UNIVERSITÄT
KOBLENZ · LANDAU
Fachbereich 4: Informatik

Erkennung zusammengesetzter Zeichen zur Interaktion auf einem handelsüblichen Whiteboard

Diplomarbeit
zur Erlangung des Grades
Diplom-Informatiker
im Studiengang Computervisualistik

vorgelegt von

Christian Latsch

Betreuer: Dipl.-Inform. Peter Decker, Institut für Computervisualistik, Fachbereich Informatik, Universität Koblenz-Landau
Erstgutachter: Dipl.-Inform. Peter Decker, Institut für Computervisualistik, Fachbereich Informatik, Universität Koblenz-Landau
Zweitgutachter: Prof. Dr.-Ing. Dietrich Paulus, Institut für Computervisualistik, Fachbereich Informatik, Universität Koblenz-Landau

Koblenz, im Mai 2009

Kurzfassung

Diese Diplomarbeit präsentiert ein interaktives System, welches die Vorzüge eines handelsüblichen Whiteboards mit denen eines Computers kombiniert. Die Inhalte des Whiteboards werden von einer Kamera aufgenommen, vom Computer verarbeitet und für eine Applikation als Eingabe verwendet bzw. durch geeignete Grafiken ergänzt. Dadurch erweitert das Whiteboard die Benutzeroberfläche des Computers. Der Anwender ist in der Lage über handgezeichnete Primitive (Viereck, Dreieck, Kreis) bzw. Kombinationen der Primitive das entwickelte interaktive Spiel zu spielen.

Abstract

This diploma thesis shows an interactive system which combines the advantages of a standard whiteboard with those of a computer. The contents from the whiteboard are recorded by a video camera, converted by a computer and used as an input for an application with graphics added thereafter. Thus the whiteboard is used as an extended user interface. The user is able to play the developed interactive game by drawing primitive shapes (triangle, circle, rectangle) or even by combining two or more of them.

Erklärung

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

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

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

Koblenz, den 14. Mai 2009

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

Einleitung

Whiteboards sind im heutigen Arbeitsleben nicht mehr wegzudenken. Die meisten Büros, Labore oder Konferenzräume sind mit ihnen ausgestattet und unterstützen beispielsweise Konversationen, Aufzeichnungen oder Präsentationen. Sie ermöglichen das Arbeiten mit mehreren Personen an einem Dokument und fördern so die Kreativität. Alles in allem bleibt das Whiteboard dabei nur ein großer Notizzettel, dessen Inhalt, wenn benötigt am Ende abfotografiert oder von Hand in den Computer übertragen werden kann.

Es existieren kommerzielle Systeme in Form von Rückprojektionsgeräten, die versuchen die Eigenschaften des Whiteboards mit den Fähigkeiten eines Computers zu erweitern. Diese sind jedoch meist sehr teuer und unflexibel. Dabei handelt es sich außerdem eher um einen großen Touchscreen als um ein Whiteboard. Andere Systeme basieren auf handelsüblichen Whiteboards und versuchen dessen Inhalt mit Kameras zu digitalisieren.

Nur wenige Systeme gehen einen Schritt weiter, verwenden das Whiteboard als Benutzeroberfläche und ermöglichen so die Interaktion mit einem Computer. Dabei wird das Whiteboard aber nur als Eingabequelle genutzt, was die Interaktion zwischen Mensch und Computer beschränkt. Diese Arbeit beschäftigt sich mit der Realisierung eines Interaktiven Whiteboardsystems, welches nicht nur Eingabesondern auch Ausgabefunktionen besitzt und somit ganz neue Möglichkeiten der Interaktion schafft.

1.1 Ziele dieser Arbeit

Es soll ein interaktives Whiteboardsystem realisiert werden, welches aus einem handelsüblichen Whiteboard, einer Kamera, einem Projektor und Computer besteht. Dieses System soll handgezeichnete Symbole des Benutzers auf dem Whiteboard er-

kennen und diese als Eingabe für ein Programm nutzen, bzw. durch den Projektor geeignete ergänzende Graphiken hinzufügen.

Dazu müssen verschiedener primitive Objekte (Kreise, Rechtecke usw.) sowie deren Lage und Größe auf dem Whiteboard automatisch erkannt, interpretiert und in einer Anwendung verarbeitet werden.

Kamera und Projektor müssen zuerst kalibriert werden, indem der Projektor manuell auf das Whiteboard ausgerichtet wird. Durch Projektion bekannter Punkte wird automatisch eine Homographie berechnet und das Kamerabild mit der Hilfe dieser Homographie entzerrt.

