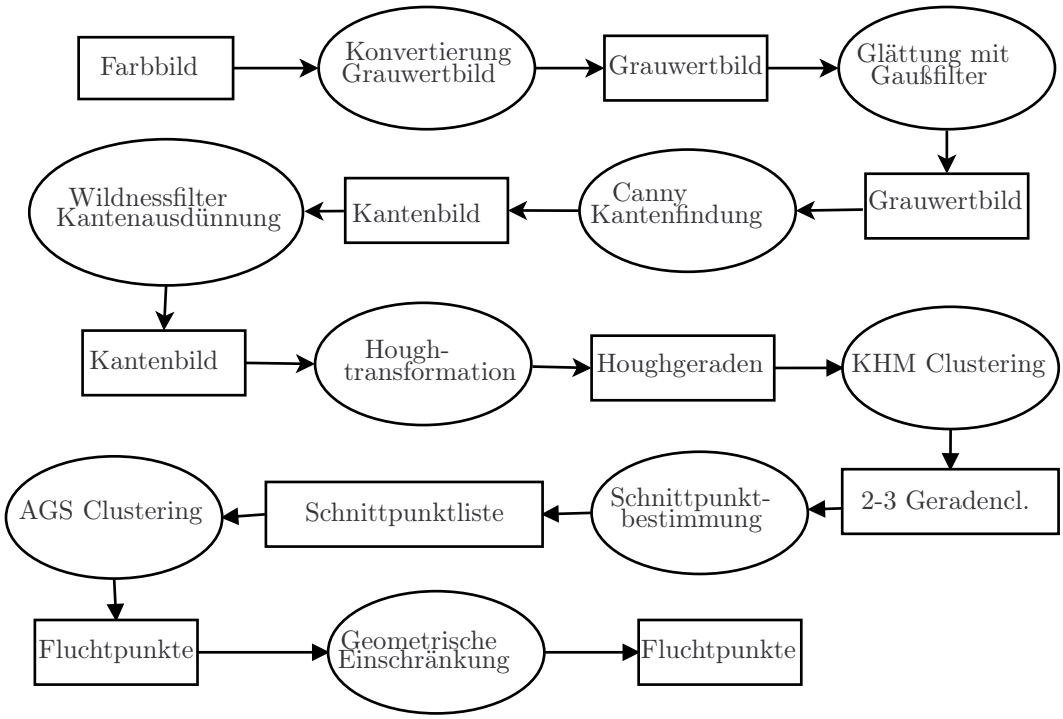


Abbildung 5.2: Datenflussdiagramm zur Fluchtpunktbestimmung von Schmitt [SP09b].

Das Verfahren ist Bestandteil der *Koblenz Image Processing Library*[1] *(KIPL/C++)* und wird mittels *Java Native Interface*[2] *(JNI)* als Komponente in das STOR-Komponentenkonzept lokal integriert.

5.2 Himmelserkennung

Das Verfahren von Schmitt [SP09a] detektiert den Himmel oberhalb von Gebäuden unabhängig von der Witterung. Bei der Gebäudeerkennung können so bereits zum Himmel gehörende Bildteile von der weiteren Untersuchung ausgeschlossen werden. Das Datenflussdiagramm in Abbildung 5.3 präsentiert den Ablauf des Verfahrens im Überblick.

Für den Himmeldetektor werden folgende Annahmen getroffen: Zum einen wird davon ausgegangen, dass die Eingabebilder waagerecht aufgenommen wurden und sich der Himmel im oberen Bereich des Bildes befindet. Weiterhin erscheint, abgesehen von Morgen- oder Abendröte, der Himmel in Blau- oder Grautönen. Rottöne

[1] `https://www.uni-koblenz-landau.de/de/koblenz/fb4/uebergreifend/er/software/imageprocessinglibraries/kipl]`

[2] `http://docs.oracle.com/javase/7/docs/technotes/guides/jni/`

werden nicht als charakteristische Himmelfarben betrachtet. Es werden daher Himmelfarben definiert, die ein bestimmtes Lichtspektrum abdecken, das von blau über grau bis weiß reicht. Zusätzlich wird davon ausgegangen, dass alle Regionen, die zum Himmel gehören, eine Verbindung zum oberen Bildrand besitzen.

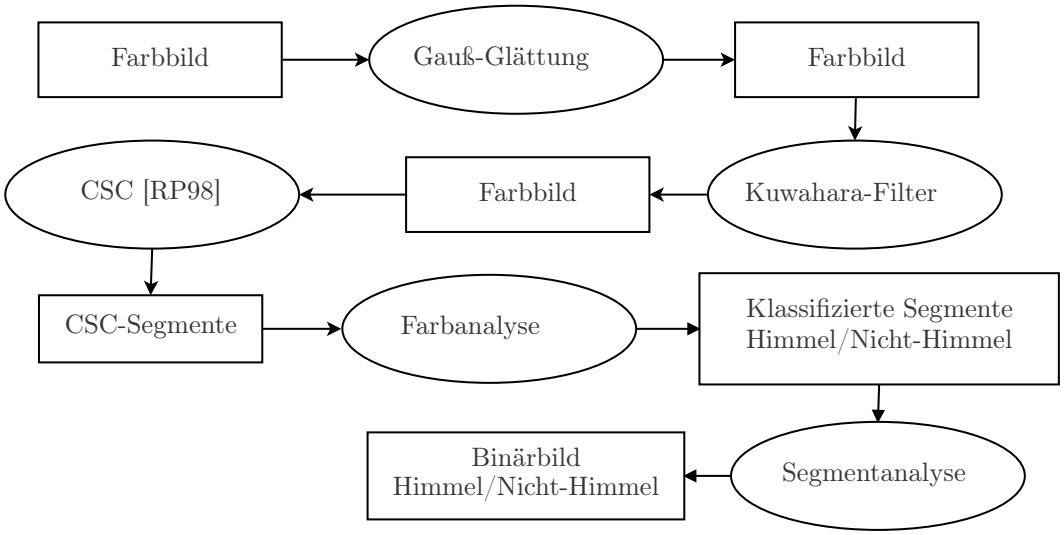


Abbildung 5.3: Datenflussdiagramm der Himmelserkennung.

Neben den charakteristischen Farben werden auch Formeigenschaften genutzt; so haben Wolken eine eher unregelmäßige Kontur und enthalten in der Kontur keine langen und zugleich geraden Kanten. Das Vorgehen gliedert sich in eine Segmentierung des Bildes und anschließende Analyse der Segmentform, -farbe und -position. Der Segmentierung geht eine Glättung mit dem *Kuwahara-Filter* voraus. Segmentiert wird mit dem *Color Structure Code*. Eine Farbanalyse der Segmente klassifiziert diese als "Himmel" oder "Nicht-Himmel". Außerdem wird die Form der Segmente untersucht, wobei eine unregelmäßige Kontur auf ein zum Himmel gehörendes Segment hinweist. Die endgültige Klassifikation des Bildes in Himmel und Nicht-Himmel beginnt mit Segmenten, die direkt an den oberen Bildrand grenzen. Besitzen diese Segmente Himmelfarben, werden sie dem Himmel zugeordnet. Alle weiteren Segmente müssen ebenfalls Himmelfarben aufweisen sowie eine unregelmäßige Grenze zu darüber liegenden Himmelssegmenten (also eine Verbindung zu bereits als Himmel klassifizierten Segmenten) haben. Manche Wolken werden jedoch fälschlicherweise als Nicht-Himmel klassifiziert. Deshalb wird nach Nicht-Himmel-Segmenten gesucht, welche Himmelfarben aufweisen und vollständig von Segmenten, die zum Himmel gehören, umgeben sind.

Das in C++ entwickelte Verfahren ist Bestandteil von KIPL und wird mittels JNI als Komponente in das STOR-Komponentenkonzept lokal integriert.

5.3 Fassadenerkennung

Im Folgenden wird das für diese Arbeit entwickelte Verfahren zur Detektion von interessanten Fassadenkanten im Detail beschrieben. Abbildung 5.4 beschreibt den Ablauf der Segmentierung in einem Datenflussdiagramm.

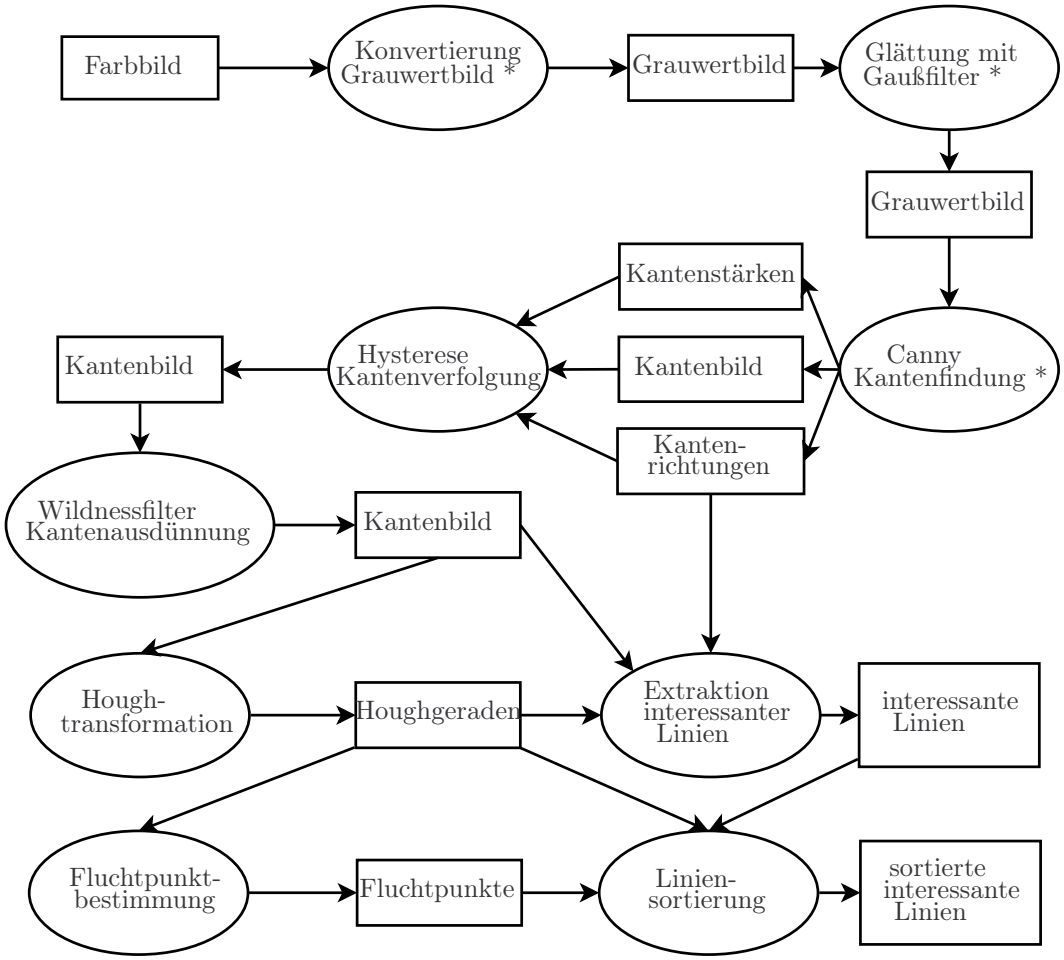


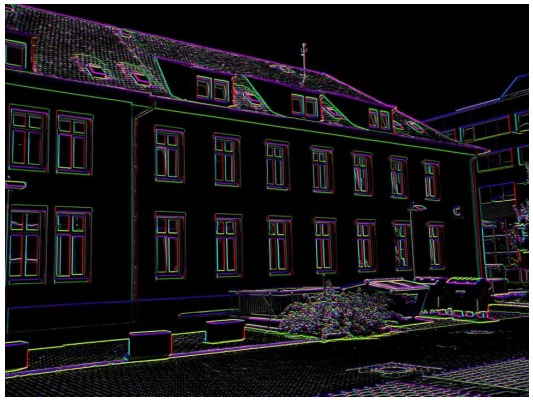
Abbildung 5.4: Datenflussdiagramm zur Segmentierung interessanter Linien eines Gebäudes.

Als Eingabe zur Segmentierung wird ein Farb- oder Grauwertbild erwartet. Handelt es sich um ein Farbbild, wird es zunächst in ein *Grauwertbild* konvertiert. Um das Bildrauschen zu mindern, wird ein *Gaußfilter* mit Maskengröße verwendet (Abbildung 5.5(a)).

Im Grauwertbild werden mittels Canny-Operator mögliche Kanten detektiert. Der *Canny-Operator* liefert als Ausgabe ein Kantenbild mit Kantenstärken und Kantenrichtungen (Abbildung 5.5(b)). Um die Menge der Kanten zu reduzieren,

werden diese mittels *Hystereseverfahren* ausgedünnt (Abbildung 5.5(c)). Das Hystereseverfahren ermittelt zusammenhängende Kantenzüge im Bild. Das Ergebnis des Hystereseverfahrens enthält viele unerwünschte Kanten, was in Abbildung 5.5(c) verdeutlicht wird. Daher wird zusätzlich der *Wildness-Filter* nach [SP09b] auf das Kantenbild angewandt (Abbildung 5.5(d)).

(a) Grauwertbild, Gauß-geglättet



(b) Gradientenrichtungen mit Canny-Operator

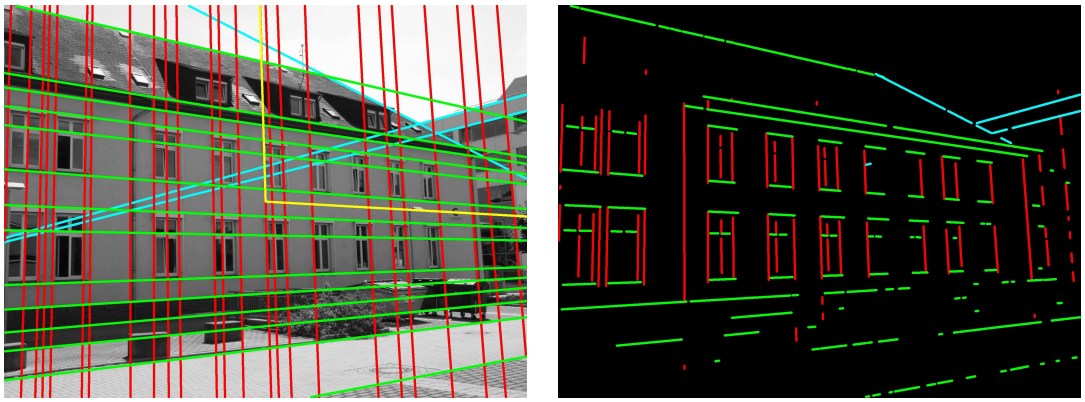
(c) Kantenbild nach Ausdünnung

(d) Kantenbild nach Wildness-Filter

Abbildung 5.5: Beispiel zur Kantenextraktion

Zur Detektion möglicher Geraden im Bild wird eine Hough-Transformation auf das bereits berechnete Kantenbild angewandt. Die Berechnung von Fluchtpunkten geschieht auf der Basis der segmentierten Houghgeraden. Hierzu wird das Verfahren von Schmitt und Priese [SP09b] (siehe Kapitel 5.1) verwendet.

Die detektierten Geraden werden dann nach Fluchtpunkten sortiert. Jedem Fluchtpunkt werden die Geraden zugeordnet, die Schnittpunkte in seiner Nachbarschaft aufweisen. Diese Zuordnung ermöglicht es, eine Ordnung auf den entsprechenden Geraden zu definieren. Die Geraden eines vertikalen Fluchtpunktes werden zum Beispiel von links nach rechts sortiert. Dies kann durch die ausschließliche



(a) Houghgeraden sortiert nach Fluchtpunkt (b) Interessante Liniensegmente

Abbildung 5.6: Beispiel zur Bestimmung von Houghgeraden und interessanter Liniensegmente. Geraden und Linien werden farblich nach Fluchtpunkten unterschieden. Die gelben Linien in (a) dienen lediglich der Kontrolle und verlaufen vom Bildzentrum zu den Fluchtpunkten.

Betrachtung ihres Eintrittspunktes ins Bild realisiert werden. Die Geraden eines seitlichen Fluchtpunktes werden analog von oben nach unten angeordnet. Hierbei kann sich auf den Vergleich der Geradenrichtungswinkel beschränkt werden.

> **Begriffsbestimmung 5.1** (Interessante Liniensegmente)**.** *Die Berechnung interessanter Liniensegmente orientiert sich an den bereits segmentierten Houghgeraden, denn es wird angenommen, dass die am stärksten ausgeprägten Geraden des Bildes die Kante des gesuchten Gebäudes enthalten. Hierzu wird im Kantenbild nach Kanten gesucht, welche in der Nähe der Houghgeraden verlaufen und deren Richtung der der Houghgeraden entspricht. Ein interessantes Liniensegment muss eine minimale Länge aufweisen und kann kleine Lücken im Kantenbild überbrücken (siehe Abbildung 5.6(b)).*

Als *Ausreißergeraden* bezeichnet man diejenigen Geraden, die keinem Fluchtpunkt zugeordnet werden können. Ihnen kann keine Ordnung bezüglich ihrer Lage im Bild zugeordnet werden. Für die spätere Verarbeitung kann es jedoch nützlich sein, sie nach ihrem Richtungswinkel zu sortieren (Abbildung 5.6(a)).

Es lässt sich für jede Houghgerade ein *Basisliniensegment* bestimmen, welches den Bereich der interessanten Liniensegmente zusammenfasst. Für eine Houghgerade können zwei verschiedene *Gewichte* definiert werden. Das erste Gewicht entspricht der Anzahl an Kantenpixel auf der Gerade. Dieser Wert wird während der Hough-Transformation automatisch berechnet. Er gibt Auskunft über die Stärke der Gerade im Bild. Über das Basisliniensegment kann ein zweites Gewicht berechnet werden. Es entsteht aus dem Verhältnis der Liniensegmente zu den verbleiben-

den Lücken. Mit diesem Gewicht lässt sich eine Aussage über die Beschaffenheit der Gerade treffen. Linienzüge mit vielen oder großen Unterbrechungen erhalten einen geringeren Wert als durchgezogene Linien. Somit könnten zum Beispiel Fenster- von Fassadenkanten unterschieden werden.

Die in Abbildung 5.4 mit Stern "*" versehenen Funktionen entsprechen Komponenten des STOR-Komponentenkonzept. Der komplette Ablauf wurde in C++ mittels KIPL umgesetzt und mittels JNI als Komponente in das STOR-Komponentenkonzept lokal integriert.

5.4 Dachkantenerkennung

Die zuvor beschriebene Detektion des Himmels im Bild ermöglicht das Bestimmen einer Horizontlinie, die zwischen dem Himmel und allen übrigen Objekten verläuft. Mit dem Verfahren von Schmitt [Sch11] werden die Horizontlinie extrahiert und dort gerade Teilstücke identifiziert, die dann Kandidaten für Dachkanten sind. Dabei wird vorausgesetzt, dass Dächer in der Regel an den Himmel grenzen. Die Berechnung der Dachlinie beruht auf der Annahme, dass Dachkanten von Gebäuden gerade sind, wohingegen nicht künstliche Objekte wie Bäume und Sträucher unregelmäßige Formen aufweisen. Die Dachkante ist auf diese Weise auch vom Sichthorizont in der Ferne zu unterscheiden, welcher eher rundlich ist. Der Dachkantenerkenner nach Schmitt [Sch11] entspricht den grauen Kästchen des Datenflussdiagramms in Abbildung 5.7 und wurde in C++ mittels KIPL umgesetzt und mittels JNI als Komponente in das STOR-Komponentenkonzept lokal integriert.

Die Strecken aus Binärbild und die möglichen Dachkanten aus den interessanten Liniensegmenten werden zunächst kombiniert und dann alle kollinearen Strecken, falls bei diesen die euklidische Distanz zwischen den Strecken im Bild einen Schwellwert nicht überschreitet, vereinigt. Für alle benachbarten Strecken wird folgende Distanz bestimmt:

$$d_{\text{roof}}(l_1, l_2) = \frac{w}{\|l_1\| + \|l_2\|} + \alpha \left(\frac{h}{y}\right) \quad , \tag{5.1}$$

mit w und h als Länge und Breite des Bildes. Der Wert y ist die y-Koordinate des Mittelpunktes von zwei benachbarten Strecken und α dient als Gewichtungsfaktor. In dieser Arbeit gilt für $\alpha = \frac{h}{w}$. Somit werden stets die Kanten des nächstliegenden Gebäudes gefunden, da dessen Dachkanten im Regelfall höher liegen und längere Einzelstrecken besitzen als weiter entfernte Gebäude. Unter Verwendung der zwei Strecken mit der geringsten Distanz zueinander (siehe Abbildung 7.5(c) auf Seite 155) ist man mit Hilfe des in Kapitel 7.2.1 beschriebenen Verfahrens in der Lage, die Rotation und Translation der Kamera zu bestimmen. Die Extraktion dieser zwei Strecken wurde als eigene Komponente umgesetzt (siehe blaue Kästchen des

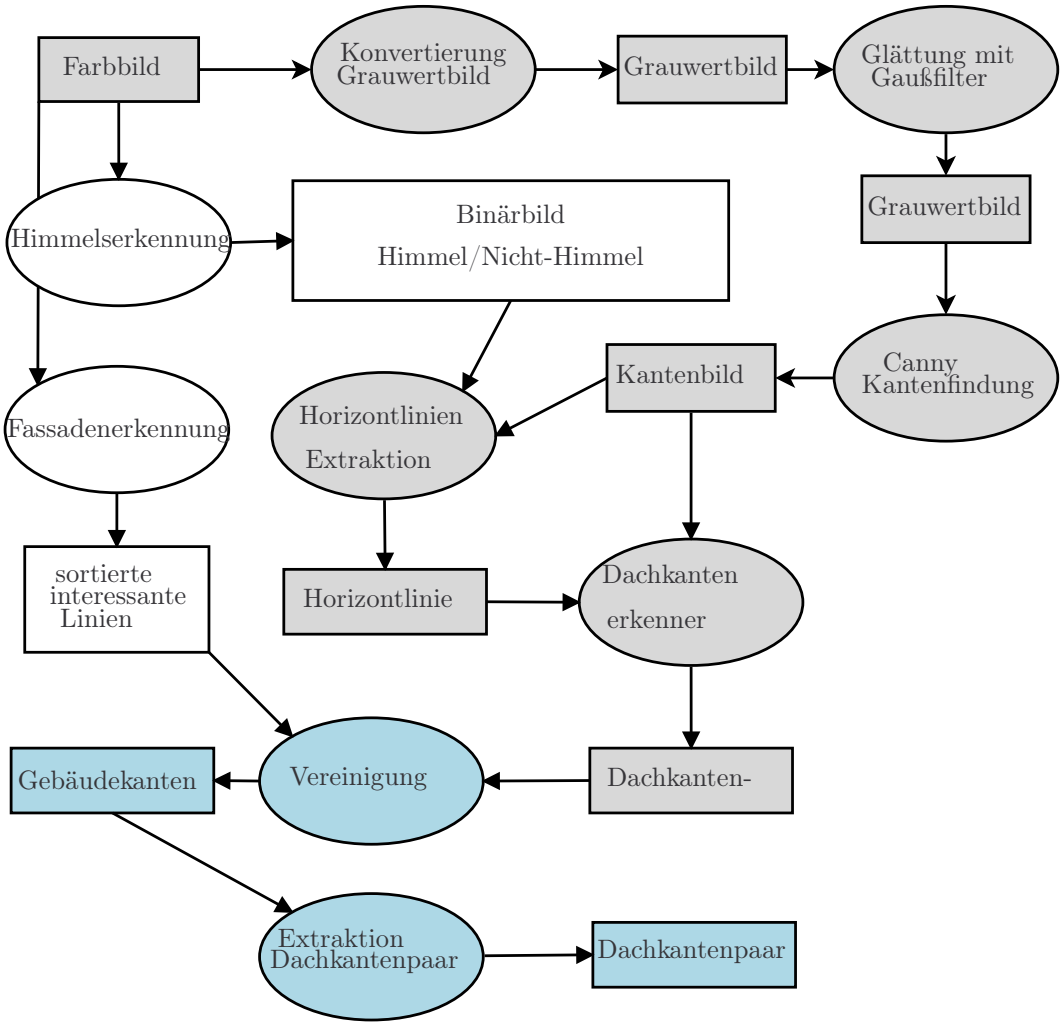


Abbildung 5.7: Datenflussdiagramm des Dachkantenerkenners.

Datenflussdiagramms in Abbildung 5.7) und nutzt als Eingaben die Ergebnisse der zuvor beschriebenen Dachkanten-, Himmels- und Fassadenerkennung.

5.5 Lokalisation von Türen und Fenstern

Fenster und Türen liefern unabhängig von den Fassaden Rückschlüsse über die Lage und die Art des Gebäudes. Daher kommt ein Verfahren zum Einsatz, das unabhängig von einer vorher zu erkennenden Fassade ist. Abbildung 5.8 skizziert dabei den Ablauf der Fenstererkennung, die trotz des Namens auch Türen erkennt.

Dabei wird das Bild zuerst mittels Gaußfilter geglättet und dann noch einmal mit dem kantenerhaltenden Kuwahara-Filter verarbeitet. Danach kommt die CSC-

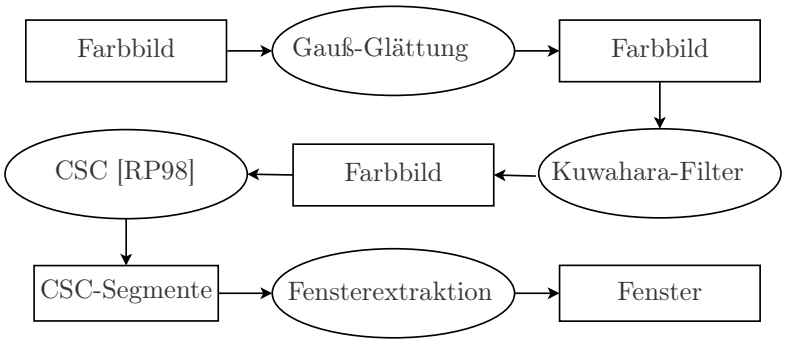


Abbildung 5.8: Datenflussdiagramm der Fenster- und Türenerkennung.

Segmentierung von Rehrmann und Priese (1998) [RP98] zum Einsatz, um so im *L*a*b*-Farbraum* farblich homogene und zusammenhängende Regionen zu erhalten. Basierend auf diesen Regionen sucht die Fensterextraktion nach rechteckigen Regionen, die sich in einer größeren Region befinden, und gibt deren Bounding-Box aus. Die Fensterextraktion ist ein von *Nicolai Wojke* entwickeltes Plugin, welches von ihm in die Koblenzer Image Processing Library (KIPL) integriert wurde. Für alle Farbsegmente im Bild werden in diesem Plugin die jeweiligen Mittelwerte bestimmt, es wird der *Freeeman-Code* aller Segmente auf den inneren Konturen berechnet und ähnliche oder nah zusammenliegende Segmente werden vereinigt. Es werden die Mittelwerte der Fensterfarbe aus den vorher berechneten Mittelwerten der zum Fenster gehörenden Segmente berechnet. Die Klassifikation der einzelnen Fensterkandidaten geschieht durch einfache Schwellwertabfragen: (i) Mittelwert der umgebenden Fassade, (ii) Vergleich der Fensterfarbe mit den Farben in einer Fenster-Datenbank, (iii) Vergleich des Größenverhältnisses mit einem vorgegebenen Größenverhältnis (Weite/Höhe). Mittels Nearest-Neighbour-Zuordnung werden Segmente, die einen gewissen Schwellwert nicht überschreiten, als Fenster klassifiziert.

Das Verfahren liefert in dieser Form wenig aussagekräftige Fenster- und Türenkandidaten, da durch Schatten, ähnliche Objekte und Verzerrungen viele falsche rechteckige Regionen in einem Bild gefunden werden. Deshalb wurde dieses Verfahren um drei *Gaußpyramiden* mit Sigma $\sigma = 0 \ldots 10$ in 1σ-Schritten erweitert. Die Pyramiden berechnen den CSC im L*a*b, im HSV oder einfach mittels Euklid. So entstehen $3 \cdot 11$ Sätze an Fenster- und Türenhypothesen. Jedes gefundene Objekt bekommt ein Vertrauensmaß $\tau(\mathit{Fenster}) = 0.2$ und $\tau(\overline{\mathit{Fenster}}) = 0.3$, sodass eine Ungewissheit $\Omega = 0.5$ bleibt. Im nächsten Schritt werden Objekte, die sich an der gleichen (mit einer gewissen Fehlertoleranz) Position befinden, zusammengefasst und die Vertrauensmaße mit der Dempster-Shafer-Kombinationsregel (siehe Kapitel 8.2.2) für dieses Objekt neu berechnet. Anschließend werden die Objekte der drei Pyramiden auf gleiche Weise zusammengefasst. Objekte, die sich in Objekten

Abbildung 5.9: Ergebnis der Fenster- und Türenerkennung.

befinden, werden gesucht und fusioniert. Dies kann auf der einen Seite zur Folge haben, dass Doppelfenster zu einem Fenster zusammengefasst werden oder dass richtige Fenster verloren gehen, aber auf der anderen Seite werden viele falsche Fenster (meist Reflexionen) verworfen und die Analyse mit weniger Fensterhypothesen belastet. Durch Wolkenstrukturen können relativ plausible Fenster/Türen im Himmel entstehen; deshalb werden mittels Himmelsextraktion [SP09a] alle Objekte im Himmel entfernt.

Der komplette Ablauf wurde in C++ mittels *(KIPL)* umgesetzt und mittels *(JNI)* als Komponente in das STOR-Komponentenkonzept lokal integriert.

Als Ergebnis erhält man zahlreiche Fenster-/Tür-Kandidaten, die alle die zwei Vertrauensmaße $\tau(Fenster)$ und $\tau(\overline{Fenster})$ besitzen. Wählt man dann nur Objekte mit einem Vertrauen von 90%, dass es sich um ein Fenster handelt, erzielt das Verfahren eine Präzision von mehr als 93,29% bei Gebäuden mit gewöhnlichen Fenstern. Das Datenflussdiagramm aus Abbildung 5.10 zeigt den Ablauf im Überblick und Abbildung 5.9 zeigt das Ergebnis der Erkennung für Fenster mit einer "Wahrscheinlichkeit" von 90%. Tabelle 5.1 gibt einen Überblick über die Ergebnisse des Fenstererkenners in den in dieser Arbeit verwendeten Datensätzen. Die Ergebnisse zeigen, dass der Erkenner gut geeignet ist, wenn die Fassaden homogen sind, aber Probleme bekommt, wenn die Fassaden verwinkelter und heterogener sind, wie im Fall vom Fertighaus der Firma Streif.

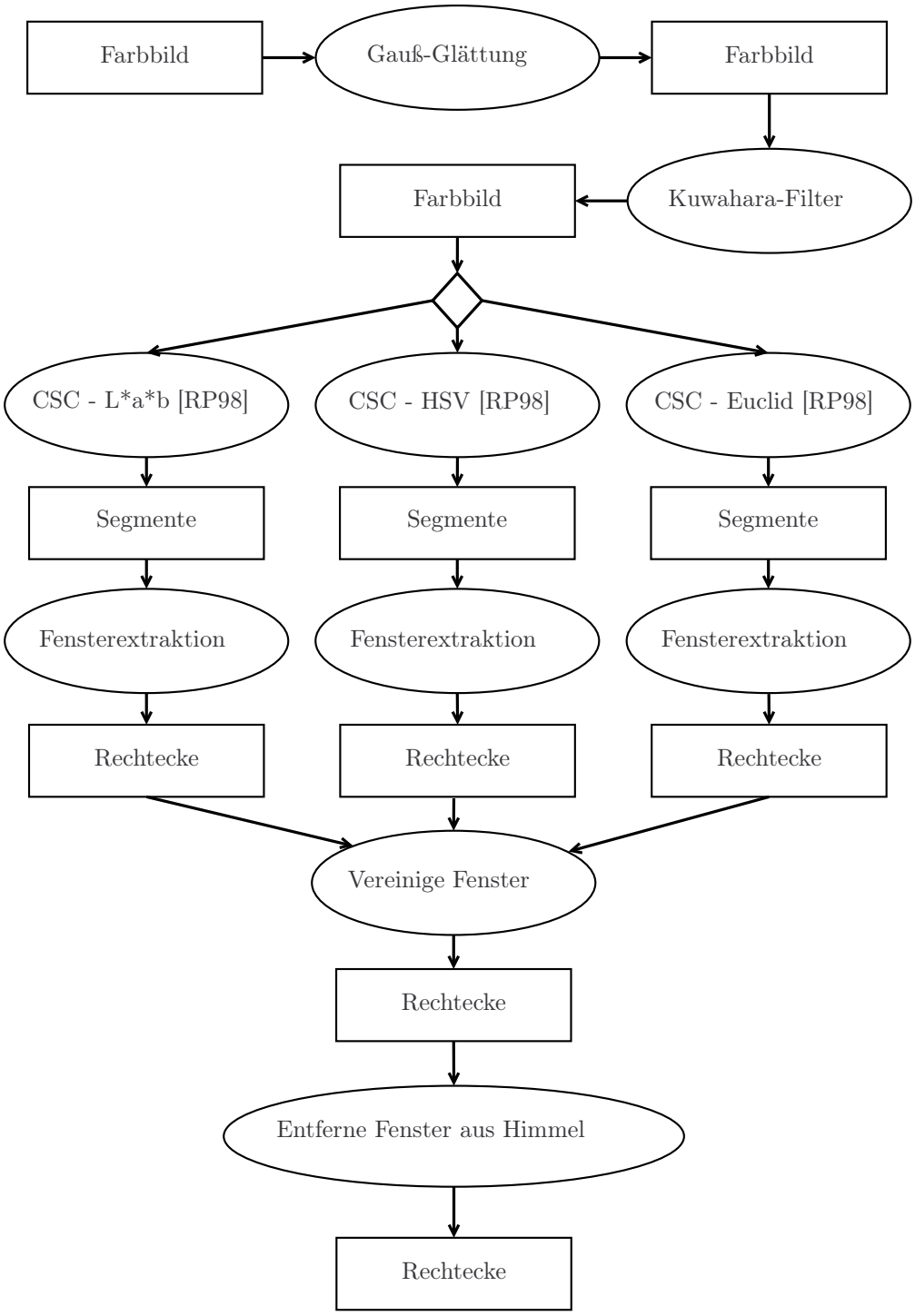


Abbildung 5.10: Datenflussdiagramm der Fenster- und Türenerkennung im Detail.

Datensatz	TP	FP	#	Recall	Precision
B-Gebäude	590	16	1013	58,24%	97,36%
Modell von Faller (H0 131277)	116	5	158	73,42%	95,87%
Modell von Auhagen (H0 11 402)	271	1	324	83,64%	99,63%
Fertighaus der Firma Massa	113	27	160	70,61%	80,71%
Fertighaus der Firma Streif	79	26	141	56,03%	75,24%
Gesamt	1169	75	1796	65,09%	93,97%

Tabelle 5.1: Ergebnisse der Lokalisation von Türen und Fenstern. TP steht für True-Positive, FP für False-Positive und # beschreibt die Gesamtzahl an Fenstern.

5.6 Diskussion

Es wurden bestehende Verfahren auf den Gebäudekontext adaptiert und weiterentwickelt, wie Fluchtpunktbestimmung, Himmelserkennung und Fassadenerkennung. Zudem wurde ein Verfahren zur Extraktion von Dachkanten und ein Verfahren zur Lokalisation von Türen und Fenstern entwickelt. Jedes dieser Verfahren liefert für sich selbst genommen durchschnittliche Ergebnisse. Sie sind dafür allerdings einfach umsetzbar. Die zu beweisende Annahme lautet: “Die Verwendung von Wissen kann Probleme bei der Erkennung kompensieren bzw. erst die Erkennung ermöglichen.”; was in Kapitel 10 belegt wird.

Kapitel 6

Kontrollstrategie

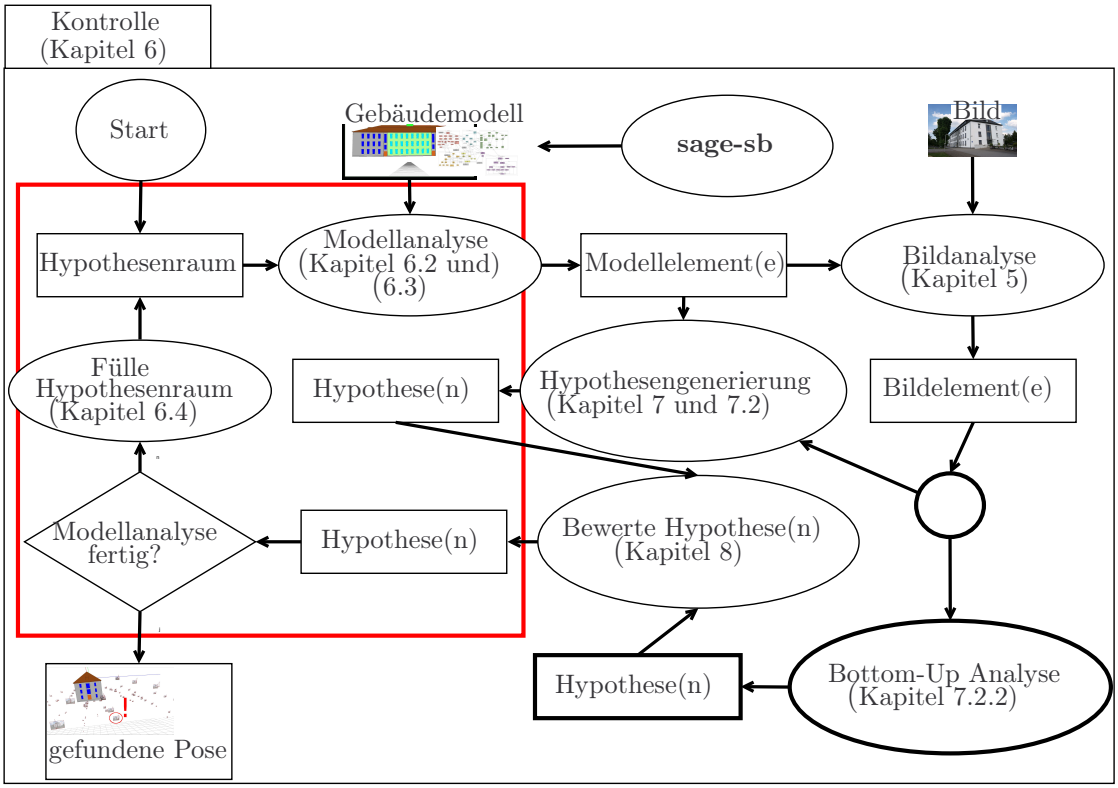


Abbildung 6.1: Eine anwendungsunabhängige Kontrolle (Abschnitte 6.2 und 6.3) steuert den Ablauf der Modellanalyse (siehe rotes Rechteck). Mit anwendungsunabhängig ist gemeint, dass jegliches rigides Objekt als geometrisches Modell zur Eingabe vorliegen kann.

Eine anwendungsunabhängige Kontrolle (Abschnitte 6.2 und 6.3) steuert den Ablauf der Modellanalyse (Abbildung 6.1). Mit anwendungsunabhängig ist ge-

meint, dass jegliches rigides Objekt als geometrisches Modell zur Eingabe vorliegen kann. Das Modell wiederum steuert über seine Bestandteile die Bildanalyse. Kapitel 5 beschreibt Verfahren, die bei der Bildanalyse zum Einsatz kommen. Im Zuge der Analyse entstehen Hypothesen (Kapitel 7) für mögliche Posen (Kapitel 7.2). Diese werden in einem Hypothesenraum verwaltet. Dies erfordert die Möglichkeit Hypothesen zu bewerten und zu verifizieren (Kapitel 8). Das Ergebnis der Analyse ist eine Pose inklusive Bewertung.

Die in Kapitel 2 beschriebenen Fallstudien besitzen aufgrund des generellen Kontrollverfahrens alle den gleichen Ablauf, sodass im Folgenden nur die Gebäudefallstudie beschrieben wird. Das Verfahren **knoPoE** besteht aus mehren Bestandteilen und wird in den nächsten Kapiteln beschrieben. Zur Übersicht wird in Abbildung 6.2 der Ablauf des Verfahrens **knoPoE** auf das wesentliche reduziert und in Beziehung zum schon eingeführten Verfahren **sage-sb** gesetzt. Die in Abbildung 6.2 gezeigten Bestandteile der Kontrollstrategie werden in den nächsten Abschnitten erklärt.

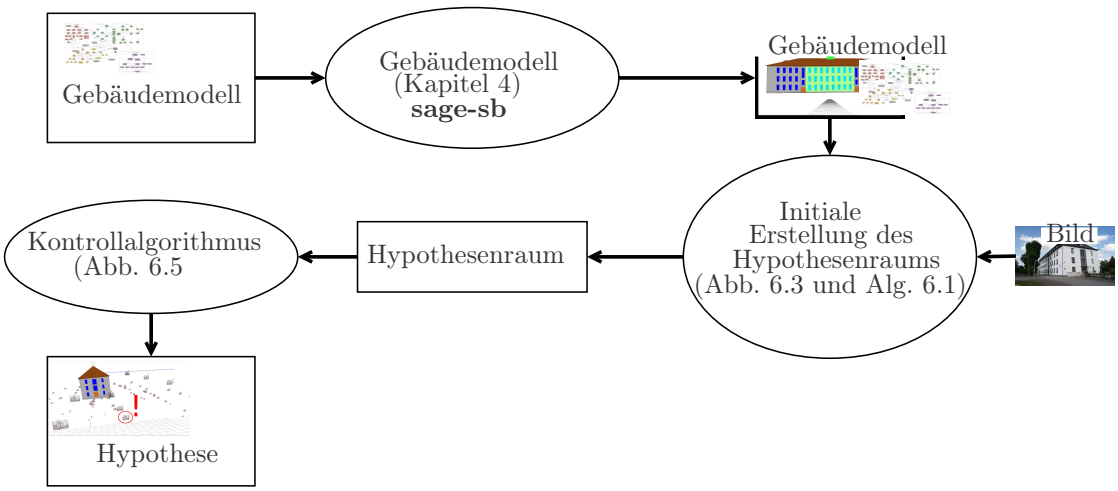


Abbildung 6.2: Zur Übersicht wird der Ablauf des Verfahrens **knoPoE** auf das wesentliche reduziert und in Beziehung zum schon eingeführten Verfahren **sage-sb** gesetzt. Die gezeigten Bestandteile der Kontrollstrategie werden in den nächsten Abschnitten erklärt.

Begriffsbestimmung 6.1 (Kontrollalgorithmus)**.** *Ein Kontrollalgorithmus im Verständnis dieser Arbeit steuert anwendungsunabhängig den Ablauf der Analyse. Die dabei entstehenden Hypothesen werden im Hypothesenraum gespeichert. In diesem Hypothesenraum kann mit Graphsuchverfahren nach der optimalen Lösung gesucht werden.*

Für eine wissensbasierte Bildanalyse sind zwei Bereiche von entscheidender Bedeutung: das *Modellwissen* und der *Kontrollalgorithmus*. Die wissensbasierte Ver-

arbeitungsstrategie war in den 80er-Jahren in vielen Bereichen populär, auch in der Bildanalyse. Ein solches System ist ERNEST [NSSK90], welches in verschiedenen Anwendungsbereichen, wie Gebäudeerkennung [KSN92] und Sprachverstehen [Sag85], Verwendung fand. Es verbindet eine Repräsentation von Konzepten in einem Graph, in dem nur wenige Kantentypen erlaubt sind, mit einer allgemeinen Kontrolle auf Basis des A^*-Algorithmus.

In dieser Arbeit werden Modelle eingesetzt, die die Geometrie der Objekte beschreiben, wofür TGraphen [Ebe08] als leicht-gewichtige Ontologie verwendet werden (siehe Kapitel 2.5). Diese Modelle enthalten neben dem deklarativen Wissen über die Semantik und Geometrie auch prozedurales Wissen (beschrieben in Kapitel 3.6). Das prozedurale Wissen beinhaltet anwendungsabhängige Funktionen unter anderem zum Finden und Initialisieren von Merkmalen im Bild. Den Modellen werden die Signaturen der entsprechenden BV-Algorithmen im STOR-Komponentensystem übergeben. Dabei wird klar zwischen Problembeschreibung bzw. Wissensbasis und Problemlösung bzw. Wissensverarbeitung getrennt. Dies hat als Konsequenz, dass das Wissen über den Anwendungsbereich direkt beschreibbar ist.

Das Verfahren **knoPoE** wurde teilweise in [WFP12, WP10, WP11] veröffentlicht und die folgende Darstellung verwendet diese Publikationen.

6.1 Einführung

Zunächst wird im Verfahren der Hypothesenraum mit dem Gebäudemodell als Starthypothese befüllt (siehe Abbildung 6.2). Aus dieser Starthypothese und der Gebäudeaufnahme werden Posehypothesen berechnet und in Hypothesen im Hypothesenraum überführt. Die Überführung von Posehypothesen in Hypothesen geschieht indem aus jeder Posehypothese eine Hypothese entsteht, bei der mittels semantischen Renderings (beschrieben in Kapitel 3.5) die sichtbare 2-D Geometrie extrahiert und jeweils im Gebäudemodell gespeichert wird. Diese Hypothesen werden wieder in den Hypothesenraum einsortiert. Abbildung 6.3 und Pseudocode 6.1 geben einen Überblick über die Initialisierung.

Der Kern der Kontrollstrategie (siehe Abbildung 6.4) nimmt den initial befüllten Hypothesenraum und analysiert iterativ die enthaltenen Modelle, welche sich durch unterschiedliche Posehypothesen und somit unterschiedlicher 2-D-Geometrie differenzieren. Bei jedem Analyseschritt entstehen neue Hypothesen. Die entstanden Hypothesen werden bewertet und in den Hypothesenraum überführt. Die Strategie (A^*-Suche) mit der die jeweils plausibelste Hypothese aus dem Hypothesenraum gesucht wird, wird in Abschnitt 6.4 beschrieben. Nach Abschluss der Analyse wird die plausibelste Hypothese ausgegeben. Die Abschnitte 6.2 und 6.3 sind in Abbildung 6.4 vereinfacht in der grauen Ellipse mit “Analysiere Modell aus Hypo-

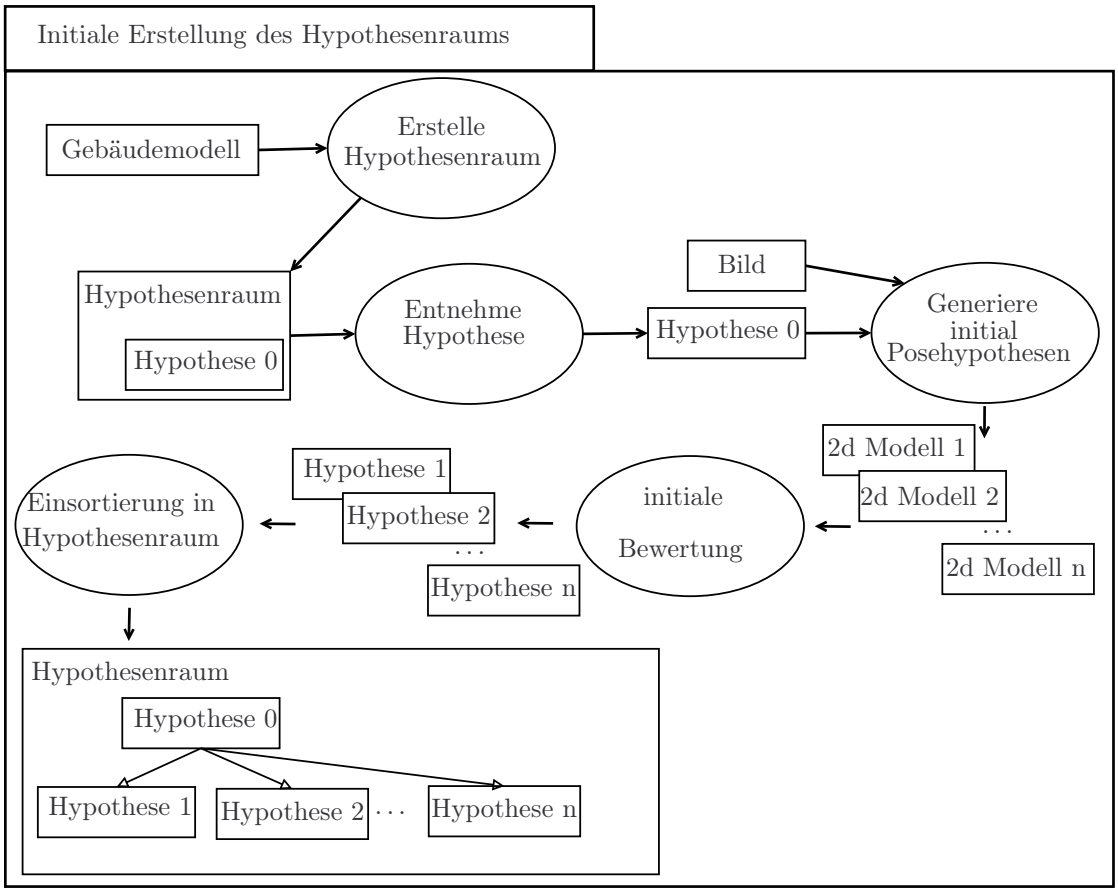


Abbildung 6.3: Initale Erstellung des Hypothesenraums mit dem Gebäudemodell als Eingabe. Der Hypothesenraum und eine Aufnahme mit dem entsprechenden Gebäude dienen als Eingabe für die Methode zur initialen Füllung des Hypothesenraums. Als Ausgabe erhält man den Hypothesenraum befüllt mit den Hypothesen entstanden aus den Posehypothesen. Dieser Hypothesenraum dient dann als Eingabe für den Kern der Analyse.

these" zusammengefasst. Diese Modellanalyse bildet den Kern des Verfahrens und wird im Folgenden beschrieben. Dabei gehen diese Beschreibungen initial von einer Analyse ohne widersprüchliche Ergebnisse aus, so dass Hypothesen und Hypothesenraum in den Abschnitten 6.2 und 6.3 so gut es geht ausgeklammert werden. Der Kreis wird dann in Abschnitt 6.4 mit einer ausführlichen Erklärung zur Suche im Hypothesenraum geschlossen.

Pseudocode 6.1 Initale Erstellung des Hypothesenraums

```
1: procedure INITIALISIERUNG(Gebäudemodell mb, Bild I)
2:     Füge mb zu Hypothese Hyp_0 hinzu
3:     Füge Hyp_0 zu Hypothesenraum H hinzu
4:     SemanticObject BU ← ENTNEHME( Wurzelknoten Building BU aus mb)
5:       ▷ //Das SemanticObject implementiert das Interface SemanticDetection
6:     Liste <Pose> Posehypothesen ←getTopDownPoseHypotheses(mb, BU, I)
7:     for all υ_i ∈ Posehypothesen do
8:         mb_i ← RENDERING(mb, υ_i)
9:         ERZEUGE (Hyp_i mit mb_i)
10:        Hyp_i speichert den Graphknoten BU als zuletzt bearbeitet
11:        BEWERTE (Hyp_i)
12:        Markiere in Hypothese_i Hypothese_0 als Vorgänger
13:        Füge Hyp_i zu Hypothesenraum H hinzu
14:    return Hypothesenraum H
```

6.2 Strategie

Es wird in einer Kombination aus *datengetriebener* (bottom up) und *modellgetriebener* (top down) Analyse nach Objekten im Bild gesucht. Top-Down und Bottom-up sind beides Strategien im Bereich der Informationsverarbeitung und Wissensanalyse.

Definition 6.1 (Top-Down Analyse)**.** *Bei der Top-Down Analyse wird mit dem allgemeinsten Modellelement (dem Wurzelknoten) mit der Analyse begonnen und sich von dort weiter in die Tiefe des Modellgraphen begeben.*

> *[Top-Down-Methode, die] "(Fachsprache) Methode, bei der man schrittweise von allgemeinen, umfassenden Strukturen zu immer spezielleren Details übergeht"* [a]

[a] `http://www.duden.de/`, Stichwort: Top-Down, Zugriff: 06.02.2015

Die Posehypothesen entstehen bei **knoPoE** Top-Down, d. h. das generellste Konzept bzw. der Wurzelknoten des Modells beschreibt, wie mögliche Posen generiert werden können. Eine Bewertung der Posen erfolgt bottom-up über die Zuordnung von Bildkanten zu den im Modell enthaltenen Polygonstrecken. Die weitere Bildanalyse wird Top-Down über die Modellelemente durchgeführt. Ein fester Regelsatz definiert exakt, in welcher Weise dies zu geschehen hat. Während

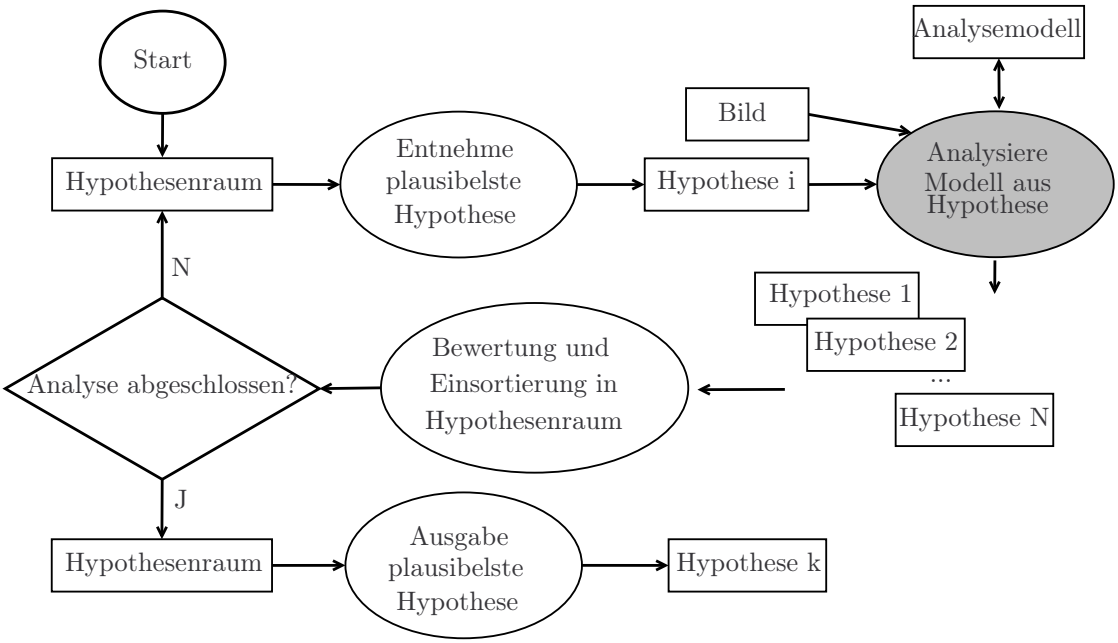


Abbildung 6.4: Ablauf der Kontrollstrategie nach Initialisierung des Hypothesenraums.

des Verfahrens werden durch extrahierte Bildelemente wieder bottom-up neue Posehypothesen generiert.

Definition 6.2 (Bottom-up-Analyse)**.** *Bei der Bottom-Up Analyse schlussfolgert die Analyse auf Basis spezifischer Modellelemente, in der Regel in der Tiefe des Modellgraphens zu finden, auf Allgemeineres und so begibt sich die Analyse von der Tiefe in die Höhe des Graphens.*

> *[Bottom-up-Methode, die] "Methode, bei der man von speziellen Details ausgeht und schrittweise über immer umfassendere Strukturen die Gesamtstruktur eines Systems errichtet"* [a]

[a] `http://www.duden.de/`, Stichwort: Bottom-up, Zugriff: 06.02.2015

Im Laufe der Bildanalyse wird bestimmten Teilen des Bildinhaltes eine Bedeutung zugeordnet. Dies geschieht, indem eine Verbindung zwischen einem Bildelement und einem Modellelement geknüpft wird. Eine solche Verbindung zwischen Bild- und Modellelement beschreibt eine *Zuordnung* und der Prozess als solches wird im Rahmen des Analyseprozesses als *Kontrollinstanziierung* bezeichnet.

Ein weiterer Analyseschritt ist die *Expansion*. Hierbei wird das bisher betrachtete Modell erweitert und das bereits erworbene Wissen an die Bestandteile weitergegeben. Vereinfacht gesagt wechselt sich Expansion und Kontrollinstanziierung im Laufe der Analyse ab.

Ein anwendungsunabhängige Kontrollalgorithmus übernimmt die Steuerung der Analyse und entscheidet, welche Schritte als nächstes durchzuführen sind. Das Verfahren ist an das Verfahren von Kummert [KSN92] angelehnt, der sechs anwendungsunabhängige Regeln verwendet. In dieser Arbeit werden *vier anwendungsunabhängigen Regeln* verwendet:

Regel 1
Prüfe ob die Methode *instantiate* eines als Bestandteil verbundenen Knotens ein Ergebnis liefert, falls nein prüfe ob beim Bestandteil eine *isRepresentedBy*-Kante vorhanden ist und führe dort nochmal die Methode *instantiate* aus. Wenn ein Ergebnis vorhanden ist, prüfe die Multiplizität (siehe Definition 3.5) des aktuellen Graphknotens. Wenn die Multiplizität größer als eins ist, führe Regel 1b mit dessen Bestandteilen aus, ansonsten nutze Regel 1a

Regel 1a
Ordne die Segmentierungsergebnisse der *instantiate*-Methode jeweils dem Graphknoten, der die Methode ausgeführt hat, zu (es entstehen ggf. mehrere neue Hypothesen).

Regel 1b
Ordne die Segmentierungsergebnisse der *instantiate*-Methode den Graphknoten, der die Methode ausgeführt hat, und seinen Geschwisterknoten vom gleichen Typ mittels Graphmatching zu (es entsteht genau eine neue Hypothese).

Regel 2
Wenn die Zuordnung bereits erfolgt ist oder die Methode *instantiate* keine Ergebnisse liefert, expandiere diesen Graphknoten, d. h. suche nach einem Bestandteil von diesem Graphknoten, der noch nicht zugeordnet wurde. Wenn alle Bestandteile schon zugeordnet wurden, gehe zum Elternknoten, dies ist der Graphknoten, der mit *consistsOf*-Kante verbunden ist und den direkten Vorgänger des Graphknotens darstellt, und wiederhole Regel 2.

Regel 3
Wenn die Kontrollinstanziierung eines Elementes erfolgreich war, wird die Bildanalyse für alle Bestandteile des Graphknotens auf dessen Region of Interest eingeschränkt.

Regel 4
Wenn ein Graphknoten nicht obligatorisch ist, versuche das Vorkommen dieses Graphknotens im Bild zu untersuchen. Sollte es nicht möglich sein, Kandidaten zu finden, wird das Fehlen nicht bestraft.

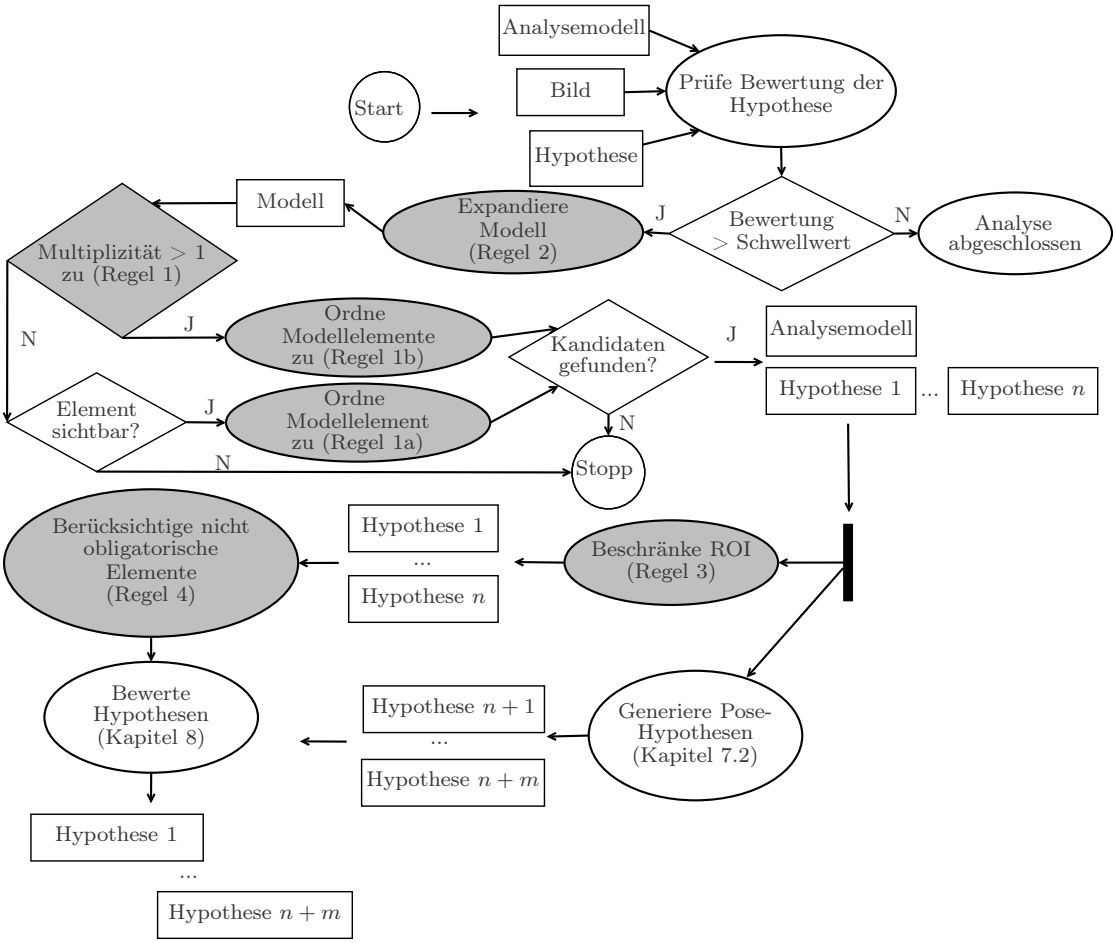


Abbildung 6.5: Die Modellanalyse der Kontrollstrategie (graue Ellipse "Analysiere Modell aus Hypothese" in Abbildung 6.4) wird hier im Detail beschrieben.

Definition 6.3 (Bipartiter Graph). *"Ein Graph $G = (V, E)$ ist bipartit, wenn die Knotenmenge V in zwei Mengen A und B partitioniert werden kann, sodass alle Kanten einen Knoten aus A mit einem Knoten aus B verbinden." [Häs10]*

Bei Anwendungen, bei denen sich Abstandsmaße für Zuordnungen zu Beginn der Analyse berechnen lassen, kann das Suchraumproblem(der Hypothesenraum

(Suchraum), wächst explosionsartig an) in eine *bipartite Graphzuordnung* überführt werden. Nach der Bewertung aller Zuordnungen muss die Zuordnung mit dem maximalen Gewicht gefunden werden. Für diese Aufgabe existieren schnelle Algorithmen wie die Ungarische Methode mit einer Komplexität von $O(n^3)$ [Jun90] (vgl. Anhang D).

Definition 6.4 (Gewichtsmaximales bipartites Zuordnungsproblem). *"Sei $G = (V, E)$ ein bipartiter Graph mit einer Gewichtsfunktion $w : E \to \mathbb{R}$. Als gewichtmaximales 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. 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 Zuordnungsmetrik zugrunde liegt (siehe [BB82] Kapitel 11 Abschnitt 3.1)." [Häs10]*

Regel 1 unterteilt sich in zwei Regeln. *Regel 1a* löst die Zuordnungen einfach kombinatorisch, so dass mehrere Hypothesen entstehen können, falls mehr als ein Segmentierungsergebnis vorliegt, und *Regel 1b* ermittelt die Zuordnung mittels Graphzuordnung (in dieser Arbeit kommt die *Ungarische Methode* [Kuh55] zum Einsatz), so kann nur eine Hypothese entstehen. Die Ungarische Methode ist in der Lage die optimalen Zuordnungen von Modell- zu Bildelementen zu finden, vorausgesetzt eine Kosten- oder Vertrauensmaßberechnung ist für jede potenzielle Zuordnung möglich. Immer wenn die Kardinalität eines Modellknotens größer als eins ist, wird diese Methode verwendet. *Regel 2* regelt die Analyse des Modells; so werden die Bestandteile eines Modellelements sequenziell expandiert. Dabei wird für ein Modellelement mit mehreren Bestandteilen ein Bestandteil nach dem anderen komplett (bis in die Blattknoten) analysiert. *Regel 3* hat zum einen die Aufgabe, den Suchbereich immer weiter einzugrenzen, und ist zum anderen dafür zuständig, unmögliche Kombinationen von Modellelementen zu verhindern. Eine Unterscheidung von obligatorischen und optionalen Elementen wird durch *Regel 4* ermöglicht. Optionale Elemente liefern einen zusätzlichen Beitrag zur Erkennung, sind aber nicht nötig, um ein Modell zu erkennen. Dies ermöglicht Elemente, die nur selten im Erkennungsprozess vorhanden sind, im Erkennungsprozess zu berücksichtigen, ohne dass deren Fehlen einen negativen Einfluss auf die Kontrolle hätte. Ein Beispiel hierfür sind Jalousien, die nicht immer heruntergelassen sind. Abbildung 6.5 zeigt, wie die Regeln in den Anwendungsablauf integriert sind, wobei die Boxen zu den korrespondierenden Regeln grau unterlegt sind. Die Fallstudien Dominostein- und Pokerkartenerkennung benötigen keine 3-D-Modelle und gene-

rieren somit auch keine Posehypothesen. Das Verfahren reduziert sich also um diese Punkte, ist aber ansonsten identisch. Der Ablauf der Kontrolle am Beispiel der Poseschätzung von Gebäuden wird nun im Detail beschrieben.

6.3 Kontrollalgorithmus

Abbildungen 6.2 gibt einen Überblick über das Verfahren **knoPoE** und stellt den Zusammenhang zum Verfahren **sage-sb** her. Die Abbildungen 6.3, 6.4 und 6.5 vertiefen die Erklärungen dazu. Im Folgenden wird die Beschreibung des Verfahrens noch einmal vertieft und es wird auch explizit auf den Zusammenhang zum Gebäudemodell eingegangen.

Pseudocode 6.2 Übersicht Kontrollstrategie

```
1: function KONTROLLSTRATEGIE(Hypothese Hyp, Bild I, Analysemodell mA, Schwellwert θ)
2:     int bel ← ENTNEHME(Bewertung von Hyp)
3:     if bel > θ then
4:         Gebäudemodell mb ← ENTNEHME(Gebäudemodell aus Hyp)
5:         Graphknoten S ← ENTNEHME(zuletzt bearbeiteter Knoten aus Hyp)
6:                                   ▷ //S ist ein Knoten aus dem Gebäudemodell
7:         Liste<Knotentyp> Typen ← EXTRAHIERETYPEN(Bestandteile verbunden mit S)
8:         for all t_i ∈ Typen do
9:             Liste<Hypothese> Hypothesen ← INITIALIZE(Hyp, S, t_i)
10:            Graphknoten E ← ENTNEHME(Endknoten aus mb über mit S verbundene Kante)
11:            Füge E in WarteSchlange zuletzt bearbeiteter Knoten hinzu
12:        Liste<Posehypothesen> Posen ← BOTTOMUPPOSEBERECHNUNG(t_i)
13:        Liste<Hypothese> Hypothesen_2 ← GENERIEREHYPOTHESEN(aus Liste<Posehypothesen>)
14:        Liste<Hypothese> Hypothesen_all ← VEREINIGELISTEN(H1 und H2)
15:        for all Hyp_j ∈ Hypothesen_all do
16:            Liste<LimitationConcept> Beschränkungen← BESCHRÄNKE-ROI(Hyp_j)
17:            Hyp_j ← SPEICHERE(Beschränkungen)
18:            BEWERTE(Hyp_j (berücksichtige auch obligatorische Elemente))
19:    return Hypothesen_all
```

Der Kern der Kontrollstrategie und Initialisierung wird im Pseudocode 6.2 und 6.3 detailliert beschrieben, da Sie zentraler Bestandteil dieser Arbeit sind. Die Beschreibung vertieft noch einmal die Beschreibung, die schon durch Abbildung 6.5 gegeben wurde, und entspricht ebenfalls der grauen Ellipse in Abbildung 6.4 "Analysiere Modell aus Hypothese". Der Kern der Kontrollstrategie beginnt mit einer Prüfung ob die Bewertung (das Vertrauensmaß (siehe Kapitel 8)) größer als ein Schwellwert ist (Zeile 2 in Pseudocode 6.2). In dieser Arbeit wurde für die Gebäudefallstudie der Schwellwert 0,6 gewählt, da Hypothesen mit einer schlechteren Bewertung wenig vielversprechend sind (siehe Kapitel 2 Ungleichung 2.35). Allerdings kann es im Zuge der Analyse dazu kommen, dass die Bewertung zunächst sinkt, bevor sie wieder steigt. Das liegt daran, dass beispielsweise zu Begin der Analyse ein Modellelement nicht zugeordnet werden kann. Eine Erhöhung des Schwellwertes führt zu weniger Hypothesen (Beschleunigung des Verfahrens), was dazu führen kann, dass die korrekte Pose nicht identifiziert werden kann, da die Bewertung temporär unter den Schwellwert fallen könnte und somit nicht weiter berücksichtigt werden würde.

Ist die Bewertung der Hypothese größer als der Schwellwert (=0,6), wird der Hypothese das zugehörige Gebäudemodell entnommen. Zu diesem Gebäudemodell gehört eine Pose und die entsprechende 2-D Geometrie. Sowohl die Pose als auch die 2-D Geometrie ist im Modell gespeichert. Die Erzeugung und Speicherung der 2-D Geometrie wird in Kapitel 3.5 beschrieben. Kapitel 7.2 beschreibt wie Posehypothesen, die Voraussetzung für die Berechnung der 2-D Geometrie sind, berechnet werden. In der Hypothese ist zudem der Stand der Analyse für diese Hypothese gespeichert, also welche Knoten des Gebäudemodells zuletzt untersucht wurden.

Ausgehend vom zuletzt bearbeiteten Knoten, werden die ausgehenden Kantentypen geprüft und gespeichert. Ist beispielsweise der Startknoten *Building*, wären die ausgehenden Kantentypen *BuildingConsistsOfWalling*, *BuildingConsistsOfRoof* und *BuildingConsistsOfGround* (siehe Abbildung 3.12 auf Seite 88). Für jeden Kantentyp wird die Initialisierung, als prozedurales Wissen über das Interface *CommonDetection* verpflichtender Bestandteil jedes Modellknotens (siehe Abbildung 3.5 auf Seite 79), aufgerufen. Wie die Initialisierung im Detail funktioniert wird am Pseudocode 6.3 erklärt. Nach der Initialisierung wird der über die Kante verbundene Knoten als zuletzt bearbeitet in der Hypothese gespeichert. Es wird eine Warteschlange zur Speicherung der zuletzt bearbeiten Knoten verwendet, da in einem Analyse- bzw. Expansionsschritt mehrere Knoten bearbeitet werden.

Auf Basis der in der Initialisierung gefunden Zuordnungen von Bild- zu Modellelementen, können neue Posehypothesen generiert werden. Alle Knoten besitzen (wieder) über das Interface *CommonDetection* eine entsprechende Methode, um Bottom-Up Posehypothesen zu erzeugen. Die Überführung von Posehypothesen

in Hypothesen geschieht indem aus jeder Posehypothese eine Hypothese entsteht, bei der mittels semantischen Renderings (beschrieben in Kapitel 3.5) die sichtbare 2-D Geometrie extrahiert und jeweils in einem Gebäudemodell gespeichert wird. Es entsteht eine Liste von Hypothesen, generiert aus den neuen Posehypothesen und ergänzt um die Hypothesen aus der neuen Initialisierung. Diese Hypothesen werden bewertet und wieder dem Hypothesenraum übergeben.

Pseudocode 6.3 Übersicht Initialisierung

```
1: function INITIALIZE(Hypothese Hyp_i, Graphknoten KnotenS, Knotentyp t_i)
2:     Liste<Hypothese> Hypothesen ← ERZEUGELISTE
3:     int n ← Anzahl Kanten von t_i ausgehend von KnotenS
4:     if n > 1 then
5:         Analysemodell mA ← BILDANALYSE(KnotenS)
6:         if Bildanalyse ohne Ergebnis then
7:             if Kante vom Typ isRepresentedBy am KnotenS vorhanden? then
8:                 mA ← BILDANALYSE(Graphknoten verbunden mittels
   isRepresentedBy-Kante)
9:         Liste<InititialisationConcept> Initialisierungen← ZUORDNUNGMI-
   TUNGARISCHERMETHODE(extrahierte Bildelemente, Modellelemente)
10:        Füge Initialisierungen zu Hyp_i hinzu
11:        Füge Hyp_i zu Hypothesen hinzu
12:    else
13:        if KnotenS sichtbar? then
14:            mA ← BILDANALYSE(KnotenS)
15:            if Bildanalyse ohne Ergebnis then
16:                if Kante vom Typ isRepresentedBy am KnotenS vorhanden?
   then
17:                    mA ← BILDANALYSE(Graphknoten verbunden mittels
   isRepresentedBy-Kante)
18:            Liste<InititialisationConcept> Initialisierungen ← DIREKTEZU-
   ORDNUNG
19:            Hypothesen ← ERZEUGEHYPOTHESENLISTE(Initialisierung)
20:    Zugeordnete Bildelemente werden mit einem GraphMarker versehen
21:    return Hypothesen
```

Die Initialisierung wird nun am am Pseudocode 6.3 beschrieben. Zu Beginn wird geprüft wie viele Knoten mit dem Startknoten(*KnotenS*) über eine Kante vom entsprechenden Kantentyp(*Knotentyp*$_i$) verbunden sind. Ist mehr als ein Knoten verbunden, erfolgt die Zuordnung von Modell- zu Bildelementen über die

Ungarische Methode. Ansonsten erfolgt die Zuordnung kombinatorisch (also über eins-zu-eins Zuordnungen).

Es müssen zunächst Bildelemente aus dem Bild extrahiert werden. Das Wissen welche Bildanalyse auszuführen ist, ist im Modellknoten als Teil der Initialisierung (Interface *CommonDetection*) hinterlegt. Scheitert die direkte Initialisierung des *SemanticObject*, wird geprüft ob eine *isRepresentedBy* verbunden ist und falls ja, wird die Initialisierungsmethode des mit dieser Kante verbundenen *GeometricObject* ausgeführt. Die Ergebnisse der Bildanalyse werden im *Analysemodell* gespeichert. Für den Fall, dass mehrere Knoten vorhanden sind, wird mittels ungarischer Methode eine Liste von Zuordnungen zwischen Elementen des Gebäudemodells und des Analysemodells als Liste gefüllt mit Elementen vom Typ *InitialisationConcept* erzeugt. Um die Ungarische Methode nutzen zu können, bietet jedes semantische Objekt (abgeleitet vom *SemanticObject* des Referenzschemas) der Dominostein-, Pokerkarten und Gebäudemodelle eine Bewertung gemäß dem Vertrauensmaß der Dempster-Shafer-Theorie [Sha76]. Diese Bewertung beschreibt wie gut ein Bildelement zu einem Modellelement zugeordnet werden kann, wobei nicht gefundene Elemente das Vertrauensmaß verschlechtern.

Auch bei der eins-zu-eins Zuordnung wird eine Liste gefüllt mit Elementen vom Typ *InitialisationConcept* erzeugt, nur dass hier vor der Zuordnung zusätzlich die Sichtbarkeit der Elemente des Gebäudemodells geprüft wird. Die Zugeordneten Bildelemente werden im Analysemodell mittels Graphmarker als zugeordnet markiert, um so Mehrfachzuordnungen zu vermeiden. Die Zuordnungen werden dann als neue Liste von Hypothesen gespeichert und zurückgegeben. Kam die ungarische Methode zur Anwendung enthält die Liste ein Element, ansonsten enthält die Liste so viele Element wie passende Bildelemente aus dem Bild extrahiert wurden.

Im Folgenden wird der Prozess der Bildanalyse bzw. der Ablauf des Kontrollalgorithmus am Beispiel einer vereinfachten (=widerspruchsfreien) Gebäudeerkennung demonstriert. Hier wird, wie zuvor, der Hypothesenraum und die Hypothesen entsprechend ausgeklammert. Anhand von Abbildung 6.6 bis 6.8 wird nun der Prozess der Bildanalyse beschrieben. Zur Beschreibung der Analyse kommt in diesem Beispiel ein vereinfachtes Gebäudemodell zum Einsatz, da ein gewöhnliches Gebäudemodell durch seine Vielzahl an Elementen nicht übersichtlich darstellbar ist (im Verfahren wird selbstverständlich das komplette Modell genutzt). Es werden im Beispiel nur semantische Objekte dargestellt und bewusst auf geometrische/regionsbasierte Objekte verzichtet. Auch die Gesamtzahl an Modellelementen wird im Beispiel stark beschränkt. Dies verändert das prinzipielle Vorgehen des Kontrollalgorithmus nicht.

Dieser Abschnitt beschreibt am Beispiel die in Abbildung 6.3 und Pseudocode 6.1 beschriebene Initialisierung des Hypothesenraums. Ausgangspunkt des Verfahrens ist ein 3-D-Modell des zu bestimmenden Gebäudes, eine Aufnahme dieses

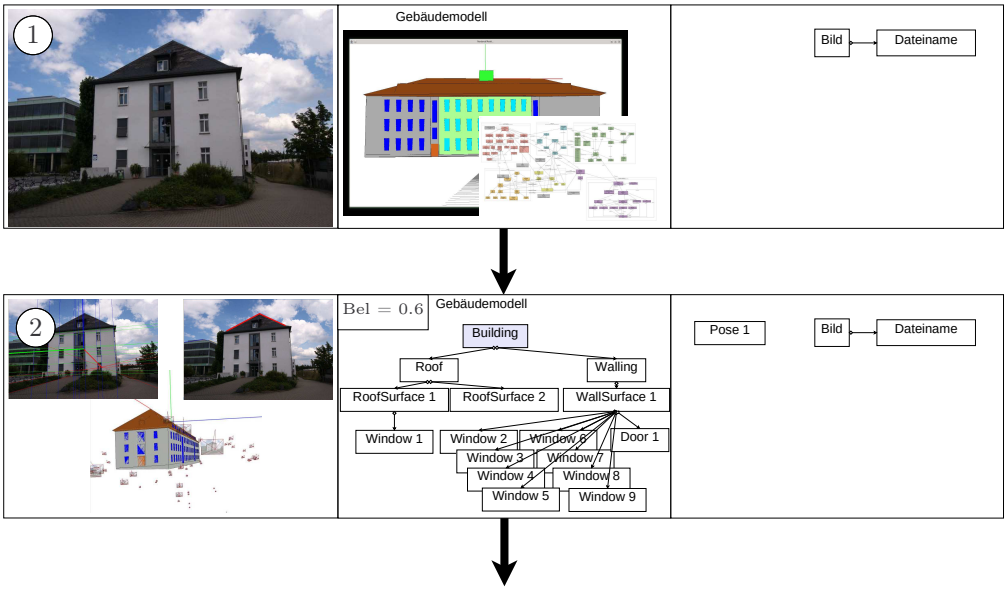


Abbildung 6.6: Beispielhafter Ablauf der Kontrollstrategie.

Gebäudes und das noch leere Analysemodell (Abbildung 6.6, Schritt 1). Zunächst wird für das 3-D-Modell eine Posehypothese berechnet. Dazu werden die Fluchtpunkte und Dachkanten des Gebäudes bestimmt. Das Verfahren zur Berechnung von Posenhypothesen wird in Kapitel 7.2 im Detail beschrieben (Abbildung 6.6, Schritt 2). Mittels der Pose wird die 2-D-Geometrie des Gebäudes erzeugt (siehe Kapitel 3.5) und dem Modell hinzugefügt, sodass nun 2-D↔2-D Zuordnungen möglich sind. Das Wissen, welche Methode zur Generierung von Posehypothesen Verwendung finden soll, ist im Knoten *Building* in der *getTopDownPoseHypotheses*-Methode verankert (vgl. Kapitel 3.4). Da sich ein Modellelement vom Typ *Building* nicht initialisieren lässt, wird das Modell nun expandiert. Initial besitzt die Hypothese passend zu dem Modell und der Pose einen Belief von 0,6.

In Schritt 3 wird in der nun anstehenden Initialisierung festgestellt, dass sich das Dach (*Roof*) zum einen nicht direkt initialisieren lässt, und zum anderen, dass es aus zwei Dachflächen (*RoofSurfaces*) besteht, die sich initialisieren lassen. Es werden die Strecken im Bild extrahiert und, da das Dach (*Roof*) mehrere Bestandteile besitzt, mittels Ungarischer Methode den Modellstrecken zugeordnet. Hier wird die Menge an Strecken zur Vereinfachung als Polygon zusammengefasst. Dabei schränken die Positionen der Modellstrecken den Suchbereich im Bild ein, was durch die beiden roten Rechtecke symbolisiert wird. Die extrahierten Polygone werden in das Analysemodell hinzugefügt, an den *Bild*-Knoten gehängt und stehen nun in der gesamten Analyse zur Verfügung. Es werden den Dachflächen (*RoofSurface1* und *RoofSurface2*) die Polygone 1 und 4 aus dem Analysemodell

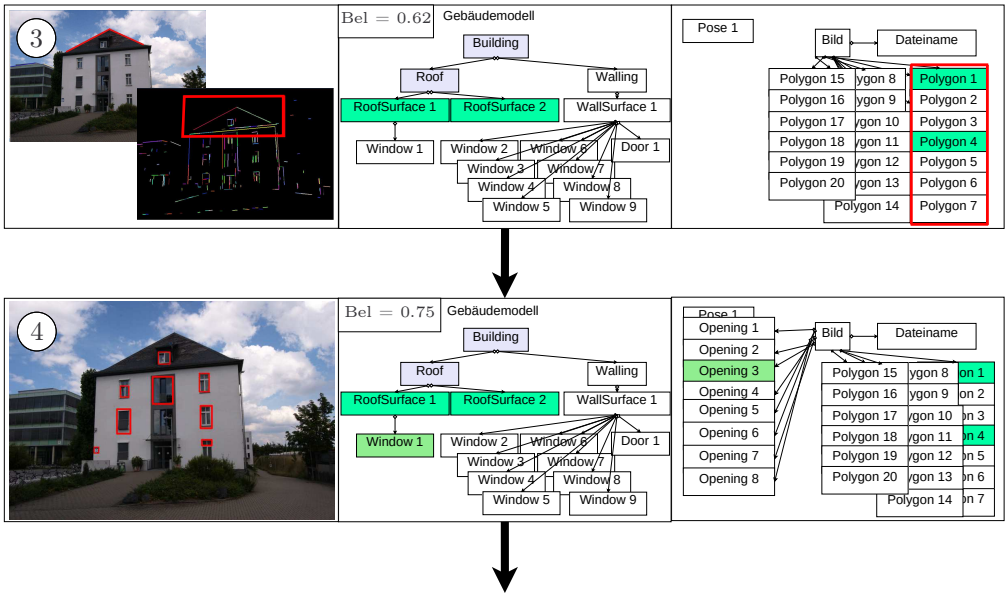


Abbildung 6.7: Fortsetzung Abbildung 6.6: Beispielhafter Ablauf der Kontrollstrategie.

zugeordnet. Die Zuordnungen werden mit einer grünen Einfärbung der Elemente kenntlich gemacht. Diese Zuordnungen erhalten jeweils einen Belief, der dann kombiniert den Belief in diese Pose und das Modell auf 0,62 erhöht. Dabei werden Modellelementen zugeordnete Bildelemente mit einem *GraphMarker* markiert, sodass Mehrfachzuordnungen ausgeschlossen werden können. Nun wird wieder expandiert und festgestellt, dass die Dachfläche (*RoofSurface 1*) als Bestandteil ein Fenster (*Window 1*) besitzt.

In Schritt 4 wird das in Kapitel 5.5 beschriebene Verfahren zur Extraktion von Fenster- und Türkandidaten (als *Opening* zusammengefasst) verwendet. Dabei werden alle Fenster- und Türkandidaten des Bildes extrahiert und in das Analysemodell eingefügt. Dieses Verfahren ist im Knotentyp *Opening*, der Fenster und Türen, in der Methode *initialize* verankert. Da die Kardinalität für die zuzuordnenden Fenster eins ist bzw. nur ein Fenster zugeordnet werden soll, wird ein Fenster direkt zugeordnet. In diesem Beispiel wird dem Modellfenster (*Window1*) das Fenster (*Opening3*) des Analysemodells zugeordnet (hell-grün markiert). Die Zuordnung erhöht den Belief auf 0,75. Nun sind für diesen Zweig des Modells alle möglichen Zuordnungen durchgeführt worden, sodass der nächste Bestandteil von *Building* expandiert wird. Dies ist in diesem Beispiel das Wandsystem (*Walling*), das auch keine eigene Initialisierungsmethode besitzt und eine Wandfläche (*WallSurfce 1*) als Bestandteil enthält.

In Schritt 5 wird aus den schon extrahierten Polygonen ein Polygon dem Modell zugeordnet (hell-grün markiert). Eine bereits erfolgte Bildanalyse (Ausführung

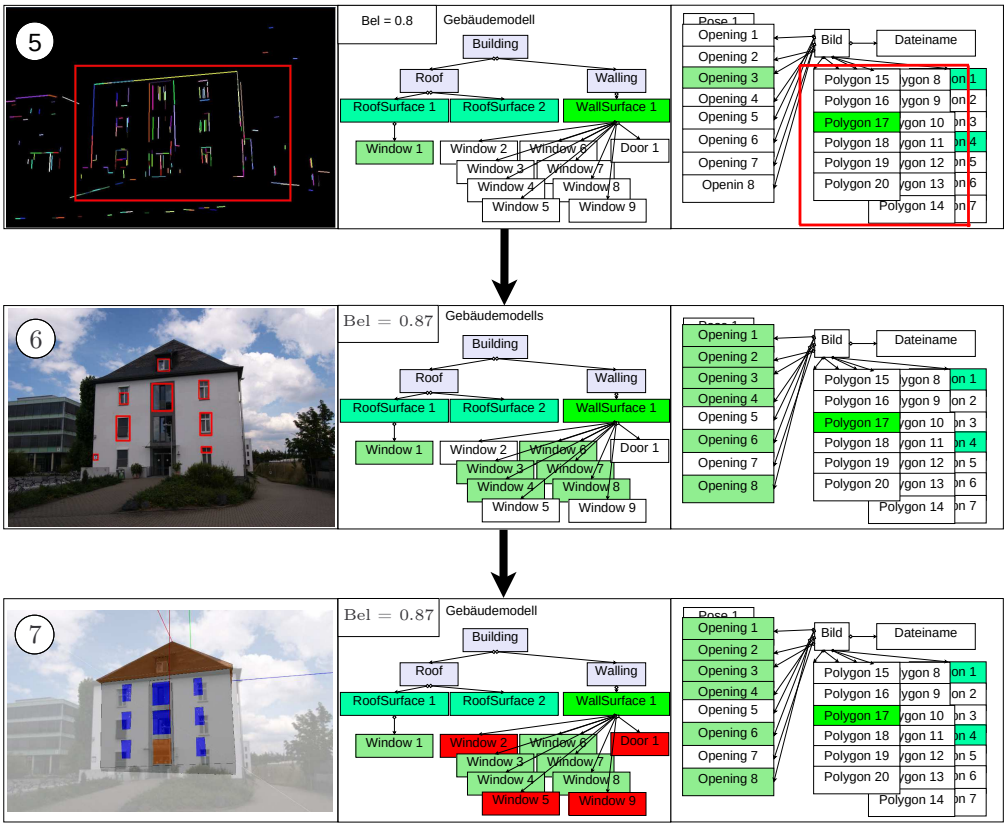


Abbildung 6.8: Fortsetzung Abbildung 6.7: Beispielhafter Ablauf der Kontrollstrategie.

initialize Methode des Knoten) eines Knotentyps wird nicht ein zweites Mal ausgeführt, wenn ein anderer Knoten des selben Typs in der Analyse an die Reihe kommt. Diese Zuordnung erhöht den Belief auf 0,81. Auch hier wird der Suchbereich über die Lage der Modellelemente eingeschränkt. Dies wird wieder durch die beiden roten Rechtecke symbolisiert. Das Modell wird expandiert und es wird festgestellt, dass die Wandfläche (*WallSurface1*) acht Fenster und eine Tür beinhaltet.

In Schritt 6 wird den Modellfenstern, mittels Ungarischer Methode, eine Teilmenge der schon extrahierten und noch nicht als zugeordnet markierten Fenster- und Türkandidaten *Openings* zugeordnet. Diese Zuordnungen (wieder grün markiert) erhöhen den Belief auf 0,87. Die Ungarische Methode kann zum Einsatz kommen, da die Kardinalität der enthaltenen Fenster und Türen neun beträgt und somit größer als eins ist. Nun wurde soweit möglich das komplette Modell zugeordnet.

Einige Bestandteile konnten nicht zugeordnet werden, was mit einer roten Markierung in Schritt 7 visualisiert wird. Trotz inkompletter Zuordnung konnte so eine plausible Pose mit Belief 0,87 berechnet und verifiziert werden.

6.4 Hypothesenraum

In dieser Arbeit kommen vier anwendungsunabhängige Regeln, kombiniert mit Suchalgorithmen, zur Bearbeitung des Kontrollproblems zum Einsatz. Dabei ordnet ein anwendungsunabhängiger und *A^*-basierter Kontrollalgorithmus* Modellelementen Bildelemente zu, füllt den Hypothesenraum mit Hypothesen und kontrolliert den Ablauf des Verfahrens, indem mittels A^*-Algorithmus die Hypothese zur weiteren Analyse ausgewählt wird, die am plausibelsten ist.

Definition 6.5 (Hypothesenraum - Konkretisierung)**.** *Der Hypothesenraum spannt einen Baum [GRRW10] mit Hypothesen auf und kann, da jede Hypothese ein Vertrauensmaß besitzt, schon während der Erstellung mit Graph-Suchalgorithmen durchsucht werden. Die Knoten des Baums bilden die Hypothesen. Jede Hypothese kennt seinen Vorgänger. Die Kontrolle nutzt den Hypothesenraum, um die Analyse zu steuern. Durch die Baumstruktur ist zu jeder Hypothese die Entstehungsgeschichte bekannt.*

Aus technischer Sicht wird der Hypothesenraum zweiteilig realisiert. Die Blattknoten des Hypothesenraums werden als PriorityQueue realisiert, da für die A^*-Suche nur die Blätter des Baums von Bedeutung sind. Um die Analyse vollständig nachzuvollziehen zu können, werden alle anderen Hypothesen in einer ArrayList abgelegt.

Im Laufe einer Bildanalyse wird es zu verschiedenen (sich widersprechenden) Zuordnungen kommen. Deshalb wird jede Kombination an Zuordnung als neue Hypothese verstanden. Jede entstehende Hypothese für eine Zuordnung wird im Hypothesenraum gespeichert.

Beispielsweise ist eine Zuordnung von Strecken zu den Polygonen des Modells in der Regel nicht eindeutig möglich (siehe Abbildung 6.6 Schritt 3 und 5), da mehrere plausible Streckenzuordnungen möglich sind. Auch im weiteren Ablauf kommt es dazu, dass eine Zuordnung mehrdeutig ist, wie auch in Schritt 4 der Abbildung 6.6 zu entnehmen ist. Hier wäre es beispielsweise denkbar, allen Fenstern ein Fenster eine Etage tiefer zuzuordnen.

Es entsteht durch die verschiedenen Hypothesen ein Baum, der sich während der Analyse immer weiter aufbaut. Die Wurzel des Baums (Hypothesenraums) ist stets die Hypothese die nur das ursprüngliche Gebäudemodell enthält. Initial werden Posehypothesen berechnet, die zu neuen Hypothesen werden und als neue Blattknoten in den Hypothesenraum eingehen (siehe Abbildung 6.3 und Pseudocode 6.1). Jede Hypothese kennt jeweils seinen Vorgänger.

Abbildung 6.9 visualisiert einen Ausschnitt aus einem Analyseprozess. Aus Analyseschritt 1 folgen drei potenziell mögliche Hypothesen (Hypothese 2, 3 und 4). Dementsprechend folgen in diesem Beispiel aus Hypothese 3 zwei Hypothesen, die Hypothesen 5 und 6. Und aus Hypothese 5 folgt Hypothese 7. Dabei gilt, dass der

Abstand von der Wurzel des Hypothesenraums zu einem Knoten angibt, wie weit die Analyse fortgeschritten ist. Je größer die Entfernung ist, umso weiter ist die Analyse. Ein fertiges Analyseergebnis befindet sich in einem Blattknoten. Der Weg von der Wurzel bis zu einer bestimmten Hypothese beschreibt alle Zwischenschritte und ermöglicht somit das Nachvollziehen der erfolgten Analyse.

Für die eigentliche Analyse ist aber nicht der ganze Baum vonnöten, sondern nur die Blattknoten des Hypothesenraums. Daher werden in der Analyse des in Abbildung 6.9 visualisierten Hypothesenraums nur die Hypothesen 2, 4, 6 und 7 für die weitere Analyse benötigt.

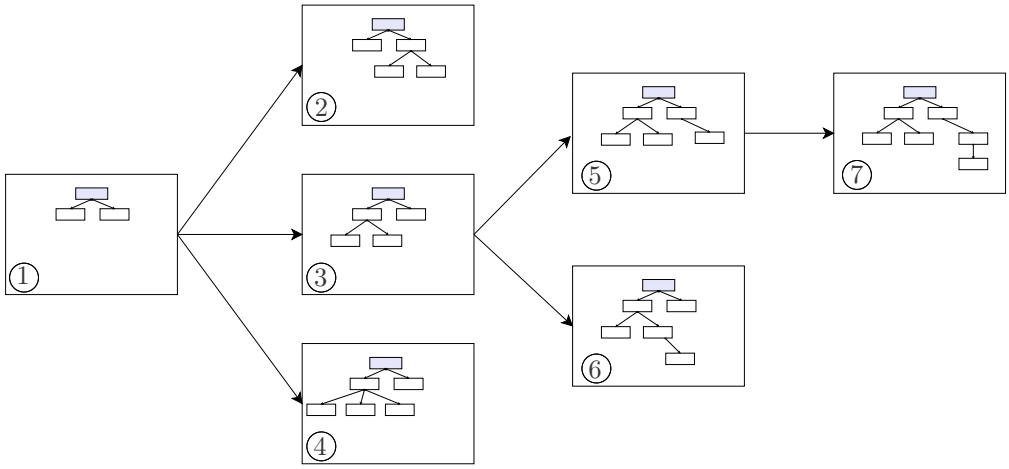


Abbildung 6.9: Hypothesenraum mit mehreren Hypothesen.

Ziel der Analyse ist es, das korrekte Ergebnis entsprechend der getätigten widerspruchsfreien Modellzuordnungen zu identifizieren. Es wird jedoch stets zu mehreren widerspruchsfreien Modellzuordnungen kommen, sodass es nicht möglich ist, exakt ein korrektes Ergebnis zu identifizieren. Daher sucht man nach dem Ergebnis, das am besten zu dem Modell passt. Dafür ist eine Suche im Hypothesenraum notwendig. Die Suche für die besten Zuordnungen kann als Suche nach dem kürzesten Pfad betrachtet werden, die sogar ausgeführt werden kann, wenn noch nicht alle Hypothesen erzeugt wurden. Da bei der Entstehung der Hypothesen noch nicht bestimmt werden kann, ob diese Hypothese in eine Sackgasse oder zum optimalen Ergebnis führt, müssen prinzipiell alle Hypothesen betrachtet werden. Eine Möglichkeit, dies zu tun, ist eine *vollständige Suche* im Hypothesenraum. Dies wäre aber eine sehr aufwändige Angelegenheit, bei der der Aufwand mit jeder weiteren Ebene exponentiell steigt.

Weitere Möglichkeiten bieten uninformierte Suchalgorithmen, wie die Breiten- oder Tiefensuche. Im Gegensatz zur *Breitensuche* wird bei der *Tiefensuche* zunächst ein Pfad vollständig analysiert, bevor abzweigende Pfade analysiert werden.

In dieser Arbeit wird auf eine *informierte Suche* zurückgegriffen. Informierte Suchen zeichnen sich dadurch aus, dass sie unter Verwendung von Heuristiken den Ablauf der Suche auf den Anwendungsfall optimieren. In dieser Arbeit wird der $\epsilon - A^*$-Algorithmus verwendet (siehe Gleichung 6.1). Dieser ist in der Lage, den kürzesten Pfad zwischen zwei Knoten in einem Graphen mit positiven Kantengewichten zu berechnen. Er kann in dieser Arbeit verwendet werden, da eine Bewertung der Hypothesen möglich ist (siehe Kapitel 8.2). Man benötigt im Hypothesenraum neben den Wahrscheinlichkeitszuordnungen eine Möglichkeit, um Hypothesen verschiedener Analyseschritte miteinander zu vergleichen. Der A^*-Algorithmus bietet die Möglichkeit Hypothesen, die unterschiedlich weit in der Analyse fortgeschritten sind, miteinander zu vergleichen. Kapitel 8 beschreibt die konkrete Realisierung der Vertrauensmaße für A^* für die Fallstudien.

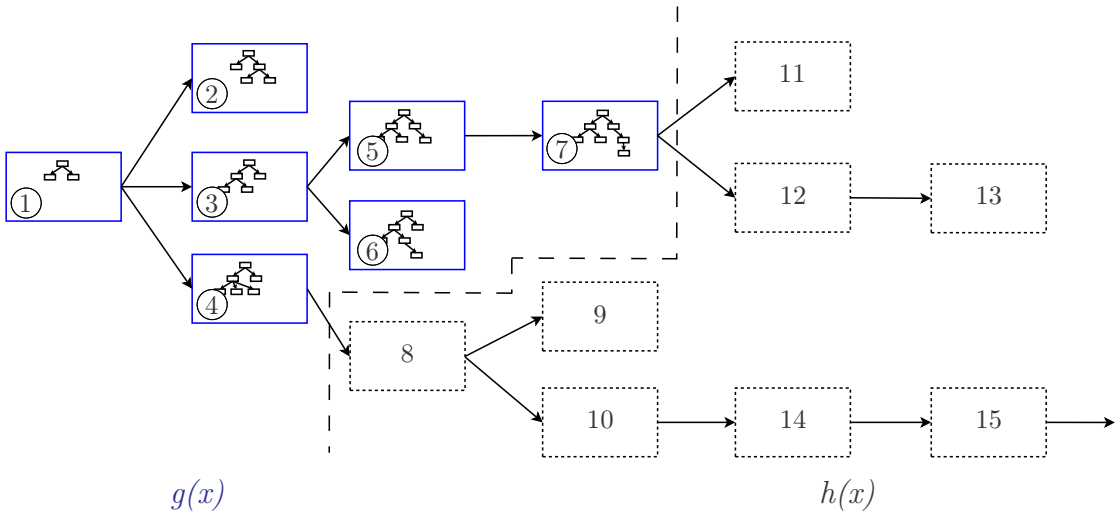


Abbildung 6.10: Hypothesen - A^*. Um die vielversprechendste Hypothese zu ermitteln, wird allen bekannten Hypothesen x jeweils ein Wert $f(x)$ zugeordnet, der angibt, wie hoch die Güte der analysierten Hypothese im günstigsten Fall ist (siehe Gleichung 6.1). Für eine Hypothese x bezeichnet $g(x)$ die bisher ermittelte Güte von der Starthypothese aus (linker Teil $g(x)$). Die optimal geschätzte Güte von x bis zur Zielhypothese (Analyseergebnis) wird mit $h(x)$ bezeichnet (rechter Teil $h(x)$).