Diese entzerrten Kamerabilder werden segmentiert. Anschließend müssen in den Segmenierungsergebnissen primitive Objekte erkannt werden. Daraufhin wird die Lage der Objekte zueinander ermittelt und als topologischer Graph in der Form "Rechteck enthält Kreis" usw. in geeigneter Form gespeichert. Modelle von möglichen Objekten werden in einer Wissensbasis gehalten. Zur Objekterkennung werden die aus dem Kamerabild erzeugten Graphen mit denen aus der Wissensbasis verglichen.

Zusammenfassend stellen sich folgende Aufgaben:

- Konzipierung eines Systems zur Interaktion mit einem handelsüblichen Whiteboard.
- Erkennung von handgezeichneten Primitiven.
- Erkennung von zusammengesetzten Zeichen.
- Entwicklung einer Applikation.

1.2 Aufbau dieser Arbeit

Die Arbeit ist folgendermaßen aufgebaut: In Kapitel 2 wird der Stand der Wissenschaft dargestellt. Dazu wird zuerst ein Überblick über Systeme zur Erkennung von grafischen Symbolen gegeben. Danach werden diverse Whiteboardsysteme beschrieben, von denen manche nur zur Digitalisierung der Inhalte des Whiteboards verwendet werden können, andere das Whiteboard als Benutzeroberfläche erweitern. Am Ende des Kapitels werden die Grundlagen der Graphentheorie sowie ein exaktes Graph-Matching Verfahren erläutert. Im darauf folgenden Kapitel 3 wird die Realisierung des interaktiven Whiteboardsystems, die dabei entstandenen Probleme und ihre Lösung beschrieben. Anschließend wird in Kapitel 4 die für das System entwickelte Applikation vorgestellt. Die durchgeführten Tests und ihrer Ergebnisse werden in Kapitel 5 präsentiert, bevor zum Schluss in Kapitel 6 die Ergebnisse zusammengefasst und einen Ausblick auf mögliche Erweiterungen gegeben werden.

Kapitel 2

Stand der Wissenschaft

In diesem Kapitel soll erläutert werden, welche Möglichkeiten zur Realisierung eines interaktiven Whiteboardsystems bestehen. Dafür wird zuerst ein allgemeiner Überblick über aktuelle Systeme zur Erkennung von grafischen Symbolen gegeben. Im darauf folgenden Abschnitt werden Systeme, die ein Whiteboard als Grundlage haben, vorgestellt. Dabei liegt der Fokus auf zwei Systemen, die das Whiteboard als Benutzeroberfläche verwenden. Zum Schluss wird eine Einführung in die Graphentheorie, sowie geeignete Algorithmen zur Zuweisung (Matching) von Graphen gegeben. Durch die Generierung und Zuweisung von Graphen können komplexe Objekte aus den einzelnen primitiven Objekte zusammengesetzt werden und erhalten somit ihre eigentliche Bedeutung.

2.1 Systeme zur Erkennung grafischer Symbole

Die Erkennung von grafischen Symbolen ist ein vielbehandeltes Thema in der Literatur. Einen guten Überblick über diese Systeme geben Chhabra und Tombre et. al [Chh98, KT96]. Symbolerkennung ist ein wichtiger Bestandteil vieler Grafikerkennugssysteme. Diese bestehen oft aus zwei Phasen. Bei der ersten Phase handelt es sich um die Vorverarbeitung des Bildes durch Operationen wie Rauschreduzierung, Binarisierung oder Ausdünnen. Dabei wird der Vordergrund (grafische Symbole) vom Hintergrund (beispielsweise ein Dokument oder Whiteboard) getrennt. Die zweite Phase ist dann die eigentliche Erkennung der Symbole. Dabei dienen Primitive wie Linien, Bögen, Regionen usw. als Eingabe. Manche Systeme verzichten auf die Vorverarbeitung und arbeiten direkt auf den Bildern.

Die grafische Symbolerkennung ist sehr anwendungsspezifisch (vgl. [Chh98]) und es gibt nur wenige Versuche sie domänunabhängig zu realisieren. Einer davon ist der von Messmer und Bunke [MB95]. Spezifische Beispiele dagegen gibt es viele. Ein Beispiel ist ein System zum automatischen Einlesen und Verstehen von

Schaltplänen [OKM+88]. Aber auch die Erkennung von Musiknoten [YPGD+96], Legenden von geografischen Karten [SS94] oder Architekturzeichnungen [LKML97] zählen dazu.