Der A^*-Algorithmus untersucht die Hypothesen (Knoten im Hypothesenraum/-baum) zuerst, die wahrscheinlich schnell zum Ziel führen. Um die vielversprechendste Hypothese zu ermitteln, wird allen bekannten Hypothesen x jeweils ein Wert $f(x)$ zugeordnet, der angibt, wie hoch die Güte der analysierten Hypothese im günstigsten Fall ist. Die Hypothese mit der höchsten Güte (f-Wert) wird als nächstes untersucht. Für eine Hypothese x bezeichnet $g(x)$ die bisher ermittelte Güte von der Starthypothese aus (siehe Abbildung 6.10 linker Teil $g(x)$). Die optimal geschätzte Güte von x bis zur Zielhypothese (Analyseendergebnis) wird mit $h(x)$

bezeichnet (siehe Abbildung 6.10 rechter Teil $h(x)$). Die verwendete Heuristik darf die Güte nie unterschätzen. Diese Einschränkung ist nicht in jedem Fall leicht zu erreichen, denn wird die Güte unterschätzt, kann es passieren, dass das optimale Ergebnis nie gefunden wird. Wird hingegen die Güte sehr weit überschätzt, kann dies zu einer enormen Ausweitung der Suche bis hin zur Breitensuche führen.

Um die Zielstrebigkeit des Verfahrens zu erhöhen, wird der $\epsilon - A^*$-*Algorithmus* [Pea84] verwendet, mit

$$f(x) = g(x) + \epsilon\ h(x) \quad , \tag{6.1}$$

wobei $0 \leq \epsilon \leq 1$ gilt. Dabei kann es nun zu einer Unterschätzung der zukünftigen Güte kommen. Pearl [Pea84] konnte aber belegen, dass der Verlust an Optimalität im ungünstigsten Fall auf $\frac{1-\epsilon}{\epsilon}$ Prozent der optimalen Lösung begrenzt ist. Die konkrete Umsetzung von Gleichung 6.1 wird in Kapitel 8.4 beschrieben.

Die A^*-Suche läuft nicht parallel zur Kontrollstrategie, sondern ist in der Bewertung der einzelnen Hypothesen enthalten. Jede neue Hypothese kann konform zu A^* bewertet werden und entsprechend einfach dann in eine priorisierte Warteschlange einsortiert werden. Die am besten bewertete Hypothese wird zur Analyse der Warteschlange entnommen, bewertet und wieder als neue Hypothese in die priorisierte Warteschlange einsortiert. Die alte Hypothese wird zur Nachvollziehbarkeit der Analyse separat gespeichert.

Definition 6.6 (Hypothese - Konkretisierung)**.** *Eine Hypothese im Verständnis dieser Arbeit beschreibt eine oder mehrere Zuordnungen zwischen Elementen des Analysemodells und eines Modells. Jede Hypothese besitzt eine Bewertung in Form eines Vertrauensmaßes. Dieses Vertrauensmaß setzt sich konform zu A^* zusammen aus $f(x) = g(x) + \epsilon\ h(x)$ (siehe Kapitel 8.4). Dabei entspricht $g(x)$ der Bewertung der bisher erfolgten Zuordnungen. Die Bewertung für $h(x)$ wird bestimmt indem alle noch offenen Zuordnung als perfekt angenommen werden. Es gilt $\epsilon = 0.7$ in dieser Arbeit.*

Aus technischer Sicht wird eine Hypothese definiert durch: eine Modell (inkl. erzeugter 2-D-Geometrie), eine Pose, eine Sammlung von Zuordnungen mittels InitialisationConcept, eine Sammlung von Suchraumeinschränkungen LimitationConcept und einer Bewertung der Hypothese. Zudem ist jeder Hypothese die Vorgängerhypothese bekannt, sodass der Analyseprozess vollständig nachvollzogen werden kann. Jede Hypothese speichert als Bewertung sowohl $g(x)$ als auch $f(x)$. $f(x)$ dient zur Sortierung der Hypothesen und $g(x)$ ist nötig, um im nächsten Schritt wieder $h(x)$ berechnen zu können.

6.5 Diskussion

Das Einbinden von Wissen in die Bildanalyse ist der Leitgedanke der gewählten Strategie. Zum einen werden Modelle verwendet, die anwendungsspezifisches Wissen sowohl im deklarativen als auch prozeduralen Bereich besitzen. Diese Modelle schränken über Bestandteils- und Konkretisierungskonzepte die Bildanalyse ein, sodass in einer frühen Phase der Analyse gewonnenes Wissen den späteren Analyseprozess beeinflusst. Zum anderen wird über die Bildanalyse selbst Wissen über das gesuchte Modell aufgebaut und wieder dazu verwendet, den Analyseprozess zu steuern. Dies geschieht beispielsweise bei der Extraktion von Fensterkandidaten (siehe Kapitel 5.5). Die Strategie ist dabei anwendungsunabhängig, was durch die verwendeten Fallstudien belegt wird, und ist durch vier Regeln definiert. Dennoch wird eine anwendungsabhängige Einschränkung getroffen. Es wird davon ausgegangen, dass es entweder durch prozedurales Wissen möglich ist, 3-D-Hypothesen zu erzeugen oder dass eine 2-D-Repräsentation schon vorliegt.

Die bei der Analyse entstehenden Hypothesen werden im Hypothesenraum gespeichert. Ein Hypothese beinhaltet dabei eine konsistente Bildinterpretation und plausible Zuordnungen zu den Modellelementen. Die Analyse schränkt die Bildinterpretation so ein, dass aus einer Hypothese nie eine inkonsistenter Hypothese erwächst. Somit kann der Endzustand der Analyse direkt als Ergebnis des Verfahrens verwendet werden. Es entstehen während des Ablaufs der Analyse eine Reihe von Zwischenergebnissen in Form von Hypothesen, die sich durchaus widersprechen können. Dadurch lässt sich aber ein Analyseergebnis Schritt für Schritt erklären, was nicht selbstverständlich ist, wenn man an Blackbox-Verfahren wie beispielsweise neuronale Netze denkt.

Zur Suche im Hypothesenraum kommt mit $\epsilon - A^*$ ein heuristisches Suchverfahren zum Einsatz. Gütefunktionen zur Steuerung des Suchverfahrens sind im Allgemeinen schwer zu ermitteln, da hierfür eine Abschätzung der Güte der noch zu entstehenden Zuordnungen gemacht werden muss. Eine Möglichkeit zur Bewertung von Hypothesen und Zuordnungen wird in Kapitel 8.2 vorgestellt. Dabei sind die Bewertungsfunktionen nicht Teil der unabhängigen Kontrolle, sondern sind fester Bestandteil des Modells. Nur ein Modellelement selbst "weiß", wie es sich zu bewerten hat. Die Entstehung von konkreten Hypothesen wird im nächsten Kapitel näher beschrieben.

Kapitel 7

Hypothesengenerierung

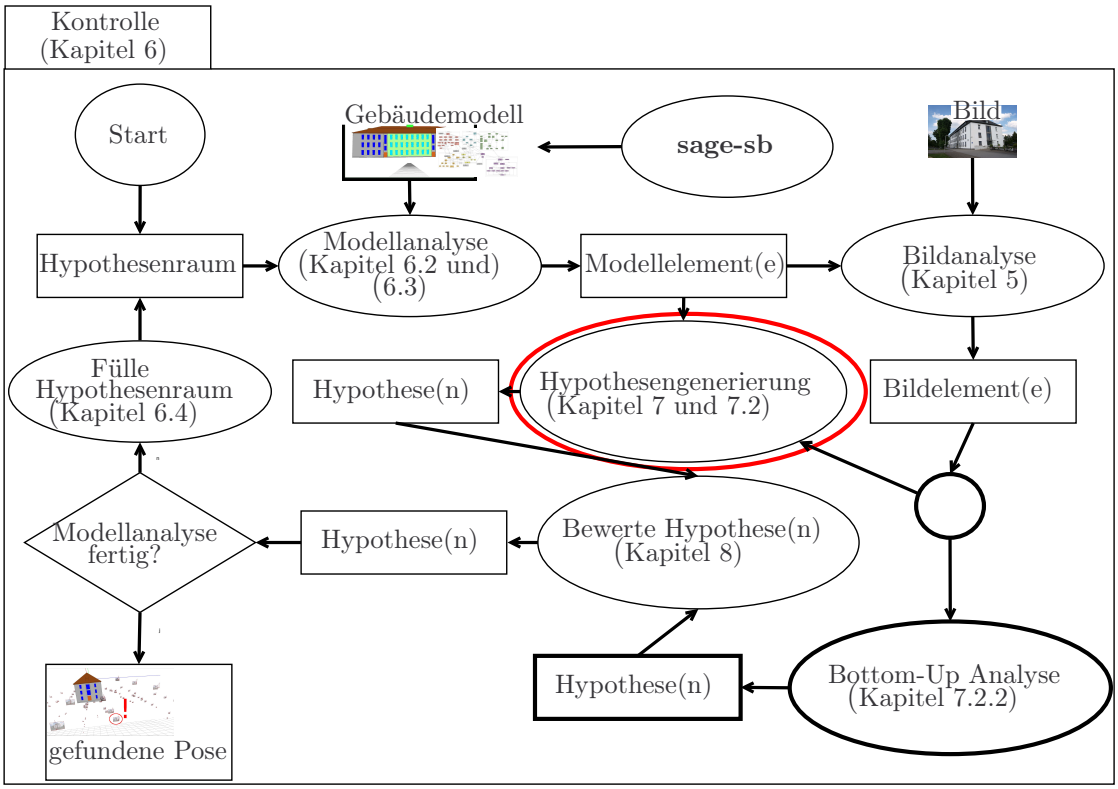


Abbildung 7.1: Im Zuge der Analyse entstehen Hypothesen (siehe rote Ellipse), wobei eine Hypothese (in der Fallstudie "Poseschätzung von Gebäuden") initial aus einer Posehypothese entsteht (Abschnitt 7.2). Diese werden in einem Hypothesenraum verwaltet.

Eine anwendungsunabhängige Kontrolle steuert den Ablauf der Modellanalyse (Abbildung 7.1). Im Zuge der Analyse entstehen Hypothesen, wobei eine Hypothese (in der Fallstudie "Poseschätzung von Gebäuden") initial aus einer Posehypothese

entsteht (Abschnitt 7.2). Diese werden in einem Hypothesenraum verwaltet. Dies erfordert die Möglichkeit Hypothesen zu bewerten und zu verifizieren (Kapitel 8). Das Ergebnis der Analyse ist eine Pose inklusive Bewertung.

Die Kontrolle (siehe Kapitel 6) generiert im Fall der Gebäudefallstudie **knoPoE** zu Beginn sowie im Laufe des Verfahrens Posehypothesen (vgl. Kapitel 2.3). Die Fallstudien zur Dominostein- und Pokerkartenerkennung benötigen diesen Schritt nicht, da ihre Geometrie schon in 2-D vorliegt. Eine *Hypothese*, im Verständnis dieser Arbeit, beschreibt eine oder mehrere Zuordnungen zwischen Elementen des Analysemodells und einem Modell. Unterschiedliche Modellelementtypen besitzen über Implementierung der Methode *instantiate* (siehe beschriebene Interfaces in Kapitel 3.4) jeweils die Möglichkeit zur Erzeugung dieser Hypothesen (beispielsweise *TwoDRectangle* oder *Opening*). Die Erzeugung einer Hypothese ist eng verknüpft mit der Extraktion der Elemente aus dem Bild, da in der Regel Elemente extrahiert werden, die eine gewisse Plausibilität besitzen. Daher ist die Erzeugung von Hypothesen, also die Zuordnung von Elementen des Analysemodells mit den Elementen der Modells, eng verknüpft mit der Extraktion der Informationen. Im Folgenden wird zum einen die Extraktion von Bildelementen - gesteuert durch Methoden der Modellelemente (siehe Interfaces in Kapitel 3.4) - und zum anderen die Entstehung von Posehypothesen betrachtet.

7.1 Dominostein- und Pokerkartenerkennung

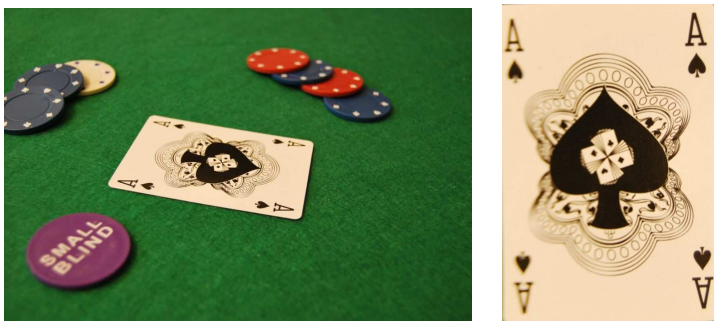
Zur Generierung der Hypothesen für die Fallstudien der Dominostein- und Pokerkartenerkennung wird eine 2-D-Welt angenommen, sodass keinerlei Aufwand in die Transformation von 3-D in 2-D entsteht. Der Schwerpunkt dieser Fallstudien liegt darin zu bewerten, ob modellbasierte Objekterkennung mit symbolischen Beschreibungen adäquat funktioniert. Hypothesen basieren auf den extrahierten geometrischen Objekten, korrespondierend zu den Elementen der entsprechenden Modelle (siehe Kapitel 3.4.1). Die entsprechend nötigen Segmentierungen werden im Folgenden beschrieben.

Die *Kreiserkennung* arbeitet mit einer Hough-Transformation und nutzt dafür die C++ PUMA-Bibliothek [Pau03]. Dabei liegt die Erkennungsrate in unserer Fallstudie bei 95,3% mit einer False-Positiv-Rate von 2,1%. Die Funktion wurde mittels JNI als Komponente in das STOR-Komponentenkonzept integriert.

Die *Viereckerkennung* detektiert Vierecke im Bild und die *Rektifizierung* berechnet die Homographie zwischen den gefundenen Vierecken und dem gegebenen Modellrechteck und projiziert das Viereck in ein Rechteck mit den Maßen des Modellrechtecks. Grundlegende Funktionen für diese Methode wurden der Bibliothek *OpenCV* entnommen und mit der Bibliothek *JavaCV*[1] in das STOR-Komponen-

[1] `code.google.com/p/javacv/`

tenkonzept integriert. Die Erkennungsrate liegt in dieser Fallstudie bei 98,4%, allerdings mit einer False-Positiv-Rate von 13,1%, die während der Bildanalyse durch die Kontrolle gefiltert werden muss. Im Laufe der Analyse werden die extrahierten Vierecke rektifiziert, indem aus den Korrespondenzen zwischen Bildviereck und Modellrechteck die Homographie bestimmt wird und das Bildviereck entsprechend projiziert wird. Abbildung 7.2 zeigt ein beispielhaftes Ergebnis der Viereckerkennung mit anschließender Rektifizierung.



(a) Pokerkarten-Bild (b) Rektifizierte Pokerkarte

Abbildung 7.2: Beispiel eines Pokerkarten-Bildes (a) und die detektierte und rektifizierte Pokerkarte (b).

Im Folgenden wird nur die *Kartenfarbenerkennung* beschrieben, da Kartenfarbenerkennung und *Buchstabenerkennung* in dieser Fallstudie äquivalent umgesetzt wurden. Zunächst werden für die Kartenfarbenerkennung manuell Regionen extrahiert und für diese Regionen die Hu-Momente und die Kreisförmigkeit berechnet. Diese Daten werden den Regionen (*Region*) der Pokerkartenmodelle passend zu den Kartenfarben (*SuitForm*) als Attribut hinzugefügt (vgl. Abbildung 3.8), sodass sichergestellt ist, dass diese Daten bei der Analyse zur Verfügung stehen.

Mittels Kontursuche werden zuerst potenzielle Regionen für die Kartenfarbe im Bild lokalisiert. Für diese Regionen wird die Farbe (rot oder schwarz) identifiziert, die *Hu-Momente* [Hu62] berechnet und die *Kreisförmigkeit* (Gleichung 7.1) bestimmt:

$$\text{Kreisförmigkeit} = \frac{4\pi \text{ Flächeninhalt}}{\text{Umfang}^2} \,. \tag{7.1}$$

Diese Merkmale ι_i werden mit den vorhandenen Daten ι_m verglichen:

$$\mathrm{d}(\{\iota_m\}, \{\iota_i\}) = \sum_k^N \left(\frac{\iota_{m_k} - \iota_{i_k}}{\iota_{m_k}} \right). \tag{7.2}$$

Die Ergebnisse für die Kartenfarbenerkennung sind mit einem Recall (False-Positiv-Rate) von 93,2%, einer Genauigkeit (positiver prädiktiver Wert) von 96,1% und einer Spezifität (True-Negativ-Rate) von 93,4% bei entzerrten Bildern gut.

Die Buchstabenerkennung in unserer Fallstudie muss nur die Buchstaben A, J, K und Q erkennen. Es tritt das Problem auf, dass die Segmentierung die feingliedrigen Strukturen der Buchstaben oft übersegmentiert und so die Buchstaben in viele kleine Regionen zerfallen. Dies ist der Grund dafür, dass die Erkennung mit einem Recall von 69,0%, einer Genauigkeit von 72,2% und einer Spezifität von 78,6% vergleichsweise schlecht ausfällt. Allerdings kann gezeigt werden, dass die modellbasierte Erkennung dies kompensieren kann.

Grundlegende Funktionen für diese Methode wurden der Bibliothek OpenCV entnommen und mit der Bibliothek *JavaCV* in das STOR-Komponentenkonzept integriert.

7.2 Poseschätzung von Gebäuden

Zur Bestimmung der Pose von komplexen 3-D-Objekten, wie sie Gebäude darstellen, ist es unumgänglich zur Generierung von Hypothesen, die im Bild enthaltenen Informationen möglichst gut zu extrahieren. Im Folgenden werden die verwendeten Verfahren beschrieben, die für die Bestimmung der Pose von Gebäuden benötigt werden, so dass man aus einem 3-D-Modell eine überschaubare Zahl Hypothesen in Form von 2-D-Abbildungen des Modells erhält.

7.2.1 Poseschätzung mit Hilfe von Fluchtpunkten

Zunächst werden zur Poseschätzung die Fluchtpunkte, wie in Abschnitt 5.1 beschrieben, berechnet (siehe Abbildung 7.5(a)). Unter Verwendung der Fluchtpunkte ist man in der Lage, die Drehung der Kamera im Verhältnis zum Modellkoordinatensystem zu bestimmen. Um die Verschiebung der Kamera zu bestimmen, benötigt man mindestens eine Kante im Bild, deren Länge man in der 3-D-Welt kennt. Aus diesem Grund wird der Himmel des Bildes [SP09a] extrahiert (siehe Abbildung 7.3(a), 7.5(b) und 7.5(c)). Es werden wie in Abschnitt 5.3 beschrieben, die interessanten Liniensegmente extrahiert (siehe Abbildung 7.3(c)).

Mit dem in Abschnitt 5.4 beschriebenen Verfahren ist man in der Lage Dachkanten zu extrahieren. Die extrahierten Fluchtpunkte und die Dachkanten dienen als Eingabe für das Verfahren zur Generierung von Posehypothesen. Da man nicht wissen kann, welche Dachkanten gefunden wurden und welche Gebäudeseite man sieht, müssen für ein Sattel- bzw. Walmdach 32 Posehypothesen bestimmt werden, da 2^3 Möglichkeiten (gefundene Kanten/Modellkanten) existieren, und das jeweils für jede der vier Gebäudeseiten.

(a) Dachkanten-Binärbild (b) Canny-Strecken (c) interessante Liniensegmente

Abbildung 7.3: Binärbild extrahiert mittels Horizontlinienerkennung [SP09a] (a), Kantenbild mittels Canny (b) und die interessanten Liniensegmente (c).

Werden drei Fluchtpunkte gefunden, sind es sogar 64 Posehypothesen, da man durch mögliche Ungenauigkeiten in der Bestimmung der Fluchtpunkte zwei mögliche Poseschätzungen erhält. Abbildung 7.4 gibt einen Überblick über den Prozess der Poseschätzung. Diese Posen werden verwendet, um die 3-D-Modelle mit der im Bild sichtbaren 2-D-Geometrie mittels semantischem Rendering zu erweitern (siehe Kapitel 3.5).

Das verwendete Verfahren von Guillou et al. [GMMB00] wird im Folgenden im Detail beschrieben und evaluiert.

Annahmen des Verfahrens Das Verfahren zur Poseschätzung [GMMB00] (graue Ellipse) trifft vier grundlegende Annahmen, die für die Berechnung der Brennweite, Rotation und Translation erfüllt sein müssen:

(i) Es müssen mindestens zwei Fluchtpunkte im Bild gefunden werden,
(ii) die Länge einer Strecke des Modells muss bekannt sein,
(iii) der Hauptpunkt ist der Mittelpunkt des Bildes und
(iv) das Seitenverhältnis des Bildes ist bekannt.

Als Kameramodell wird die Lochkamera angenommen. Das *Kamerazentrum* befindet sich in 0^c und die *Projektion des Kamerazentrums* entspricht dem Hauptpunkt H. Zwei Mengen paralleler Geraden definieren zwei Fluchtpunkte. Dabei wird außerdem angenommen, dass die Linien aus der einen Menge senkrecht auf der anderen Menge stehen. Werden nun aus jeder Menge zwei Geraden ausgewählt, bilden deren Schnittpunkte ein Rechteck $a^c b^c c^c d^c$. Die Richtungsvektoren der Geraden $\overline{\boldsymbol{a}^m \boldsymbol{d}^m}$ und $\overline{\boldsymbol{a}^m \boldsymbol{b}^m}$ sind im Folgenden $\boldsymbol{u}$ und $\boldsymbol{v}$. Die Geraden $\overline{\mathsf{a}^c\mathsf{d}^c}$ und $\overline{\mathsf{b}^c\mathsf{c}^c}$ haben denselben Fluchtpunkt $vp_u = \overline{\mathsf{a}^c\mathsf{d}^c} \cap \overline{\mathsf{b}^c\mathsf{c}^c}$. Genauso haben die Geraden $\overline{\mathsf{a}^c\mathsf{b}^c}$ und $\overline{\mathsf{d}^c\mathsf{c}^c}$ denselben Fluchtpunkt $vp_v = \overline{\mathsf{a}^c\mathsf{b}^c} \cap \overline{\mathsf{d}^c\mathsf{c}^c}$.

Das Verfahren von Guillou et al. [GMMB00] ist in der Lage die Brennweite zu bestimmen (Siehe Anhang B). In dieser Arbeit wird allerdings das in Kapitel 3.1 beschriebene Verfahren zur Bestimmung der Brennweite verwendet.

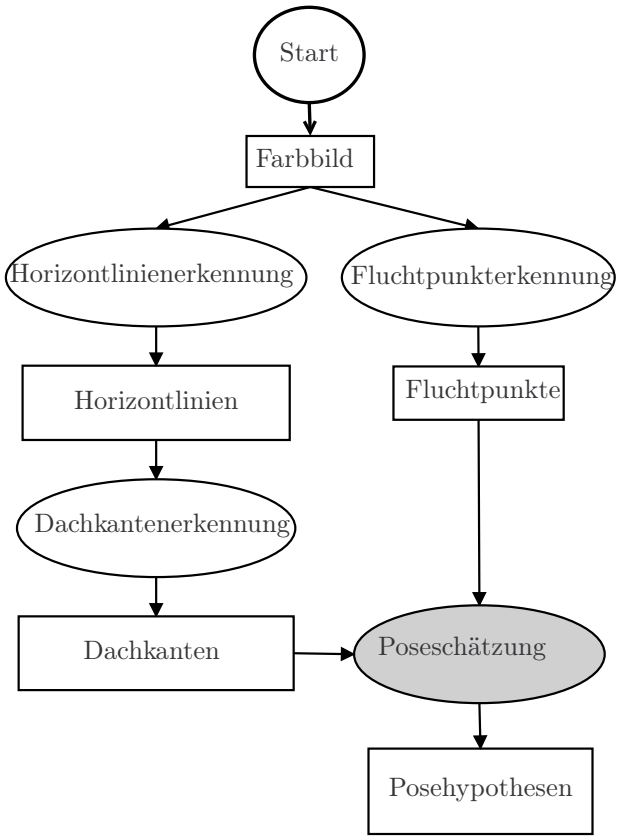
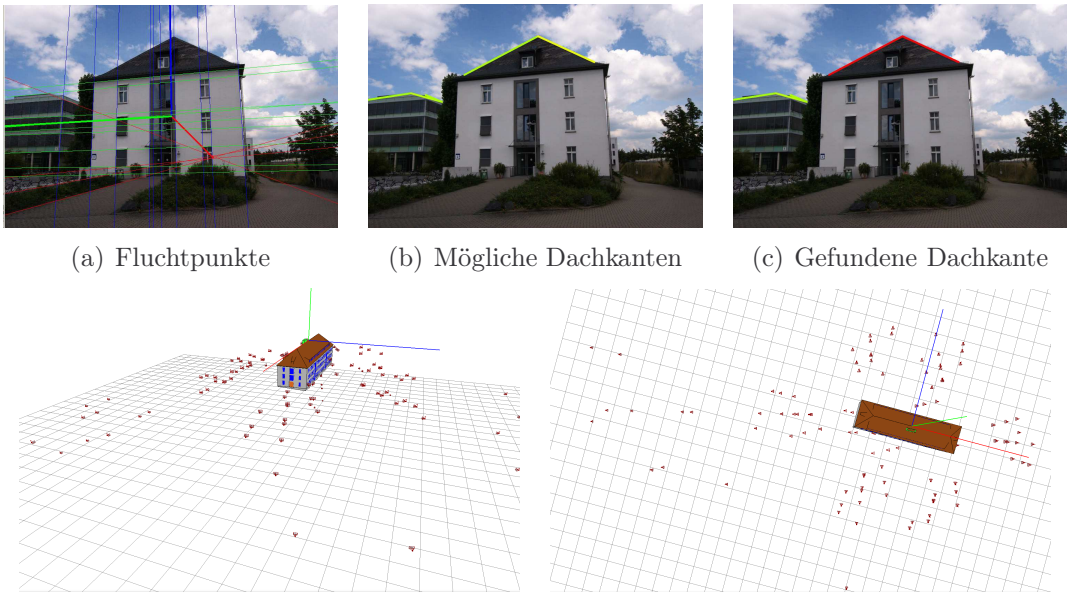


Abbildung 7.4: Datenflussdiagramm der nötigen Vorarbeiten zur Poseschätzung (graue Ellipse).

Berechnung der Rotationsmatrix Bei der Bestimmung der Rotation zwischen Kamerakoordinatensystem und Modellkoordinatensystem bildet ein beliebiger Punkt des Modellkoordinatensystems den Ursprung des Koordinatensystems, beispielsweise $\boldsymbol{a}^{\mathrm{m}}$. Es wird das Modellkoordinatensystem durch die drei Basisvektoren $\boldsymbol{u}$, $\boldsymbol{v}$ und $\boldsymbol{w} = \boldsymbol{u} \times \boldsymbol{v}$ mit Ursprung im Punkt $\boldsymbol{a}^{\mathrm{m}}$ definiert und das Kamerakoordinatensystem über die drei Basisvektoren $\boldsymbol{i}$, $\boldsymbol{j}$ und $\boldsymbol{k}$ und Ursprung im Punkt 0^{c} (siehe Abbildung 7.6). Das Ziel ist es nun, eine Rotationsmatrix zu bestimmen, sodass gilt:

$$\boldsymbol{R}\boldsymbol{i} = \boldsymbol{u},\ \boldsymbol{R}\boldsymbol{j} = \boldsymbol{v},\ \boldsymbol{R}\boldsymbol{k} = \boldsymbol{w}. \tag{7.3}$$

Da die beiden Geraden mit dem Richtungsvektor $\boldsymbol{u}'$ beziehungsweise $\boldsymbol{v}'$ und Punkt im Ursprung 0^{c} entweder durch den Fluchtpunkt ${vp_u}^{\mathrm{c}}$ oder ${vp_v}^{\mathrm{c}}$ führen, entsprechen die Basisvektoren ($\boldsymbol{u}$, $\boldsymbol{v}$, $\boldsymbol{w}$) den Basisvektoren ($\boldsymbol{u}'$, $\boldsymbol{v}'$, $\boldsymbol{w}'$). Daher ist die Rotation zwischen dem Kamerakoordinatensystem (0^{c}, $\boldsymbol{i}$, $\boldsymbol{j}$, $\boldsymbol{k}$) und (A^{c}, $\boldsymbol{u}$, $\boldsymbol{v}$, $\boldsymbol{w}$) sowie zwischen Kamerakoordinatensystem (0^{c}, $\boldsymbol{i}$, $\boldsymbol{j}$, $\boldsymbol{k}$) und (0^{c}, $\boldsymbol{u}'$, $\boldsymbol{v}'$, $\boldsymbol{w}'$) identisch. Die Vektoren $\boldsymbol{u}'$, $\boldsymbol{v}'$ und $\boldsymbol{w}'$ sind im Kamerakoordinatensystem bekannt:



(a) Fluchtpunkte (b) Mögliche Dachkanten (c) Gefundene Dachkante

(d) Mögliche Hypothesen (e) Mögliche Hypothesen

Abbildung 7.5: Vorberechnungen für die Poseschätzung: erst Fluchtpunktbestimmung (a), mögliche Dachkanten in gelb (b) und die erkannten Dachkanten in rot (c). Die Poseschätzung ist dann in der Lage Posehypothesen zu generieren (d)(e).

$$\boldsymbol{u}' = \frac{{vp_u}^{\mathrm{c}}}{\|{vp_u}^{\mathrm{c}}\|} \tag{7.4}$$

$$\boldsymbol{v}' = \frac{{vp_v}^{\mathrm{c}}}{\|{vp_v}^{\mathrm{c}}\|} \tag{7.5}$$

$$\boldsymbol{w}' = \boldsymbol{u}' \times \boldsymbol{v}'. \tag{7.6}$$

Daher gilt auch:

$$\boldsymbol{Ri} = \boldsymbol{u}',\ \boldsymbol{Rj} = \boldsymbol{v}',\ \boldsymbol{Rk} = \boldsymbol{w}'. \tag{7.7}$$

Da bekannt ist, dass $\boldsymbol{i} = (1,0,0)$, $\boldsymbol{j} = (0,1,0)$ und $\boldsymbol{k} = (0,0,1)$ gilt, können die Gleichungen 7.4, 7.5, 7.6 und 7.7 genutzt werden, um die Rotationsmatrix zu bestimmen:

$$\boldsymbol{R} = \begin{pmatrix} \frac{vp_{u_i}}{\sqrt{{vp_{u_i}}^2+{vp_{u_j}}^2+f^2}} & \frac{vp_{v_i}}{\sqrt{{vp_{u_i}}^2+{vp_{u_j}}^2+f^2}} & \boldsymbol{w}'_i \\ \frac{vp_{u_j}}{\sqrt{{vp_{u_i}}^2+{vp_{u_j}}^2+f^2}} & \frac{vp_{v_j}}{\sqrt{{vp_{u_i}}^2+{vp_{u_j}}^2+f^2}} & \boldsymbol{w}'_j \\ \frac{f}{\sqrt{{vp_{u_i}}^2+{vp_{u_j}}^2+f^2}} & \frac{f}{\sqrt{{vp_{u_i}}^2+{vp_{u_j}}^2+f^2}} & \boldsymbol{w}'_k \end{pmatrix}. \tag{7.8}$$

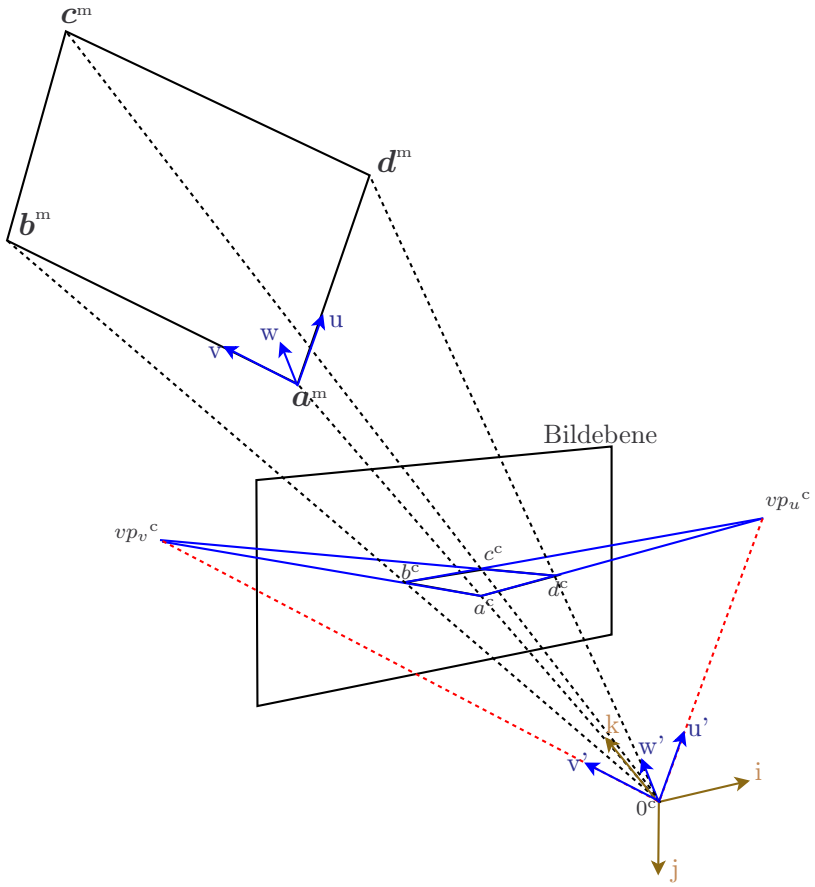


Abbildung 7.6: Berechnung der Rotationsmatrix.

Berechnung des Translationsvektors Nun gilt es, die Translation zwischen dem Kamerakoordinatensystem und dem Modellkoordinatensystem zu bestimmen. Dies veranschaulicht Abbildung 7.7. Dabei stellt a'^c die perspektivische Projektion von a^c und $\overline{\mathsf{a'^c p'^c}}$ die perspektivische Projektion der Strecke $\overline{\mathsf{a^c p^c}}$, die parallel zur Achse $\boldsymbol{u}$ verläuft, dar. Um die Translation zu bestimmen, wird angenommen, dass die Länge l der Strecke $\overline{\mathsf{a^c p^c}}$ bekannt ist. Andernfalls ist die Translation nur bis zu einem Skalierungsfaktor bestimmbar.

Der Schnittpunkt der Strecke $\overline{\mathsf{0^c p^c}}$ mit der Strecke $\overline{\mathsf{a'^c p''^c}}$ ist p''^c. Daher gilt nach dem *Strahlensatz* beziehungsweise *Satz des Thales* für die Dreiecke $0^c a'^c p''^c$ und $0^c a^c p^c$:

$$\frac{\left\|\overline{\mathsf{a'^c p''^c}}\right\|}{\left\|\overline{\mathsf{a^c p^c}}\right\|} = \frac{\left\|\overline{\mathsf{0^c a'^c}}\right\|}{\left\|\overline{\mathsf{0^c a^c}}\right\|} \tag{7.9}$$

oder umgeformt

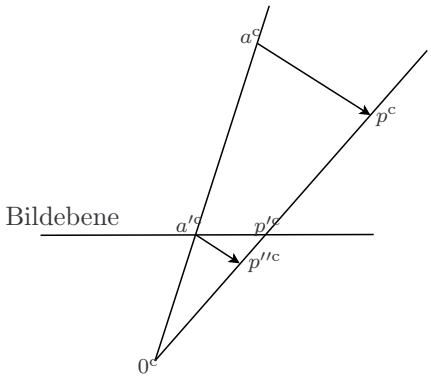


Abbildung 7.7: Berechnung des Translationsvektors.

$$\left\|\overline{0^c a^c}\right\| = \frac{\left\|\overline{0^c a'^c}\right\| \left\|\overline{a^c p^c}\right\|}{\left\|\overline{a'^c p''^c}\right\|}. \tag{7.10}$$

Hieraus folgt für die Translation:

$$\boldsymbol{t} = a^c - 0^c = \left\|\overline{0^c a^c}\right\| \frac{\boldsymbol{a}}{\left\|\overline{0^c a'^c}\right\|}, \text{ mit } \boldsymbol{a} = \boldsymbol{a'}^m - \boldsymbol{0}^m. \tag{7.11}$$

Alternative Berechnung des Translationsvektors Basierend auf der Idee von Prof. Dr. Frank[2] wird hier eine Alternative zur Bestimmung der Translation beschrieben. Basis der Idee ist das Doppelverhältnis (siehe Abbildung 7.8):

$$\mathrm{DV}(A^c, B^c, C^c, D^c) = \frac{\overline{A^c C^c}}{\overline{B^c C^c}} : \frac{\overline{A^c D^c}}{\overline{B^c D^c}}. \tag{7.12}$$

Das *Doppelverhältnis* ist eine Invariante jeder projektiven Abbildung. Falls man drei Punkte auf einer Strecke und die entsprechenden Abstände auf der anderen Strecke kennt, kann man dies nutzen, um die Ausrichtung und die korrespondierenden Punkte zu bestimmen. Dies liegt an der der Parallelität der Strecken $\Rightarrow \overline{0D} \parallel \overline{A'C'}$ (siehe Abbildung 7.9).

$$\mathrm{DV}(A^c, B^c, C^c, D^c) = \mathrm{DV}(A'^c, B'^c, C'^c, D'^c) \tag{7.13}$$

$$\mathrm{DV}(A'^c, B'^c, C'^c, \infty) = \frac{\overline{A'^c C'^c}}{\overline{B'^c C'^c}} = \mathrm{TV}(A'^c, B'^c, C'^c) \tag{7.14}$$

$$\mathrm{DV}(A'^c, B'^c, C'^c, \infty) = \mathrm{DV}(A^c, B^c, C^c, D^c) \tag{7.15}$$

[2] `https://www.uni-koblenz-landau.de/de/koblenz/fb3/mathe/mitglieder/rolfdieter-frank`

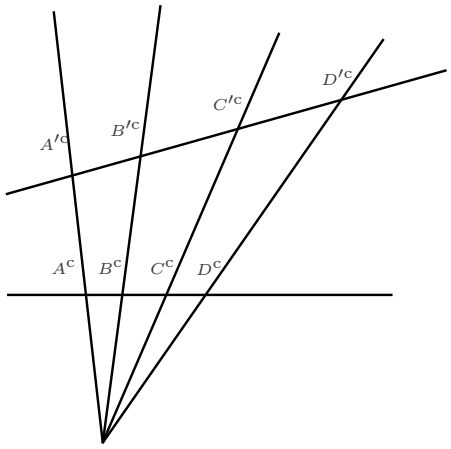


Abbildung 7.8: Darstellung des Doppelverhältnis.

Aus den Gleichungen 7.13, 7.14 und 7.15 kann der gesuchte Punkt D^c bestimmt werden.

Evaluation Zur Evaluation wird ein einfaches Haus aus drei Entfernungen und Sichtwinkeln von 0° bis 70° gerendert. Bei dem Haus selbst handelt es sich um ein fiktives Haus. Für diese 24 Häuser werden manuell die Fluchtpunkte bestimmt, um einen störenden Einfluss bei der Evaluation des Verfahrens durch Ungenauigkeiten bei der Fluchtpunktschätzung zu vermeiden.

In der Evaluation wird untersucht, wie gut das Verfahren die Pose bestimmt. Dafür wird der Rückprojektionsfehler verwendet. Der Rückprojektionsfehler misst den euklidischen Abstand zwischen den mittels berechneter Rotation und Translation in das Bild projizierten Punkten und den korrespondierenden Punkten im

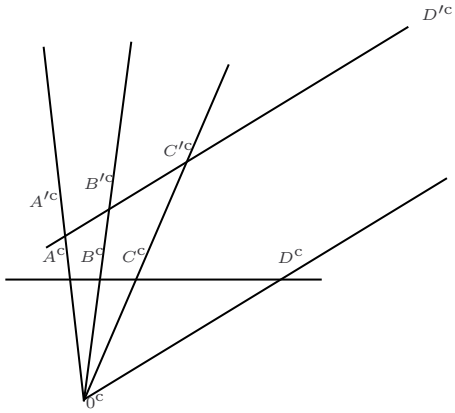
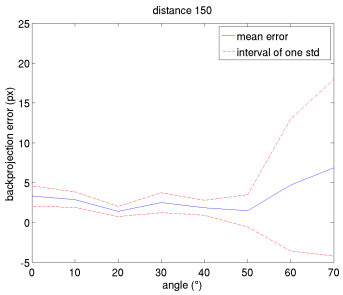
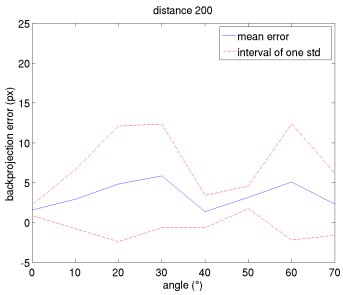


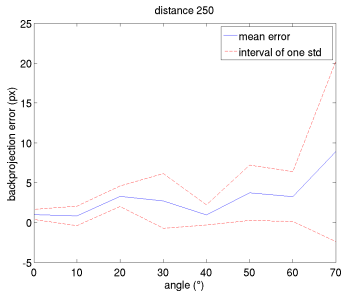
Abbildung 7.9: Alternative Berechnung des Translationsvektors.



(a) Rotationen mit einer Entfernung von 150.



(b) Rotationen mit einer Entfernung von 200.

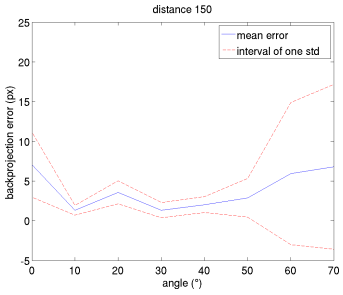


(c) Rotationen mit einer Entfernung von 250.

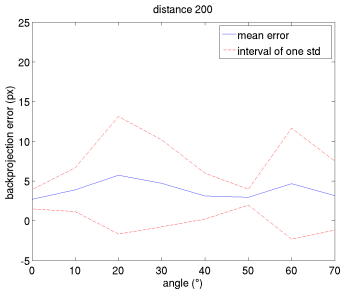
Abbildung 7.10: Rückprojektionsfehler (in Pixeln) in Abhängigkeit zur Rotation (in Grad) um die $\boldsymbol{v}$-Achse. Dabei wurden in dieser Evaluation sowohl Rotation als auch Translation berechnet.

Bild. Dies geschieht mit den 18 in allen Bildern sichtbaren Eckpunkten des Dachs, der Fassade, Fenster und Türen. Für diese Punkte wird der Mittelwert der Fehler und die Standardabweichung bestimmt. Abbildung 7.10 zeigt, dass der mittlere Fehler bei allen Winkeln und Abständen ungefähr 5 Pixel beträgt und die Standardabweichung in den meisten Fällen nicht größer als 10 wird. Diese Fehler sind für eine erste Posehypothese akzeptabel.

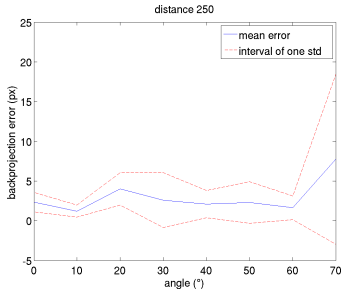
Außerdem ist interessant zu evaluieren, welchen Einfluss die Berechnung der Rotation und Translation unabhängig voneinander besitzen. Zur Evaluation der einzelnen Einflüsse wird nun die jeweilige andere Größe auf den richtigen Wert gesetzt. Der unveränderte mittlere Fehler in Abbildung 7.11 und der etwas niedrigere mittlere Fehler mit einer eindeutig kleineren Streuung in Abbildung 7.12 zeigen,



(a) Rotationen mit einer Entfernung von 150.

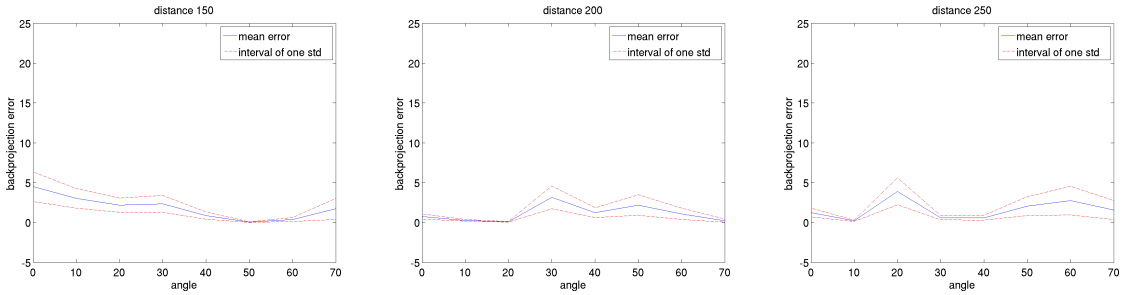


(b) Rotationen mit einer Entfernung von 200.



(c) Rotationen mit einer Entfernung von 250.

Abbildung 7.11: Rückprojektionsfehler (in Pixeln) in Abhängigkeit zur Rotation (in Grad) um die $\boldsymbol{v}$-Achse. Dabei wurde in dieser Evaluation die Rotation berechnet und die Original-Translation verwendet.



(a) Rotationen mit einer Entfernung von 150.

(b) Rotationen mit einer Entfernung von 200.

(c) Rotationen mit einer Entfernung von 250.

Abbildung 7.12: Rückprojektionsfehler (in Pixeln) in Abhängigkeit zur Rotation (in Grad) um die $\boldsymbol{v}$-Achse. Dabei wurde in dieser Evaluation die originale Rotation verwendet und die Translation berechnet.

dass die Berechnung der Translation einen kleineren Einfluss auf den Rückprojektionsfehler besitzt als die Rotation.

7.2.2 Neue Posehypothesen durch Verfeinerung der Pose

Im Bild gefundene Objekte können nicht nur dazu genutzt werden, die Stimmigkeit der vorliegenden Hypothese zu evaluieren, sondern auch zum Generieren neuer Hypothesen dienen.

In dieser Arbeit werden hierzu die erkannten und zugeordneten Fenster bzw. deren Eckpunkte genutzt (siehe Kapitel 5.5). Die im Bild erkannten Boundingboxen mit Fensterkandidaten müssen dazu weiter verarbeitet werden. Die Fensterkandidaten wurden in einer Gaußpyramide in der Regel auf verschiedenen Ebenen (unterschiedlichen Glättungen) detektiert. Die höchste Ebene (stärkste Glättung), in der das entsprechende Fenster detektiert werden konnte, wird genutzt, um eine Vordergrund-Hintergrund-Trennung mittels morphologischer Operationen durchzuführen. Es wird ausgenutzt, dass die gefundenen Fensterflächen in der entsprechenden Weichzeichnungsstufe eine homogene Farbregion bilden. Diese Region wird extrahiert und mittels Regression ein Viereck in diese Region mit kleinsten quadratischem Abstand eingepasst. Auf diese Weise können die Eckpunkte des Fensters im Bild den Eckpunkten im Modell zugeordnet werden.

Mit den 2-D $\leftrightarrow$ 3-D-Korrespondenzen kann mittels *angepasstem RANSAC* [FB81] und dem Verfahren von Fiore [Fio01] (siehe Kapitel 4.4 auf Seite 103) eine neue Pose bestimmt werden. Das Modellwissen ermöglicht es einen *informierten RANSAC* zu verwenden (siehe Pseudocode 7.1, der die RANSAC Poseschätzung in Abbildung 7.13 beschreibt). Es werden im angepassten RANSAC-Verfahren nicht zufällig Merkmale zur Hypothesenerstellung genutzt, sondern stets auf den be-

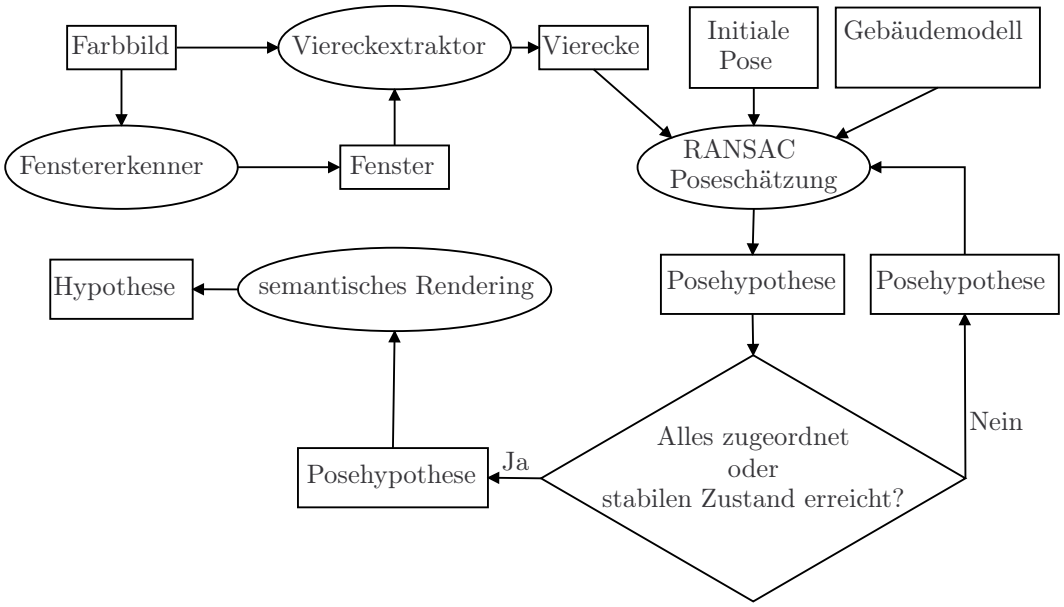


Abbildung 7.13: Datenflussdiagramm der Poseverfeinerung.

stehenden Modell-Bild-Zuordnungen gearbeitet. Die zugeordneten Fenster werden initial zur Posebestimmung verwendet.

(a) Bsp. Poseverfeinerung

(b) Bsp. Poseverfeinerung

Abbildung 7.14: Beispiel für Poseverfeinerung. In schwarz die gefundenen Fenster, in blau die Position der Fenster aus der ursprünglich bestimmten Pose mittels Rückprojektion und in rot die Fenster nach Rückprojektion von der mit dem Verfahren von Fiore neu berechneten Pose.

Unter Verwendung der berechneten Pose werden die Fenster neu zugeordnet und auf diesen Zuordnungen wieder eine neue Pose bestimmt. Die Zuordnung erfolgt mittels Ungarischer Methode. Wenn weniger als drei Fenster zugeordnet wurden, wird im nächsten Schritt Nichtzuordnen stärker bestraft und wurden mehr als die Hälfte zugeordnet, wird im nächsten Schritt Nichtzuordnen weniger stark

Pseudocode 7.1 Beschreibung des Prinzips vom informierten RANSAC

```
1: procedure INFORMIERTERRANSAC(Liste<Modellfenster> OP_M, Liste<Analysemodellfenster> OP_S, Pose υ)
2:     initiale Zuordnung Z_1 ← ZUORDNUNGMITUNGARISCHERMETHODE(OP_M,OP_S,υ)
3:     for op_i ∈ OP_M do
4:         for op_j ∈ OP_S do
5:             Ordne Eckpunkte von op_i und op_j zu
6:             Pose υ_k ← FIORE(mit zugeordneten Eckpunkten von op_i und op_j)
7:             Zuordnung Z_k ← ZUORDNUNGMITUNGARISCHERMETHODE(OP_M,OP_S,υ_k)
8:             if Zuordnung Z_k besser als Zuordnung Z_1 then
9:                 Pose υ_l ← FIORE(Z_k)
10:                Zuordnung Z_l ← ZUORDNUNGMITUNGARISCHERMETHODE(OP_M,OP_S,υ_l)
11:                if Zuordnung Z_k besser als Zuordnung Z_l then
12:                    υ ← υ_k
13:                else
14:                    υ ← υ_l
15:                Zuordnung Z_1 ← ZUORDNUNGMITUNGARISCHERMETHODE(OP_M,OP_S,υ)
16:    return υ
```

bestraft. Dies wird solange durchgeführt bis alle Fenster zugeordnet sind und der Rückprojektionsfehler entsprechend klein ist oder bis ein stabiler Zustand (beispielsweise drei Zustände, die sich jeweils abwechseln) erreicht wird. In dem Fall, dass nicht alle Fenster zugeordnet werden können und der stabile Zustand des Verfahrens mehrere Posehypothesen beinhaltet, wird die Posehypothese mit den meisten Fensterzuordnungen und kleinstem Rückprojektionsfehler gewählt. Die berechnete Pose wird mittels semantischem Rendering gerendert (siehe Kapitel 3.5) und geht als komplett neue unbearbeitete Hypothese in den Hypothesenraum ein. Abbildung 7.13 fasst das Verfahren in einem Datenflussdiagramm zusammen und Abbildung 7.14 zeigt zwei Beispiele, bei denen die Neuberechnung der Pose geholfen hat, die Pose zu verbessern.

7.3 Diskussion

In diesem Kapitel konnte gezeigt werden, dass die Kombination von einfachen Extraktionsverfahren mit explizitem Modellwissen die Generierung von Hypothe-

sen ermöglicht. Eine einfache Extraktion von geometrischen Formen, Farben und Regionen ermöglicht die Erstellung von Hypothesen für 2-D-Erkennungsprobleme. Dazu wurden Erkenner für Vierecke bzw. Rechtecke, Kreise, Kartensymbole und die Buchstaben A, J, K und Q implementiert.

In einer komplexen 3-D-Anwendung muss zunächst das 3-D-Erkennungsproblem in ein 2-D-Erkennungsproblem überführt werden. Dazu sind relativ einfache Extraktionsverfahren nötig, die in Kombination mit dem Modellwissen in der Lage sind, dies zu bewerkstelligen. Die zuvor beschrieben Verfahren zur Extraktion von Dachkanten und Fluchtpunkten ermöglichen die Berechnung einer überschaubaren Zahl an Posehypothesen. Die Zahl an Posehypothesen könnte mit einem genaueren Fluchtpunkt- und Dachkantenerkenner reduziert werden. Darauf lag aber nicht der Schwerpunkt der Arbeit, sondern auf dem Umgang mit unpräzisem Wissen. Ein weiteres Verfahren extrahiert aus vorher gefundenen Fensterregionen Fensterecken und berechnet mit diesen Eckpunkten weitere Posehypothesen. Hier könnte ein besserer Fenstererkenner die Bestimmung der Posehypothesen präzisieren und die Robustheit steigern.

Auch liefern diese Verfahren für sich selbst genommen durchschnittliche Ergebnisse. Hier sei wieder auf die in Kapitel 10 belegte These der Arbeit verwiesen: "Die Verwendung von Wissen kann Probleme bei der Erkennung kompensieren bzw. erst die Erkennung ermöglichen.". Neben der in diesem Kapitel beschriebenen Erzeugung von Hypothesen ist es für die Steuerung des Verfahrens nötig, diese Hypothesen auch zu bewerten. Möglichkeiten Zuordnungen und Hypothesen zu bewerten und die für diese Arbeit gewählte Form der Bewertung werden im nächsten Kapitel beschrieben.

Kapitel 8

Hypothesenbewertung

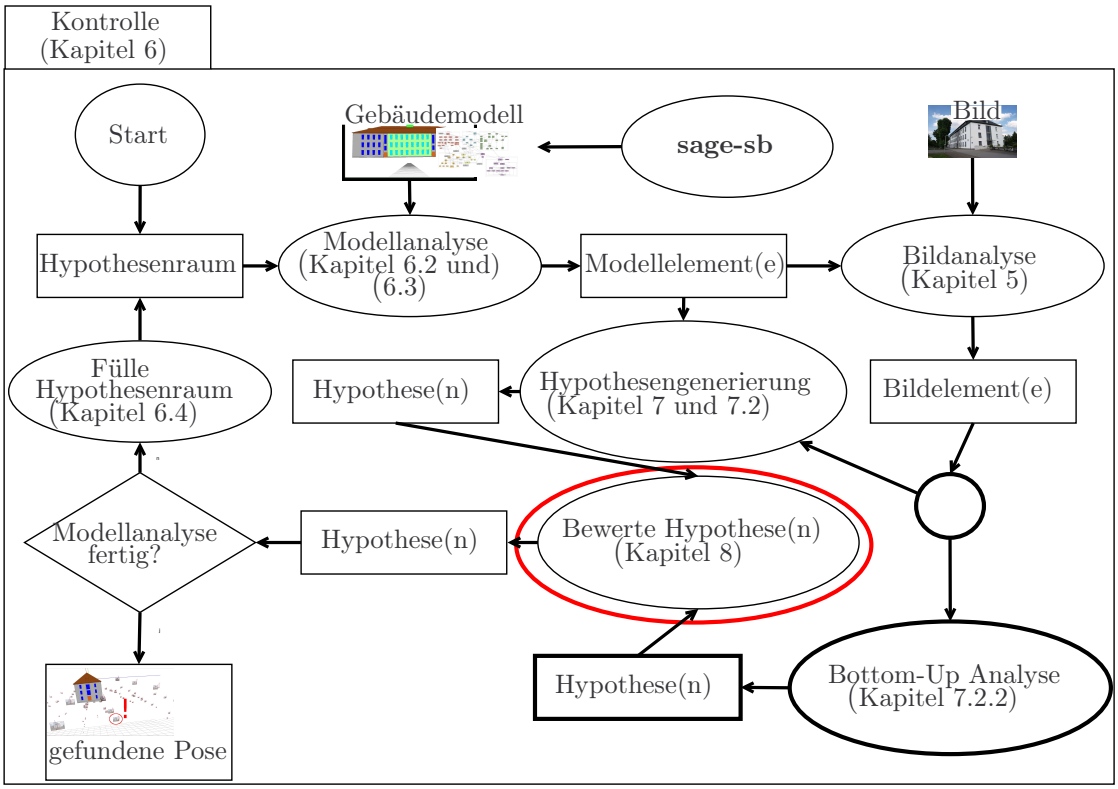


Abbildung 8.1: Jede Zuordnung und jede Hypothese besitzt eine Bewertung in Form eines *Vertrauensmaßes* (siehe rote Ellipse). Das Ergebnis der Analyse ist eine Pose inklusive Bewertung.

Eine anwendungsunabhängige Kontrolle steuert den Ablauf der Modellanalyse (Abbildung 8.1). Im Zuge der Analyse entstehen Hypothesen (Kapitel 7). Konkurrierende *Zuordnungen* werden in unterschiedlichen *Hypothesen* gespeichert, welche

wiederum in einem sogenannten *Hypothesenraum* verwaltet werden. Jede Zuordnung und jede Hypothese besitzt eine Bewertung in Form eines *Vertrauensmaßes*. Das Ergebnis der Analyse ist eine Pose inklusive Bewertung.

Zur Bewertung von Hypothesen, aber auch einzelner Zuordnungen, wird ein Maß aus zweierlei Gründen benötigt. Zum einen benötigt man zur Steuerung der Analyse die Möglichkeit die Hypothesen zu sortieren, um so stets mit der Hypothese fortzufahren, die die höchste Bewertung besitzt. Daher benötigt man möglichst monotone Funktionen zur Berechnung der Bewertung. Diese Funktionen müssen weder nach unten noch nach oben begrenzt sein. Zum anderen ist ein Maß für die Güte der Hypothesen wünschenswert. Dafür eignet sich ein Vertrauensmaß, das das subjektive Vertrauen eines Menschen in das Eintreffen eines Ereignisses beschreibt. Dabei handelt es sich um eine heuristisch bestimmte Zahl aus einem festen Wertebereich. Je größer die Zahl ist, desto höher ist die Güte der Hypothese und umgekehrt. Das Vertrauensmaß ermöglicht es, verschiedene Hypothesen desselben Systems zu vergleichen, nicht aber Hypothesen verschiedener Systeme, da diese verschiedene Berechnungen des Vertrauensmaßes verwenden.

8.1 Vertrauensmaße und Wahrscheinlichkeiten

> [Vertrauensmaß, das] "Metaqualitätselement, das die Richtigkeit einer Qualitätsinformation beschreibt." [1]

Ein Vertrauensmaß ist, wie schon beschrieben, eine subjektive Maßzahl für das Eintreffen eines Ereignisses. Die Bewertung von Zuordnungen wird sich von Mensch zu Mensch unterscheiden. Beispielsweise können bei der Zuordnung von Fenstern unterschiedliche Kriterien angewandt werden. Man könnte beispielsweise den Polygonabstand messen, die Schwerpunkte vergleichen oder den Abstand der Eckpunkte akkumulieren. Eine perfekte Zuordnung wird es in der Realität nie geben, sodass es dem Entwickler überlassen bleibt, wie er die Funktionen skaliert und gewichtet.

Definition 8.1 (Ereignisraum)**.** *Die Menge $\Omega\{A_k\}$ heißt Ergebnisraum Ω mit $k = 1, \ldots, K$ und $A \subseteq \Omega$, wobei A_k beliebige Ereignisse aus diesem Ereignisraum sind. Jedes mögliche Ergebnis kommt genau einmal in Ω vor.*

Ein elementares Ereignis im Sinne dieser Arbeit ist entweder eine Zuordnung (eines Elements des Analysemodells zu einem Element eines Objektmodells) oder das Auffinden eines Objektes im Bild.

Für ein Vertrauensmaß wird in der Regel der Wertebereich von 0 bis 1 oder von 0 bis 100 gewählt. Dies entspricht dem Wertebereich von Wahrscheinlichkeiten,

[1] `http://www.geoinformatik.uni-rostock.de/suche.asp`, Stichwort: Vertrauensmaß, Zugriff: 05.03.2015

sodass ein Vertrauensmaß leicht mit einer Wahrscheinlichkeit verwechselt werden kann. Nach Laplace ist die Wahrscheinlichkeit eines Ereignisses das Verhältnis der günstigen Ergebnisse zu allen möglichen Ergebnissen. Die Wahrscheinlichkeit ist also ein objektives Maß dafür, wie wahrscheinlich es ist, dass ein Ereignis eintritt; bzw. beschreibt die Chance, dass ein Ereignis eintritt. Um Wahrscheinlichkeiten im klassischen Verständnis einsetzen zu können, müssen sowohl der Ereignisraum als auch das Experiment klar spezifiziert und alle Randbedingungen benannt werden. Dies ist aber gerade im Kontext der Poseschätzung von Gebäuden nahezu unmöglich, da alleine die Modellierung unterschiedlicher Wetterbedingungen und Jahreszeiten und die damit einhergehenden Beleuchtungsänderungen eine höchst komplexes Modell erfordern. Daher scheint es nicht möglich, objektive Maßzahlen für Hypothesen, die durch Zuordnungen von Bild- zu Modellelementen entstanden sind, zu ermitteln. Dies hat zur Folge, dass in dieser Arbeit Vertrauensmaße verwendet werden, deren Formalismus im Folgenden beschrieben wird.

8.2 Bayessche und Dempster-Shafer-Theorie

Bei der Analyse muss mit unsicherem Wissen umgegangen werden, da Segmentierungsergebnisse nicht eindeutig sind. *Possibilistische Theorien*, wie *Bayes* [TK09], *Dempster-Shafer* [BKI06] oder *Fuzzy Mengen* [BKI06], definieren, wie man mit Ungewissheit, Ungenauigkeiten und vagem Wissen umgehen kann. Sie bieten eine formale Repräsentation für unsichere Informationen und Schlussfolgerungsstrategien. Dempster-Shafer wird als *Vertrauensmaßfortpflanzung* in Hois [Hoi07] benutzt, wohingegen Neumann *bayesische Kompositionshierachien* verwendet [Neu08].

8.2.1 Bayessche Theorie

Gegeben sei der *Ereignisraum* $\Omega = \{A_k\}$ mit $k = 1, \ldots, K$ und $A \subseteq \Omega$, wobei A_k beliebige Ereignisse aus diesem Ereignisraum sind. Das *Vertrauensmaß* $P(A_k)$ für ein Ereignis A_k folgt den Axiomen der Wahrscheinlichkeitslehre mit der Voraussetzung, dass $\Omega \neq 0$ für die Ergebnismenge eines Zufallsexperimentes gilt. Eine auf der Ereignismenge $P(\Omega)$ definierte Funktion $P : P(\Omega) \rightarrow [0; 1]$ heißt Wahrscheinlichkeitsmaß nach Kolmogorow [FPT03], falls gilt:

$$P(\emptyset) = 0, \qquad P(\Omega) = 1 \tag{8.1}$$

$$P(A_k) \geq 0 \quad \text{für alle } A_k \subseteq \Omega \tag{8.2}$$

$$P(A_k \cup A_j) \quad = P(A_k) + P(A_j) - P(A_k \cap A_j). \tag{8.3}$$

Daraus folgt, dass jede Funktion P, die den oben beschriebenen Axiomen folgt, ein Wahrscheinlichkeitsmaß darstellt. Dies hat zur Folge, dass bei der Erstellung

von P auf keinerlei statistische Befunde (wie beispielsweise Häufigkeiten) Bezug genommen werden muss. Gleichung 8.1 beschreibt in Kombination mit Gleichung 8.2 die *Nichtnegativität* und *Normiertheit* auf dem Intervall $[0, 1]$. Das entstehende Vertrauen bei der Vereinigung zweier Ereignisse (*Additivität*) wird in Gleichung 8.3 beschrieben. Dabei ist zu betonen, dass es sich bei P um ein Vertrauensmaß handelt, das zwar den Axiomen der Wahrscheinlichkeitslehre folgt, aber bei dem es sich nicht um eine objektive Wahrscheinlichkeit, wie der frequentistischen Wahrscheinlichkeit, handelt.

8.2.2 Dempster-Shafer-Theorie

Ein Basismaß, das das Vertrauen in exakt ein Ereignis bzw. eine Aussage beschreibt ist nach Dempster-Shafer [BKI06] definiert als Vertrauensdichte τ. Die *Vertrauensdichte* τ ist eine Funktion $\tau : 2^\Omega \to [0, 1]$ und erfüllt die folgenden Bedingungen:

$$\tau(\emptyset) = 0 \tag{8.4}$$

$$\sum_{A_k \subseteq \Omega} \tau(A_k) = 1. \tag{8.5}$$

Da es nur durch diese beiden Bedingen definiert wird, muss es weder monoton noch additiv sein und ist daher geeignet, um Funktionen heuristisch zu modellieren. Sind nun k Elementarereignisse im Ereignisraum definiert, müssen 2^k Werte der Vertrauensdichte spezifiziert werden. Wohingegen bei der Bayesschen Theorie nur k Werte benötigt werden.

Das zur Bayesschen Theorie vergleichbare Vertrauensmaß wird in der Dempster-Shafer-Theorie mit *Belief* bel bezeichnet. Es folgt, ähnlich zu Bayes, den in den Gleichungen 8.1 - 8.3 beschriebenen Axiomen der Wahrscheinlichkeitstheorie:

$$\text{bel}(\emptyset) = 0, \qquad \text{bel}(\Omega) = 1 \tag{8.6}$$

$$\text{bel}(A_k) \geq 0 \quad \text{für alle } A_k \subseteq \Omega \tag{8.7}$$

$$\text{bel}(A_k \cup A_j) \geq \text{bel}(A_k) + \text{bel}(A_j) - \text{bel}(A_k \cap A_j). \tag{8.8}$$

Dabei ist der einzige Unterschied das "Größer-gleich-Zeichen" in Ungleichung 8.8. Das *Vertrauensmaß* errechnet sich aus der Summe der Vertrauensdichten aller Teilmengen des Ereignisses:

$$\text{bel}(A_k) = \sum_{A_i \subseteq A_k} \tau(A_i). \tag{8.9}$$

Neben dem Vertrauensmaß definiert Dempster-Shafer auch eine *Plausibilitätsfunktion*. Diese beschreibt die optimistische Schätzung des Beliefs in ein Ereignis.

Dazu wird jeglicher Belief in ein Ereignis mit einbezogen oder anders formuliert: Alle Ereignisse, die nicht gegen das Ereignis sprechen, sprechen dafür. Die Plausibilität ist definiert als Funktion $\mathrm{pl} : 2^{\Omega} \to [0,1]$, für die gilt:

$$\mathrm{pl}(A_k) = \sum_{A_i \cap A_k \neq \emptyset} \tau(A_i). \tag{8.10}$$

Beierle und Kern-Isberner haben zum besseren Verständnis des Dempster-Shafer-Vertrauensmaßes einige grundlegende Beziehungen zusammengefasst [BKI06]:

$$\begin{aligned} \mathrm{bel}(\emptyset) &= 0 &&= \mathrm{pl}(\emptyset) && (8.11)\\ \mathrm{bel}(\Omega) &= 1 &&= \mathrm{pl}(\Omega) && (8.12)\\ \mathrm{bel}(A_k) &\leq \mathrm{pl}(A_k) && && (8.13)\\ \mathrm{bel}(A_k) + \mathrm{bel}(\neg A_k) &\leq 1 && && (8.14)\\ \mathrm{pl}(A_k) + \mathrm{pl}(\neg A_k) &\geq 1. && && (8.15) \end{aligned}$$

Es ist möglich, sowohl dem Ereignis A_k als auch $\neg A_k$ das Vertrauensmaß 0 zuzuweisen:

$$\mathrm{bel}(A_k) = 0 \Leftrightarrow \mathrm{bel}(\neg A_k) = 0. \tag{8.16}$$

Gleichung 8.16 drückt aus, dass über das Ereignis A_k nichts bekannt ist. Es herrscht also komplette Unwissenheit. Dies wäre in der Wahrscheinlichkeitstheorie nicht möglich, da $P(A_k) = 0$ $P(\neg A_k) = 1$ erzwingen würde. Der "wahre" Belief in das Ereignis A_k liegt im Intervall $[\mathrm{bel}(A_k), \mathrm{pl}(A_k)]$ und die Differenz zwischen $\mathrm{pl}(A_k)$ und $\mathrm{bel}(A_k)$ beschreibt die Unwissenheit bzgl. des Ereignisses A_k. Stimmen Plausibilitätsfunktion und Vertrauensmaß überein, entsprechen sie dem Vertrauensmaß der Bayesschen Theorie.

Dempster-Shafer stellt zudem auch eine geeignete Kombinationsregel für unabhängige Vertrauensmaße bereit. Die *Kombinationsregel* der Dempster-Shafer-Theorie lautet:

$$\tau_1 \oplus \tau_2 = \tau(A_k) = \begin{cases} 0 & : \quad A_k = \emptyset \\ \frac{\sum_{A_i \cap A_j = A_k} \tau_1(A_i)\tau_2(A_j)}{\sum_{A_i \cap A_j \neq \emptyset} \tau_1(A_i)\tau_2(A_j)} & : \quad \emptyset \neq A_k \subseteq \Omega \end{cases} . \tag{8.17}$$

Die Dempster-Shafer-Regel ist kommutativ und assoziativ. Demnach ist es möglich, verschiedene Quellen zu kombinieren, wobei die Quellen unabhängig sein müssen, denn im Allgemeinen gilt $\tau \oplus \tau \neq \tau$.

In der Analyse sollen auch Vertrauensmaße verschiedener Ereignisräume, die einen Beitrag zum Erkennen eines Objektes in einem Bild liefern, miteinander kombiniert werden. Dafür wird die Kombinationsregel nach Dempster-Shafer, in der gleichen Weise wie Quint sie in [Qui95] verwendet, genutzt:

"Zur Fortpflanzung der Vertrauensmaße, d. h. für die Berechnung der Vertrauensdichte auf dem Ereignisraum Ω_H bezüglich der Informationsquelle E_x ", schlägt Quint "folgende Gleichungen vor:" (nach [Qui95, Seite 72])

$$\tau_{\mathrm{Hyp}_A}(\emptyset|\mathrm{E}_A) = 0 \tag{8.18}$$

$$\tau_{\mathrm{Hyp}_A}(\mathrm{Hyp}_i|\mathrm{E}_A) = \sum_{A_k \subseteq \Omega_A} \tau_{\mathrm{Hyp}|A}(\mathrm{Hyp}_i|A_k)\tau_A(A_k|\mathrm{E}_A). \tag{8.19}$$

Da die in Gleichung 8.17 beschriebene Regel kommutativ und assoziativ ist, "kann die Kombination auch für mehrere Informationsquellen vorgenommen werden, indem sie schrittweise auf Paare angewendet wird:" (nach [Qui95, Seite 73])

$$\tau_{\mathrm{Hyp}}(\mathrm{Hyp}_k|\mathrm{E}_A,\mathrm{E}_B) = \frac{\sum_{\mathrm{Hyp}_i \cap \mathrm{Hyp}_j = \mathrm{Hyp}_k} \tau_{\mathrm{Hyp}_A}(\mathrm{Hyp}_i|\mathrm{E}_A)\tau_{\mathrm{Hyp}_B}(\mathrm{Hyp}_j|\mathrm{E}_B)}{\sum_{\mathrm{Hyp}_i \cap \mathrm{Hyp}_j \neq \emptyset} \tau_{\mathrm{Hyp}_A}(\mathrm{Hyp}_i|\mathrm{E}_A)\tau_{\mathrm{Hyp}_B}(\mathrm{Hyp}_j|\mathrm{E}_B)}. \tag{8.20}$$

"Mit der bekannten resultierenden Vertrauensdichte auf dem Ereignisraum Ω_H bezüglich aller Informationsquellen E kann das Vertrauensmaß für eine Hypothese Hyp_k gemäß" Gleichung 8.9 "berechnet werden:"(nach [Qui95, Seite 73])

$$\mathrm{bel}(\mathrm{Hyp}_k|\{\mathrm{E}\}) = \sum_{\mathrm{Hyp}_i \subseteq \mathrm{Hyp}_k} \tau_{\mathrm{Hyp}}(\mathrm{Hyp}_i|\{\mathrm{E}\}). \tag{8.21}$$

Ein detaillierter Vergleich von Verfahren zur Fortpflanzung von Vertrauensmaßen wird in [Qui95] präsentiert.

8.2.3 Vergleich

Bei der Bayesschen Theorie kommen ebenso wie bei der Dempster-Shafer-Theorie Vertrauensmaße zum Einsatz, die einem definierten Regelwerk folgen. Dabei folgt Bayes den Axiomen der Wahrscheinlichkeitslehre und die Vertrauensmaße weisen daher starke Ähnlichkeit zu Wahrscheinlichkeiten auf. Allerdings gibt es bei Bayes für die Kombination von Vertrauensmaßen keine Entsprechung zu Gleichung 8.17. Dempster-Shafer folgt auch den Axiomen der Wahrscheinlichkeitslehre, aber erweitert das dritte Axiom (Ungleichung 8.8). Daher (aber auch aus den anderen beschriebenen Punkten) kann man bei Dempster-Shafer von einer Erweiterung der Bayesschen Theorie sprechen. Bei Dempster-Shafer ist es möglich, einem Ereignis mehrere (unterschiedliche) Vertrauensmaße zuzuordnen, was bei Bayes so nicht möglich ist, da nach Bayes einem Ereignis stets genau ein subjektives Vertrauen anhaftet. Allerdings ist es bei Bayes möglich, mit bedingten Wahrscheinlichkeiten ähnlich Dinge zu modellieren. Dempster-Shafer ermöglicht die explizite

Modellierung von Unwissenheit und zwingt nicht zur Annahme "zweiseitiger Wetten", sodass aus der Modellierung des Vertrauens in ein Ereignis nicht sofort das Vertrauen in das Gegenereignis folgt.

Diese beschriebenen Dempster-Shafer-Eigenschaften sowie die Möglichkeit zur Kombination von Vertrauensmaßen werden in der Arbeit genutzt, um damit die Bewertung der Hypothesen zu ermöglichen und den A^*-Algorithmus zu steuern.

8.3 Bewertung der Modelle

Dieser Ansatz basiert auf drei generellen Bewertungsfunktionen, die für jedes semantische Objekt (*SemanticObject*) SOb (siehe Kapitel 3.4) der vorgestellten Objektmodelle gelten. Die folgenden Bezeichner entsprechen den in Kapitel 3.4 beschriebenen Graphelementen des STOR-Referenzschemas sowie der speziellen Modelle. Als erstes wird eine Funktion benötigt, die Dempster-Shafer-Vertrauensmaße τ aus unterschiedlichen Ereignisräumen kombiniert (siehe Gleichung 8.20). Die Qualität des Vertrauens, dass ein semantisches Objekt SOb im Bild gefunden wurde, wird modelliert mit Gleichung 8.22:

$$\begin{aligned} &\tau(\mathrm{SOb}|\mathrm{E}_{\mathrm{parts}}, \mathrm{E}_{\mathrm{rep}}) = \\ &\quad \kappa_1\ \tau(\mathrm{SOb}|\mathrm{E}_{\mathrm{parts}}) \oplus \kappa_2\ \tau(\mathrm{SOb}|\mathrm{E}_{\mathrm{rep}})\ . \end{aligned} \tag{8.22}$$

Dabei können mit κ_1 und κ_2 die einzelnen Vertrauensmaße gewichtet werden. Hierbei gilt $\kappa_1, \kappa_2 \in [0, 1]$. Der Ereignisraum E_{parts} beschreibt die Beobachtungen der Bestandteile eines semantischen Objekts und der Ereignisraum $\mathrm{E}_{\mathrm{rep}}$ spezifiziert die Beobachtung eines geometrischen Objektes (*GeometricObject*) oder einer Region (*Region*), die ein semantische Objekt repräsentiert.

Außerdem wird ein Vertrauensmaß benötigt, das das Vertrauen in ein semantisches Objekt beschreibt, gegeben alle beobachteten Bestandteilsarten (Ereignisraum E_{parts}). Dafür werden alle Bestandteilsarten des semantischen Objektes SOb kombiniert mit Gleichung 8.23:

$$\begin{aligned} &\tau(\mathrm{SOb}|\mathrm{E}_{\mathrm{parts}}) = \\ &\omega_1\ \tau(\mathrm{SOb}|\{\mathrm{parts}\}_1) \oplus \omega_2\ \tau(\mathrm{SOb}|\{\mathrm{parts}\}_2) \oplus \\ &\qquad \ldots \oplus \omega_N\ \tau(\mathrm{SOb}|\{\mathrm{parts}\}_N)\ . \end{aligned} \tag{8.23}$$

Dabei können mit $\omega_1 \ldots \omega_N \in [0, 1]$ die einzelnen Vertrauensmaße gewichtet werden. Des Weiteren wird das Vertrauensmaß für alle Bestandteile einer Bestandteilsart benötigt:

$$\tau(\mathrm{SOb}|\{\mathrm{parts}\}) = \frac{1}{N}\sum_{i}^{N}\tau(\mathrm{SOb}|\mathrm{parts}_i)\tau(\mathrm{parts}_i|\mathrm{E}_{\mathrm{parts}},\mathrm{E}_{\mathrm{rep}}) \ . \tag{8.24}$$

Die drei Gleichungen 8.22 - 8.24 gelten für jedes semantische Objekt und werden ergänzt durch Vertrauensmaße für den konkreten Anwendungsfall, beispielsweise wie hoch das Vertrauen ist, dass ein gefundenes Objekt ein Rechteck ist.

Jedes extrahierte Bildelement und jede extrahierte Bildregion bewertet sich selbst und gibt damit das Vertrauen in sich selbst an. Also beispielsweise wie wahrscheinlich handelt es sich wirklich um eine Fensterregion oder einen Kreis. Dieser Belief kann in Gleichung 8.22 integriert und gewichtet werden. In der Regel wird dabei nur eine schwache Gewichtung gewählt, da die Relation der Elemente einen stärkeren Einfluss auf die Wiedererkennung des Modells hat als der Belief in das Element selbst.

8.3.1 Dominostein- und Pokerkartenerkennung

Im Folgenden werden Beispiele für die Standardfunktionen (Gleichung 8.22 - 8.24) gegeben und Funktionen für die Initialisierung gezeigt.
Ein Beispiel für Gleichung 8.22 ist:

$$\tau(\mathrm{DT}|\mathrm{E}_{\mathrm{DTrect}_i},\mathrm{E}_{\mathrm{R}}) = 0.7\,\tau(\mathrm{DT}|\mathrm{E}_{\mathrm{R}}) \oplus \tau(\mathrm{DT}|\mathrm{E}_{\mathrm{DTrect}_i}), i \in \{2,3\}. \tag{8.25}$$

Diese Gleichung beschreibt das Vertrauen in einen Dominostein DT gegeben die Informationsquellen $\mathrm{E}_{\mathrm{DTrect}_i}$: beobachtete Dominosteinhälften, und E_{R}: beobachtete Rechtecke. Das Rechteck selbst liefert einen kleineren Beitrag in das Vertrauen, dass wirklich ein Dominostein im Bild gefunden wurde, als mögliche gefundene Dominostein-Hälften daher ist $\kappa_1 = 0.7 \wedge \kappa_2 = 1$.

Das Vertrauensmaß für ein semantisches Objekt gegeben die Informationsquelle E_{parts} (Gleichung 8.23) wird beispielsweise für einen Dominostein DT (Gleichung 8.26) umgesetzt:

$$\tau(\mathrm{DT}|\mathrm{E}_{\mathrm{DTrect}}) = \tau(\mathrm{DT}|\{\mathrm{DTrect}_1,\mathrm{DTrect}_2\}) \ . \tag{8.26}$$

Dabei besitzt ein Dominostein zwei Hälften DTrect_1 und DTrect_2, sodass sich für Gleichung 8.24 die Umsetzung

$$\tau(\mathrm{DT}|\{\mathrm{DTrect}\}) = \frac{1}{2}\sum_{i=1}^{2}\tau(\mathrm{DT}|\mathrm{DTrect}_i) \quad \text{ergibt.} \tag{8.27}$$

Im Folgenden werden nun mit den Gleichungen 8.28 - 8.31 einige wichtige Vertrauensmaße für die Dominostein- und Pokerkarten-Fallstudie vorgestellt.
Das Vertrauensmaß einer Zuordnung von einem segmentierten Rechteck zu einem Modellrechteck ist definiert als

$$\tau(\mathrm{DTrect}_1|\mathrm{E_R}) = \begin{cases} \frac{\mathrm{ratio}_r}{\mathrm{ratio}_{\mathrm{DTrect}_1}} & : \quad \mathrm{ratio}_{\mathrm{DTrect}_1} > \mathrm{ratio}_r, \quad r \in \mathrm{R} \\ \frac{\mathrm{ratio}_{\mathrm{DTrect}_1}}{\mathrm{ratio}_r} & : \quad \mathrm{ratio}_{\mathrm{DTrect}_1} < \mathrm{ratio}_r, \quad r \in \mathrm{R} \end{cases} . \tag{8.28}$$

Für die Berechnung von $\tau(\mathrm{DTrect}_i|\mathrm{E_R})$ wird das bereits gefundene Rechteck DTrect_1 eines Dominosteins in seine zwei Hälften geteilt und mit dem Kandidaten für die jeweilige Hälfte DTrect_i verglichen. Hierfür wird die Schnittfläche einer Hälfte mit DTrect_i berechnet und anschließend durch die gesamte Fläche der beiden Flächen geteilt:

$$\tau(\mathrm{DTrect}_i|\mathrm{E_R}) = \frac{\text{intersection area of } \mathrm{DTrect}_i \text{ and R}}{\text{entire area of } \mathrm{DTrect}_i \text{ and R}} \text{ with } i \in \{2,3\} . \tag{8.29}$$

Die Vertrauensmaße einer Rechteckzuordnung in (8.28) und (8.29) sind nötig, um Dominosteine von anderen Objekten unterscheiden zu können. Außerdem wird eine Funktion benötigt, die fehlende Zuordnungen von Modellelementen und/oder Segmentierungsobjekten bestraft:

$$\begin{aligned} &\tau(\mathrm{DTrect}_i|\mathrm{PIP}) = e^{-\frac{1}{2}x^2} \\ &x = ||\mathrm{C}| + |\{\text{gegeben Punkte}\} \in \mathrm{PIP}| - |\{\text{zugeordnet Punkte}\} \in \mathrm{PIP}|| . \end{aligned} \tag{8.30}$$

Dabei beschreibt |C| die Anzahl segmentierter Kreise und PIP definiert die Menge der im Modell vorhandenen Punkte.

Das Vertrauensmaß für die Zuordnung eines Punktes Pip zu einem Kreis c, wobei (c_x, c_y) und $(\mathrm{Pip}_x, \mathrm{Pip}_y)$ die Mittelpunkte der Kreise und c_r und Pip_r die Radien sind, ist sehr wichtig für die Analyse. Diese Gleichung verhindert, dass falsche und weit entfernte Punkte zugeordnet werden:

$$\begin{aligned} \tau(\mathrm{Pip} \in \mathrm{PIP}|c \in \mathrm{C}) &= 101 - 200^q - 50|c_r - \mathrm{Pip}_r|, \\ q &= \sqrt{(\mathrm{Pip}_x - c_x)^2 + (\mathrm{Pip}_y - c_y)^2} . \end{aligned} \tag{8.31}$$

Die Funktion, die einer Kartenfarbe CO (Pik, Kreuz, Herz oder Karo) eine Region Reg zuordnet, benötigt nur die Zentren (x_M, y_M) und (x_S, y_S) der Modell- und der segmentierten Bildregion zum Vergleich. Da bei Bildregionen kein Radius verfügbar ist, kommt hier eine vereinfachte Version von Gleichung 8.31 zum Einsatz:

$$\tau(\text{CO}|\text{Reg}) = e^{-a\sqrt{(x_M - x_S)^2 + (y_M - y_S)^2}}. \tag{8.32}$$

Der Faktor a hängt von der Größe des extrahierten Teilbildes ab. Im Fall der Pokerkartenerkennung beträgt diese 300x450 und das Ausgangsbild hat eine Auflösung von 3872x2592. Der Faktor a berechnet sich dann aus: $a = 300/3872 \approx 0,08$.

Die Wahl der Funktionen (8.30) und (8.32) reguliert, ob es besser ist eine Zuordnung wegzulassen oder nicht. Umso mehr Unterschiede zwischen den Punkten und Kreisen von der Funktion $\tau(\text{Pip} \in \text{PIP}|c \in \text{C})$ verziehen werden, umso mehr Segmentierungsobjekte werden zugeordnet und umgekehrt. Für die Berechnung der Vertrauensmaße (8.29), (8.30) und (8.31) werden die Rechtecke und dazu im Verhältnis auch die Kreise in den Ursprung des Koordinatensystems verschoben, rotiert und skaliert.

Mit diesen Funktionen hat man nun die Möglichkeit, die Zuordnungen von Modellelementen und Segmentierungsobjekten zu bewerten. Weiterhin liefert die Kombination der Funktionen den Gesamtbelief, dass ein spezifisches Modell in einem Bild erkannt wurde.

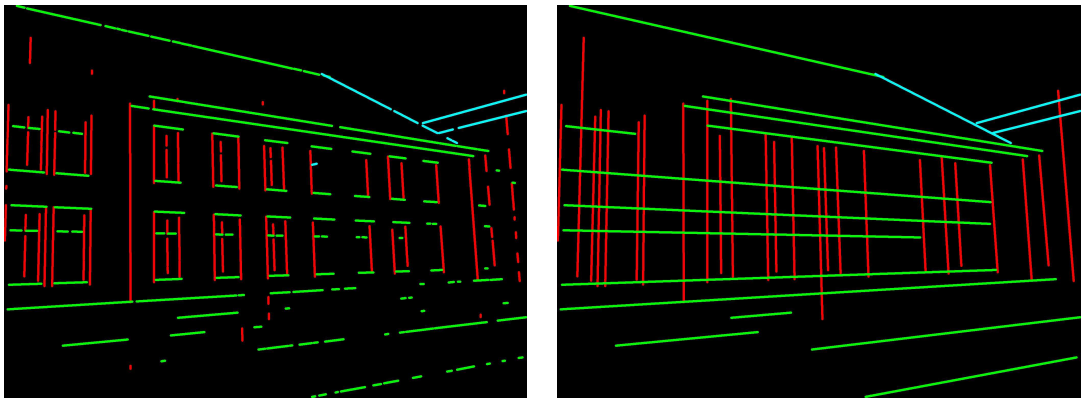
8.3.2 Poseschätzung von Gebäuden

Durch die in Kapitel 7.2 beschriebene Hypothesengenerierung besitzt das Modell eine 2-D-Geometriedarstellung, die das Modell aus Sicht der Kamera beschreibt. Aus den Geometriedarstellungen werden alle Strecken extrahiert, die ROI berechnet und die kürzeste Strecke bestimmt. Es werden dabei nicht alle Strecken auf einmal berücksichtigt, sondern konform zum Modell (Kapitel 3.3) separat jeweils die Strecken des Daches, der Fassade und der Grundfläche.

Aus dem Bild werden die interessanten Liniensegmente (Kapitel 5.3) extrahiert und alle Strecken verworfen, die nicht in der Modell-ROI liegen, die sich im vorher bestimmten Himmel befinden oder weniger als ein Drittel der Länge der vorher bestimmten kürzesten Modellstrecke besitzen. Diese werden dann zu *erweiterten Strecken* zusammengefasst. Eine *erweiterte Strecke* beschreibt eine Strecke, die dicht beieinanderliegende und annähernd kollineare Strecken in einer Strecke zusammenfasst (beispielsweise Strecken einer Fensterreihe oder durch Segmentierungsfehler entstandene Lücken) (Abbildung 8.2).

Diese Strecken werden den erweiterten Strecken der 2-D-Projektion des 3-D-Modells zugeordnet, über die Distanzfunktion mit den Modellstrecken verglichen und mittels Ungarischer Methode zugeordnet (siehe Anhang D).

Als Distanzfunktionen wurden die (i) Hausdorff-Distanz, (ii) Trucco-Distanz, (iii) Modified line segment Hausdorff-Distanz, (iv) Perpendicular-Distanz, (v) Mid-



(a) interessante Liniensegmente

(b) Erweiterte-Strecken der interessanten Liniensegmente

Abbildung 8.2: Interessante Liniensegmente und deren erweiterte Strecken.

point-Distanz, (vi) Closest point-Distanz und (vii) unsere eigene Strecken-Geraden-Distanz evaluiert.

Wir verwenden unsere eigene Distanz in dieser Fallstudie, da sie sich als sehr gut geeignet für diesen Anwendungsfall erwiesen hat. Diese Evaluation wurde auf der OGRW 2014 [WP15, WP16] veröffentlicht und die detaillierten Ergebnisse der Studie befinden sich im Anhang E.

Die Strecken-Geraden-Distanz nutzt $\mathrm{d}_1 = \|p_{1,l_1} - p_{1,l_2}\|$, $\mathrm{d}_2 = \|p_{1,l_1} - p_{2,l_2}\|$, $\mathrm{d}_3 = \|p_{2,l_1} - p_{1,l_2}\|$ und $\mathrm{d}_4 = \|p_{2,l_1} - p_{2,l_2}\|$ und verbindet diese in:

$$\mathrm{d}_{\mathrm{translation}}(l_1, l_2) = \frac{\mathrm{d}_1 + \mathrm{d}_2 + \mathrm{d}_3 + \mathrm{d}_4}{4} - \frac{\|l_1\| + \|l_2\|}{4} \tag{8.33}$$

$$\mathrm{d}_{\mathrm{ST}}(l_1, l_2) = \mathrm{d}_{\mathrm{closestpoint}} + 0.25\ \mathrm{d}_{\Theta}(l_1, l_2) + \mathrm{d}_{\mathrm{translation}} \tag{8.34}$$

$$\mathrm{d}_{\mathrm{straightLine}}(l_1, l_2) = \min(\mathrm{d}_{\mathrm{ST}}(l_1, l_2), \mathrm{d}_{\mathrm{ST}}(l_2, l_1)). \tag{8.35}$$

Die Standardfunktionen wurden in der Gebäudefallstudie genauso umgesetzt wie in der Fallstudie für Dominosteine und Pokerkarten (siehe Gleichung 8.22 bis 8.24 und Abschnitt 8.3.1). Die Tabellen 8.1 - 8.4 listen die verwendeten Vertrauensmaße auf.

Tabelle 8.1 listet die Vertrauensmaße passend zu Gleichung 8.22 auf. In Tabelle 8.2 werden die Vertrauensmaße passend zu Gleichung 8.23 aufgelistet und Tabelle 8.3 listet die Vertrauensmaße passend zu Gleichung 8.24 auf.

Die Gleichungen aus den Tabellen 8.1 bis 8.3 bedürfen keiner individuellen Anpassung, sondern ergeben sich aus den beschriebenen Standardfunktionen. Dahingegen listet Tabelle 8.4 anwendungsabhängige Funktionen auf, die eigens für

$\tau(\mathrm{BU}\|\mathrm{E}_{\mathrm{parts}}, \mathrm{E}_{\mathrm{rep}})$	Berechnet den Belief in ein gefundenes Gebäude, gegeben seine Bestandteile *Walling*, *Roof* und *Ground*, sowie seine Repräsentation.
$\tau(\mathrm{WS}\|\mathrm{E}_{\mathrm{parts}}, \mathrm{E}_{\mathrm{rep}})$	Berechnet den Belief in eine gefundene Wandfläche, gegeben seine Bestandteile *Opening* und seine Repräsentation.
$\tau(\mathrm{RS}\|\mathrm{E}_{\mathrm{parts}}, \mathrm{E}_{\mathrm{rep}})$	Berechnet den Belief in eine gefundene Dachfläche, gegeben seine Bestandteile *Opening* und seine Repräsentation.
$\tau(\mathrm{GS}\|\mathrm{E}_{\mathrm{parts}}, \mathrm{E}_{\mathrm{rep}})$	Berechnet den Belief in eine gefundene Kellerfläche, gegeben seine Bestandteile *Opening* und seine Repräsentation.

Tabelle 8.1: Die verwendeten anwendungsunabhängigen Vertrauensmaße passend zu Gleichung 8.22.

die Poseschätzung von Gebäuden modelliert wurden. Man sieht, dass eine überschaubare Anzahl Funktionen benötigt werden.

Bei der in diesem Kapitel beschriebenen Zuordnung von Strecken (siehe Seite 175) wird für jede Zuordnung ein Kostenwert mittels Gleichung 8.35 ermittelt. Dies ermöglicht es, ein Vertrauensmaß für die gesamte Zuordnung zu berechnen. Dafür werden zunächst die durchschnittlichen Kosten mittels Gleichung 8.36 bestimmt:

$$\mathrm{avgcost} = \frac{1}{N} \sum_{i}^{N} \mathrm{d}_{\mathrm{straightLine\_i}}. \tag{8.36}$$

Aus diesen Durchschnittskosten lässt sich nun ein Vertrauensmaß für ein Objekt, gegeben seine Repräsentation (in diesem Fall als Menge LS der gefundenen Strecken) bestimmen:

$$\tau(\mathrm{SOb}|\mathrm{E}_{\mathrm{LS}}) = e^{-\frac{\mathrm{avgcost}}{100}} \ . \tag{8.37}$$

Das Vertrauensmaß in die Zuordnung eines Fensters/Tür (*Opening*) OP zu einer Bildregion Reg eines Fenster-/Türkandidaten (Gleichung 8.38), wobei (c_x, c_y) und (o_x, o_y) die Mittelpunkte der Regionen sind, orientiert sich an Gleichung 8.32, normiert gleichzeitig Größenunterschiede und verhindert, dass falsche und weit entfernte Punkte zugeordnet werden:

$$\begin{aligned} \tau(\mathrm{OP}|\mathrm{E}_{\mathrm{Reg}}) &= 1 - \frac{0,95 * q}{50 + q}, \\ q &= \sqrt{(o_x - c_x)^2 + (o_y - c_y)^2} \ . \end{aligned} \tag{8.38}$$

$\tau(\mathrm{BU}\|\mathrm{E}_{\mathrm{parts}})$	Berechnet den Belief in ein gefundenes Gebäude, gegeben seine Bestandteile.
$\tau(\mathrm{WA}\|\mathrm{E}_{\mathrm{parts}})$	Berechnet den Belief in die gefundene Gebäudefassade, gegeben seine Bestandteile.
$\tau(\mathrm{RO}\|\mathrm{E}_{\mathrm{parts}})$	Berechnet den Belief in das gefundene Dach, gegeben seine Bestandteile.
$\tau(\mathrm{GR}\|\mathrm{E}_{\mathrm{parts}})$	Berechnet den Belief in den gefundenen Keller, gegeben seine Bestandteile.
$\tau(\mathrm{WS}\|\mathrm{E}_{\mathrm{parts}})$	Berechnet den Belief in eine gefundene Wandfläche, gegeben seine Bestandteile.
$\tau(\mathrm{RS}\|\mathrm{E}_{\mathrm{parts}})$	Berechnet den Belief in eine gefundene Dachfläche, gegeben seine Bestandteile.
$\tau(\mathrm{GS}\|\mathrm{E}_{\mathrm{parts}})$	Berechnet den Belief in eine gefundene Kellerfläche, gegeben seine Bestandteile.

Tabelle 8.2: Die verwendeten anwendungsunabhängigen Vertrauensmaße passend zu Gleichung 8.23.

$\tau(\mathrm{BU}\|\{\mathrm{WA}\})$	Berechnet den Belief in ein gefundenes Gebäude, gegeben die Fassade (*Walling* im Modell genannt).
$\tau(\mathrm{BU}\|\{\mathrm{RO}\})$	Berechnet den Belief in ein gefundenes Gebäude, gegeben das Dach (*Roof* im Modell genannt).
$\tau(\mathrm{BU}\|\{\mathrm{GR}\})$	Berechnet den Belief in ein gefundenes Gebäude, gegeben der Keller (*Ground* im Modell genannt).
$\tau(\mathrm{WA}\|\{\mathrm{WS}\})$	Berechnet den Belief in die gefundene Gebäudefassade, gegeben seine Bestandteile.
$\tau(\mathrm{RO}\|\{\mathrm{RS}\})$	Berechnet den Belief in das gefundene Dach, gegeben seine Bestandteile.
$\tau(\mathrm{GR}\|\{\mathrm{GS}\})$	Berechnet den Belief in den gefundenen Keller, gegeben seine Bestandteile.
$\tau(\mathrm{WS}\|\{\mathrm{OP}\})$	Berechnet den Belief in eine gefundene Wandfläche, gegeben ihre Bestandteile.
$\tau(\mathrm{RS}\|\{\mathrm{OP}\})$	Berechnet den Belief in eine gefundene Dachfläche, gegeben ihre Bestandteile.
$\tau(\mathrm{GS}\|\{\mathrm{OP}\})$	Berechnet den Belief in eine gefundene Kellerfläche, gegeben ihre Bestandteile.

Tabelle 8.3: Die verwendeten anwendungsunabhängigen Vertrauensmaße passend zu Gleichung 8.24.

$\tau(\text{BU}\|\text{E}_{\text{rep}})$	Berechnet den Belief in ein gefundenes Gebäude gegeben seine Repräsentation.
$\tau(\text{WS}\|\text{E}_{\text{rep}})$	Berechnet den Belief in eine gefundene Wandfläche gegeben ihre Repräsentation.
$\tau(\text{RS}\|\text{E}_{\text{rep}})$	Berechnet den Belief in eine gefundene Dachfläche gegeben ihre Repräsentation.
$\tau(\text{GS}\|\text{E}_{\text{rep}})$	Berechnet den Belief in eine gefundene Kellerfläche gegeben ihre Repräsentation.
$\tau(\text{OP}\|\text{E}_{\text{rep}})$	Berechnet den Belief in ein gefundenes Fenster bzw. gefundene Tür gegeben seine/ihre Repräsentation.

Tabelle 8.4: $\tau(\text{BU}|\text{E}_{\text{rep}})$, $\tau(\text{WS}|\text{E}_{\text{rep}})$, $\tau(\text{RS}|\text{E}_{\text{rep}})$ und $\tau(\text{GS}|\text{E}_{\text{rep}})$ verwenden alle dasselbe Vertrauensmaß aus Gleichung 8.37. Einzig für $\tau(\text{OP}|\text{E}_{\text{rep}})$ wird mit Gleichung 8.38 eine eigene Gleichung verwendet.