Einige Arbeiten wie die von Lopez et. al [LKML97] zur Erkennung von handgezeichneten Flurplänen benutzen in ihrem Ansatz attributiertes Graph-Matching zur Erkennung von Symbolen aus einem bekannten Modell. Die Knoten des Graphen repräsentieren die charakteristischen Punkte der Linien (Verbindungen oder Endpunkte der Linien) der zuvor vektorisierten Daten. Die Attribute der Knoten sind die Position, die Anzahl der Linien die, mit dem Knoten verbunden sind und die Winkel zwischen den einzelnen Linien. Die Kanten repräsentieren Segmente der Verbindungen an den charakteristischen Punkten und die Attribute der Kanten bestehen aus der Länge und den Parametern, die entweder die gerade Linie oder den Bogen der Linie beschreiben. Es können dabei nur Symbole erkannt werden, die zuvor im Modell festgelegt worden sind. Ah-Soon und Tombre [ST01] beschäftigen sich in ihrer Arbeit auch mit der Erkennung von Architekturzeichnungen. Jedoch beschreiben sie ein flexibleres Modell, welches inkrementell erstellt und aktualisiert werden kann.

Wieloch et. al [JKW07] beschreiben eine Methode des entwicklungsmäßigen visuellen Lernens zur Erkennung handgezeichneter Formen mit der Hilfe der Generative Genetische Programmierung.

2.2 Whiteboardsysteme

Es existieren viele Systeme, die versuchen die Möglichkeiten des Whiteboards mit der Unterstützung eines Computers zu erweitern. Li-wei He et. al [wHLZ02] stellen ein System zur automatischen Speicherung des Inhalte eines Whiteboards über eine Kamera und der Gespräche mit Hilfe eines Mikrofons. Dabei werden Veränderungen des Inhalts durch den Benutzer mittels Analysieren der Bildsequenzen in From von *Keyframes* gespeichert. So ist es später möglich den ganzen Inhalt des Whiteboards (dazu zählen auch schon wieder gelöschte Inhalte) über einem visuellen Index aufzurufen. Der Inhalt des Whiteboard wird dabei jedoch nicht näher interpretiert.

2.2.1 Automatische Schrifterkennung auf dem Whiteboard

In der Literatur lassen sich mehrere Ansätze finden, die sich mit der Erkennung handgeschriebener Texte auf dem Whiteboard beschäftigen. Einer davon ist der von Black und Jepson [BJ98]. Sie versuchen einen speziellen Stift mit sehr auffälliger Farbe zu verfolgen und anhand dessen Trajektorie die einzelnen Buchstaben

mittels *On-Line*-Erkennung[1] zu bestimmen. Es existieren aber auch Systeme, die normale Stifte verfolgen (siehe Munich und Perona [PM96]).

On-Line Systeme können nur an Stellen verwendet werden, bei denen der Stift immer sichtbar ist. Deshalb macht es sie für den Einsatz am Whiteboard ungeeignet, da der Stift oft von der schreibenden Person verdeckt wird. Anders sieht es bei *Off-Line* Systemen aus. Bei denen der Prozess des Schreibens nicht wichtig ist und somit vom Benutzer verdeckt werden kann. Die Erkennung des Textes findet anhand der geschriebenen Wörter statt. Das von Wienecke et. al [WFS05] beschriebene System extrahiert die schreibende Person in einer Sequenz von Aufnahmen und analysiert die auf dem Whiteboard beschriebenen Bereiche.

2.2.2 Whiteboard als Benutzeroberfläche

Im folgenden werden zwei Systeme vorgestellt, die einen Schritt weiter gehen und das Whiteboard in eine Benutzeroberfläche verwandeln. Die Interaktion zwischen Anwender und Computer findet in Form von Symbolen, die vom Anwender auf das Whiteboard gezeichnet werden, statt.

BrightBoard

Das System *BrightBoard* [SFR96] macht es möglich, die gezeichneten Bilder auf dem Whiteboard beispielsweise zu speichern, zu drucken oder zu versenden, indem bestimmte Markierungen auf das Board gemalt werden. Weiter lassen sich bestimme Bereiche selektieren oder umfangreichere Operationen wie Steuerung der Lichtschalter ausführen.