8.4 Hypothesenraumbewertung

Kapitel 6.4 beschreibt die in dieser Arbeit verwendete Suchraumanalyse und den verwendeten $\epsilon - A^*$-Algorithmus mit Gleichung 6.1: $f(x) = g(x) + \epsilon\, h(x)$. Um die vielversprechendste Hypothese zu ermitteln, wird allen bekannten Hypothesen x jeweils ein Wert $f(x)$ zugeordnet, der angibt, wie hoch die Güte der analysierten Hypothese im günstigsten Fall ist. Die Hypothese mit der höchsten Güte (f-Wert) wird als nächstes untersucht. Für eine Hypothese x bezeichnet $g(x)$ die bisher ermittelte Güte von der Starthypothese aus und die optimal geschätzte Güte von x bis zur Zielhypothese (Analyseergebnis) ist $h(x)$.

In den Fallstudien entspricht $g(x)$ dem Belief in einen Dominostein (DT) (Gleichung 8.25) bzw. in eine Pokerkarte (CARD) oder ein Gebäude (BU) und $h(x)$ entspricht dem Belief in einen Dominostein bzw. in eine Pokerkarte oder ein Gebäude, bei dem alle noch nicht zugeordneten Elemente (Knoten des Objektmodells zu Elementen des Analysemodells) als perfekt zugeordnet mit einem Belief von 100% angenommen werden.

8.5 Diskussion

In diesem Kapitel werden verschiedene Verfahren für Vertrauensmaße präsentiert. Auch wenn sich die Bayes-Theorie ähnlich einer Wahrscheinlichkeit verhält und auch die gleichen Symbole verwendet, handelt es sich dennoch um ein subjektives Vertrauensmaß, das den Axiomen der Wahrscheinlichkeitslehre folgt. In dieser Arbeit kommt die Dempster-Shafer-Theorie zum Einsatz, da sie nicht zur Annahme von “zweiseitigen” Wetten zwingt, mehrere Informationsquellen das gleiche Er-

eignis unterschiedlich beurteilen dürfen und eine komfortable Kombinationsregel Vertrauensmaße aus verschiedenen Quellen wieder zu einem Belief zusammenfasst.

Drei anwendungsunabhängige Vertrauensmaße führen zu einer konsistenten Berechnung des Beliefs in Objekte. Dennoch benötigt jedes Modell anwendungsabhängige Vertrauensmaße, die der Systemarchitekt mit modellieren muss. Dabei handelt es sich in der Regel nur um einige wenige Maße. Jedes extrahierte Bildelement weiß für sich, wie hoch der Belief daran ist, dass es korrekt im Bild gefunden wurde. Diese Information fließt zusätzlich in die Bewertung mit ein, sodass Objekte, die möglicherweise falsch im Bild erkannt wurden, weniger Einfluss auf die Analyse besitzen. Die umgesetzten Bewertungen ermöglichen die Bewertung jeder Hypothese, die Bewertung des Endergebnisses der Analyse und den Vergleich von Hypothesen, die unterschiedlich weit in der Analyse fortgeschritten sind. Auf diese Weise kann mittels ϵ-A^*-Algorithmus die Analyse gesteuert werden und der Suchraum erst während der Analyse aufgebaut werden (siehe Hypothesendefinition auf Seite 146).

Mit der Möglichkeit der Verifikation von Hypothesen sind nun alle Bestandteile (Modell, Modellgenerierung, Kontrollverfahren und Hypothesengenerierung) zusammen, die zur modellbasierten Poseschätzung vonnöten sind. Im nächsten Kapitel wird die Implementation dieser Bestandteile beschrieben.

Kapitel 9

Implementation

Dieses Kapitel beschreibt die Implementation des Verfahrens **knoPoE**. Das Verfahren **sage-sb** wurde größtenteils in C++ implementiert und nur die spätere Modellverbesserung ist in das Komponentenkonzept integriert worden. Die Aspekte der Implementierung werden bzgl. der in Kapitel 3.3 beschriebenen Modelle in Abschnitt 9.1, bzgl. des in Kapitel 3.6 beschriebenen Komponentenkonzepts in Abschnitt 9.2 und bzgl. der in Abschnitt 6 beschriebenen Kontrolle in Kapitel 9.3 beschrieben. Abschnitt 9.4 verdeutlicht an ausgewählten STOR-Komponenten das Zusammenspiel von Modellen, Komponentenkonzept und STOR-Komponenten.

9.1 Modell

Kapitel 3.3 beschreibt im Detail das von Frau Falkowski entwickelte STOR (Software Techniques for Object Recognition)-Referenzschema [FE11] (siehe Definition 2.5).

Abbildung 3.4 auf Seite 77 gibt einen groben Überblick über die Schemapakete und deren Beziehungen.

Die TGraphen ermöglichen eine modellgetriebene Softwareentwicklung, weshalb sie perfekt zu dem Ansatz dieser Arbeit passen. Die mittels UML-Werkzeug entwickelten grUML-Schema-Diagramme werden zunächst als XMI-Datei (XML Metadata Interchange[1]) gespeichert. Dieses Modell wird dann in das sogenannte TG-Format (eine TGraph-Schema-Datei) mittels XML-Transformation überführt.

Der bereits erwähnte *Java Graph Laboratory (JGraLab)* Codegenerator[2] überführt das Schema dann in eine objektorientierte API für Graphen, die komfortablen Zugriff auf die Modelle gestattet [ERW08]. JGraLab ist eine von der Arbeitsgruppe Ebert entwickelte Javabibliothek und stellt eine effiziente API für die Verarbei-

[1] `http://www.omg.org/spec/XMI/`
[2] `http://jgralab.uni-koblenz.de`

tung von TGraphen zur Verfügung. Die Verarbeitung der TGraphen erfolgt mittels JGraLab in Kombination mit einer Graph-Abfragesprache *graph query language (GReQL)* und dem dazugehörigen UML-basierenden *Metamodellierungsansatz grUML* [ERW08]. Schemas werden in grUML spezifiziert und sind dadurch formal definiert [DEF+98]. grUML bildet eine Teilmenge der UML-Klassendiagramme und besitzt eine formale TGraph-Semantik. So werden beispielsweise nur die Elemente verwendet, die auch in einem Graph dargestellt werden können. Klassen entsprechen Knotentypen, Assoziationen entsprechen Kantentypen, Spezialisierungen und Generalisierungen führen zu Hierarchien von Typen und Attribute verfeinern die Informationen eines Knoten- oder Kantentyps. Multiplizitäten entsprechen Einschränkungen im Knotengrad. (Der Inhalt dieses Absatzes entstammt in Auszügen den Veröffentlichungen [ERW08, FE09b])

Das STOR-Referenzschema wird durch die Klasse *STORGraph* repräsentiert und alle Modelle sind somit vom Typ *STORGraph*. Die Klasse STORGraph ist eine Spezialisierung der Klasse *de.uni_koblenz.jgralab.Graph* und besitzt Methoden zur Erzeugung, zum Löschen und zum in Beziehung setzen der Elemente des STOR-Referenzschemas.

Es wurden Interfaces für verpflichtende Methodenimplementierungen entwickelt (siehe Abbildung 3.5 auf Seite 79). Bei den Objektmodellen muss jedes Element des Typs *SemanticObjects* des Referenzschemas das Interface *SemanticDetection* implementieren. Jedes Element des Typs *Geometry* implementiert das Interface *GeometricDetection* und jede *Region* implementiert *RegionDetection*. Elemente vom Typ *SemanticObject*, *Geometry* oder *Region* implementieren alle *CommonDetection*. Dies hat als Konsequenz, dass nun jedes Objekt des zuvor genannten Typs Methoden besitzt, um sich selbst im Bild zu lokalisieren bzw. zu initialisieren, zu bewerten und auf Basis bereits erfolgter Zuordnungen neue *Posehypothesen* (siehe Kapitel 2.3) zu erzeugen. Die Methode *instantiate(STORGraph, HypotheseImpl, TwoDImage)* des Interface *CommonDetection* mit den Argumenten *STORGraph* für das Analysemodell, *HypotheseImpl* für die Hypothese und *TwoDImage* für das Eingabebild ist für die Lokalisierung/Initialisierung zuständig und gibt die getätigten Zuordnungen als *List<InitialisationConcept>* zurück. Die Methode *getBottomUpPoseHypotheses* erzeugt Bottom-Up Posehypothesen und benötigt als Argumente die durch die Instanziierung entstandenen Zuordnungen als *List<InitialisationConcept>* und die Hypothese als *HypotheseImpl* und gibt eine neue Hypothese als Ausgabe zurück. Die Konzepte *InitialisationConcept* und *LimitationConcept* (siehe Abbildung 9.1 und die Begriffserklärungen 3.4 und 3.5) verwalten und beschränken die Zuordnungen von Modellelementen mit Elementen aus dem Analysemodell (siehe Abschnitt 3.4.3).

Die Methoden des Interface *SemanticDetection* verwalten das Wissen über Sichtbarkeit und ob ein Element obligatorisch für die Erkennung ist. Zusätzlich

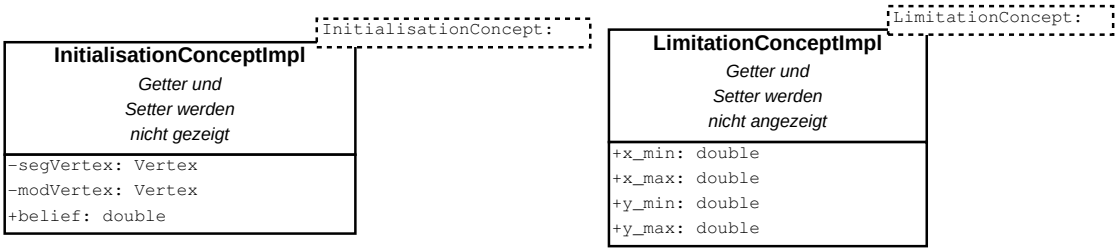


Abbildung 9.1: Ein *InitialisationConcept* beschreibt die konkrete Zuordnung eines Modellelementes zu einem Bildelement und besitzt als Attribute das Modellelement, das segmentierte Bildelement aus dem Analysemodell und eine Bewertung der Zuordnung. Ein *LimitationConcept* beschreibt Beschränkungen des Suchraums im Bild, sodass Objekte, die in einem bereits gefundenen Objekt enthalten sind, nur noch in der entsprechenden ROI gesucht werden.

steuert es "Top-Down" die Erzeugung von Posehypothesen. Die Interfaces *GeometricDetection* und *RegionDetection* erzeugen die aus der Segmentierung entstehenden Begrenzungen des Suchraums. Es ist für die Ablaufkontrolle nicht erforderlich, dass alle Methoden und Attribute definiert werden.

Es wurden eine Reihe eigener Klassen für Elemente des STOR-Referenzschemas bezogen auf die Gebäudefallstudie umgesetzt. Die Beziehung zwischen der Spezifikation eines Graphknotens (als Interface) und einer eigenen Implementierung wird mittels der GraphFactory von jGraLab und der Methode setVertexImplementationClass hergestellt (siehe Tabellen 9.1 und 9.2).

Tabelle 9.1: Eigene Klassen für Elemente des STOR-Referenzschemas bezogen auf die Gebäudefallstudie. Diese Beziehung wird mittels der *GraphFactory* von jGraLab und der Methode *setVertexImplementationClass* hergestellt. Alle Klassen implementieren das Interface STORObject (siehe Abbildung 9.2).

Graphknoten	Eigene Klasse	Beschreibung
Building	BuildingExImpl	Implementiert Interface *SemanticObject* für Gebäude
Walling	WallingExImpl	Implementiert Interface *SemanticObject* für den Gebäudekorpus, bestehend aus den Außenwänden
Roof	RoofExImpl	Implementiert Interface *SemanticObject* für das Dach
Ground	GroundExImpl	Implementiert Interface *SemanticObject* für den Gebäudeboden
WallSurface	WallSurfaceExImpl	Implementiert Interface *SemanticObject* für Wandflächen
RoofSurface	RoofSurfaceExImpl	Implementiert Interface *SemanticObject* für die Dachflächen

Tabelle 9.1: Fortsetzung Eigene Klassen für Elemente des STOR-Referenzschemas

Graphknoten	Eigene Klasse	Beschreibung
GroundSurface	GroundSurfaceExImpl	Implementiert Interface *SemanticObject* für die Bodenflächen
Door	DoorExImpl	Implementiert Interface *SemanticObject* für Türen
Window	WindowExImpl	Implementiert Interface *SemanticObject* für Fenster
OuterBuildingInstallation	OuterBuildingInstallationExImpl	Implementiert Interface *SemanticObject* für Gebäudeinstallationen
RealCamera	RealCameraExImpl	Implementiert Attribute und Methoden bzgl. des Kameramodells Kapitel 3.1
Pose	PoseExImpl	Implementiert Attribute und Methoden bzgl. der Pose im Verständnis dieser ArbeitKapitel 3.1
TwoDPolygonalFaceList	TwoDPolygonalFaceListExImpl	Implementiert Interface *GeometricObject* für eine Liste von 2-D-Flächen
TwoDLineSegmentList	TwoDLineSegmentListExImpl	Implementiert Interface *GeometricObject* für eine 2-D-Fläche
TwoDLineSegment	TwoDLineSegmentExImpl	Implementiert Interface *GeometricObject* für eine 2-D-Strecke
ThreeDPolygonalFaceList	ThreeDPolygonalFaceListExImpl	Implementiert Interface *GeometricObject* für eine Liste von 3-D-Flächen
ThreeDQuadrangle	ThreeDQuadrangleExImpl	Implementiert Interface *GeometricObject* für ein 3-D-Viereck
ThreeDTriangle	ThreeDTriangleExImpl	Implementiert Interface *GeometricObject* für ein 3-D-Dreieck
ThreeDPolygonalFace	ThreeDPolygonalFaceExImpl	Implementiert Interface *GeometricObject* für eine 3-D-Fläche

Tabelle 9.2: Eigene Klassen für Elemente des STOR-Referenzschemas bezogen auf die Fallstudien zur Dominosteinerkennung und Pokerkartenerkennung. Diese Beziehung wird mittels der *GraphFactory* von jGraLab und der Methode *setVertexImplementationClass* hergestellt. Alle Klassen implementieren das Interface STORObject (siehe Abbildung 9.2).

Graphknoten	Eigene Klasse	Beschreibung
CardForm	CardFormImplIM	Implementiert Interface *SemanticObject* für Kartenfarben

Tabelle 9.2: Fortsetzung Eigene Klassen für Elemente des STOR-Referenzschemas

Graphknoten	Eigene Klasse	Beschreibung
Card	CardImplIM	Implementiert Interface *SemanticObject* für die Spielkarte
CharcterForm	CharcterFormImplIM	Implementiert Interface *SemanticObject* für die Buchstaben von Spielkarten
DominoTile	DominoTileImplIM	Implementiert Interface *SemanticObject* für Dominosteine
DominoTilePart	DominoTilePartImplIM	Implementiert Interface *SemanticObject* für die Dominosteinhälften.
FigureCard	FigureCardImplIM	Implementiert Interface *SemanticObject* für Zahlenkarten
NumberForm	NumberFormImplIM	Implementiert Interface *SemanticObject* für Zahlen der Spielkarten
PictureCard	PictureCardImpl	Implementiert Interface *SemanticObject* für Figurenkarten
Pips	PipsImplIM	Implementiert Interface *SemanticObject* für Dominosteinpunkte
Region	RegionImplIM	Implementiert Interface *RegionObject* für Regionen
TwoDCircle	TwoDCircleImplIM	Implementiert Interface *GeometricObject* für 2-D-Kreise
TwodQuadrangle	TwodQuadrangleImplIM	Implementiert Interface *GeometricObject* für 2-D-Vierecke
TwodRectangle	TwodRectangleImplIM	Implementiert Interface *GeometricObject* für 2-D-Rechtecke (Dominosteinerkennung)
TwodRectangle	TwodRectanglePerspectiveImplIM	Implementiert Interface *GeometricObject* für perspektivische 2-D-Rechtecke (Pokerkartenerkennung)

9.2 Komponentenkonzept

Das STOR-Komponentenkonzept wurde als wissenschaftliches Komponentenkonzept konzipiert, um verschiedene Techniken und Datenstrukturen zur Objekterkennung nutzen zu können. Für dieses Komponentenkonzept wurde für das Frontend die Programmiersprache Java gewählt und größtenteils C++ (teilweise auch Java) für das Backend. Da Java im Gegensatz zu C++ einige wichtige Merkmale (Reflektion, Serialisierbarkeit, Multi-Threading) Out-of-the-box bietet, wurde sich für diesen Kombinationssprachenansatz entschieden, obwohl dies die Verarbeitungs-

geschwindigkeit verringern könnte. Abbildung 9.2 gibt einen Überblick über die im Folgenden beschriebenen Zusammenhänge zwischen Komponenten und Datenstrukturen sowie zwischen Java Komponenten/Datenstrukturen und C++ Komponenten/Datenstrukturen.

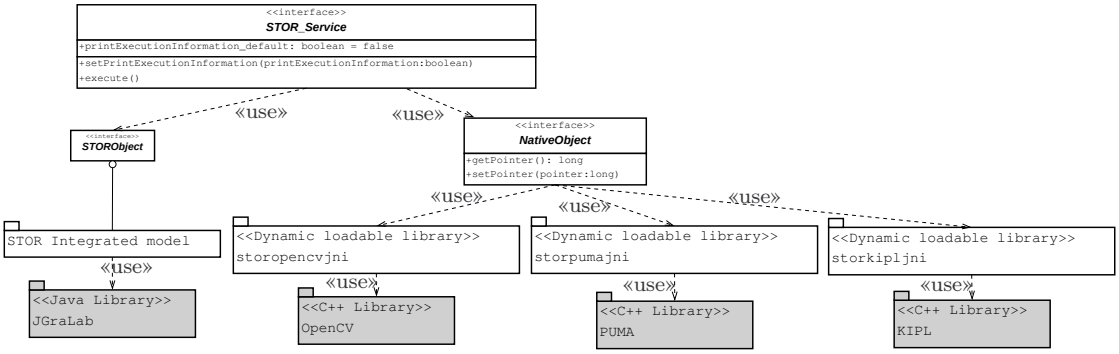


Abbildung 9.2: Übersicht über die STOR Entitäten, die in Beziehung zum *STOR_Service* stehen. Das Interface *STOR_Service* liefert die Grundbeschreibung aller Komponenten. Interne Komponenten von STOR werden weiß dargestellt und Externe Beziehungen werden grau dargestellt. Die Zeichnung basiert auf Abbildung 1 in [FE09b].

Eine *STOR-Datenstruktur* besteht aus einer Spezifikation in Form eines Java Interfaces und einer Implementierung als Java Klasse. Alle Interfaces für STOR-Datenstrukturen erweitern das Interface *STORObject*. Dies gilt auch für die in Tabelle 9.1 und 9.2 beschriebenen Datenstrukturen. Die Interfaces für Datenstrukturen werden in der Regel für mehr als eine Datenstruktur implementiert, beispielsweise implementiert das Interface *TwoDImage* das Interface *STORObject* und die Interfaces *TwoDBinaryImage*, *TwoEdgeImage*, *TwoDGrayScaleImage* und *TwoDRGBImage* implementieren alle das Interface *TwoDImage*. Eine Klasse für eine STOR-Datenstruktur implementiert mindestens ein Interface, das vom Interface *STORObject* abhängt und besitzt mindestens einen *public Konstruktor*. Gewöhnlich ist der Name einer Datenstrukturklasse der Name des Interface erweitert um die Endung *Impl*. (Der Inhalt dieses Absatzes entstammt in Auszügen der Veröffentlichung [FE09b])

Eine *STOR-Komponente* besteht wie eine *STOR-Datenstruktur* aus einer Spezifikation in Form eines Java Interfaces und einer Implementierung als Java Klasse. Alle Interfaces für STOR-Komponenten erweitern das Interface *STOR_Service*. Die Java Interfaces werden durch spezielle Annotationen ergänzt, die es einer Komponente ermöglichen ihre syntaktischen Eigenschaften und zum Teil auch ihre Semantik zur Laufzeit selbst zu beschreiben. Ein Service wird portbasiert beschrieben, d. h. für jede benötigte Entität (Datum oder Service) gibt es einen Eingabeport in Form einer `set()`-Methode, für jedes bereitgestellte Datum einen

Ausgabeport in Form einer `get()`-Methode. Darüber hinaus besitzt jede Komponente eine `execute()`-Methode, die die Komponente anstoßen kann. Die portbasierte Beschreibung ermöglicht die Trennung von Daten- und Kontrollfluss zur Assemblierungs- und Ausführungszeit. Eine vollständige Beschreibung findet sich in Kapitel 3.6.

Es kommen in dieser Arbeit die drei Bibliotheken OpenCV, PUMA und KIPL zum Einsatz. Die Bibliotheken PUMA und KIPL wurden an der Universität Koblenz-Landau entwickelt.

Begriffsbestimmung 9.1 (OpenCV)**.** *"OpenCV ist eine freie Programmbibliothek mit Algorithmen für die Bildverarbeitung und maschinelles Sehen. Sie ist für die Programmiersprachen C und C++ geschrieben und steht als freie Software unter den Bedingungen der BSD-Lizenz. Das "CV" im Namen steht für englisch "Computer Vision". Die Entwicklung der Bibliothek wurde von Intel initiiert und wird heute hauptsächlich von Willow Garage gepflegt.*

Die Stärke von OpenCV liegt in ihrer Geschwindigkeit und in der großen Menge der Algorithmen aus neuesten Forschungsergebnissen." [a]

[a] `http://www.wikipedia.de`, Stichwort: OpenCV, Zugriff: 19.05.2017

Begriffsbestimmung 9.2 (PUMA)**.** *"Die Programmierumgebung für die Musteranalyse (PUMA)* [a] *[Pau92, PH03] ist eine API und wurde von der Arbeitsgruppe Aktives Sehen der Universität Koblenz-Landau entwickelt. Sie bietet Algorithmen, Datenstrukturen und kleine spezialisierte ausführbare Programme zur Objekterkennung, für Bildverarbeitung und Computervision. PUMA wurde in C/C++ entwickelt und basiert auf* Koblenz-NIHCL (KONIHCL) *einer Variante des* National Institute of Health Class Library (NIHCL) *[GOP90] und ist verfügbar für Mac OS X, Linux und Windows. PUMA wird seit 1992 hauptsächlich entwickelt von Prof. Dr.-Ing. Dietrich Paulus, seinen Mitarbeitern und Studenten." (aus dem Englischen [FE09b, Seite 14])*

[a] English: Programming environment for pattern recognition, `http://www.uni-koblenz-landau.de/koblenz/fb4/institute/icv/agpaulus/agas-projekte/puma`

Begriffsbestimmung 9.3 (KIPL)**.** *"Die Koblenzer Image Processing Library (KIPL) ist eine API für die Bibliothek Image Recognition Laboratory der Universität Koblenz-Landau für Algorithmen und Datenstrukturen im Bereich Bildverarbeitung und Objekterkennung. KIPL wurde hauptsächlich von Frank Schmitt und Patrick Sturm unter der Leitung von Prof. Dr. Lutz Priese in C/C++ entwickelt und ist verfügbar für Mac OS X, Linux und Windows. KIPL wird in einem Plugin-Framework [Bal06] von Dirk Balthasar verwendet, so dass die gekapselten Algorithmen in einem eigenen Frontend visualisiert und*

zur Laufzeit zusammengestellt oder als plugin in die freie Software GIMP[a] *integriert werden können.“ (aus dem Englischen [FE09b, Seite 14])*

[a] `http://www.gimp.org`

Diese Bibliotheken werden mittels *Java Native Interface* an die STOR-Komponenten angebunden. Das Interface *NativeObject* stellt auf Komponentenseite die minimale Spezifikation der Datenstrukturen der nativen Bibliotheken dar.

Begriffsbestimmung 9.4 (Java Native Interface)**.** *”*Java Native Interface (JNI) *[Lia99] ist eine standardisierte Anwendungsprogrammierschnittstelle (API), die die Möglichkeit schafft, aus der Programmiersprache Java heraus Plattform-spezifische Funktionen bzw. Methoden aufzurufen. Es stellt die Schnittstelle zu anderen Programmiersprachen wie C oder C++ dar. Umgekehrt können “native” Programme Java-Methoden über JNI aufrufen oder eine JVM ausführen. Im Gegensatz zu einfachen Java Programmen, sind Java Programme, welche native Bibliotheken via JNI nutzen, nicht notwendigerweise plattformunabhängig.“ (adaptiert aus dem Englischen [FE09b, Seite 15])*

Alle Datenstrukturen nativer Bibliotheken werden entsprechend ihrer Anforderungen um weitere Interfaces erweitert und als Java Klasse implementiert. Für OpenCV wird in der Regel nicht der Weg beschritten OpenCV-Programme über JNI anzubinden, sondern es werden OpenCV-Programme über die OpenCV-Wrapper-Bibliothek JavaCV[3] direkt in Java realisiert.

Begriffsbestimmung 9.5 (JavaCV)**.** *”JavaCV uses wrappers from the JavaCPP Presets of commonly used libraries by researchers in the field of computer vision (OpenCV, FFmpeg, libdc1394, PGR FlyCapture, OpenKinect, librealsense, CL PS3 Eye Driver, videoInput, ARToolKitPlus, and flandmark), and provides utility classes to make their functionality easier to use on the Java platform, including Android.“*[a]

[a] `https://github.com/bytedeco/javacv`

Äquivalent zu reinen Java Datenstrukturklassen ist gewöhnlich der Name einer Komponente der Name des Interface erweitert um die Endung *Impl*. Eine Auflistung der in dieser Arbeit entstandenen STOR-Komponenten gibt Tabelle 9.3.

9.3 Kontrolle

Der Kern der Kontrollstrategie nimmt den initial befüllten Hypothesenraum und analysiert iterativ die enthaltenen Modelle, welche sich durch unterschiedliche Pose-

[3] `https://github.com/bytedeco/javacv`

hypothesen und somit unterschiedlicher 2-D-Geometrie differenzieren. Im Zuge der Analyse entstehen Hypothesen für mögliche Posen. Die entstandenen Hypothesen werden bewertet und in den Hypothesenraum überführt. Die Strategie (A^*-Suche) mit der die jeweils plausibelste Hypothese aus dem Hypothesenraum gesucht wird, wird in Abschnitt 6.4 beschrieben. Nach Abschluss der Analyse wird die plausibelste Hypothese ausgegeben. Abbildung 9.3 gibt einen Überblick über den Ablauf der Kontrolle, die in der STOR-Komponente *ActivityControlImpl* gekapselt wird. Die verwendeten Datenstrukturen werden in Abbildung 9.4 in Verbindung zur STOR-Komponente *ActivityControlImpl* gesetzt. Abbildung 9.3 und Abbildung 9.4 beschreiben über Einfärbung mit Grüntönen gleiche Konzepte.

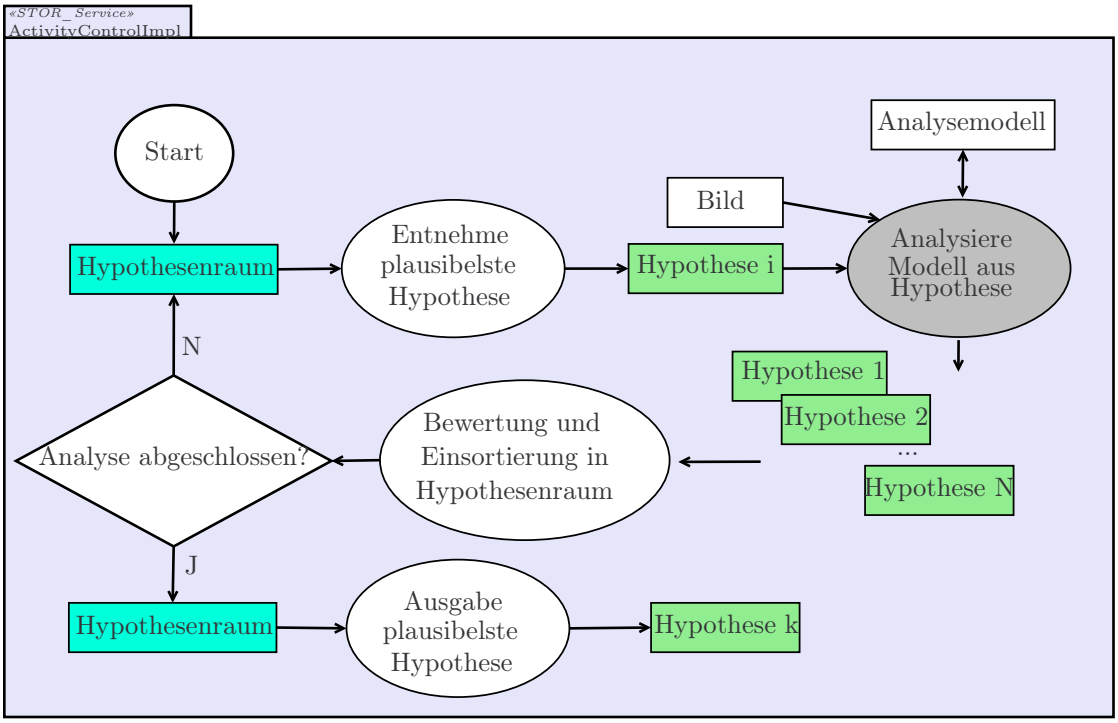


Abbildung 9.3: Übersicht über den Kontrollalgorithmus. Konzepte, die Konzepten aus Abbildung 9.4 entsprechen, werden über eine identische Einfärbung signalisiert.

Die Hypothese (siehe Hypothesendefinition auf Seite 146) wird als *StateImpl* mit einer eindeutigen *id* umgesetzt und besitzt entsprechend der Fallstudienbeschreibung Attribute für den Belief (*belief*, umgesetzt als Feld, um A* zu genügen (speichert in belief[0]=$f(x)$ und in belief[1]=$g(x)$)), für das zugehörige Modell mb_υ (*STORGraph*), für die Pose υ (*pose*), für die Zuordnungen (*init*) sowie die Beschränkungen (*limit*), für die Vorgängerhypothese (*prior*) und für die Steuerung der Analyse, die hinzugefügten, aber noch nicht bearbeiteten Knoten des Modells (*lastAddedVertex*). Der Hypothesenraum (*StateSpaceImpl*) verwaltet die Hypothe-

sen in zwei Listen. Die Liste *openList* beinhaltet alle noch nicht abgeschlossenen Hypothesen, also die Blattknoten des Baums, der durch den Hypothesenraum aufgespannt wird (siehe Kapitel 6.4). Diese Hypothesen sind sortiert nach ihrem Belief, so dass in der Analyse stets die plausibelste Hypothese weiter verfolgt wird. Die Liste *closeList* speichert alle fertig analysierten Hypothesen. Die STOR-Komponente *ActivityControlImpl* beinhaltet das *Analysemodell*, das *Eingabebild*, den *Schwellwert* θ_3 als untere Schranke des Beliefs für die Akzeptanz von Hypothesen, den *GraphMarker segvertexini* zur Markierung schon zugeordneter Elemente des Analysemodells und eine Liste von Modellen (*modelle*), die als Eingabe für die Analyse dienen.

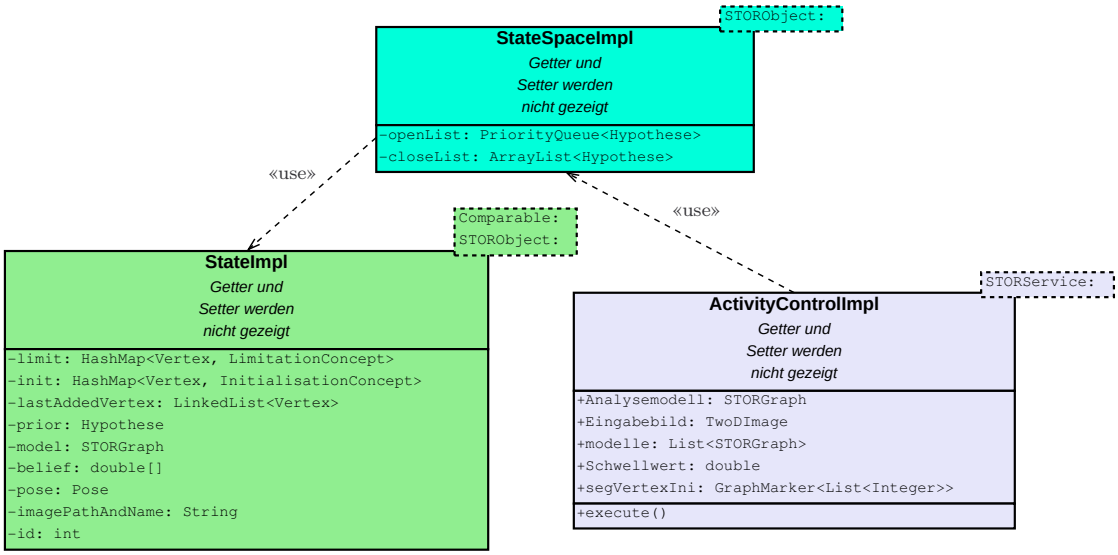


Abbildung 9.4: Umsetzung der Kontrolle. Zur Übersicht wurden die getter und setter Methoden der Attribute weggelassen. Es werden zudem die Datentypspezifischen Methoden zur Handhabung der Datentypen implementiert.

Der prinzipielle Ablauf der Kontrolle wird durch den Pseudocode 6.2 auf Seite 136 beschrieben.

9.4 Komponenten in der Praxis

Am Beispiel der Klasse *WindowExImpl*, das das Interface *Window* implementiert, sollen die Komponenten beschrieben werden, die in der Methode *instantiate* verwendet werden. Der Pseudocode 9.1 zeigt den prinzipiellen Ablauf der Methode *instantiate* der Klasse *WindowExImpl*. Abbildung 9.5 gibt einen Überblick über die in der *instantiate* Methode verwendeten Komponenten.

Die Klasse *WindowExImpl* implementiert das Interface *SemanticDetection*, das das Interface *CommonDetection* spezialisiert. Zudem spezialisiert *WindowExImpl* die automatisch mit JGraLab erzeugte Klasse *WindowImpl*. Die Klasse *WindowImpl* enthält automatisch generierte Methoden zur Nutzung der per UML-Diagramm für Fenster (*Windows*) definierten Attribute und Relationen. Im Zuge der Analyse wird die *instantiate* Methode der Klasse *WindowExImpl* aufgerufen (siehe Regel 1 in Abbildung 6.5 in Kapitel 6.2). Der Pseudocode 9.1 beschreibt den Ablauf der Instanziierung von Fenstern/Türen.

Pseudocode 9.1 Instanziierung Fenster

```
1: function INSTANTIATE(Analysemodell mA, Hypothese Hyp, Bild I)
2:     if Ist noch nicht instanziiert worden? then
3:         Liste<Analysemodellfenster> OP_S ← OPENINGDETECTIONANDEXTRACTION.EXECUTE(mA, I)
4:         Füge gefundene Fenster/Türen OP_S zu mA hinzu
5:         Liste<Modellfenster> OP_M ← ENTNEHME(Modellfenster aus Hyp)
6:         Liste<InitialisationConcept> Initialisierungen← ZUORDNUNGMITUNGARISCHERMETHODE(OP_M,OP_S)
7:         return Initialisierungen
```

Wie im Pseudocode 9.1 zu sehen, wird die *execute* Methode der STOR-Komponente *OpeningDetectionAndExtractionImpl* aufgerufen. Die Komponente *OpeningDetectionAndExtractionImpl* implementiert die Interfaces *OpeningDetectionAndExtraction* (nicht in Abbildung 9.5 enthalten) sowie *STOR_Service* und enthält als Attribute das Eingabebild *TwoDImage* (Element des Referenzschemas (*STORGraph*)), das Analysemodell als *STORGraph* und einen Vektor zur Rückgabe von gefundenen Fenstern. Die gefundenen Fenster werden als *WindowROI* repräsentiert. *WindowROI* implementiert das Interface *STORObject* und verwaltet die Fensterkandidaten als achsenparallele Rechtecke. Es wird in *WindowROI* gespeichert, in welchem der Farbräume (CSC-L*a*b, CSC-HSV, CSC-Euclid), mit welcher Weichzeichenstufe *sigma* (falls das Fenster in mehreren Weichzeichenstufen erkannt wurde, wird das kleinste *sigma* gewählt) und in wie vielen Weichenzeichenstufen dieses Fenster gefunden wurde (*frequency*). Zudem beinhaltet *WindowROI* einen Belief (*beliefFoundWindow*), dass es sich um ein Fenster handelt, und einen Belief (*beliefWrongFoundWindow*), dass es sich um kein Fenster handelt.

Der Ablauf der *execute* Methode der STOR-Komponente *OpeningDetectionAndExtractionImpl* wird im Pseudocode 9.2 beschrieben. Zunächst wird die *execute* Methode der STOR-Komponente *WindowDetectionComplexImpl* aufgerufen. *WindowDetectionComplexImpl* implementiert die Interfaces *WindowDetectionComplex* (nicht in Abbildung 9.2 enthalten) und *STOR_Service*. Die Komponente *WindowDetectionComplexImpl* steuert den Ablauf (siehe Pseudocode 9.3)

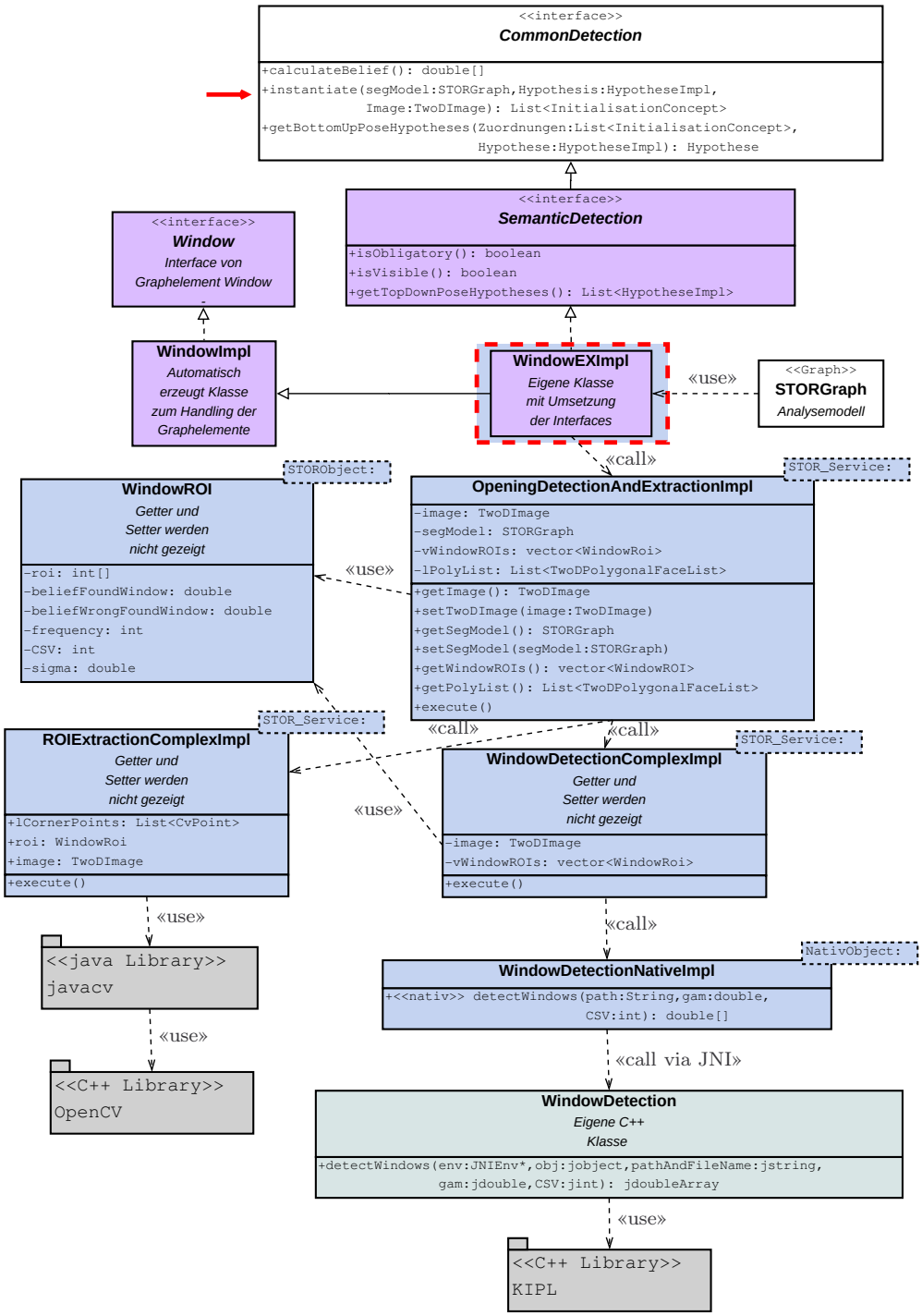


Abbildung 9.5: Komponenten am Beispiel der Methode *instantiate*. Den Ausgangspunkt des Beispieles bildet die *instantiate* Funktion (roter Pfeil) der Komponente *WindowEXImpl* (rot umrandet). Alle blau eingefärbten Komponenten sind in Java im Zuge dieser Arbeit geschrieben. Die hellblaue Komponente ist in C++ im Zuge dieser Arbeit entwickelt worden und bei den grau eingefärbten Elementen handelt es sich um externe Bibliotheken. Die Getter und Setter werden zur Übersichtlichkeit größtenteils weggelassen.

Pseudocode 9.2 Lokalisierung der Fenster im Bild

```
1: function OpeningDetectionAndExtractionImpl.execute(Bild I)
2:     Vektor<FensterROI> FensterROIs ← WindowDetectionComplexImpl.execute(I)
3:     for Fenster_i ∈ FensterROIs do
4:         Liste<2D-Punkt> Eckpunkte ← ROIExtractionComplexImpl.execute(I, Fenster_i)
5:         Liste<2D-Punkt> Viereck ← PasseViereckEin(Eckpunkte)
6:         Einfügen der 2D-Punkte des Vierecks als TwoDPolygonalFaceList in List<TwoDPolygonalFaceList>
```

der Fenstererkennung, wie sie in Kapitel 5.5 beschrieben wird. Es wird für die Verarbeitung der drei verwendeten Farbräume jeweils ein Thread gestartet, um die Verarbeitung zu beschleunigen. Die Kombination der extrahierten Fenster erfolgt, wenn Fenster aus verschiedenen Farbräumen und/oder Weichzeichenstufen an der gleichen Stelle im Bild zu finden sind. Bei der Fensterkombination wird der Belief des kombinierten Fensters (*WindowROI*) mittels der Dempster-Shafer-Kombinationsregel neu berechnet und die *frequency* (Attribut von *WindowROI*) entsprechend erhöht.

Pseudocode 9.3 Übersicht Fenstererkennung

```
1: function WindowDetectionComplexImpl.execute(Bild I)
2:     ▷ Starte Threads JoinerThread:
3:     JoinerThread.run(I, CSC-L*a*b)
4:     JoinerThread.run(I, CSC-HSV)
5:     JoinerThread.run(I, CSC-Euclid)
6:     if Threads sind fertig? then
7:         Vektor<FensterROI> FensterROIs ← erkannte Fenster der Threads
8:         FensterROIs ← kombiniere Fenster
9:         FensterROIs ← behandle Fenster in Fenstern
10:        FensterROIs ← entferne Fenster aus dem Himmel
11:    return FensterROIs
12: function JoinerThread.run(Bild, CSV)
13:    for int sigma 0 bis 8 do
14:        detectWindowsNativ(Bild, sigma, CSV)        ▷ (siehe Kapitel 5.5)
```

Die native Fenstererkennung erfolgt über die Klasse *WindowDetectionNativeImpl*, welche das Interface *NativeObject* implementiert. Die Funktion *detectWindows* der Klasse *WindowDetectionNativeImpl* ruft via *Java Native Interface* die Metho-

de *detectWindows* der C++ Klasse *WindowDetection* auf. Das Fenstererkennungsverfahren wurde im *graph-plugin* von KIPL zusammengestellt und dann per *export*-Funktionalität in C++ Code konvertiert (siehe Abbildung 9.6). Die Übergabe der Argumente und Rückgabewerte erfolgt über spezielle JNI-Datentypen.

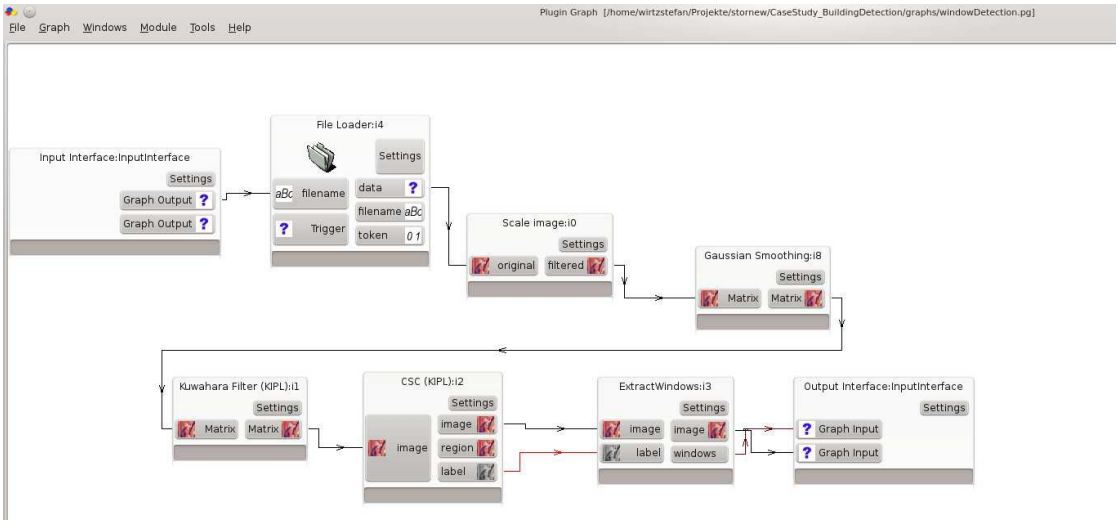


Abbildung 9.6: Hier wird die C++ Fenstererkennung im KIPL *graph-plugin* visualisiert. Das Verfahren wurde im *graph-plugin* zusammengestellt und dann per *export*-Funktionalität in C++ Code konvertiert.

Wie im Pseudocode 9.2 beschrieben, erfolgt nach dem Aufruf der *execute* Methode der STOR-Komponente *WindowDetectionComplexImpl* der Aufruf der *execute* Methode der STOR-Komponente *ROIExtractionComplexImpl*. *ROIExtractionComplexImpl* implementiert die Interfaces *ROIExtractionComplex* (nicht in Abbildung 9.5 enthalten) und *STOR_Service*. In *ROIExtractionComplexImpl* werden zu einem achsenparallelen Fensterkandidaten (*WindowRoi roi*) die Eckpunkte des Fensters (*List<CvPoint> lCornerPoints*) gesucht. Die Eckpunkte können durch die perspektivische Verzerrung relativ weit von der ROI (*roi*) abweichen. Pseudocode 9.4 beschreibt die Extraktion der Eckpunkte des Fensters (*List<CvPoint> lCornerPoints*) aus dem Bild.

Wie im Pseudocode 9.2 beschrieben, wird im Anschluss an die Extraktion der Fenstereckpunkte ein Viereck in diese Punktliste eingepasst und die gefundenen Fenstereckpunkte in *List<TwoDPolygonalFaceList>* gespeichert.

Eine Auflistung der in dieser Arbeit entstandenen STOR-Komponenten gibt Tabelle 9.3.

Pseudocode 9.4 Extrahiere Fenstereckpunkte auf der Basis von einer ROI

```
1: function ROIExtractionComplexImpl.execute(Bild I, FensterROI roi)
2:     Bild I_roi ← Extrahiere roi aus Bild
3:     Binarisiere I_roi
4:     Closing auf binarisierter I_roi
5:     Extrahiere Kontur aus binarisierter I_roi
6:     Reduziere Kontur auf konvexe Hülle
7:     Wandle konvexe Hülle in Liste<2D-Punkt> Eckpunkte
8:     return Eckpunkte
```

Tabelle 9.3: Auflistung der im Rahmen dieser Arbeit entstandenen STOR-Komponenten. Grau eingefärbt sind die in der Arbeit letztendlich verwendeten Komponenten. Der vollständige Name des Package ist immer de.uni_koblenz.stor.components.<Package>.

Package	Name	Beschreibung
distancecalculation.impl	AngleDistanceImpl	siehe Anhang E
distancecalculation.impl	ClosestPointDistance-Impl	siehe Anhang E
distancecalculation.impl	HausdorffDistanceImpl	siehe Anhang E
distancecalculation.impl	HausdorfLineSegment-DistanceImpl	siehe Anhang E
distancecalculation.impl	StraightLineDistan-ceImpl	siehe Anhang E
distancecalculation.impl	MidPointDistanceImpl	siehe Anhang E
distancecalculation.impl	PerpendicularDistan-ceImpl	siehe Anhang E
distancecalculation.impl	TruccoDistanceImpl	siehe Anhang E
distancecalculation.impl	MaurWirtzDistanceIm-pl	siehe "eigene Distanzfunktion" in Anhang E
distancecalculation.impl	MahalanobisDistan-ceImpl	Mahalanobisdistanz
distancecalculation.impl	DistanceCalculation-_Function1Impl	Distanz zwischen Hu-Momenten
distancecalculation.impl	DistanceCalculation-_Function2Impl	Distanz zwischen Hu-Momenten
distancecalculation.impl	DistanceCalculation-_Function3Impl	Distanz zwischen Hu-Momenten für die Zeichen- und Kartenfarbenerkennung
informationdetection.impl	BluePrintImpl	Erstellung Blueprint (siehe Kapitel 5.3)
informationdetection.impl	SkyDetectionImpl	Himmelserkennung

Tabelle 9.3: Fortsetzung Auflistung der im Rahmen dieser Arbeit entstandenen STOR-Komponenten

Package	Name	Beschreibung
informationdetection.impl	LineSegmentationAndExtensionImpl	Extraktion Strecken aus Bild
informationdetection.impl	LineSegmentationAndExtensionBPImpl	Extraktion Strecken aus Blueprint
informationdetection.impl	OpeningDetectionAndExtractionImpl	Finde Fenster
informationdetection.impl	ROIExtractionImpl	Finde Eckpunkte von Fenster in ROI
informationdetection.impl	ROIExtractionComplexImpl	Finde Eckpunkte von Fenster in ROI
informationdetection.impl	RoofSegmentationDetectionImpl	Extrahiere Dachkanten
informationdetection.impl	SkyDetectionUsingBinaryImageImpl	Extraktion Himmel
informationdetection.impl	SkyDetectionUsingBluePrintImpl	Extraktion Himmel
informationdetection.impl	SkyDetectionUsingColorImageImpl	Extraktion Himmel
informationdetection.impl	WindowDetectionImpl	Extraktion Fenster
informationdetection.impl	WindowDetectionComplexImpl	Extraktion Fenster
informationdetection.impl	WindowDetectionNativeImpl	Extraktion Fenster
informationdetection.impl	CircleDetectionContour_PUMAImpl	Kreiserkennung
informationdetection.impl	RectangleDetectionContour_PUMAImpl	Rechteckerkennung
informationdetection.impl	QuadrangleDetection_OpenCVImpl	Viereckerkennung
informationdetection.impl	SubImageRectification_OpenCVImpl	Rektifizierung
control.impl	ActivityControlImpl	siehe Kapitel 6
control.impl	BuildingPreLimitationImpl	Initiale Einschränkung des Suchbereichs
error.impl	PoseModelImpl	Posemodell für RANSAC
matching.impl	WindowMatchingImpl	Zuordnung von Fenstern
matching.impl	WindowMatchingNewImpl	Zuordnung von Fenstern
matching.impl	LineMatchingImpl	Zuordnung von Strecken

Tabelle 9.3: Fortsetzung Auflistung der im Rahmen dieser Arbeit entstandenen STOR-Komponenten

Package	Name	Beschreibung
matching.impl	LineMatchingUsingROIImpl	Zuordnung von Strecken in ROI
matching.impl	QuadrangleImpl	Zuordnung von Vierecken
geometry.impl	DistancePointToPolygonImpl	Abstand Punkt zu Polygon
geometry.impl	ModelTranslationRoofVertexImpl	Verschieben des Modells in Ursprung linker/oberer Dachpunkt
geometry.impl	ModelTranslationAndRotationRoofVertexImpl	Verschieben/Drehen des Modells in Ursprung linker/oberer Dachpunkt
geometry.impl	RectifyOpeningsImpl	Rektifizierung von Fenstern
geometry.impl	RectifyOpeningsInBuildingModelsImpl	Rektifizierung von Fenstern
urbanobjects.impl	CityGMLModel2STORModelForceBuildingImpl	für **sage-sb**
urbanobjects.impl	CityGMLToSimpleStorModelImpl	für **sage-sb**
urbanobjects.impl	DrawAndShowUrbanModelImpl	für **sage-sb**
urbanobjects.impl	SemiAutomaticModelImprovementImpl	für **sage-sb**
posecalculation.impl	AbsolutOrientationImpl	Poseberechnung
posecalculation.impl	BackprojectionImpl	Rückprojektion
posecalculation.impl	LinearPoseFioreImpl	Poseberechnung nach Fiore
posecalculation.impl	RANSACImpl	RANSAC
posecalculation.impl	SemanticRANSACImpl	RANSAC für Posebestimmung mit Fenstern
posehypotheses.impl	GenerateSemanticRenderingsImpl	semantisches Rendering
posehypotheses.impl	GenerateSemanticRenderingsWithListImpl	semantisches Rendering
posehypotheses.impl	PoseHypothesisForComplexPitchedRoofBuildingsImpl	Posehypothesen
posehypotheses.impl	PoseHypothesisForPitchedRoofBuildingsImpl	Posehypothesen

Tabelle 9.3: Fortsetzung Auflistung der im Rahmen dieser Arbeit entstandenen STOR-Komponenten

Package	Name	Beschreibung
posehypotheses.impl	PoseHypothesisWith-ListOfModelWindow-CorrespondecesImpl	Posehypothesen
posehypotheses.impl	PoseWithVanashing-PointsImpl	Posehypothesen
rendering.impl	ExtractGeometryAnd-AssignColorSimpleImpl	Rendering
rendering.impl	ModelAndPoseVisuali-sation_OpenGLImpl	Rendering
rendering.impl	SemanticModelRende-ringImpl	semantisches Rendering
rendering.impl	SemanticUrbanModel-RenderingImpl	semantisches Rendering
utilities.impl	KPermutationImpl	K-Permutation für RANSAC
regionformdetection.impl	CardCharacterFormDe-tection_OpenCVImpl	Erkennung Buchstaben der Spielkarten
regionformdetection.impl	CardColorFormDetecti-on_OpenCVImpl	Erkennung Kartenfarben der Spielkarten
regionformdetection.impl	NumberFormDetecti-on_OpenCVImpl	Erkennung Zahlen der Spiel-karten
regionformdetection.impl	CardCharacterFormDe-tection_OpenCVImpl	Erkennung Buchstaben der Spielkarten

Kapitel 10

Evaluation des Gesamtsystems

Zur Erkennung von Objekten werden Objektmodelle eingesetzt, deren Bestandteile mit Merkmalen im Bild in Beziehung gesetzt werden. Drei verschiedene Anwendungen dieser Erkennungsstrategie wurden spezifiziert (siehe Kapitel 2). Diese Anwendungen unterscheiden sich in der Komplexität und der Dimension der Objekte. Sie nutzen eine gemeinsame Ablaufkontrolle, die unabhängig von der Anwendung ist. Unter Verwendung der anwendungsabhängigen Modelle werden geeignete Merkmalsdetektionsverfahren durch die Kontrolle ausgelöst. Im einfachsten Beispiel der Dominosteinerkennung sind dies beispielsweise Kreis- und Rechteckdetektoren. Für die Erkennung von Gebäuden in einer Modellumgebung oder in der realen Umwelt werden Geraden, Strecken, Fluchtpunkte und auch höherwertige Merkmale wie Fenster und Dächer gesucht. Bei den Außenaufnahmen stört Vegetation die Erkennung kaum. In allen drei Szenarien werden sehr gute Erkennungsraten erzielt, was in den Experimenten belegt wird.

10.1 Dominostein- und Pokerkartenerkennung

In diesen Fallstudien wird gezeigt, dass die modellbasierte Objekterkennung gute Ergebnisse liefert. Es kann gezeigt werden, dass modellbasierte Objekterkennung konkurrenzfähig ist mit dem gleichzeitigen Vorteil, dass sowohl der Belief jedes Modells als auch jeder Zuordnung bekannt ist.

10.1.1 Experiment

Die Bilder des Testdatensatzes wurden mit einem Drehteller in einer *Lichtbox* erstellt (JUST Pantone Colour Viewing Box 1). Für das Experiment wurden 489 Bilder von einzelnen Dominosteinen vor homogenem Hintergrund erzeugt und ver-

wendet. Eine Kamera[1], die nahezu orthogonal zu den Dominosteinen angeordnet war, wurde zur Bildaufnahme verwendet. Außerdem wurden die Dominosteine in 22-Grad-Schritten mittels Drehteller gedreht. Es wurden 8, 12 oder 16 Lichtquellen pro Aufnahme aktiviert, um unterschiedliche Beleuchtungssituationen zu erhalten. Zusätzlich zu den automatisch erzeugten Bildern wurden Bilder unter direkter und indirekter Sonneneinstrahlung erzeugt.

Die Aufnahmen der Pokerkarten erfolgten hingegen auf einem Spieltisch mit einer Spiegelreflexkamera[2]. Die Bilder haben eine Auflösung von 3872 x 2592 Pixeln und sind im JPEG-Format gespeichert. Es wurden 3 Verzerrungsstufen über den Winkel zwischen Kamera und Spielkarte erzeugt: starke Verzerrung: ca. 30°, wenig Verzerrung: ca. 45°, keine Verzerrung: ca. 90°. Anhang A zeigt eine Teilmenge der zur Evaluation verwendeten Datensätze.

10.1.2 Ergebnisse

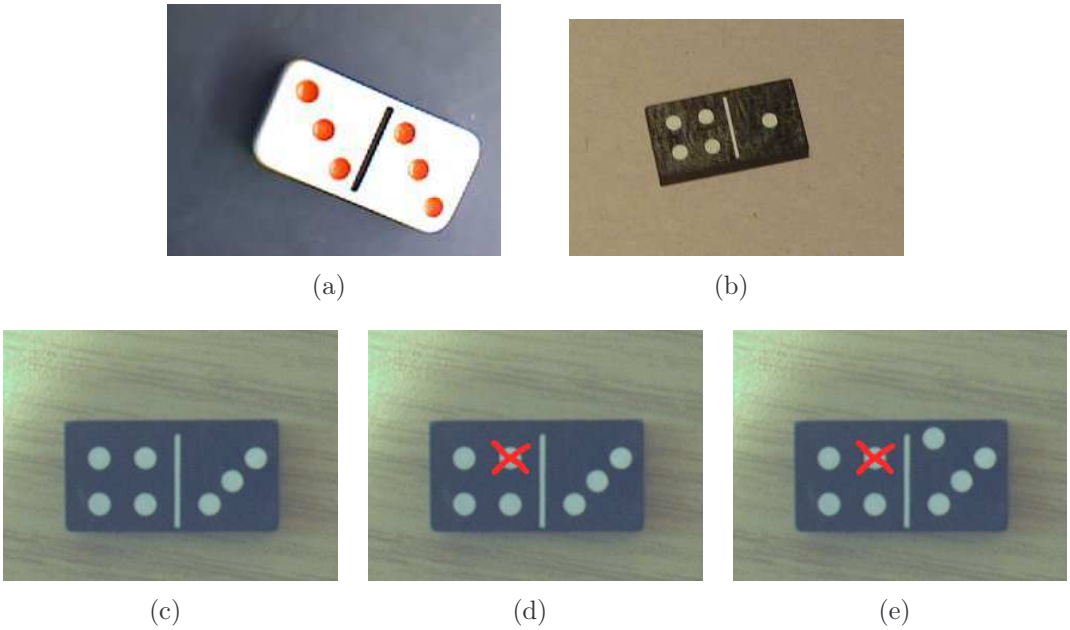
	Recall	Genauigkeit	Erkennungsrate
Pokerzahlenkarten	98,0%	94,4%	92,7%
Pokerbildkarten	98,0%	89,0%	87,4%
Alle Pokerkarten	98,0%	93,2%	91,5%
Dominosteine	90,4%	97,8%	88,6%

Tabelle 10.1: Recall, Genauigkeit und Erkennungsrate für Dominosteine und Pokerkarten. Die Pokerkarten werden nochmal unterteilt in Zahlenkarten und Bildkarten.

Die Erkennungsrate für Dominosteine und Pokerkarten ist mit 90,6% auf den kompletten Datensatz gut. Die Werte für Recall (98,0%/90,4%) und Genauigkeit (93,2%/97,8%) zeigen, dass die Klassifikation eine gute Balance zwischen Recall und Genauigkeit besitzt. Die Ergebnisse insgesamt (siehe Tabelle 10.1) belegen die Möglichkeit, das Verfahren auch zur Poseschätzung von Gebäuden zu nutzen. Ein Grund für die etwas schlechtere Leistung bei der Klassifikation von Dominosteinen im Vergleich zu Pokerkarten ist deren 3-D Struktur und die wenigen Merkmale. Diese Fallstudie zeigt, dass der modellbasierte Ansatz in der Lage ist mit fehlenden und ungenauen Daten umzugehen. Die gezielte Suche in spezifischen Regionen, basierend auf zuvor zugeordneten Modellelementen, verbessert außerdem die Segmentierungsergebnisse gegenüber der Analyse des gesamten Bildes.

Abbildung 10.1(c) zeigt ein Bild des Datensatzes, bei der die Klassifikation korrekt den Dominostein als 4-3-Dominostein mit einem Belief von 97,3% erkennt. Um die Klassifikation zu erschweren, wurde in Abbildung 10.1(d) und 10.1(e) ein

[1]Panasonic Lumix DMC-FS6 Digitalkamera, Auflösung um die 1000x800
[2]Nikon D60, AF-S DX NIKKOR 18-55mm f/3.5-5.6G VR



(a) (b)

(c) (d) (e)

Abbildung 10.1: Beispielbilder: Die Bilder (a) und (b) von Dominosteinen sind aus zwei unterschiedlichen Datensätzen der Bilddatenbank entnommen. Das Bild (c) ist eins der Bilder der Datenbank, welche unter Sonnenlicht aufgenommen wurden. Dieses Bild wurde zusätzlich händisch verfälscht, um die Klassifikation zu erproben. Es wurde in (d) und (e) ein Kreis maskiert und in (e) ein zusätzlicher Kreis hinzugefügt.

Kreis für die Erkenner unkenntlich gemacht und in Abbildung 10.1(e) ein zusätzlicher Kreis hinzugefügt. Der Dominostein aus Abbildung 10.1(d) liefert dennoch eine richtige Klassifikation mit einem Belief von 84,9%. Die Klassifikation ist in diesem Fall in der Lage, den richtigen Dominostein zu erkennen, obwohl die Segmentierung unvollständig ist, da die Position der Kreise stärker gewichtet wird als deren Anzahl. Für den Dominostein aus Abbildung 10.1(e) schlägt die Klassifikation allerdings fehl. Die Klassifikation liefert als Ergebnis einen 4-4-Dominostein mit einem Belief von 68,9%. Dies geschieht, da es wahrscheinlicher ist, dass ein Kreis, vor allem mit korrektem Radius, eher an der falschen Position liegt, als dass er falsch segmentiert wurde. Wie zu erwarten, ist der Belief für diesen Dominostein niedrig. Neben dem Gesamtbelief kennen wir auch die Beliefs für die einzelnen Zuordnungen. In diesem Fall beträgt der Belief für die Zuordnung des Kreises mit dem Modellkreis 41,5%, wohingegen die Beliefs der anderen Kreiszuordnungen im Durchschnitt bei einem Wert von 98% liegen.

Abbildung 10.2(a) zeigt eine Pokerkarte des Testkartensatzes. Die zwei mit der höchsten Wahrscheinlichkeit gefundenen Klassen für diese Karte sind Herz-Sieben mit einem Vertrauen von 79,6% und die Herz-Sechs mit einem Vertrau-

(a) Herz-7 des Datensatzes. Korrekt klassifiziert.

(b) Ein fehlendes Herz. Korrekt klassifiziert.

(c) Zwei fehlende Herzen. Korrekt klassifiziert.

(d) Drei fehlende Herzen. Falsch klassifiziert.

Abbildung 10.2: Beispiel, bei dem die Pokerkartenerkennung funktioniert und bei dem sie nicht funktioniert.

en von 76,1%. Wenn eins der Herzen nicht erkannt werden kann, wie in Abbildung 10.2(b), sind die wahrscheinlichsten Klassen wieder die Herz-Sieben (74,1%) und Herz-Sechs (73,1%). Selbst wenn zwei Herzen nicht erkannt werden können (Abbildung 10.2(c)), ist die Klassifikation mit einem Vertrauen von 68,6% korrekt. Wird allerdings das für die Herz-Sieben charakteristische Herz in der Mitte darüber hinaus nicht detektiert, scheitert die Klassifikation. In diesem Fall ist die wahrscheinlichste Klasse die Herz-Vier mit 71,3%.

10.2 Poseschätzung von Gebäuden

Ziel dieser Fallstudie ist es, mit einem anwendungsunabhängigen Kontrollverfahren modellbasiert die Pose von Gebäuden zu bestimmen. Es wurde, wie in Kapitel 4 beschrieben, für die verwendeten Gebäude jeweils ein Gebäudemodell erstellt. Abbildung A.3, A.4, A.5, A.6 und A.7 in Anhang A zeigen die erstellten Modelle und die zur Evaluation verwendeten Datensätze.

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

Abbildung 10.3: In dieser Fallstudie wurden zur Evaluation fünf Gebäude verwendet. Davon sind drei Gebäude real und zwei sind Modellhäuschen im Maßstab H0, die auch unter realen Bedingungen aufgenommen wurden.

10.2.1 Experiment

In dieser Fallstudie wurden zur Evaluation fünf Gebäude modelliert und jeweils aus verschiedensten Ansichten Aufnahmen erstellt. Drei Gebäude sind real und bei zwei Gebäuden handelt es sich um Modellhäuschen im Maßstab H0 (siehe Abbildung 10.3). Die Modellhäuschen wurden vor künstlichem Himmel auf einer künstlichen Wiese mit einer künstlichen Straße unter Tageslichteinfall aufgenommen (siehe Abbildung 10.4).

(a) (b)

Abbildung 10.4: Aufbau für die Aufnahme der Modellhäuschen.

Die Aufnahmen der Datensätze wurden auf Bodenhöhe mit einer Spiegelreflexkamera[3] gemacht. Die Bilder haben eine Auflösung von 3872 x 2592 Pixeln und sind im JPEG-Format gespeichert. Die in der Spezifikation festgelegten Rahmenbedingungen wurden eingehalten (siehe Kapitel 2.3).

10.2.2 Ergebnisse

Zur Evaluation wird zunächst quantitativ geprüft, wie viele Posen richtig erkannt wurden. Dazu wird manuell jede gefundene Pose mit dem Ausgangsbild verglichen und entschieden, ob das aus der Pose gerenderte Bild dem Ursprungsbild nahezu entspricht. Tabelle 10.2 gibt einen Überblick über die Ergebnisse der quantitativen Evaluation.

Modellname	# Bilder	# richtiger Posen	Erkennungsrate
Haus von Faller	16	15	93,75%
Haus von Auhagen	18	18	100,00%
B-Gebäude	25	24	96,00%
Fertighaus von Massa	27	25	92,59%
Fertighaus von Streif	22	21	95,46%
Gesamt	108	103	95,37%

Tabelle 10.2: Ergebnisse der quantitativen Analyse.

Eine Erkennungsrate von 95,37% über alle Datensätze zeigt, dass die Fallstudie ihr Ziel gut erfüllt. Schlechtere Erkennungsraten gibt es beim Fertighaus von Massa.

[3]Nikon D60, AF-S DX NIKKOR 18-55mm f/3.5-5.6G VR

Der Grund dafür liegt an der geringen Anzahl Fenster. Je weniger Fenster das Gebäude besitzt, umso schlechter können nicht aufgefundene Fenster kompensiert werden. Erschwerend kommt hinzu, dass der Fenstererkenner bei diesem Gebäude auch schlechtere Erkennungsraten erzielt als bei den Häusern "B-Gebäude", "Haus von Faller" und "Haus von Auhagen" (siehe Tabelle 5.1 auf Seite 126).

Modellname	μ	σ	Min	Max
Haus von Faller	37,59	24,67	1,00	120,67
Haus von Auhagen	44,80	29,28	3,16	168,01
B-Gebäude	32,63	26,66	2,00	197,85
Fertighaus von Massa	40,76	24,59	1,41	156,21
Fertighaus von Streif	39,45	27,44	5,39	147,96
Gesamt	40,70	27,39	1,00	197,85

Tabelle 10.3: Rückprojektionsfehler in Pixeln ohne Bottom-Up-Hypothesengenerierung.

In einer qualitativen Evaluation mittels Rückprojektionsfehler wird die Qualität der Posen näher beleuchtet. Es wird zunächst die äußere Kontur der Bilder des Datensatzes mit einem selbst geschriebenen Annotationstool annotiert. Den Annotationen werden die Punkte aus dem Modell mit der gefundenen Pose automatisch zugeordnet. Die Posen, die als "falsch" in der quantitativen Evaluation bewertet wurden, werden in der qualitativen Evaluation nicht verwendet, da es in solchen Fällen nicht nur zu Fehlern in der Translation, sondern auch in der Rotation gekommen sein könnte. Dies hätte zur Folge, dass eine komplett unterschiedliche Fassadenseite zu sehen wäre und diese sich dem Bild nicht sinnvoll zuordnen lassen würde.

Modellname	μ	σ	Min	Max	# Posen	# Gesamt
Haus von Faller	26,84	16,89	1,00	86,02	10	15
Haus von Auhagen	29,03	24,04	2,00	128,69	14	18
B-Gebäude	29,28	18,95	2,00	102,40	6	24
Fertighaus von Massa	34,15	17,47	1,41	108,42	7	25
Fertighaus von Streif	39,45	27,44	5,39	147,96	1	21
Gesamt	31,44	21,66	1,00	147,96	38	103

Tabelle 10.4: Rückprojektionsfehler in Pixeln unter Einbeziehung der Bottom-Up-Hypothesengenerierung aus Kapitel 4.

Tabelle 10.3 präsentiert die Verteilung der Rückprojektionsfehler auf die einzelnen Modelle und auf alle Modelle insgesamt. Wir erzielen insgesamt einen durchschnittlichen Fehler von 40,7 Pixeln mit einer Standardabweichung von 27,39 Pi-

xeln. Am besten schneidet das B-Gebäude ab und am schlechtesten das Modell von Auhagen. Prinzipiell liegen aber alle Modelle in der gleichen Fehlerskala.

Modellname	μ	σ	Min	Max
Haus von Faller	97,84%	2,09%	93,63%	99,59%
Haus von Auhagen	97,96%	1,25%	94,48%	99,67%
B-Gebäude	88,16%	11,53%	62,52%	99,23%
Fertighaus von Massa	95,99%	2,59%	88,44%	99,36%
Fertighaus von Streif	96,07%	4,49%	81,42%	99,99%
Gesamt	94,79%	7,11%	62,52%	99,99%

Tabelle 10.5: Die Verteilung der Beliefs für die erkannten Posen.

Zur Analyse wurde die Bottom-Up-Hypothesengenerierung nicht angewendet, um ihren tatsächlichen Einfluss auf die Qualität des Verfahrens zu verifizieren. Tabelle 10.4 zeigt die Werte unter Verwendung der Bottom-Up-Hypothesengenerierung. In diesem Fall dienen die gefundenen Fenster dazu, mittels adaptierten RANSAC und Fiore wieder neue Posehypothesen zu generieren (siehe Kapitel 7.2.2). Die Gesamtbewertung zeigt einen kleineren Mittelwert (um 9 Pixel) und eine kleinere Standardabweichung (um 5,5 Pixel). Es wurde in 38 von 103 Fällen die Bottom-Up entstandene Posehypothese gewählt. Das heißt in 36,9% der Fälle konnte Bottom-Up die Pose verbessert werden und die Methode so ihren Nutzen unter Beweis stellen. Im Fall des Fertighauses von Streif wird nur einmal Bottom-Up eine Verbesserung erzielt und die Verbesserung dabei war marginal, was mit der mäßigen Leistung (Recall=56,03% und Precision=75,24%, siehe Tabelle 5.1 auf Seite 126) des Fenstererkenners für dieses Gebäude zu begründen ist. In keinem Fall konnte eine Pose nur Bottom-Up erkannt werden, sodass die in Tabelle 10.2 aufgelisteten Ergebnisse unverändert gelten.

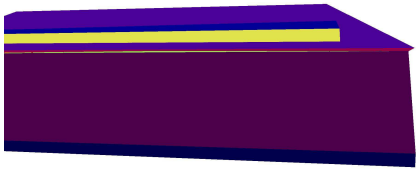
Modellname	μ	σ	Min	Max
Haus von Faller	37,59	24,67	1	120,67
Haus von Faller (manuell)	44,45	26,87	4	115,43

Tabelle 10.6: Vergleich des Rückprojektionsfehlers in Pixeln vom Haus von Faller mit generiertem und manuell erstelltem Haus.

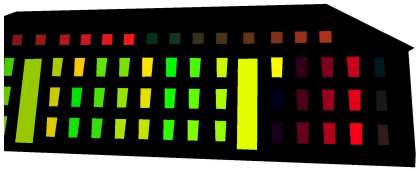
Die erkannten Posen besitzen im Durchschnitt ein Vertrauen von 94,79% mit einer Standardabweichung von 7,11%. Tabelle 10.5 listet die Beliefs für alle Modelle separat auf und zeigt ein hohes Vertrauen in die jeweiligen Analyseergebnisse der Modelle.

Da alle Modelle schon in einem halbautomatischen Prozess entstanden sind und diese wie in Kapitel 4 beschrieben einen durchschnittlichen Fehler von 11 cm

(a)



(b)



(c)

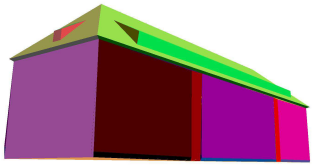
Abbildung 10.5: Beispiel, bei dem das Verfahren die richtige Pose nicht finden konnte. Abbildung (b) und (c) visualisieren die vermutete Pose mit projizierten Gebäudeflächen (b) und projizierten Fenstern (c).

aufweisen, muss untersucht werden, inwiefern diese die Poseschätzung beeinflussen. Dazu wurde das Modell des Hauses von Faller manuell ausgemessen und modelliert. Dieses Modell diente in Kapitel 4 zur Überprüfung des Verfahrens. Es wurde quantitativ das gleiche Ergebnis erzielt wie mit dem generierten Modell: 15 von 16 Posen wurden erkannt; das entspricht einer Erkennungsrate von 93,75%. Qualitativ zeigt sich allerdings eine Verschlechterung beim manuellen Modell von 7 Pixeln im Mittel und 2 in der Standardabweichung (siehe Tabelle 10.6). Dies ist erstaunlich und lässt sich nur durch numerische Probleme bei der Poseberechnung mit dem "perfekt" modellierten Haus erklären[4].

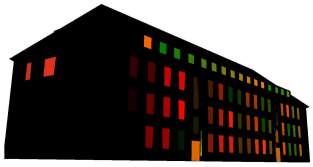
Abbildung 10.5 präsentiert ein Beispiel, bei dem das Verfahren die richtige Pose nicht finden konnte. Ähnlich zu den Beispielen der Dominostein- und Pokerkartenerkennung kommt es hier zu Verdeckungen. Diese Verdeckungen können nur

[4]Alternativ könnte die größere Abweichung auch in Ungenauigkeiten in der Vermessung des realen Modellhauses begründet sein. Die Vermessung wurde überprüft und ist im Millimeterbereich korrekt. Mögliche Abweichngen zwischen Messergebnis und Realität sind dementsprechend kleiner als 0.5 mm.

(a)



(b)



(c)

Abbildung 10.6: Beispiel, bei dem das Verfahren die richtige Pose finden konnte. Abbildung (b) und (c) visualisieren die vermutete Pose mit projizierten Gebäudeflächen (b) und projizierten Fenstern (c).

ausgeglichen werden, wenn der Fenstererkenner möglichst viele der restlichen Fenster erkennt. Da der eingesetzte Fenstererkenner nur wenige Fenster (diese aber präzise) erkennt, wird in diesem Beispiel die um 180 Grad gedrehte Pose als die richtige Pose angenommen. Dabei ist der Belief für die erkannte Pose 86,58%. Der Belief ist so hoch, da das Gebäude symmetrisch ist und das um 180 Grad gedrehte Gebäude sich in dieser Ansicht nur in der Position und Anzahl der Fenster bzw. Türen unterscheidet. Abbildung 10.6 zeigt ein Gebäude, bei dem die Erkennung trotz Verdeckungen gut funktioniert. Der Belief ist mit 75,46% eher niedrig, da große Teile des Gebäudes nicht zu sehen und somit auch nicht zuordenbar sind.

Kapitel 11

Schlussfolgerung und Ausblick

Dieses Kapitel fasst die Ergebnisse der Arbeit zusammen und stellt die wichtigsten Erkenntnisse heraus. Das Kapitel schließt mit einem Ausblick auf weitere Themen in diesem Bereich.

Das Einbinden von Wissen in die Bildanalyse ist der Leitgedanke der gewählten Strategie. Zum einen werden Modelle verwendet, die anwendungsspezifisches Wissen sowohl im deklarativen als auch prozeduralen Bereich besitzen. Zum anderen wird über die Bildanalyse selbst Wissen über die gesuchten Modelle aufgebaut und wieder dazu verwendet, den Analyseprozess zu steuern. Die Strategie ist dabei anwendungsunabhängig und ist durch vier Regeln definiert. Die bei der Analyse entstehenden Hypothesen werden im Hypothesenraum gespeichert. Die Analyse schränkt die Bildinterpretation so ein, dass aus einer Hypothese keine inkonsistente Hypothese erwachsen kann. Somit kann der Endzustand der Analyse direkt als Ergebnis des Verfahrens verwendet werden. Während des Ablaufs der Analyse entsteht eine Reihe von Zwischenergebnissen in Form von Hypothesen, die sich durchaus widersprechen können. Neben den Beiträgen auf Systemebene wurden auch Beiträge in Teilbereichen erzielt, auch wenn der Fokus der Arbeit auf dem Gesamtsystem liegt.

In Fallstudien zur Dominostein- und Pokerkartenerkennung wurde ein Ansatz beschrieben, der symbolische Beschreibungen verwendet, um 2-D-Objekte in perspektivischen Bildern zu erkennen. Die Suche im Hypothesenraum wird mit einem Graphmatchingalgorithmus, der Ungarischen Methode, kombiniert, um die Zahl der Hypothesen im Vergleich zu einem rein kombinatorischem Ansatz klein zu halten und so eine effiziente Analyse zu ermöglichen. Eine einfache Extraktion von geometrischen Formen, Farben und Regionen ermöglicht die Erstellung von Hypothesen für 2-D-Erkennungsprobleme. Dazu wurden Erkenner für Vierecke bzw. Rechtecke, Kreise, Kartensymbole und die Buchstaben A, J, K und Q implementiert. Die Bewertung der Modelle und ihrer Zuordnung durch das Dempster-Shafer-

Kalkül berücksichtigt Unsicherheiten und berechnet den Belief in das Auffinden eines Modells im Bild.

Dominostein- und Pokerkartenerkennung

Vorbereitend für **knoPoE** wurde die Erkennung von Dominosteinen und Pokerkarten mittels der auch in **knoPoE** genutzten anwendungsunabhängigen Kontrollstrategie präsentiert. Die wissensbasierte Objekterkennung mit symbolischen Beschreibungen zeigt sehr gute Ergebnisse bei perspektivischen Bildern, die **Erkennungsrate** beträgt **90,6%** und ist in der Lage mit Unsicherheiten und fehlenden Daten umzugehen.

Zur Generierung des Gebäudemodells mittels des Verfahrens **sage-sb** werden Bilder registriert, Punkte rekonstruiert und teil-interaktiv zu dreidimensionalen Modellen zusammengefügt, die semantisch annotiert werden. Im Falle der Häuser entstehen so Modelle, in denen beispielsweise Fenster und Fassaden und deren Beziehung zueinander modelliert werden. Dieses Verfahren besitzt eine ergonomische Benutzeroberfläche, die eine einfache Mensch-Maschine-Interaktion erlaubt. Die Modelle lassen sich in Standardformate konvertieren und aus solchen Formaten erzeugen. Es lassen sich aus Modellen und den Posehypothesen die entsprechenden Ansichten rendern, wobei für die erzeugten Projektionen der Bezug zur Modellinformation und die zugrundeliegende 3D-Geomentrie erhalten bleibt.

Verfahren sage-sb zur Modellgenerierung

Der Aufwand der Erstellung eines 3-D-Modells und die Interaktion mit dem System konnte auf ein Minimum reduziert werden. Der durchschnittliche **Punkt-zu-Punkt-Fehler liegt bei 1,3 mm** für die getesteten Modelle; das entspricht **11 cm in der Realität** bei einem Maßstab von 1:87. Je mehr Bilder und Punktkorrespondenzen verwendet werden, desto besser wird die Qualität der berechneten Punkte und Posen. Wird dabei das gesamte Gebäude modelliert, kommt es idealerweise zu einem "Schleifenschluss" und somit zu einer Steigerung der Qualität des Gebäudemodells durch das Optimierungsverfahren.

In einer komplexen 3-D-Anwendung muss zunächst das 3-D-Erkennungsproblem in ein 2-D-Erkennungsproblem überführt werden. Es konnte gezeigt werden, dass die Kombination von einfachen Extraktionsverfahren mit explizitem Modellwissen die Generation von Posehypothesen ermöglicht. Um dieses Vorhaben zu ermöglichen, wurden Komponenten konform zum STOR-Komponentenkonzept zur Merkmalsextraktion und der Interpretation dieser Merkmale als Basis für das **knoPoE**-Verfahren entwickelt.

Middle-Level-Merkmale für das Verfahren knoPoE

Ein Verfahren zur **Extraktion von Dachkanten** mit anschließender Berechnung einer überschaubaren Zahl an Posehypothesen wurde zu diesem Zweck entwickelt. Ein weiteres Verfahren extrahiert aus vorher gefundenen **Fensterregionen Fensterecken** und erzeugt mit diesen Eckpunkten weitere Posehypothesen.

Beim Verfahren **knoPoE** wird eine anwendungsunabhängigen Kontrollstrategie zur Extraktion der Pose bei Gebäudeaufnahmen genutzt. Als Wissensbasis des zweiten Verfahrens dienen die Gebäudemodelle, die mit dem Verfahren **sage-sb** generiert wurden. Es konnte gezeigt werden, dass das Verfahren sehr gut mit fehlenden und ungenauen Analysedaten umgehen kann und in diesen Fällen in der Lage ist, eine korrekte Pose zu bestimmen.

Modellbasierte Poseschätzung mit knoPoE

Es wurden **95,37% der Posen korrekt** erkannt, was für die verwendeten Datensätze ein akzeptables Ergebnis ist, da einfachste Erkenner nur mit Hilfe von Modellwissen in die Lage versetzt wurden, diese teilweise sehr komplexen Szenen richtig zu interpretieren. Dabei beträgt der **durchschnittliche Fehler 40,7 Pixel**, wenn ausschließlich **Top-Down** ausgewertet wurde. Dies konnte mit der Integration einer zusätzlichen **Bottom-Up-Analyse** in das Verfahren um **9,3 Pixel** auf einen durchschnittlichen Fehler von **31,4 Pixel verbessert** werden. Setzt man dies in Verhältnis zur Auflösung von 3872x2592 der Bilder und dem Fehler von 11 cm, der bei der Modellerstellung entsteht, erweist sich die erreichte Qualität als gut. Eine händische "perfekte" Modellierung konnte die Qualität des Verfahrens nicht verbessern.

Neben der Erzeugung von Hypothesen ist für die Steuerung des Verfahrens die Bewertung dieser Hypothesen erforderlich. Die Dempster-Shafer-Theorie kommt zum Einsatz, da sie nicht zur Annahme von "zweiseitigen" Wetten zwingt, mehrere Informationsquellen das gleiche Ereignis unterschiedlich beurteilen dürfen und Vertrauensmaße aus verschiedenen Quellen durch eine komfortable Kombinationsregel wieder zu einem Vertrauen zusammengefasst werden. Drei anwendungsunabhängige Vertrauensmaße führen zu einer konsistenten Berechnung des Beliefs in Objekte. Dennoch benötigt jedes Modell anwendungsabhängige Vertrauensmaße, die der Systemarchitekt modellieren muss. Dabei handelt es sich in der Regel nur um einige wenige Maße. Jedes extrahierte Bildelement weiß für sich, wie hoch der Belief darin ist, dass es korrekt im Bild gefunden wurde. Diese Information fließt zusätzlich in die Bewertung mit ein und reduziert den Einfluss eventuell falsch erkannter Objekte im Bild auf die Analyse. Die umgesetzten Bewertungen ermöglichen die

Bewertung der Hypothesen, die Bewertung des Endergebnisses der Analyse und den Vergleich von Hypothesen, die unterschiedlich weit in der Analyse fortgeschritten sind. Auf diese Weise kann mittels ϵ-A^*-Algorithmus die Analyse gesteuert und der Hypothesenraum während der Analyse aufgebaut werden.

Modellbasierte Poseschätzung mit knoPoE

Die berechneten Posen erlangen im Durchschnitt einen **Belief von 94,79%**.

Die vorgestellte Arbeit ist in sich geschlossen, dennoch bietet das Feld der modellbasierten Poseschätzung weitere Themen. Eine automatische Modellgenerierung ist beispielsweise wünschenswert, um die Akzeptanz des vorgestellten Verfahrens **sage-sb** zu erhöhen. Mittels Struktur aus Bewegungstechniken kann automatisch ein 3-D-Flächenmodell erzeugt werden. Spezialisierte Erkenner können dann die gesuchten semantischen Objekte automatisch extrahieren und über die berechneten Kameraposen ins Modell eintragen. Auf diese Weise kann der Aufbau von "Enthaltensein"-Relationen über die Fusion der Daten im 3-D-Raum automatisch erfolgen und über ein einheitliches Optimierungsverfahren die Genauigkeit des Modells weiter erhöht werden.

Die Qualität der gefundenen Posen steigt mit der Qualität der gefundenen Bestandteile. Dies liegt zum einen daran, dass Objekte sich dann leichter identifizieren lassen, und zum anderen ist die Qualität der Posehypothesen mit der Genauigkeit der dafür extrahierten Bildmerkmale korreliert. Die Erkennung über Boosting-Verfahren, Random-Forests oder mittels Deep-Learning ist zurzeit im Fokus der Forschung und es wurden vielversprechende Ergebnisse erzielt. Daher würde der Einsatz dieser Verfahren die Qualität weiter steigern. Verschiedenste dieser Bottom-Up-Ansätze könnten - parallel eingesetzt - neue Posen generieren. Durch diese Ansätze könnte das vorgestellte Verfahren weiterentwickelt werden und seine vielversprechenden Ergebnisse könnten die Relevanz für unterschiedlichste Anwendungsgebiete erhöhen.

Anhang A

Datensätze

Dieses Kapitel präsentiert alle Datensätze und die zu den entsprechenden Datensätzen erstellten Modelle. Die Bilder zur Erstellung der Modelle und die Datensätze zur Evaluation des Verfahrens besitzen keine Schnittmenge.

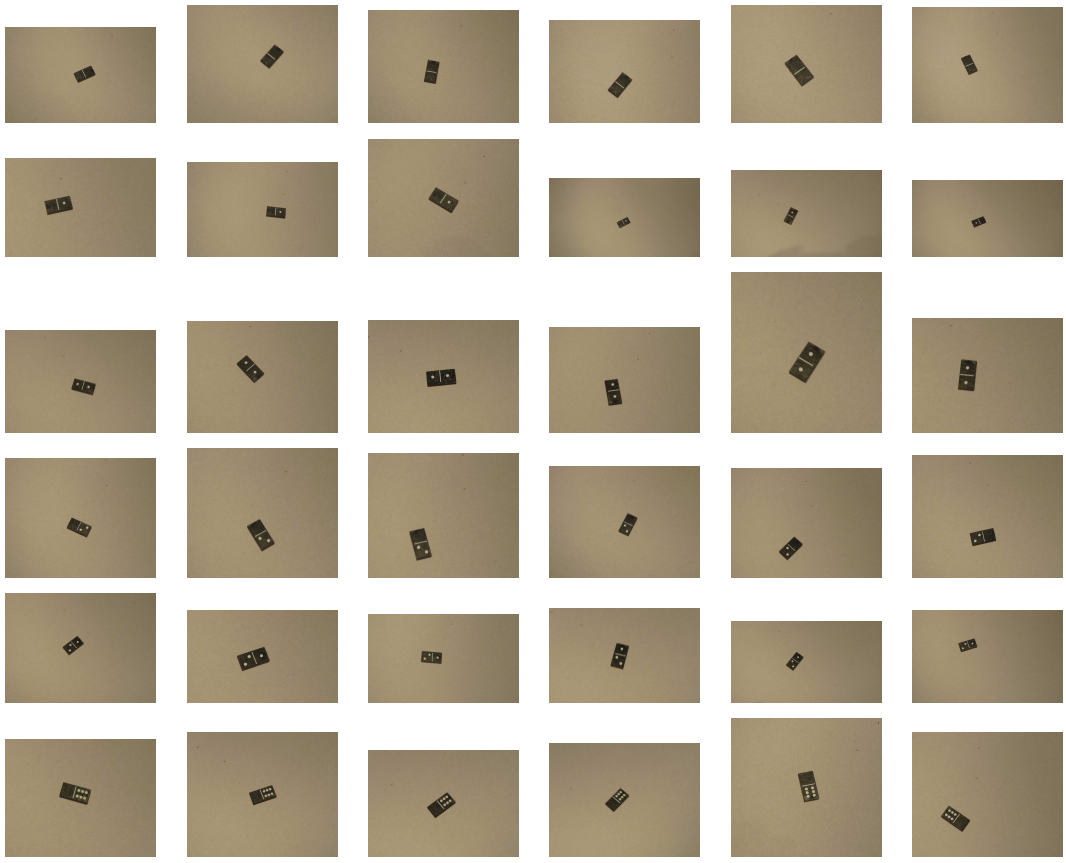


Abbildung A.1: Beispielbilder aus dem Datensatz der Dominosteinerkennung.

Abbildung A.1 zeigt einen Ausschnitt des Datensatzes, der zur Evaluation der Dominosteinerkennung verwendet wurde. Beispielbilder aus dem Datensatz der Pokerkartenerkennung werden in Abbildung A.2 präsentiert. Abbildung A.3 zeigt den kompletten Datensatz für das B-Gebäude des Universitätscampus Koblenz und das erzeugte Modell. Der komplette Datensatz für das Fertighaus der Firma Massa und für das der Firma Streif sowie das jeweils erzeugte Modell werden in Abbildung A.4 und A.5 präsentiert. Abbildung A.6 präsentiert den kompletten Datensatz für das Zweifamilienhaus von Faller (H0 131277), das erzeugte Modell und das händisch generierte Modell. Der komplette Datensatz für das Mehrfamilienhaus von Auhagen (H0 11 402) und das erzeugte Modell wird in Abbildung A.7 gezeigt.

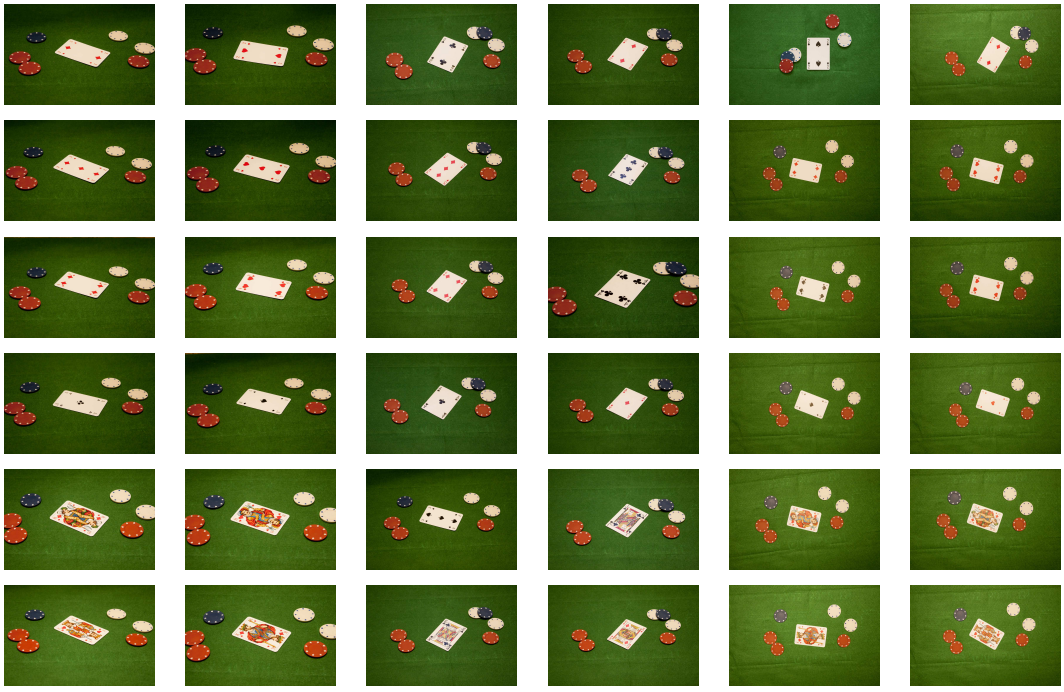


Abbildung A.2: Beispielbilder aus dem Datensatz der Pokerkartenerkennung.

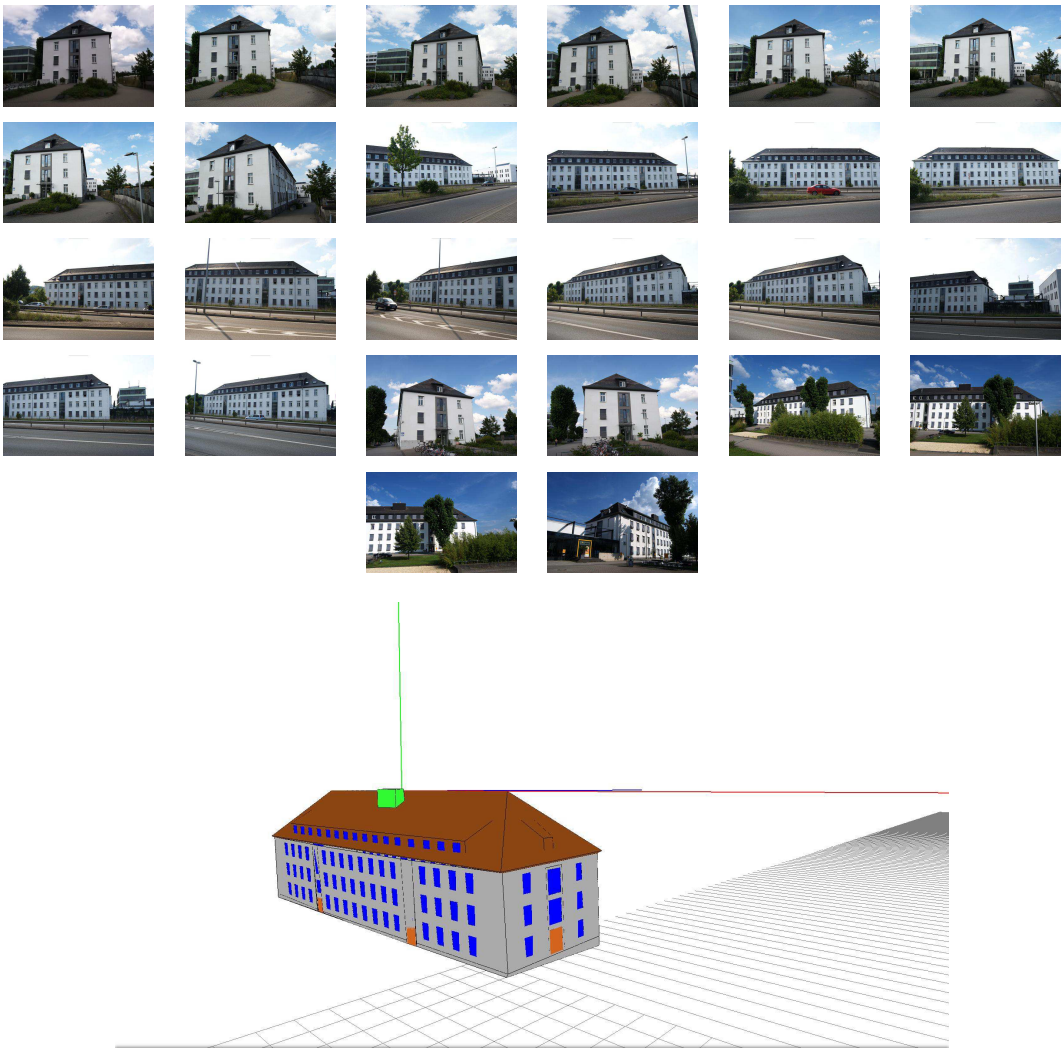


Abbildung A.3: Der komplette Datensatz für das B-Gebäude des Universitätscampus Koblenz und das erzeugte Modell.

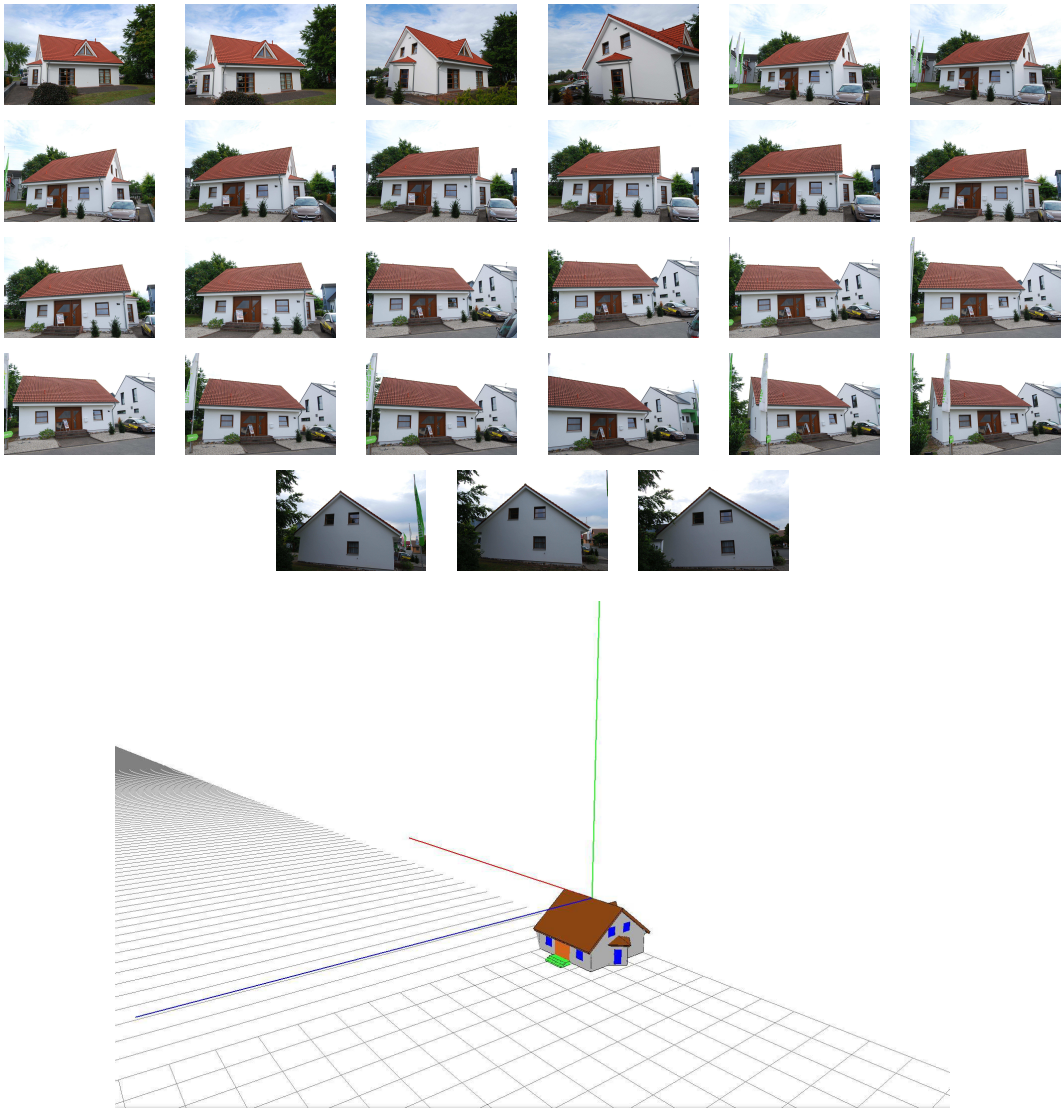


Abbildung A.4: Der komplette Datensatz für das Fertighaus der Firma Massa und das erzeugte Modell.

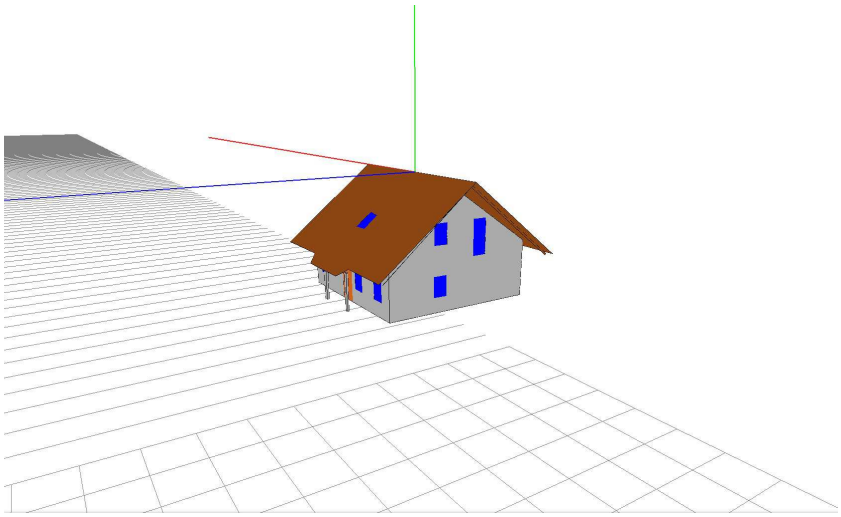
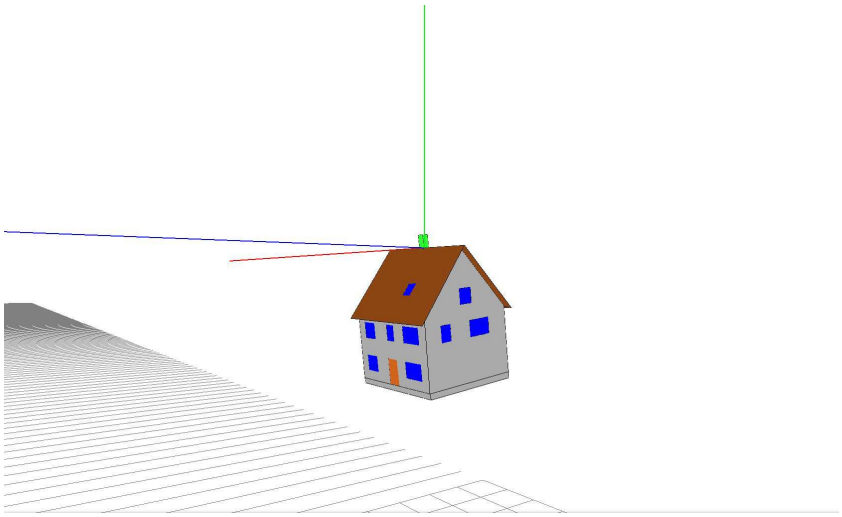
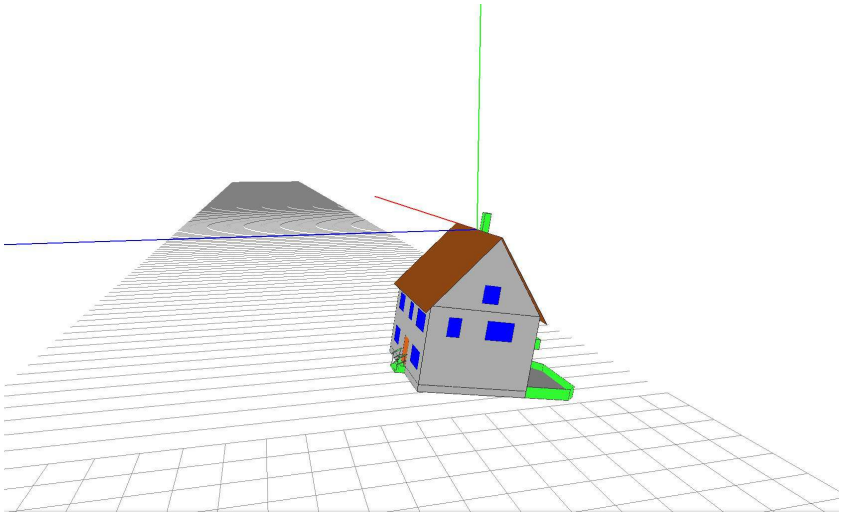


Abbildung A.5: Der komplette Datensatz für das Fertighaus der Firma Streif und das erzeugte Modell.



(a) mit sage-sb erstelltes Modell



(b) manuell erstelltes Modell

Abbildung A.6: Der komplette Datensatz für das Zweifamilienmodellhaus von Faller (H0 131277), das erzeugte Modell (a) und das händisch generierte Modell (b).

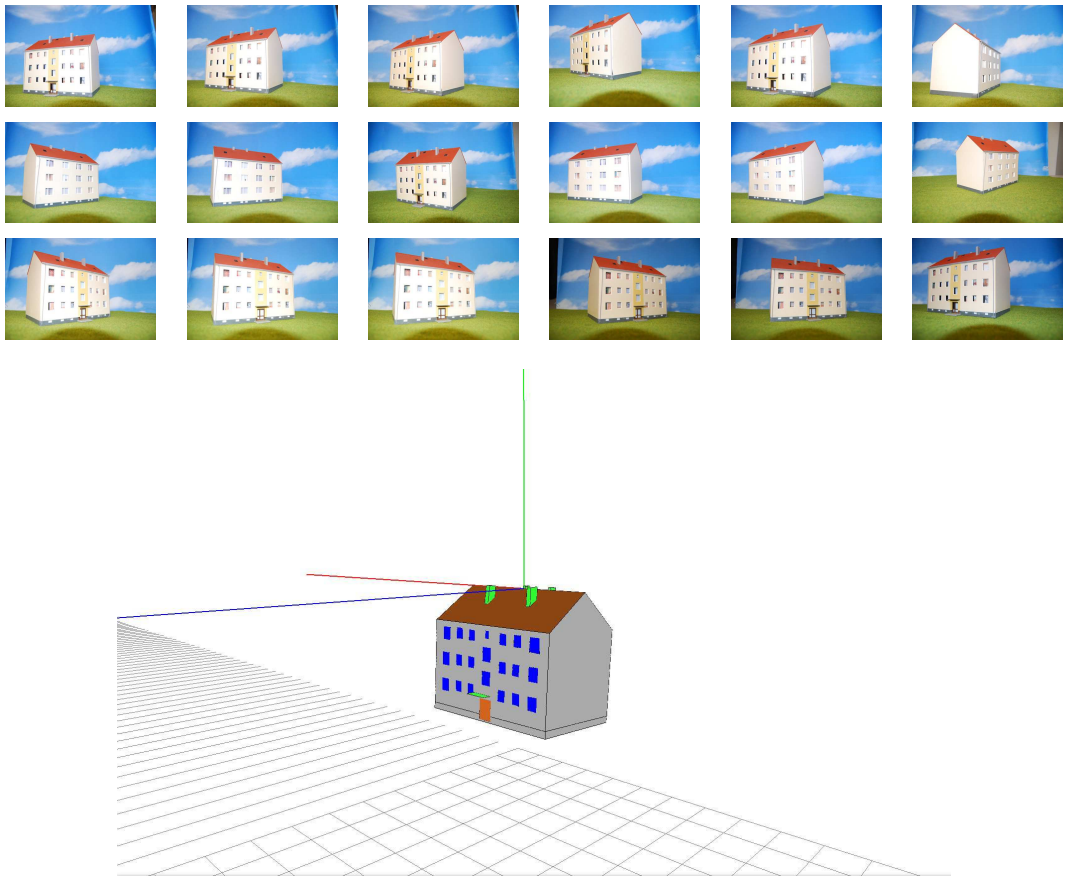


Abbildung A.7: Der komplette Datensatz für das Mehrfamilienmodellhaus von Auhagen (H0 11 402) und das erzeugte Modell.

Anhang B

Brennweiteschätzung

Das Verfahren von Guillou et al. [GMMB00] ist neben der in Kapitel 7.2.1 vorgestellten Berechnung der Rotation und Translation auch in der Lage, die Brennweite zu bestimmen. In dieser Arbeit wird allerdings das in Kapitel 3.1 beschriebene Verfahren zur Bestimmung der Brennweite verwendet. Im Folgenden wird das Verfahren von Guillou et al. näher beschrieben. Zusätzlich zu den Annahmen aus Kapitel 7.2.1 wird für die Berechnung der Brennweite ausgeschlossen, dass sich das Rechteck $(\boldsymbol{a}^{\mathrm{m}}, \boldsymbol{b}^{\mathrm{m}}, \boldsymbol{c}^{\mathrm{m}}, \boldsymbol{d}^{\mathrm{m}})$ parallel zur Bildebene befindet.

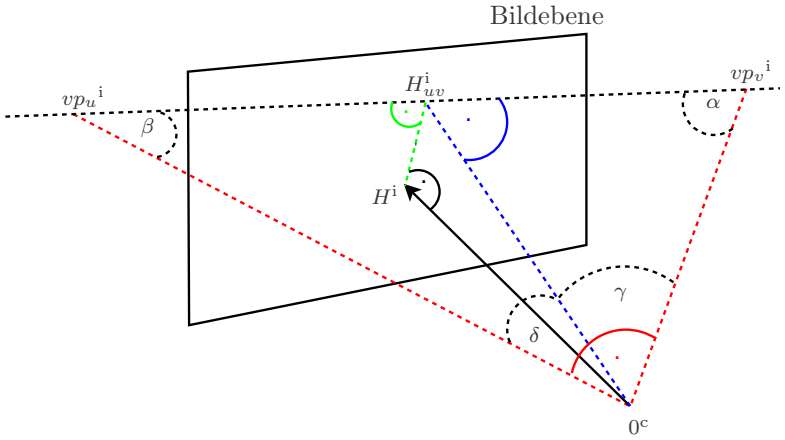


Abbildung B.1: Berechnung der Brennweite.

Zunächst müssen die Fluchtpunkte im Bildkoordinatensystem bestimmt werden. Die Fluchtgerade $\overline{\mathsf{vp_v{}^c vp_u{}^c}}$, die durch das Rechteck $(a^{\mathrm{c}}, b^{\mathrm{c}}, c^{\mathrm{c}}, d^{\mathrm{c}})$ definiert wird, ist außerdem die Schnittgerade zwischen der Fluchtebene $\Pi(0^{\mathrm{c}}, vp_v{}^{\mathrm{c}}, vp_u{}^{\mathrm{c}})$ und der Bildebene. Es gilt

$$f = \|H^{\mathrm{c}} - 0^{\mathrm{c}}\| = \sqrt{\overline{\mathsf{0^c H^c}}^2_{\mathrm{uv}} - \overline{\mathsf{H^i H^i}}^2_{\mathrm{uv}}}. \tag{B.1}$$

Da die Bildkoordinaten von H^{i}, $vp_u{}^{\mathrm{i}}$, $vp_v{}^{\mathrm{i}}$ und die Länge von $\overline{\mathsf{H^i H^i}}_{\mathrm{uv}}$ bekannt sind, muss nur die Länge von $\overline{\mathsf{0^c H^c}}_{\mathrm{uv}}$ bestimmt werden, um die Brennweite berechnen zu können. Es wird angenommen, dass die Dreiecke $\triangle_1 = (0^{\mathrm{c}}, vp_u{}^{\mathrm{c}}, vp_v{}^{\mathrm{c}})$, $\triangle_2 = (0^{\mathrm{c}}, vp_u{}^{\mathrm{c}}, H^{\mathrm{c}}_{uv})$ und $\triangle_3 = (0^{\mathrm{c}}, vp_v{}^{\mathrm{c}}, H^{\mathrm{c}}_{uv})$ rechtwinklig sind mit rechtem Winkeln bei 0^{c}, H^{c}_{uv} und H^{c}_{uv} (siehe Abbildung B.1). Daher gilt:

$$\begin{array}{rcl} \alpha + \gamma & = & \frac{\pi}{2} \\ \beta + \delta & = & \frac{\pi}{2} \\ \gamma + \delta & = & \frac{\pi}{2}. \end{array} \tag{B.2}$$

Aus Gleichung B.2 folgt:

$$\begin{array}{rcl} \alpha & = & \delta \\ \gamma & = & \beta. \end{array} \tag{B.3}$$

Da $\alpha = \delta$ und die Dreiecke $(0^{\mathrm{c}}, vp_u{}^{\mathrm{c}}, H^{\mathrm{c}}_{uv})$ und $(0^{\mathrm{c}}, vp_v{}^{\mathrm{c}}, H^{\mathrm{c}}_{uv})$ im Punkt H^{c}_{uv} rechtwinklig sind, folgt

$$\frac{\left\|\overline{\mathsf{0^c H^c}}_{\mathrm{uv}}\right\|}{\left\|\overline{\mathsf{H^c}_{\mathrm{uv}}\mathsf{vp_u}{}^{\mathrm{c}}}\right\|} = \frac{\left\|\overline{\mathsf{vp_u}{}^{\mathrm{c}}\mathsf{H^c}}_{\mathrm{uv}}\right\|}{\left\|\overline{\mathsf{0^c H^c}}_{\mathrm{uv}}\right\|}. \tag{B.4}$$

Woraus

$$\left\|\overline{\mathsf{0^c H^c}}_{\mathrm{uv}}\right\| = \sqrt{\left\|\overline{\mathsf{vp_v}{}^{\mathrm{c}}\mathsf{H^c}}_{\mathrm{uv}}\right\| \left\|\overline{\mathsf{H^c}_{\mathrm{uv}}\mathsf{vp_u}{}^{\mathrm{c}}}\right\|} \text{ folgt.} \tag{B.5}$$

Beweis auf Rechtwinkligkeit der Dreiecke $\triangle_1$, $\triangle_2$ und $\triangle_3$. Solange die Fluchtpunkte vp_u (bzw. vp_v) mit den Richtungsvektoren $\boldsymbol{u}$ (bzw. $\boldsymbol{v}$) assoziiert sind, gilt $\boldsymbol{u} * \boldsymbol{v} = 0$. Daraus folgt für die Vektoren $\boldsymbol{f_u} = vp_u{}^{\mathrm{c}} - 0^{\mathrm{c}}$ und $\boldsymbol{f_v} = vp_v{}^{\mathrm{c}} - 0^{\mathrm{c}}$, dass $\boldsymbol{f_u} * \boldsymbol{f_v} = 0$. Somit gilt $\boldsymbol{f_u} \perp \boldsymbol{f_v}$, was beweist, dass das Dreieck $(0^{\mathrm{c}} vp_u{}^{\mathrm{c}} vp_v{}^{\mathrm{c}})$ rechtwinklig mit rechten Winkel bei 0^{c} ist.

Vorgegeben durch die Konstruktion der Dreiecke ist für die Vektoren $\boldsymbol{h} = H^{\mathrm{c}} - 0^{\mathrm{c}}$, $\boldsymbol{h_{(uv)}} = H^{\mathrm{c}}_{uv} - 0^{\mathrm{c}}$, $\boldsymbol{a_{(HH_{uv})}} = H^{\mathrm{c}}_{uv} - H^{\mathrm{c}}$ und $\boldsymbol{a_{(vp_u vp_v)}} = vp_v{}^{\mathrm{c}} - vp_v{}^{\mathrm{c}}$:

$$\boldsymbol{h_{(uv)}}\ \boldsymbol{a_{(vp_u vp_v)}} = [\boldsymbol{h} + \boldsymbol{a_{(HH_{uv})}}]\, \boldsymbol{a_{(vp_u vp_v)}} = \boldsymbol{h}\ \boldsymbol{a_{(vp_u vp_v)}} + \boldsymbol{a_{(HH_{uv})}}\ \boldsymbol{a_{(vp_u vp_v)}} = 0.$$

Daher gilt: $\boldsymbol{h_{(uv)}} \perp \boldsymbol{a_{(vp_u vp_v)}}$ und die Geraden $\overline{\mathsf{0H}}_{\mathrm{uv}}$ und $\overline{\mathsf{vp_u vp_v}}$ sind rechtwinklig zueinander. Daraus folgt, dass die Dreiecke $(0^{\mathrm{c}} vp_u{}^{\mathrm{c}} H^{\mathrm{c}}_{uv})$ und $(0^{\mathrm{c}} vp_v{}^{\mathrm{c}} H^{\mathrm{c}}_{uv})$ rechtwinklig im Punkt H^{c}_{uv} sind. □

Anhang C

Effiziente lineare Berechnung der externen Orientierung

Der Ansatz von *Fiore* setzt voraus, dass die 3-D-Modellpunkte $\boldsymbol{p}_i^{\mathrm{m}}$ und die korrespondierenden Bildpunkte $\boldsymbol{p}_i^{\mathrm{i}}$ der Kamera bekannt sind. Gesucht werden dann der Translationsvektor $\boldsymbol{t}$, die Rotationsmatrix $\boldsymbol{R}$, die Skalierung λ und die projektiven Parameter $\boldsymbol{l}_i$, sodass sie folgende Gleichung optimal erfüllen:

$$\boldsymbol{l}_i \tilde{\boldsymbol{p}}_i^{\mathrm{i}} = \lambda \boldsymbol{R}(\boldsymbol{p}_i^{\mathrm{m}} + \boldsymbol{t}), \quad i = 1, \ldots, N. \tag{C.1}$$

Die Datenmatrix $\boldsymbol{D}_1 = [\tilde{\boldsymbol{p}}_1^{\mathrm{m}} \ldots \tilde{\boldsymbol{p}}_N^{\mathrm{m}}]$ wird definiert, um die Notation zu vereinfachen. Man kann zeigen, dass

$$\boldsymbol{l} = {\boldsymbol{D}_1}^{\mathrm{T}} \boldsymbol{\alpha} \tag{C.2}$$

für einen unbekannten 4×1 Vektor $\boldsymbol{\alpha}$ gilt. Es werden $N \geq 6$ Punkte benötigt, um einen eindeutigen minimalen Vektor $\boldsymbol{\alpha}$ zu finden. Wenn wir die linke Seite der Gleichung (C.1) mit $\boldsymbol{b_i} = \boldsymbol{l}\tilde{\boldsymbol{p}}^{\mathrm{i}}$ substituieren, wird Gleichung (C.1) zu

$$\boldsymbol{b_i} = \lambda \boldsymbol{R}(\boldsymbol{p}_i^{\mathrm{m}} + \boldsymbol{t}), \quad i = 1, \ldots, N. \tag{C.3}$$

Die Punkte $\widehat{\boldsymbol{b}}_i$ und $\widehat{\boldsymbol{p}^{\mathrm{m}}}_i$ wurden um deren Schwerpunkt verschoben, sodass die optimale kleinste quadratische Lösung für die Skalierung λ mit

$$\lambda = \frac{\sum_i^N \left\|\widehat{\boldsymbol{p}^{\mathrm{m}}}_i\right\| \left\|\widehat{\boldsymbol{b}}_i\right\|}{\sum_i^N \left\|\widehat{\boldsymbol{p}^{\mathrm{m}}}_i\right\|} \tag{C.4}$$

berechnet werden kann.

$\boldsymbol{B}$ ist die $3 \times N$ Matrix, in der die Punkte $\widehat{\boldsymbol{b}}_i$ nebeneinander eingefügt wurden, und $\boldsymbol{D_2}$ entspricht der Matrix, in der die skalierten Punkte $\widehat{\boldsymbol{p}^{\mathrm{m}}}_i$ ebenso eingefügt

wurden. Es gilt nun $\sum_{i=1}^{N} \left\| \widehat{\boldsymbol{b}}_i - \lambda \boldsymbol{R} \widehat{\boldsymbol{p}^{\mathrm{m}}}_i \right\|_2^2 = \|\boldsymbol{B} - \boldsymbol{R}\boldsymbol{D_2}\|$ zu minimieren, wobei die Lösung bekannt ist als

$$\boldsymbol{R} = \boldsymbol{V}_R {\boldsymbol{U}_R}^{\mathrm{T}}. \tag{C.5}$$

Die SVD der Matrix $\boldsymbol{D_2}\boldsymbol{B}^{\mathrm{T}}$ liefert mit $\boldsymbol{U}_R \boldsymbol{S}_R \boldsymbol{V}_R$ die entsprechenden linken und rechten Singulärvektoren $\boldsymbol{U}_R$ und $\boldsymbol{V}_R$.

Die Gleichungen (C.2-C.5) bieten die Möglichkeit die Kamerapose zu berechnen.

Anhang D

Ungarische Methode

Die folgende Beschreibung des Lösungsansatzes des gewichtsmaximalen bipartiten Zuordnungsproblems entstammt der im Rahmen dieser Arbeit betreuten Abschlussarbeit von Häselich [Häs10]. Sie wurde aus den Kapiteln 2 und 4 der Abschlussarbeit zusammengefasst und entsprechend der mathematischen Notation dieser Arbeit angepasst.

Das Problem der Bestimmung einer Zuordnung mit maximaler Bewertung bezüglich einer Abstandsfunktion kann direkt auf das gewichtsmaximale bipartite Zuordnungsproblem zurückgeführt werden (siehe [Win94] Kapitel 5.6.1). Zur Lösung des 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. Die Implementierung erfolgte auf Grundlage von [Jun90]. Eine Validierung und die Bestimmung der Komplexität $O(n^3)$ sind in [Jun90] Kapitel 11 zu finden.

Jeder *bipartite Graph* lässt sich durch eine Matrix repräsentieren.

Definition D.1 (Bipartiter Graph)**.** *Ein Graph $G = (V, E)$ ist bipartit, wenn die Knotenmenge V in zwei Mengen A und B partitioniert werden kann, so dass alle Kanten einen Knoten aus A mit einem Knoten aus B verbinden.*

Definition D.2 (vollständig bipartiter Graph)**.** *Ein vollständiger bipartiter Graph besteht aus zwei Mengen A und B mit m bzw. n Knoten, wobei jeder Knoten aus A mit jedem Knoten aus B durch eine Kante verbunden ist.*

Die Werte dieser Zuordnungsmatrix werden dabei von der Kosten- bzw. Straffunktion berechnet. Die Aufgabe der Zuordnung, und damit der Ungarischen Methode besteht demnach darin, für eine gegebene Zuordnungsmatrix das Ergebnis in Form einer *Adjazenzmatrix* zu bestimmen.

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

Abbildung D.1: Darstellung einer Zuordnungsmatrix mit ihrer korrespondierenden Adjazenzmatrix.

Definition D.3 (Adjazenzmatrix). *Jeder bipartite Graph G lässt sich durch eine Adjazenzmatrix (auch Nachbarschaftsmatrix genannt) darstellen. Dabei werden die Knoten V 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 D.2 (rechte Seite) zu sehen.*

Abbildung D.1 zeigt ein Beispiel für eine beliebige Zuordnungsmatrix auf der linken Seite und ihrer korrespondierenden Adjazenzmatrix auf der rechten Seite.

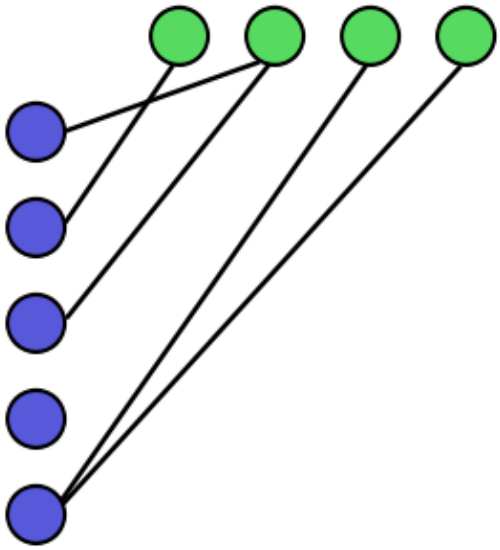
Aus der Adjazenzmatrix lässt sich die Bewertung des Modell für das Segmentierungsobjekt einfach errechnen, indem alle in der Adjazenzmatrix mit 1 belegten Positionen den Werten der Zuordnungsmatrix zugeordnet, aufaddiert und durch die Dimension der Matrix dividiert werden.

Im Folgenden werden Begrifflichkeiten zum Verständnis der Ungarischen Methode geklärt.

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

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



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

Abbildung D.2: 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 Graphen korrespondierende Adjazenzmatrix. Der Menge n_1 wurden dabei die Zeilen zugeordnet, der Menge n_2 die Spalten.

Pfad In einem Graphen $G = (V, E)$ mit zwei Knoten $a, b \in V$ 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 $0 \leq i < k$ (siehe [Lux04] Kapitel 2).

Alternierender Pfad Für einen Graphen $G = (V, E)$ 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 $0 \leq i \leq k-2$ (siehe [Lux04] Kapitel 2).

Augmentierender Pfad Für einen Graphen $G = (V, E)$ ist ein alternierender Pfad p ein augmentierender Pfad bezüglich einer Zuordnung M, falls die Endknoten von p ungepaart sind (siehe [Lux04] Kapitel 2).

Gewichtsmaximales bipartites Zuordnungsproblem Sei $G = (V, E)$ ein bipartiter Graph mit einer Gewichtsfunktion $w : E \to \mathbb{R}$. Als *gewichtmaximales 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. 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 Zuordnungsmetrik zugrunde liegt (siehe [BB82] Kapitel 11 Abschnitt 3.1).

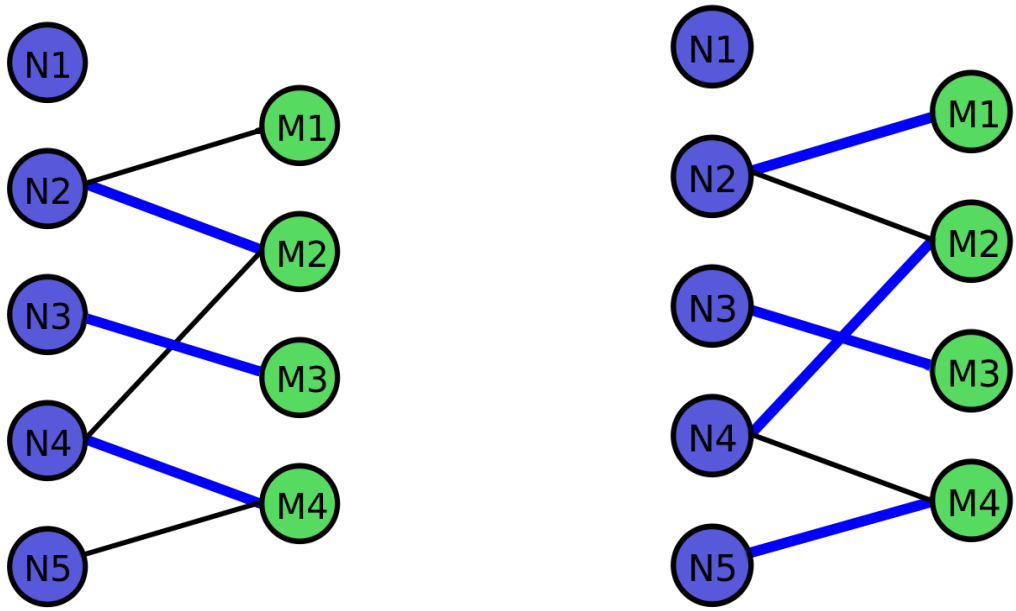
Augmentierende Pfade sind der Dreh- und Angelpunkt der Ungarischen Methode. Zur Erklärung der Augmentierung (Vergrößerung) fehlen jedoch noch zwei weitere Definitionen, die für das Konzept des Algorithmus benötigt werden:

Kennzeichnung Sei $G = (V, E)$ ein bipartiter Graph mit $V = v_1 \cup v_2 \wedge v_1 \cap v_2 = \varnothing$ 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: $\forall a \in v_1 \, \forall b \in v_2 : l(a) + l(b) \geq w(e(a, b))$.

Gleichheitsgraph Sei $G = (V, E)$ ein bipartiter Graph mit einer Gewichtsfunktion $w : E \to \mathbb{R}$ und $G_t = (V, E_t)$ ein Teilgraph von G. Daher ist 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 D.3 kann die Erweiterung eines augmentierenden Pfades nachvollzogen werden. Belegte Kanten sind in der Abbildung in blau eingezeichnet, freie Kanten in schwarz. Die zweite Phase des Algorithmus kommt 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 $G = (V, E)$ ein bipartiter Graph mit $V = v_1 \cup v_2 \wedge v_1 \cap v_2 = \varnothing$, dann ist

$$\Delta = \min_{a \in v_1, b \in v_2} l(a) + l(b) - w(e(a, b))$$



(a) (b)

Abbildung D.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 (a) vergrößert sich die Anzahl der belegten Kanten um eins (b).

Für die Kennzahl $l(x) = l'(x)$ eines Knotens $x \in V$ gilt

$$l'(x) = \begin{cases} l(x) + \Delta, & x \in v_1 \\ l(x) - \Delta, & x \in v_2 \\ l(x), & sonst \end{cases}$$

Da sich die Kennzahl in beiden Knoten der Teilgraphen identisch vergrößert bzw. verkleinert, bleibt die Kennzeichnung zulässig, weil nach wie vor die Bedingung $\forall a \in v_1 \, \forall b \in v_2 : l(a) + l(b) \geq w(e(a, b))$ 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.

Anhang E

Evaluation von etablierten Distanzfunktionen für Strecken

Als Vorbedingung für die Evaluation wird angenommen, dass zwei Streckensätze vorhanden sind. Die Entstehung dieser Streckensätze, sei es beispielsweise durch den Canny Kantenerkenner oder durch ein 3-D CAD-Modell, ist nicht relevant für die Arbeit, da die Distanzfunktionen unabhängig vom Extraktionsmechanismus evaluiert werden sollen.

Diese Studie arbeitet mit Pixelkoordinaten und eine Strecke l ist definiert als $\mathrm{l}_i = p_{1,\mathrm{l}_i} \times p_{2,\mathrm{l}_i}$. Ein geometrisches Objekt GEO ist die Verallgemeinerung von einer Menge von Strecken LS. Eine Distanzfunktion für Strecken wird beschrieben als $\mathrm{d}^{(\mathrm{l})} : \mathrm{l} \times \mathrm{l} \mapsto \mathbb{R}^+$ und misst die Unähnlichkeit zwischen Strecken, wobei ein Wert von null die Identität zweier Strecken bedeutet.

Distanzfunktionen

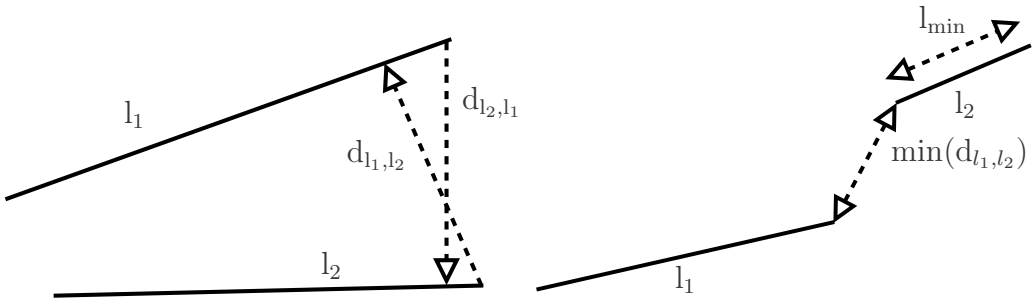
In der Evaluation werden die Funktionen (i) Hausdorff-Distanz, (ii) Trucco-Distanz, (iii) Modifizierte-Strecken-Hausdorff-Distanz, (iv) Modifizierte-Lot-Hausdorff-Distanz, (v) Mittelpunkt-Distanz, (vi) Nächster-Punkt-Distanz und (vii) die selbst entwickelte Strecken-Distanz betrachtet. Diese messen alle die Distanz zwischen zwei Strecken l_1 und l_2.

Die *(i) Hausdorff-Distanz* [ABB95] wird in Abbildung E.1(a) illustriert und wird wie folgt beschrieben:

$$\mathrm{d}_{\mathrm{l}_1,\mathrm{l}_2}(\mathrm{l}_1,\mathrm{l}_2) = \max_{\boldsymbol{p}\in\mathrm{l}_1} \min_{\boldsymbol{q}\in\mathrm{l}_2} \|\boldsymbol{p}-\boldsymbol{q}\| \tag{E.1}$$

$$\mathrm{d}_{\mathrm{l}_2,\mathrm{l}_1}(\mathrm{l}_1,\mathrm{l}_2) = \max_{\boldsymbol{p}\in\mathrm{l}_2} \min_{\boldsymbol{q}\in\mathrm{l}_1} \|\boldsymbol{p}-\boldsymbol{q}\| \tag{E.2}$$

$$\mathrm{d}_{\mathrm{Hausdorff}}(\mathrm{l}_1,\mathrm{l}_2) = \max(\mathrm{d}_{\mathrm{l}_1,\mathrm{l}_2}, \mathrm{d}_{\mathrm{l}_2,\mathrm{l}_1}). \tag{E.3}$$



(a) Hausdorff-Distanz (b) Trucco-Distanz

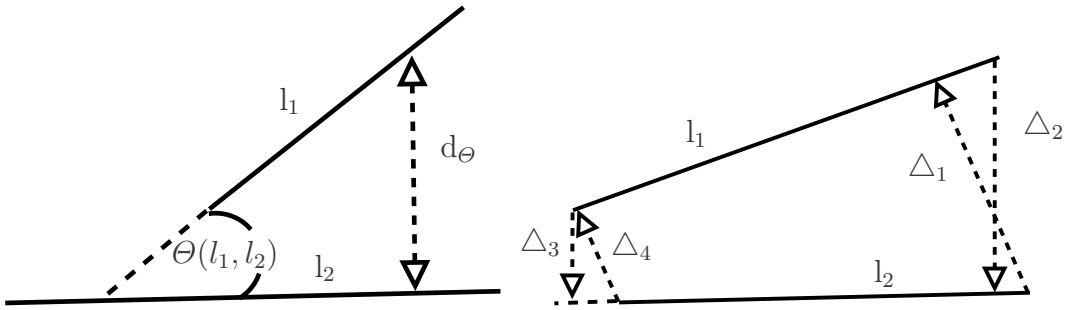
Abbildung E.1: Visualisierung der Hausdorff- und Trucco-Distanz.

Die *(ii) Trucco-Distanz* [TV98] wird in Abbildung E.1(b) illustriert und ist definiert als:

$$\mathrm{l}_{\min} = \min(\|l_1\|, \|l_2\|) \tag{E.4}$$

$$\min(\mathrm{d}_{l_1,l_2}) = \min(\|p_{1,l_1} - p_{1,l_2}\|\|p_{1,l_1} - p_{2,l_2}\|, \|p_{2,l_1} - p_{1,l_2}\|, \|p_{2,l_1} - p_{2,l_2}\|) \tag{E.5}$$

$$\mathrm{d}_{\mathrm{Trucco}}(l_1, l_2) = \left(\frac{\mathrm{l}_{\min}}{\min(\mathrm{d}_{l_1,l_2})}\right)^2. \tag{E.6}$$



(a) Modifizierte-Strecken-Hausdorff-Distanz (b) Modifizierte-Lot-Hausdorff-Distanz

Abbildung E.2: Visualisierung der Modifizierten-Strecken Hausdorff-Distanz und Modifizierten-Lot-Hausdorff-Distanz mit $\triangle_1 = \max(\mathrm{d}_{\perp,\mathrm{l}_1,1}, \mathrm{d}_{\perp,\mathrm{l}_1,2})$, $\triangle_2 = \max(\mathrm{d}_{\perp,\mathrm{l}_2,1}, \mathrm{d}_{\perp,\mathrm{l}_2,2})$ $\triangle_3 = \min(\mathrm{d}_{\perp,\mathrm{l}_1,1}, \mathrm{d}_{\perp,\mathrm{l}_1,2})$ und $\triangle_4 = \min(\mathrm{d}_{\perp,\mathrm{l}_2,1}, \mathrm{d}_{\perp,\mathrm{l}_2,2})$.

Die *(iii) Modifizierte-Strecken-Hausdorff-Distanz* [CLG03] wird in Abbildung E.2(a) illustriert, nutzt die Winkelfunktion $\Theta(l_1, l_2) : \mathbb{R}^2 \times \mathbb{R}^2 \mapsto \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$ und integriert diese in

$$\mathrm{d}_{\Theta}(l_1, l_2) = \min(\|l_1\|, \|l_2\|)\ \sin(\Theta(l_1, l_2)). \tag{E.7}$$

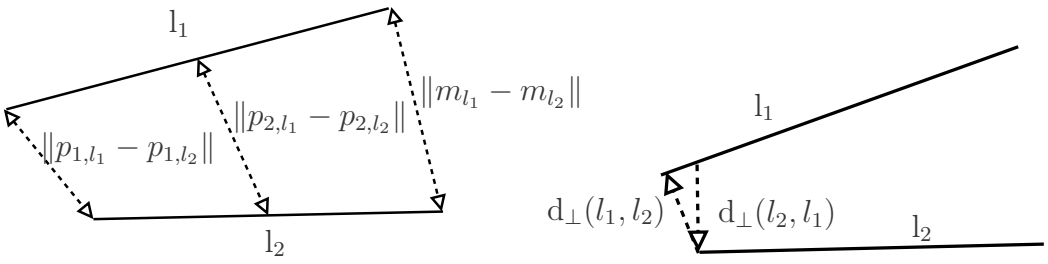
Die *(iv) Modifizierte-Lot-Hausdorff-Distanz* [SL04] wird in Abbildung E.2(b) illustriert, nutzt die Lotfunktion $\mathrm{d}_\perp(l_1, l_2) = s_\perp$ und integriert diese in

$$s_{\perp,1} = \min(\max(\mathrm{d}_{\perp,\mathrm{l}_1,1}, \mathrm{d}_{\perp,\mathrm{l}_1,2}), \max(\mathrm{d}_{\perp,\mathrm{l}_2,1}, \mathrm{d}_{\perp,\mathrm{l}_2,2})) \tag{E.8}$$

$$s_{\perp,2} = \min(\min(\mathrm{d}_{\perp,\mathrm{l}_1,1}, \mathrm{d}_{\perp,\mathrm{l}_1,2}), \min(\mathrm{d}_{\perp,\mathrm{l}_2,1}, \mathrm{d}_{\perp,\mathrm{l}_2,2})) \tag{E.9}$$

$$w_i = \frac{s_{\perp,i}}{(s_{\perp,1} + s_{\perp,2})} \text{ with } i = \{1, 2\} \tag{E.10}$$

$$\mathrm{d}_{\perp\mathrm{mod}}(l_1, l_2) = \frac{1}{2}(w_1\ s_{\perp,1} + w_2\ s_{\perp,2}). \tag{E.11}$$



(a) Mittelpunkt-Distanz (b) Nächster-Punkt-Distanz

Abbildung E.3: Visualisierung der Mittelpunkt- und Nächster-Punkt-Distanz.

Die *(v) Mittelpunkt-Distanz* wird in Abbildung E.3(a) illustriert. Diese nutzt zusätzlich den Mittelpunkt der Strecken $m_{l_i} = (p_{2,l_i} - p_{1,l_i})/2$ und integriert dies in

$$\mathrm{d}_{\mathrm{midpoint}}(l_1, l_2) = \|p_{1,l_1} - p_{1,l_2}\| + \|p_{2,l_1} - p_{2,l_2}\| + 3\|m_{l_1} - m_{l_2}\|. \tag{E.12}$$

Die *(vi) Nächster-Punkt-Distanz* [WZ02] wird in Abbildung E.3(b) illustriert und beschrieben als

$$\mathrm{d}_{\mathrm{closestpoint}}(l_1, l_2) = min(\mathrm{d}_\perp(l_1, l_2), \mathrm{d}_\perp(l_2, l_1)). \tag{E.13}$$

Die eigene *(vii) Strecken-Distanz* nutzt $\mathrm{d}_1 = \|p_{1,l_1} - p_{1,l_2}\|$, $\mathrm{d}_2 = \|p_{1,l_1} - p_{2,l_2}\|$, $\mathrm{d}_3 = \|p_{2,l_1} - p_{1,l_2}\|$ und $\mathrm{d}_4 = \|p_{2,l_1} - p_{2,l_2}\|$; und fasst diese Funktionen in einer zusammen:

$$\mathrm{d}_t(l_1, l_2) = \frac{\mathrm{d}_1 + \mathrm{d}_2 + \mathrm{d}_3 + \mathrm{d}_4}{4} - \frac{\|l_1\| + \|l_2\|}{4} \tag{E.14}$$

$$\mathrm{d}_{\mathrm{ST}}(l_1, l_2) = \mathrm{d}_{\mathrm{closestpoint}} + 0.25\ \mathrm{d}_\Theta(l_1, l_2) + \mathrm{d}_t \tag{E.15}$$

$$\mathrm{d}_{straightLine}(l_1, l_2) = min(\mathrm{d}_{\mathrm{ST}}(l_1, l_2), \mathrm{d}_{\mathrm{ST}}(l_2, l_1)). \tag{E.16}$$

Evaluation

Die Distanzfunktionen sollen auf Basis der Parameter einer Strecke (Position, Länge und Orientierung) Unterschiede zwischen Strecken quantifizieren.

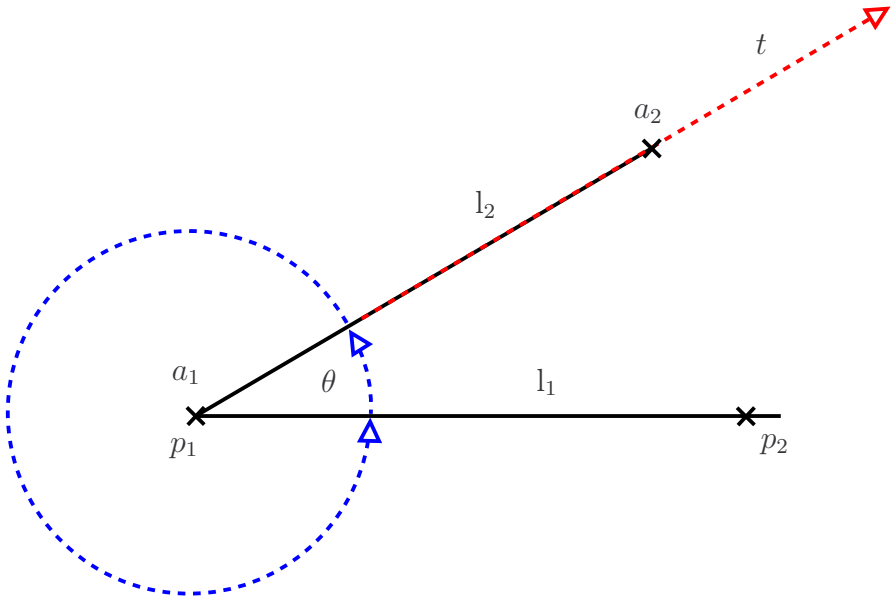


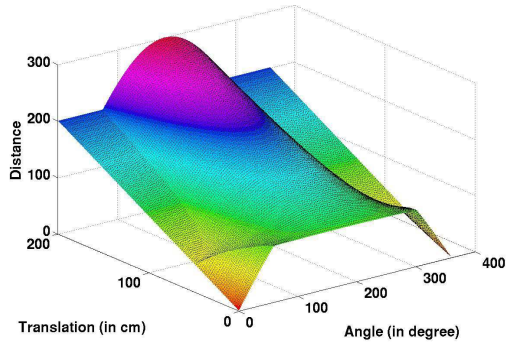
Abbildung E.4: Beschreibung des Evaluationsprozesses.

In welcher Metrik die Strecken vorliegen beeinflusst die Distanzfunktionen nicht, da diese unabhängig davon agieren. Für diese Arbeit werden Zentimeter als Metrik gewählt. Es könnten genauso Pixel als Metrik gewählt werden, da aber in Bildern verschiedenste Auflösungen, angefangen bei VGA *(=640x480)* bis zu Auflösungen mit mehreren tausend Pixeln, möglich sind, fiel die Wahl auf Zentimeter.

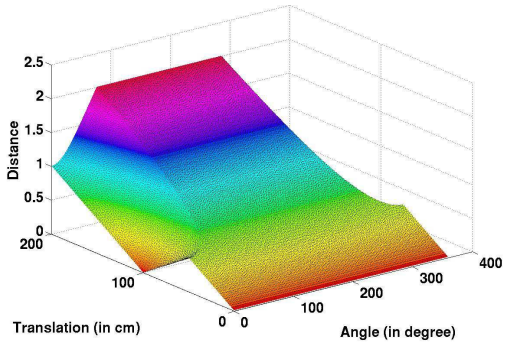
Es wurden zwei Strecken l_1 und l_2 mit Länge 100 cm generiert. Die Strecke l_2 wird um die Strecke l_1 rotiert, in Richtung von l_2 verschoben und zudem skaliert, wobei der Punkt p_1 gleich bleibt. Es wird rotiert im Bereich von $\theta = [0,\ 2\pi]$, verschoben von $t = [0\text{cm},\ 200\text{cm}]$ und skaliert von $[1\text{cm},\ 200\text{cm}]$ (siehe Abbildung E.4). In Abschnitt E werden die Ergebnisse beschrieben und diskutiert.

Ergebnisse

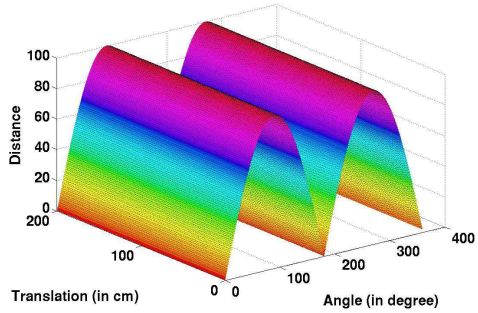
Das Ergebnis der Evaluation wird in Abbildung E.5(a) bis E.5(g) gezeigt. Abbildung E.5 macht die Bedeutung der Verschiebung und Rotation für die Distanzfunktionen klar. Die Trucco- und die Nächste-Punkt-Distanz (Abbildung E.5(b) und E.5(f)) sind ungeeignet, um Unterschiede in der Orientierung zu quantifizieren. Zudem ist die Trucco-Distanz nicht monoton steigend. Monotonie ist wichtig,



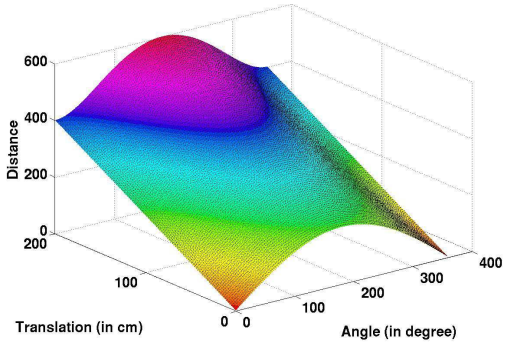
(a) Hausdorff-Distanz



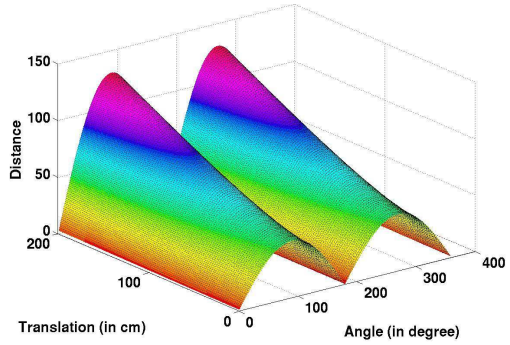
(b) Trucco-Distanz



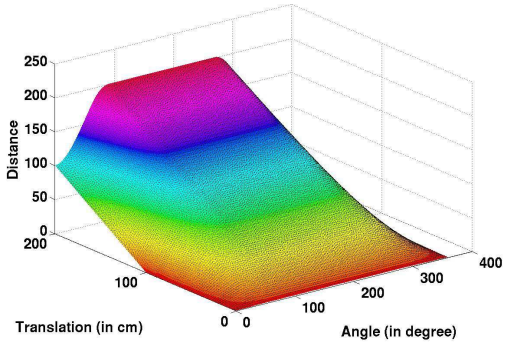
(c) Modifizierte-Strecken-Hausdorff-Distanz



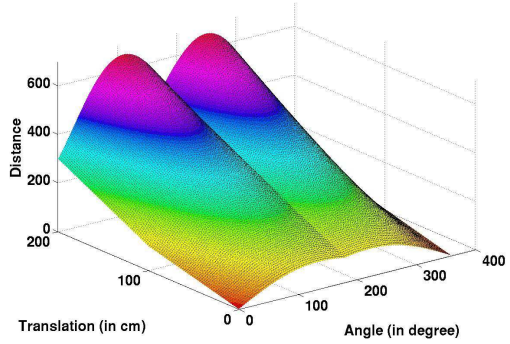
(d) Mittelpunkt-Distanz



(e) Modifizierte-Lot-Hausdorff-Distanz

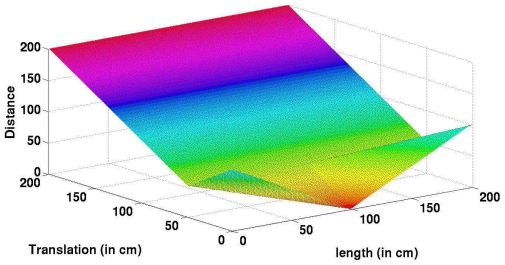


(f) Nächster-Punkt-Distanz

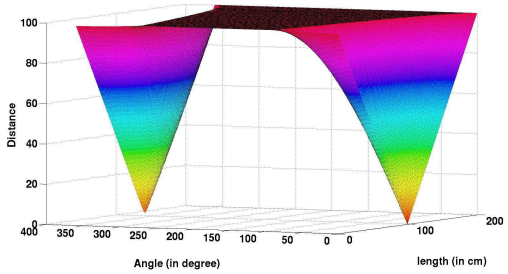


(g) Strecken-Distanz

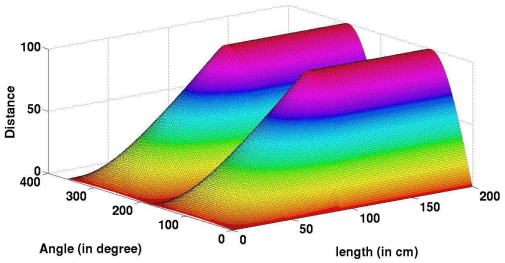
Abbildung E.5: Distanzfunktion mit Winkel in Grad auf der x-Achse, der Translation in Zentimeter auf der y-Achse und der Distanz auf der z-Achse.



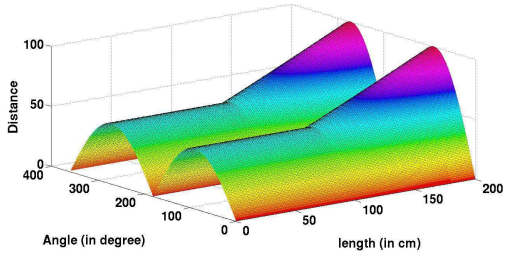
(a) Hausdorff-Distanz - Translation zu Länge



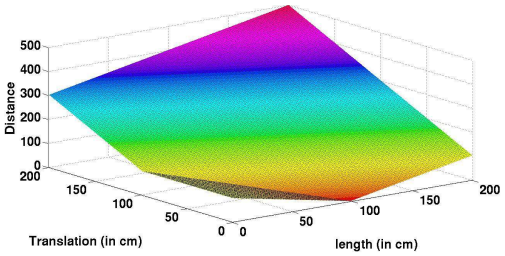
(b) Hausdorff-Distanz - Winkel zu Länge



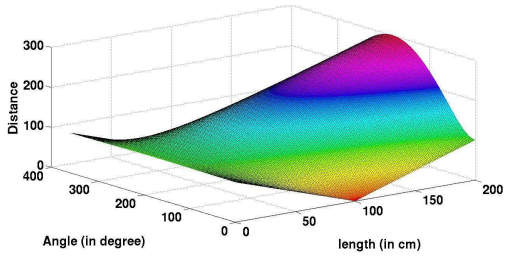
(c) Modifizierte-Strecken-Hausdorff-Distanz - Winkel zu Länge



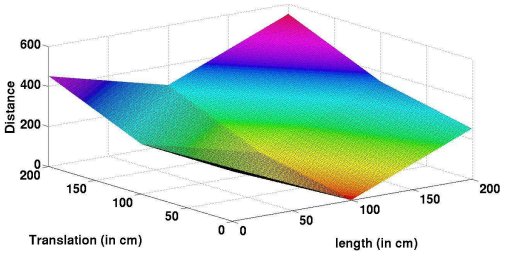
(d) Modifizierte-Lot-Hausdorff-Distanz - Winkel zu Länge



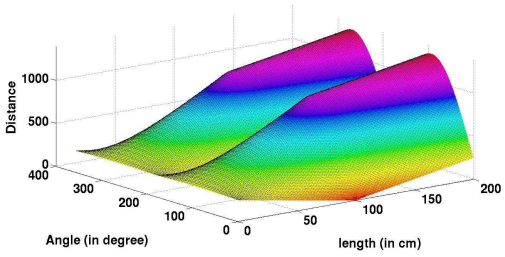
(e) Mittelpunkt-Distanz - Translation zu Länge



(f) Mittelpunkt-Distanz - Winkel zu Länge



(g) Strecken-Distanz - Translation-zu Länge



(h) Strecken-Distanz - Winkel zu Länge

Abbildung E.6: Distanzfunktionen, die von der Streckenlänge beeinflusst werden, werden mit der Länge auf der x-Achse, dem Winkel in Grad oder der Verschiebung in Zentimeter auf der y-Achse und der Distanz auf der z-Achse beschrieben. Der jeweils nicht verwendete Wert, der Verschiebung oder der Winkel, werden auf null gesetzt.

	HD	TD	MHD	MPHD	MD	CD	SD
Winkel	$\oplus$	$\ominus$	$\oplus$	$\oplus$	$\oplus$	$\ominus$	$\oplus$
Verschiebung	$\oplus$	$\oplus$	$\ominus$	$\oplus$	$\oplus$	$\oplus$	$\oplus$
Skalierung	$\odot$	$\ominus$	$\odot$	$\odot$	$\oplus$	$\ominus$	$\oplus$

Tabelle E.1: Es werden die Abhängigkeiten bzgl. Orientierung, Position und Skalierung aufgelistet. Das Zeichen $\oplus$ steht für Abhängigkeit, $\odot$ steht für Abhängigkeit mit Abstrichen und $\ominus$ steht für Unabhängigkeit von dieser Größe.

falls die Werte der Distanzfunktionen als Maß für die Unähnlichkeit verwendet werden sollen.

Die Modifizierte-Strecken-Hausdorff-Distanz- und die Modifizierte-Lot-Hausdorff-Distanz (Abbildung E.5(c) und E.5(e)) unterscheiden sehr gut Winkelunterschiede, können aber benachbarte Strecken nicht unterscheiden, was bei einem Winkel von 180° augenscheinlich wird.

Die Hausdorff-Distanz (Abbildung E.5(a)) genauso wie die Trucco-Distanz steigt zwar nicht monoton, unterscheidet dafür gut Winkel- und Verschiebungsunterschiede. Die fehlende Monotonieeigenschaft führt dazu, dass die Ergebnisse der Distanzfunktion nicht klar interpretierbar sind. Die verbleibende Mittelpunkt- und Strecken-Distanz (Abbildung E.5(d) und E.5(g)) sind gut geeignet, um Winkel- und Verschiebungsunterschiede zu quantifizieren. Zudem sind sie monoton steigend.

Neben Unterschieden in Position und Orientierung, die beispielsweise durch unpräzise Daten bei der Segmentierung entstehen können, müssen auch unvollständige Daten, die in unterschiedliche Streckenlänge resultieren, behandelt werden. Abbildung E.6(a) bis E.6(h) präsentieren die Funktionen, die von Streckenlängen beeinflusst werden.

Die Nächster-Punkt- und Trucco-Distanz sind unabhängig von der Streckenlänge, wobei die Trucco-Distanz zur Berechnung die Streckenlängen normalisiert und daher unabhängig ist. Diesen Strecken steht die Hausdorff-Distanz gegenüber, diese hängt von der Streckenlänge ab, allerdings spielt dies ab einem bestimmten Fehler in der Verschiebung oder Drehung keine Rolle mehr (siehe Abbildung E.6(a) und E.6(b)). Die Modifizierte-Strecken-Hausdorff-Distanz nutzt die kürzeste Strecke, so dass kürzere Strecken die Distanz verkleinern (siehe Abbildung E.6(c)). Dies könnte im Extremfall dazu führen, dass zwei Strecken als sehr ähnlich angesehen werden, die senkrecht zueinander stehen, wenn die eine Strecke sehr lang und die andere sehr kurz ist. Das gegenteilige Verhalten ist bei der Modifizierten-Lot-Hausdorff-Distanz zu beobachten, dort beeinflusst nur die längste Strecke das Ergebnis der Distanzfunktion (siehe Abbildung E.6(d)).

Für die Modifizierte-Strecken-Hausdorff-Distanz wird nur die Abbildung zum Winkelverhalten gezeigt, da die Länge unabhängig von der Verschiebung ist. Die Mittelpunkt- und Strecken-Distanz sind beide abhängig von den Streckenlängen, was in Abbildung E.6(e) bis E.6(h) gezeigt wird. Tabelle E.1 gibt einen Überblick über die Eigenschaften der Distanzfunktionen.

Tabellenverzeichnis

Abbildungsverzeichnis

Pseudocodeverzeichnis

Begriffs- und Definitionsverzeichnis

Eigene Veröffentlichungen

[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 (ifgiPrints), S. 101–110

[PKS+16] Piao, Songlin ; Kiekbusch, Lisa ; Schmidt, Daniel ; Berns, Karsten ; Hering, Daniel ; Wirtz, Stefan ; Hering, Nils ; Weiland, Jürgen: Real-time multi-platform pedestrian detection in a heavy duty driver assistance system. In: *Commercial Vehicle Technology 2016 – Proceedings of the Commercial Vehicle Technology Symposium (CVT 2016)*, 2016, S. 61–70

[WDP11] Wirtz, Stefan ; Decker, Peter ; Paulus, Dietrich: Semiautomatic generation of semantic building models from image series. In: *Three-Dimensional Image Processing (3DIP) and Applications II* Bd. 8290, SPIE, 2011, S. 8290191–8290198

[WFP12] Wirtz, Stefan ; Falkowski, Kerstin ; Paulus, Dietrich: Model-based recognition of 2D objects under perspective images. In: *Pattern Recognition and Image Analysis* Bd. 22, Springer-Verlag New York, Inc., 2012. – ISSN 1054–6618, S. 419–432

[WHP10] Wirtz, Stefan ; Häselich, Marcel ; Paulus, Dietrich: Model-Based Recognition of Domino Tiles using TGraphs. In: *Pattern Recognition, Proceedings of the 32nd DAGM* Bd. 6376, Springer Berlin Heidelberg, 2010 (LNCS). – ISBN 978–3–642–15985–5, S. 101–110

[WP10] Wirtz, Stefan ; Paulus, Dietrich: Model-based recognition of 2D objects in perspective images. In: *Proceedings of the 10th International Conference on Pattern Recognition and Image Analysis: New Information Technologies (PRIA-10-2010)*, Springer MAIK Nauka/Interperiodica, 2010. – ISBN 978–5–7325–0972–4, S. 259–261

[WP11] Wirtz, Stefan ; Paulus, Dietrich: Model-based recognition of 2D objects in perspective images. In: *Pattern Recognition and Image Analysis* 21 (2011), Nr. 2, S. 361–364. – ISSN 1054–6618

[WP15] Wirtz, Stefan ; Paulus, Dietrich: Evaluation of Established Line Segment Distance Functions. In: Paulus, Dietrich (Hrsg.) ; Fuchs, Christian (Hrsg.) ; Droege, Detlev (Hrsg.): *9th Open German-Russian Workshop on Pattern Recognition and Image Understanding (OGRW 2014), Proceedings*. Koblenz : University of Koblenz-Landau, 2015, S. 89–93

[WP16] Wirtz, S. ; Paulus, D.: Evaluation of established line segment distance functions. In: *Pattern Recognition and Image Analysis* 26 (2016), Nr. 2, S. 354–359. `http://dx.doi.org/10.1134/S1054661816020267`. – DOI 10.1134/S1054661816020267. – ISSN 1555–6212

Literaturverzeichnis

[ABB95] Alt, Helmut ; Behrends, Bernd ; Blömer, Johannes: Approximate Matching of Polygonal Shapes. In: *Annals of Mathematics and Artificial Intelligence* 13 (1995), Nr. 3–4, S. 251–265. – This is the full version of SoCG'91.

[AFD+99] Ahlrichs, Ulrike ; Fischer, Julia ; Denzler, Joachim ; Drexler, Chr. ; Niemann, Heinrich ; Nöth, E. ; Paulus, Dietrich: Knowledge Based Image and Speech Analysis for Service Robots. In: *Integration of Speech and Image Understanding.* Los Alamitos, USA, 1999. – ISBN 0–7695–0471–X, S. 21–47. – Proceedings Integration of Speech and Image Understanding, 21 September 1999. Corfu, Greece. Seiten 21-47. IEEE Computer Society, Los Alamitos, USA. ISBN: 0-7695-0471-X.

[AS09] Arai, Yuki ; Saito, Hideo: Complemental Use of Multiple Cameras for Stable Tracking of Multiple Markers. In: *VMR '09: Proceedings of the 3rd International Conference on Virtual and Mixed Reality.* Berlin, Heidelberg : Springer-Verlag, 2009. – ISBN 978–3–642– 02770–3, S. 413–420

[ASJ+07] Ali, H. ; Seifert, Christin ; Jindal, N. ; Paletta, Lucas ; Paar, G.: Window Detection in Facades. In: *Image Analysis and Processing, International Conference on* 0 (2007), S. 837–842. ISBN 0–7695– 2877–5

[Bal06] Balthasar, Dirk: *Drei neue Verfahren zum Matching und zur Klassifikation unter Echtzeitbedingungen*, Diss., 2006. `http://www.uni-koblenz.de/~lb/publications/Balthasar2006DNV_slides.pdf`

[BB82] Ballard, Dana H. ; Brown, Chris. M.: *Computer Vision.* Englewood Cliffs, NJ : Prentice-Hall, 1982

[BCM+03] Baader, Franz (Hrsg.) ; Calvanese, Diego (Hrsg.) ; McGuinness, Deborah L. (Hrsg.) ; Nardi, Daniele (Hrsg.) ; Patel-Schneider, Peter F. (Hrsg.): *The description logic handbook: theory, implementation, and applications.* New York, NY, USA : Cambridge University Press, 2003. – ISBN 0–521–78176– 0

[BKI06] Beierle, C. ; Kern-Isberner, G.: *Methoden wissensbasierter Systeme - Grundlagen, Algorithmen, Anwendungen.* 1. Auflage : Vieweg, 2006

[BMP02] Belongie, Serge ; Malik, Jitendra ; Puzicha, Jan: Shape Matching and Object Recognition Using Shape Contexts. In: *IEEE Trans. Pattern Anal. Mach. Intell* 24 (2002), Nr. 4, S. 509–522

[Bou08] Bouguet, J. Y.: *Camera calibration toolbox for Matlab.* 2008

[BSS+15] Brust, Clemens-Alexander ; Sickert, Sven ; Simon, Marcel ; Rodner, Erik ; Denzler, Joachim: Convolutional Patch Networks with Spatial Prior for Road Detection and Urban Scene Understanding. In: *Proceedings of the International Conference on Computer Vision Theory and Applications (VISAPP)*, 2015

[BTP09] Burochin, J.P. ; Tournaire, O. ; Paparoditis, Nicolas: An Unsupervised Hierarchical Segmentation of a Facade Building Image in Elementary 2D-Models. In: *CMRT09*, 2009, S. 223–228

[BVTP10] Burochin, Jean-Pascal ; Vallet, Bruno ; Tournaire, O. ; Paparoditis, Nicolas: A Formulation For Unsupervised Hierarchical Segmentation Of Façade Images With Periodic Models. In: *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences* 38 (2010), S. 227–232

[CDL+01] Cox, Simon ; Daisey, Paul ; Lake, Ron ; Portele, Clemens ; Whiteside, Arliss: OpenGIS Geography Markup Language (GML) Implementation Specification / Open Geospatial Consortium, Inc. 2001 (3.1.1). – Forschungsbericht

[CLG03] Chen, Jingying ; Leung, Maylor K. ; Gao, Yongsheng: Noisy logo recognition using line segment Hausdorff distance. In: *Pattern Recognition* 36 (2003), Nr. 4, S. 943–955

[Con14] Consortium, Open G.: *OGC City Geography Markup Language (CityGML) Encoding Standard*. `https://portal.opengeospatial.org/files/?artifact_id=47842`. Version: 2014. – Zugriff 25.10.2014

[CRBSF09] Collet Romea, Alvaro ; Berenson, Dmitry ; Srinivasa, Siddhartha ; Ferguson, David: Object Recognition and Full Pose Registration from a Single Image for Robotic Manipulation. In: *IEEE International Conference on Robotics and Automation (ICRA '09)*, 2009

[CSS06] Cremers, Daniel ; Sochen, Nir ; Schnörr, Christoph: A Multiphase Dynamic Labeling Model for Variational Recognition-driven Image Segmentation. In: *International Journal of Computer Vision* 66 (2006), Nr. 1, S. 67–81

[CSW01] Cremers, Daniel ; Schnorr, Christoph ; Weickert, Joachim: Diffusion-Snakes: Combining Statistical Shape Knowledge and Image Information in a Variational Framework. In: *IEEE Workshop on Variational and Level Set Methods (VLSM'01)* (2001), S. 137. ISBN 0–7695– 1278–X

[DD92] DeMenthon, Daniel ; Davis, Larry S.: Model-Based Object Pose in 25 Lines of Code. In: *ECCV '92: Proceedings of the Second European Conference on Computer Vision*. London, UK : Springer-Verlag, 1992. – ISBN 3–540–55426– 2, S. 335 343

[Dec12] Decker, Peter: *Modellbasierte Kameraposebestimmung aus Bildern*. Der Andere Verlag, 2012. – ISBN 978–3–86247– 307–6

[DEF+98] Dahm, Peter ; Ebert, Jürgen ; Franzke, Angelika ; Kamp, Manfred ; Winter, Andreas: TGraphen und EER-Schemata - formale Grundlagen / Universität Koblenz-Landau, Institut für Informatik. Koblenz, 1998 (12/98). – Forschungsbericht

[DEL94] Dahm, Peter ; Ebert, Jürgen ; Litauer, Christoph: Das EMS-Graphenlabor 3.0 - Benutzerhandbuch / Universität Koblenz-Landau, Institut für Softwaretechnik. Koblenz, 1994 (3/94). – Forschungsbericht

[DMB+06] Dzbor, Martin ; Motta, Enrico ; Buil, Carlos ; Gomez, Jose ; Görlitz, Olaf ; Lewen, Holger: Developing ontologies in OWL: An observational study. In: *Proceedings of the OWL: Experiences and Directions workshop*, 2006

[DTC04] Dick, A. R. ; Torr, P. H. S. ; Cipolla, R.: Modelling and Interpretation of Architecture from Several Images. In: *Int. J. Comput. Vision* 60 (2004), Nr. 2, S. 111–134. – ISSN 0920–5691

[DTM96] Debevec, Paul E. ; Taylor ; Malik, Jitendra: Modeling and rendering architecture from photographs: a hybrid geometry- and image-based approach. In: *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. New York, NY, USA : ACM, 1996. – ISBN 0–89791–746–4, S. 11–20

[Ebe08] Ebert, Jürgen: Metamodels Taken Seriously: The TGraph Approach. In: Kontogiannis, Kostas (Hrsg.) ; Tjortjis, Christos (Hrsg.) ; Winter, Andreas (Hrsg.): *12th European Conference on Software Maintenance and Reengineering*. Piscataway, NJ, 2008

[ERW08] Ebert, Jürgen ; Riediger, Volker ; Winter, Andreas: Graph Technology in Reverse Engineering, The TGraph Approach. In: Gimnich, Rainer (Hrsg.) ; Kaiser, Uwe (Hrsg.) ; Quante, Jochen (Hrsg.) ; Winter, Andreas (Hrsg.): *10th Workshop Software Reengineering (WSR 2008)* Bd. 126. Bonn : GI, 2008 (GI Lecture Notes in Informatics), S. 67–81

[ES09] Enomoto, Akihito ; Saito, Hideo: AR Display for Observing Sports Events Based on Camera Tracking Using Pattern of Ground. In: *VMR '09: Proceedings of the 3rd International Conference on Virtual and Mixed Reality*. Berlin, Heidelberg : Springer-Verlag, 2009. – ISBN 978–3–642– 02770–3, S. 421–430

[EWD+96] Ebert, Jürgen ; Winter, Andreas ; Dahm, Peter ; Franzke, Angelika ; Süttenbach, Roger: Graph Based Modeling and Implementation with EER/GRAL. In: Thalheim, B. (Hrsg.): *ER'96 - Proceedings of the 15th International Conference on Conceptual Modeling*. Berlin : Springer Verlag, 1996 (LNCS 1157), S. 163–178

[Fal10] Falkowski, Kerstin: A component concept for scientific experiments - focused on versatile visual component assembling. In: *Proceedings of the Fifteenth International Workshop on Component-Oriented Programming (WCOP) 2010*, 2010, S. 31–38

[FB81] Fischler, Martin A. ; Bolles, Robert C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. In: *Communications of the ACM* 24 (1981), Nr. 6, S. 381–395. – ISSN 0001–0782

[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. – ISSN 1682–1750, S. 115 – 120

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

[FE11] Falkowski, Kerstin ; Ebert, Jürgen: A reference schema for interoperability between geo data and 3d models. In: *Geoinformatik 2011 - Geochange*, 2011. – Accepted

[FG87] Förstner, Wolfgang ; Gülch, E.: A fast operator for detection and precise location of distinct points, corners, and centers of circular features. In: *Proc. Intercommision Conf. on Fast Processing of Photogrammetric Data*, 1987, S. 281–305

[Fio01] Fiore, Paul D.: Efficient linear solution of exterior orientation. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 23 (2001), Nr. 2, S. 140–148. – ISSN 0162–8828

[FNN98] Fischer, J. ; Niemann, Heinrich ; Noeth, E.: A Real-Time and Any-Time Approach for a Dialog System. In: *Proc. International Workshop Speech and Computer (SPECOM'98)*. St. Petersburg, 1998, S. 85–90

[FPT03] Fahrmeir, Ludwig ; Pigeot, Iris ; Tutz, Gerhard: *Statistik. Der Weg zur Datenanalyse*. München : Springer Verlag, 2003

[FRD10] Frohlich, Bjorn ; Rodner, Erik ; Denzler, Joachim: A Fast Approach for Pixelwise Labeling of Facade Images. In: *Proceedings of the 2010 20th International Conference on Pattern Recognition*. Washington, DC, USA : IEEE Computer Society, 2010 (ICPR '10). – ISBN 978–0–7695– 4109–9, S. 3029–3032

[FRKD12] Frohlich, Bjorn ; Rodner, Erik ; Kemmler, Michael ; Denzler, Joachim: Large-Scale Gaussian Process Classification using Random Decision Forests. In: *Pattern Recognition and Image Analysis* 22 (2012), Nr. 1, S. 113–120

[GKCN08] Groeger, Gerhard ; Kolbe, Thomas H. ; Czerwinski, Angela ; Nagel, Claus: OpenGIS City Geography Markup Language (CityGML) Encoding Standard / Open Geospatial Consortium Inc. 2008 (1.0.0). – Forschungsbericht

[GMMB00] Guillou, E ; Meneveaux, D ; Maisel, E ; Bouatouch, K: Using vanishing points for camera calibration and coarse 3D reconstruction from a single image. In: *The Visual Computer, Springer Berlin* (2000), Nr. 16, S. 396–410

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

[GRRW10] Gurski, Frank ; Rothe, Irene ; Rothe, Jörg ; Wanke, Egon: *Exakte Algorithmen für schwere Graphenprobleme*. Springer, 2010 (eXamen.press). – I–XII, 1–331 S. – ISBN 978–3–642–04499–1

[GW01] Gonzalez, Rafael C. ; Woods, Richard E.: *Digital Image Processing*. 2. Prentice Hall, 2001

[GWFM+13] Goodfellow ; Warde-Farley ; Mirza ; Courville ; Bengio: Maxout Networks. In: *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013* Bd. 28, JMLR.org, 2013 (JMLR Proceedings), S. 1319–1327

[GXLP05] Guo, Xiaoxin ; Xu, Zhiwen ; Lu, Yinan ; Pang, Yunjie: An Application of Fourier-Mellin Transform in Image Registration. In: *CIT*, 2005. – ISBN 0–7695–2432–X, S. 619–623

[Har97] Hartley, Richard I.: In Defense of the Eight-Point Algorithm. In: *Pattern Analysis and Machine Intelligence* 19 (1997), Nr. 6, S. 580–593

[Häs10] Häselich, Marcel: *Erkennen von Dominosteinen mit Hilfe von wissensbasierten Bildanalyse-Verfahren*, Universität Koblenz-Landau, Institut für Computervisualistik, Arbeitsgruppe Aktives Sehen, Diplomarbeit, 2010

[HKRS08] Hitzler, Pascal ; Krötzsch, Markus ; Rudolph, Sebastian ; Sure, York: *Semantic Web: Grundlagen*. Berlin : Springer, 2008. – ISBN 978–3–540– 33993–9

[HL13] Hartmann ; Lehner: *Technische Expertensysteme: Grundlagen, Programmiersprachen, Anwendungen*. Springer Berlin Heidelberg, 2013. – ISBN 9783642934537

[HMW03] Horrocks, Ian ; McGuinness, Deborah L. ; Welty, Christopher A.: Digital libraries and web-based information systems. In: [BCM+03], S. 427–449

[Hoi07] Hois, Johana: Towards Combining Ontologies and Uncertain Knowledge. In: *progic07: The Third Workshop on Combining Probability and Logic* (2007). – University of Kent, Canterbury, UK

[HPFP09] Haugeard, J.E. ; Philipp Foliguet, S. ; Precioso, F.: Windows and facades retrieval using similarity on graph of contours. In: *IEEE International Conference on Image Processing (ICIP 09)*, 2009, S. 269–272

[HSK+12] Hinton, Geoffrey E. ; Srivastava ; Krizhevsky, Alex ; Sutskever ; Salakhutdinov, Ruslan R.: Improving neural networks by preventing co-adaptation of feature detectors. In: *CoRR* abs/1207.0580 (2012)

[HST99] Horrocks, Ian ; Sattler, Ulrike ; Tobies, Stephan: Practical Reasoning for Expressive Description Logics. In: *LPAR '99: Proceedings of the 6th International Conference on Logic Programming and Automated Reasoning*. London, UK : Springer-Verlag, 1999. – ISBN 3–540–66492– 0, S. 161–180

[Hu62] Hu, M.K.: Visual Pattern Recognition by Moment Invariants. In: *IEEE Transactions on Information Theory* 8 (1962), Nr. 2, S. 179–187

[HZ03] Hartley, Richard I. ; Zisserman, Andrew: *Multiple View Geometry in Computer Vision*. 2. Cambridge : Cambridge University Press, 2003

[HZ04] Hartley, Richard I. ; Zisserman, Andrew: *Multiple View Geometry in Computer Vision*. 2. Cambridge University Press, 2004

[JP08] Jahangiri, M ; Petrou, M: Fully bottom-up blob extraction in building facades. In: *Proceedings of PRIA* (2008), S. 1–8

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

[Jun02] Jung, Steffen: *Erarbeitung eines wissensbasierten Systems für die Modellierung von Mehrkomponenten-Mehrphasen-Gleichgewichten*. Universitäts- und Landesbibliothek, 2002

[KH94] Kumar, Rakesh ; Hanson, Allen R.: Robust methods for estimating pose and a sensitivity analysis. In: *CVGIP: Image Underst.* 60 (1994), Nr. 3, S. 313 342. – ISSN 1049–9660

[Krä03] Krämer, J.: *Delaunay-Triangulierungen in zwei und drei Dimensionen*. Diplom.de, 2003 https://books.google.de/books?id=sYh6AQAAQBAJ. – ISBN 9783832466466

[KSH12] Krizhevsky, Alex ; Sutskever ; Hinton: ImageNet Classification with Deep Convolutional Neural Networks. In: Pereira (Hrsg.) ; Burges (Hrsg.) ; Bottou (Hrsg.) ; Weinberger (Hrsg.): *Advances in Neural Information Processing Systems 25*. Curran Associates, Inc., 2012, S. 1097–1105

[KSN92] In: Kummert, F ; Sagerer, G. ; Niemann, Heinrich: *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

[Kum92] Kummert, Franz: *Flexible Steuerung eines sprach-verstehenden Systems mit homogener Wissensbasis*. 1992

[Kur04] In: Kurlbaum, Jörg: *Levenberg-Marquardt Algorithmus, Nichtlineare Ausgleichsrechnung*. Universität Bremen, 2004, S. 27

[KWBE02] Kühne, Gerald ; Weickert, Joachim ; Beier, Markus ; Effelsberg, Wolfgang: Fast Implicit Active Contour Models. In: *Proceedings Pattern Recognition (24th DAGM Symposium)*. Zürich, Schweiz, 2002, S. 133–140

[KZ05] Kosecka, J. ; Zhang, Wei: Extraction, matching, and pose recovery based on dominant rectangular structures. In: *Comput. Vis. Image Underst.* 100 (2005), S. 274–293. – ISSN 1077–3142

[LA09] Lourakis, M.I.A. ; Argyros, A.A.: SBA: A Software Package for Generic Sparse Bundle Adjustment. In: *ACM Trans. Math. Software* 36 (2009), Nr. 1, S. 1–30

[LA13] Lafarge, Florent ; Alliez, Pierre: Surface reconstruction through point set structuring. In: *Proc. of Eurographics*. Girona, Spain, 2013

[LH96] Lenz, R. ; Homma, K.: Rotational Symmetry: The Lie Group SO(3) and Its Representations. In: *Proceedings of the International Conference on Image Processing (ICIP 1996)*, 1996, S. III: 203–206

[LHN00] Lee, SC ; Huertas, A ; Nevatia, R: Modeling 3-D complex buildings with user assistance. In: *Applications of Computer Vision, 2000, Fifth IEEE Workshop on.*, IEEE, 2000, S. 170–177

[Lia99] Liang, S: *The Java Native Interface*. Addison-Wesley, 1999 http://books.google.it/books?id=NFnBRcu8DHEC. – ISBN 0–201–32577–2

[LJN02] Lee, S.C. ; Jung, S.K. ; Nevatia, R.: Automatic integration of facade textures into 3D building models with a projective geometry based line clustering. In: *Computer Graphics Forum* Bd. 21, Wiley Online Library, 2002, S. 511–519

[LKBH13] Lafarge, Florent ; Keriven, Renaud ; Brédif, Mathieu ; Hiep, Vu: A hybrid multi-view stereo algorithm for modeling urban scenes. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35 (2013), Nr. 1, S. 5–17

[LLF05] Lepetit, Vincent ; Lagger, Pascal ; Fua, Pascal: Randomized Trees for Real-Time Keypoint Recognition. In: *Conference on Computer Vision and Pattern Recognition, San Diego, CA*, 2005, S. 775–781

[LLS04] Leibe, Bastian ; Leonardis, Ales ; Schiele, Bernt: Combined Object Categorization and Segmentation With An Implicit Shape Model. In: *ECCV' 04 Workshop on Statistical Learning in Computer Vision*, 2004, S. 17–32

[LLS06] Leibe, Bastian ; Leonardis, Ales ; Schiele, Bernt: An Implicit Shape Model for Combined Object Categorization and Segmentation. In: *Toward Category-Level Object Recognition*, 2006, S. 508–524

[LN04] Lee, Sung C. ; Nevatia, R.: Extraction and Integration of Window in a 3D Building Model from Ground View Images. In: *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on* 2 (2004), S. 113–120. – ISSN 1063–6919

[Low04] Lowe, David G.: Distinctive Image Features from Scale-Invariant Keypoints. In: *International Journal of Computer Vision* 60 (2004), Nr. 2, S. 91–110

[LS02] Li, Yi ; Shapiro, Linda G.: Consistent Line Clusters for Building Recognition in CBIR. In: *ICPR '02: Proceedings of the 16 th International Conference on Pattern Recognition (ICPR'02) Volume 3*. Washington, DC, USA : IEEE Computer Society, 2002. – ISBN 0–7695– 1695–X, S. 30952

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

[MBCM99] Marchand, Éric ; Bouthemy, Patrick ; Chaumette, François ; Moreau, Valérie: Robust Real-Time Visual Tracking using a 2D-3D Model-based Approach. In: *International Conference on Computer Vision*, 1999, S. 262–268

[MMS93] Mineau, Guy W. (Hrsg.) ; Moulin, Bernard (Hrsg.) ; Sowa, John F. (Hrsg.): *Proceedings on Conceptual Graphs for Knowledge Representation*. Bd. *699*. London, UK : Springer, 1993 (Lecture Notes in Computer Science). – ISBN 3–540–56979–0

[MMVG16] Mathias ; Martinović ; Van Gool, Luc: ATLAS: A Three-Layered Approach to Facade Parsing. In: *International Journal of Computer Vision* 118 (2016), Nr. 1, S. 22–48. ISBN 1573–1405

[MR05] Mayer, H ; Reznik, S: Building façade interpretation from image sequences. In: *of ISPRS Workshop CMRT* Bd. XXXVI, 2005, S. 55–60

[MRM+10] Musialski, Przemyslaw ; Recheis, Meinrad ; Maierhofer, Stefan ; Wonka, Peter ; Purgathofer, Werner: Tiling of ortho-rectified facade images. In: *Proceedings of the 26th Spring Conference on Computer Graphics*. New York, NY, USA : ACM, 2010 (SCCG '10). – ISBN 978–1–4503– 0558–7, S. 117–126

[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

[MZWG07] Müller, P. ; Zeng, Gang ; Wonka, Peter ; Gool, Luc J. V.: Image-based procedural modeling of facades. In: *ACM Transactions on Graphics* 26 (2007), Nr. 3, S. 85. – ISSN 0730–0301

[NDM13] Nguatem, W. ; Drauschke, M. ; Mayer, H.: Roof Reconstruction from Point Clouds using Importance Sampling. In: *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences* (2013), Nr. 3, S. 73–78

[NDM14] Nguatem, W. ; Drauschke, M. ; Mayer, H.: Localization of Windows and Doors in 3d Point Clouds of Facades. In: *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences* (2014), S. 87–94

[Neu99] Neuweg, Georg H.: *Könnerschaft und implizites Wissen. Zur lehrlerntheoretischen Bedeutung der Erkenntnis- und Wissenstheorie Michael Polanyis.* Waxmann, 1999. – ISBN 3893257535

[Neu08] Neumann, B.: Bayesian Compositional Hierarchies - A Probabilistic Structure for Scene Interpretation / Department of Informatics, Hamburg University. 2008 (FBI-HH-B-282/08). – Forschungsbericht

[NFH07] Nischwitz, Alfred ; Fischer, Max ; Haberäcker, Peter: *Computergrafik und Bildverarbeitung.* 2. Wiesbaden : Vieweg, 2007. – ISBN 978–3–8348– 0186–9

[NFPF96] Niemann, Heinrich ; Fischer, V. ; Paulus, Dietrich ; Fischer, J.: Knowledge Based Image Understanding by Iterative Optimization. In: Görz, G. (Hrsg.) ; Hölldobler, St. (Hrsg.): *KI 96: Advances in Artificial Intelligence* Bd. 1137 (Lecture Notes in Artificial Intelligence). Berlin : Springer, 1996, S. 287 301

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

[Nil80] Nilsson, N.: *Principles of Artificial Intelligence.* San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 1980. – ISBN 0–934613– 10–9

[NSSK90] Niemann, Heinrich ; Sagerer, G. ; Schröder, S. ; Kummert, F.: ERNEST: A Semantic Network System for Pattern Understanding. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 9 (1990), S. 883–905

[OLA13] Oesau, Sven ; Lafarge, Florent ; Alliez, Pierre: Indoor Scene Reconstruction using Primitive-driven Space Partitioning and Graph-cut. In: *Eurographics Workshop on Urban Data Modelling and Visualisation.* Girona, Espagne, 2013

[OQT05] In: Oosterom, Peter van ; Quak, Wilko ; Tijssen, Theo: *About Invalid, Valid and Clean Polygons.* Berlin, Heidelberg : Springer Berlin Heidelberg, 2005. – ISBN 978–3–540–26772–0, 1–16

[Pau92] Paulus, Dietrich: *Objektorientierte und wissensbasierte Bildverarbeitung.* Braunschweig : Vieweg, 1992 `https://www.uni-koblenz.de/~agas/Public/Paulus1992OUW.pdf`. – ISBN 3–528–05270– 8

[Pau01] Paulus, Dietrich: *Aktives Bildverstehen.* Osnabrück : Der Andere Verlag, 2001

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

[Pea84] Pearl, Judea: *Heuristics Intelligent search strategies for computer problem solving.* Addison-Wesley, 1984

[PH03] Paulus, Dietrich ; Hornegger, Joachim: *Applied pattern recognition: A practical introduction to image and speech processing in C++.* 4. Braunschweig : Vieweg, 2003 (Advanced Studies in Computer Science)

[Pol67] Polanyi, Michael: *The Tacit Dimension.* London, Routledge & K. Paul, 1967

[PPDS09] Paulus, Dietrich ; Priese, Lutz ; Decker, Peter ; Schmitt, Frank: Pose-Tracking Forschungsbericht. Version: 2009. `http://www.uni-koblenz.de/~fb4reports/2009/2009_17_Arbeitsberichte.pdf`. 2009 (17/2009). – Forschungsbericht

[Pri15] Priese, Lutz: *Computer Vision - EinfÃ¼hrung in die Verarbeitung und Analyse digitaler Bilder*. Springer Vieweg, 2015

[Qui68] Quillian, M. R.: Semantic Memory. In: Minsky, M. (Hrsg.): *Semantic Information Processing*. Cambridge, Mass. : MIT Press, 1968, S. 227–270

[Qui95] Quint, Franz: Aerial image understanding using digital map based semantic models. In: *Euroconference GIS, Union Europeenne, Project Capital Humain et Mobilite, ENSG*. Saint-Mande, France, 1995

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

[RM07] Reznik, Sergej ; Mayer, Helmut: Implicit Shape Models, Self-Diagnosis, and Model Selection for 3D Facade Interpretation. In: *In the International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* (2007), Nr. XXXVI, S. 173–178

[RN04] Rajashekahar, S.C. ; Namboodiri, V.P.: Image retrieval based on projective invariance. In: *Image Processing, 2004. ICIP '04. 2004 International Conference on* Bd. 1, 2004. – ISSN 1522–4880, S. 405–408

[RP98] Rehrmann, Volker ; Priese, Lutz: Fast and Robust Segmentation of Natural Color Scenes. In: Chin, Roland T. (Hrsg.) ; Pong, Ting-Chuen (Hrsg.): *3rd Asian Conference on Computer Vision (ACCV'98)*, Springer Verlag, 1998 (LNCS 1351), S. 598–606

[RSM+08] Rusu, Radu B. ; Sundaresan, Aravind ; Morisset, Benoit ; Agrawal, Motilal ; Beetz, Michael: Leaving Flatland: Realtime 3D Stereo Semantic Reconstruction. In: *ICIRA '08: Proceedings of the First International Conference on Intelligent Robotics and Applications*. Berlin, Heidelberg : Springer-Verlag, 2008. – ISBN 978–3–540– 88512–2, S. 921–932

[Sag85] Sagerer, G.: *Darstellung und Nutzung von Expertenwissen für ein Bildanalysesystem*. Berlin : Springer, 1985

[Sal95] Salzbrunn, Richard: *Wissensbasierte Erkennung und Lokalisierung von Objekten*. Aachen : Shaker Verlag, 1995

[SC96] Schiele, Bernt ; Crowley, James L.: Object Recognition Using Multidimensional Receptive Field Histograms. In: *ECCV '96: Proceedings of the 4th European Conference on Computer Vision-Volume I*. London, UK : Springer-Verlag, 1996. – ISBN 3–540–61122– 3, S. 610–619

[Sch11] Schmitt, Frank: *Semantik aus Segmenten. Zwei neue Verfahren zur Detektion und Identifikation von Objekten im 2-D und 3-D*. Der Andere Verlag, Uelvesbüll, Universität Koblenz-Landau, Diss., 2011

[Sch14] Schmidhuber: Deep Learning in Neural Networks: An Overview. In: *CoRR* abs/1404.7828 (2014)

[Sel04] Selig, J.M.: Lie Groups and Lie algebras in Robotics. In: Byrnes, J. (Hrsg.): *Computational Noncommutative Algebra and Applications: Proceedings of the NATO Advanced Study Institute*, Kluwer, 2004, S. 101–125

[Sha76] Shafer, Glenn: *A Mathematical Theory of Evidence*. Princeton : Princeton University Press, 1976

[SL04] Supervisor, Yu X. ; Leung, Maylor: Hausdorff Distance for Shape Matching. In: *The 4th LASTED International Conference on visualization image and image processing* (2004), S. 1–25

[SLS06] Seemann, Edgar ; Leibe, Bastian ; Schiele, Bernt: Multi-Aspect Detection of Articulated Objects. In: *onference on Computer Vision and Pattern Recognition - Volume 2 (CVPR'06)* 2 (2006), S. 1582–1588. – ISSN 1063–6919

[SN97] Sagerer, G. ; Niemann, Heinrich: *Semantic Networks for Understanding Scenes*. New York and London : Plenum Press, 1997 (Advances in Computer Vision and Machine Intelligen)

[Sow92] Sowa, John F.: Conceptual graphs. In: *Knowledged-Based Systems* 5 (1992), Nr. 3, S. 171–172

[SP09a] Schmitt, Frank ; Priese, Lutz: Sky detection in CSC-segmented color images. In: *Fourth International Conference on Computer Vision Theory and Applications (VISAPP) 2009, Lisboa, Portugal* Bd. 2, 2009, S. 101–106

[SP09b] Schmitt, Frank ; Priese, Lutz: Vanishing Point Detection with an Intersection Point Neighborhood. In: *Discrete Geometry for Computer Imagery* Bd. 5810, Springer Berlin / Heidelberg, 2009 (Lecture Notes in Computer Science). – ISBN 978–3–642– 04396–3, S. 132–143

[SS09] Staab, Steffen ; Studer, R. ; Studer, R. (Hrsg.): *Handbook on Ontologies - International Handbooks on Information Systems*. 2nd Edition. Springer Verlag, 2009

[SSQTMN13] Simo-Serra, E. ; Quattoni, A. ; Torras, Carme ; Moreno-Noguer, F.: A Joint Model for 2D and 3D Pose Estimation from a Single Image. In: *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, 2013. – ISSN 1063–6919, S. 3634–3641

[SSRA+12] Simo-Serra, E. ; Ramisa, A. ; Alenyà, Guillem ; Torras, Carme ; Moreno-Noguer, F.: Single image 3D human pose estimation from noisy observations. In: *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, 2012. – ISSN 1063–6919, S. 2673–2680

[SSS91] Schmidt-Schauss, Manfred ; Smolka, Gert: Attributive concept descriptions with complements. In: *Artificial Intelligence* 48 (1991), Nr. 1, S. 1–26. – ISSN 0004–3702

[Sta73] Stachowiak, H.: *Allgemeine Modelltheorie*. Wien : Springer, 1973

[STKP11] Simon, Loïc ; Teboul, Olivier ; Koutsourakis, Panagiotis ; Paragios, Nikos: Random Exploration of the Procedural Space for Single- View 3D Modeling of Buildings. In: *International Journal of Computer Vision* 93 (2011), Nr. 2, S. 253–271

[TK09] Theodoridis, Sergios ; Koutroumbas, Konstantinos: *Pattern Recognition*. 4. Academic Press, 2009

[TKJ08] Trinh, Hoang-Hon ; Kim, Dae-Nyeon ; Jo, Kang-Hyun: Cross Ratio-Based Refinement of Local Features for Building Recognition. In: Huang, De-Shuang (Hrsg.) ; II, Donald C. W. (Hrsg.) ; Lavine, D. (Hrsg.) ; Jo, Kang-Hyun (Hrsg.): *ICIC (1)* Bd. 5226, Springer, 2008 (Lecture Notes in Computer Science). – ISBN 978–3–540–87440–9, S. 544–551

[TKS+11] Teboul, Olivier ; Kokkinos ; Simon ; Koutsourakis, Panagiotis ; Paragios, Nikos: Shape grammar parsing via Reinforcement Learning. In: *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, 2011. – ISBN 1063–6919, S. 2273–2280

[TV98] Trucco, Emanuele ; Verri, Alessandro: *Introductory Techniques for 3-D Computer Vision*. New York : Prentice Hall, 1998

[VAB10] Vanegas, Carlos A. ; Aliaga, Daniel G. ; Benes, Bedrich: Building reconstruction using manhattan-world grammars. In: *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on* 0 (2010), S. 358–365. ISBN 978–1–4244– 6984–0

[VJ01] Viola, Paul ; Jones, Michael: Rapid object detection using a boosted cascade of. In: *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* Bd. 1. Kauai, Hawaii : IEEE Computer Society, 2001. – ISBN 0–7695–1272–0, S. 511–518

[W3C04a] W3C: RDF Vocabulary Description Language 1.0: RDF Schema. 2004. – Forschungsbericht. – W3C Recommendation 10 02 2004

[W3C04b] W3C: Resource Description Framework (RDF): Concepts and Abstract Syntax / W3C. 2004. – Forschungsbericht. – W3C Recommendation 10 02 2004

[WDF08] Wenzel, S. ; Drauschke, M. ; Förstner, W.: Detection of repeated structures in facade images. In: *Pattern Recognition and Image Analysis* 18 (2008), Nr. 3, S. 406–411. – ISSN 1054–6618

[WFFP08] Wu, Changchang ; Fraundorfer, Friedrich ; Frahm, Jan-Michael ; Pollefeys, Marc: 3D model search and pose estimation from single images using VIP features. In: *Computer Vision and Pattern Recognition Workshop* 0 (2008), S. 1–8. ISBN 978–1–4244– 2339–2

[WFP12] Wirtz, Stefan ; Falkowski, Kerstin ; Paulus, Dietrich: Model-based recognition of 2D objects under perspective images. In: *Pattern Recognition and Image Analysis* Bd. 22, Springer-Verlag New York, Inc., 2012. – ISSN 1054–6618, S. 419–432

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

[Win00] Winter, Andreas: *Referenz-Metaschema für visuelle Modellierungssprachen*. Wiesbaden : Deutscher Universitätsverlag, 2000. – zugl. Dissertation, Institut für Informatik. Universität Koblenz-Landau.

[WSR10] Walter, Tobias ; Schwarz, Hannes ; Ren, Yuan: Establishing a Bridge from Graph-based Modeling Languages to Ontology Languages. In: *Proceedings of the of the Third Workshop on Transforming and Weaving Ontologies in Model Driven Engineering (TWOMDE) at TOOLS*, 2010

[WWL+14] Wang, Chunyu ; Wang, Yizhou ; Lin, Zhouchen ; Yuille, Alan ; Gao, Wen: Robust Estimation of 3D Human Poses from a Single Image. In: *CoRR* abs/1406.2282 (2014)

[WZ02] Werner, Thomas ; Zisserman, Andrew: New Techniques for Automated Architecture Reconstruction from Photographs. In: *European Conference on Computer Vision* Bd. 2, Springer-Verlag, 2002

[XH10] Xiong, Xuehan ; Huber, Daniel: Using Context to Create Semantic 3D Models of Indoor Environments. In: *Proc. BMVC*, 2010

[ZDY+14] Zhu, Menglong ; Derpanis, Konstantinos G. ; Yang, Yinfei ; Brahmbhatt, Samarth ; Zhang, Mabel ; Phillips, Cody ; Lecce, Matthieu ; Daniilidis, Kostas: Single Image 3D Object Detection and Pose Estimation for Grasping. In: *International Conference on Robotics and Automation* (2014)

[Zha00] Zhang, Zhengyou: A flexible new technique for camera calibration. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22 (2000), Nr. 11, S. 1330–1334

[ZHD01] Zhang, Bin ; Hsu, Meichun ; Dayal, Umeshwar: K-Harmonic Means - A Spatial Clustering Algorithm with Boosting. In: *Proceedings of the First International Workshop on Temporal, Spatial, and Spatio-Temporal Data Mining-Revised Papers*. London, UK, UK : Springer-Verlag, 2001 (TSDM '00). – ISBN 3–540–41773– 7, S. 31–45

[ZK05] Zhang, Wei ; Kosecka, J.: Localization Based on Building Recognition. In: *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Workshops*. Washington, DC, USA, 2005. – ISBN 0–7695– 2372–2–3, S. 21

[ZSS13] Zia, M. Z. ; Stark, Michael ; Schindler, Konrad: Explicit Occlusion Modeling for 3D Object Class Representations. In: *CVPR*, IEEE, 2013, S. 3326–3333

[ZSS14a] Zia, M. Z. ; Stark, Michael ; Schindler, Konrad: Towards Scene Understanding with Detailed 3D Object Representations. In: *CoRR* abs/1411.5935 (2014)

[ZSS14b] Zia, M.Zeeshan ; Stark, Michael ; Schindler, Konrad: Are Cars Just 3D Boxes? - Jointly Estimating the 3D Shape of Multiple Objects. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014

[ZSSS13] Zia, Zeeshan ; Stark, Michael ; Schiele, Bernt ; Schindler, Konrad: Detailed 3D Representations for Object Recognition and Modeling. In: *IEEE Transactions on Patterm Analysis and Machine Intelligence (PAMI)* 35 (2013), Nr. 11, S. 2608 – 2623

[ZZD15] Zhu, Menglong ; Zhou, Xiaowei ; Daniilidis, Kostas: Pose and Shape Estimation with Discriminatively Learned Parts. In: *CoRR* abs/1502.00192 (2015)