Die nötigen Arbeitsschritte des Systems werden von Robinson et. al [SFR96] wie folgt beschrieben:

1. Ansteuerung der Kamera, wenn sich keine Person vor dem Whiteboard befindet.

2. Vorverarbeitung der Kamerabilder mittels Schwellwertverfahren.

3. Erkennung der Markierungen.

4. Analyse der gefunden Markierungen.

5. Ausführung bestimmter Aktionen auf der Basis der Analyse.

[1] Dabei werden die Buchstaben durch die Bewegung des Stiftes erkannt, anders als bei der Off-Line-Erkennung, bei man die Informationen durch Bildverarbeitung erhält.

Das System nimmt jede halbe Sekunde ein Bild mit geringer Auflösung auf. Damit das System keine Bilder analysiert, auf dem sich Personen befinden, werden vorher definierte Bereiche des aufgenommen Bildes mit den äquivalenten Bereichen des vorherigen Bildes verglichen. Die Abweichungen dürfen einen gewissen Schwellwert nicht überschreiten. Dieser Schwellwert wurde bei der Kalibrierung des Systems ermittelt und basiert statistisch aus dem Rauschen des Pixel. Somit werden rauschanfälligere Regionen (z. B. dunklere Bereiche oder flackerndere Monitore) kompensiert.

Die Vorverarbeitung basiert auf einem Schwellwertalgorithmus von Wellner [Wel93]. Dieses adaptive Verfahren wurde für echtzeitfähige Dokumentenerkennung entworfen und trennt die Zeichnungen vom Hintergrund. In einem Durchgang werden die Pixelreihen durchlaufen, wobei sich nach jeder Reihe die Richtung ändert. Dabei wird laufend ein Mittelwert der Pixel bestimmt. Jeder Pixel, der signifikant dunkler als der Durchschnittswert ist, wird als Vordergrund gewertet, die restlichen gehören zum Hintergrund. Dieses einfache Verfahren liefert gute Ergebnisse, auch wenn sich verdunkelte und erhellte Stellen im Bild befinden. Dies ist bei Videoaufnahmen des Whiteboards üblich.

Im nächsten Schritt werden alle dem Vordergrund zugeordnete Bereiche untersucht. Damit nicht jeder Pixel einzeln überprüft werden muss und gleichzeitig Rauschen gefiltert werden kann, wird nur jeder dritte oder vierte Pixel in jede Richtung untersucht. Wird ein Pixel gefunden, wird die dazugehörig Region mittels *Flutfüllung* gesucht. Dabei werden alle Pixel als untersucht markiert und müssen später nicht mehr überprüft werden. Regionen, deren Größe unter- oder oberhalb eines bestimmten Wertes liegen, werden nicht weiter betrachtet, da es sich in den meisten Fällen nicht um Regionen von Zeichnungen handelt. Für jede Region werden Statistiken gesammelt, die beispielsweise die Anzahl der Pixel, die Boundingbox, die Verteilung der Pixel in jede Richtung usw. beinhaltet.

Aus den statistischen Daten der Regionen werden Merkmalsvektoren errechnet. Bei einem Merkmalsvektor handelt es sich um eine Menge realer Zahlen, die charakteristische Eigenschaften einer Region repräsentieren. Man kann sie sich als Koordinaten in einem n-dimensionalen Raum vorstellen. Ein Wert z. B. steht für das Verhältnis der Anzahl schwarzer Pixel mit weißen Pixel darüber zu der Anzahl schwarzer Pixel mit weißen Pixel rechts daneben. Dadurch kann man grob das Verhältnis von horizontalen gegenüber vertikalen Linien bestimmen. Diese Merkmalsvektoren werden nun mit Prototypen verglichen. Die Prototypen wurden durch eine Menge an Trainingsbilder erstellt. Zur Klassifizierung wird der Prototyp im n-dimensionalen Raum gesucht, der am nächsten an der Region liegt. Somit entspräche die Region dem selben Typ des Prototypen. Dabei existieren vier Einschränkungen (s. [SFR96]):

1. Die Skalierung der Dimensionen stehen nicht in Beziehung zueinander und so kann eine vermeintlich weitere Distanz in einer Dimension signifikanter als ein näher erscheinende Distanz einer anderen Dimension sein. Daher wurde die skaleninvariante *Mahalanobis-Distanz* [Mah36] anstelle der Euklidischen-Distanz verwendet.

2. Es fehlt eine Verwerfungsbedingung. Das bedeutet alle Regionen können immer zugewiesen werden, was meistens unwahrscheinlich ist. Mit Hilfe der Mahalanobis-Distanz kann ein Mindestabstand überprüft werden. Dadurch können Zuweisungen auch verworfen werden.

3. In vielen Dimensionen überlappen Gruppen verschiedener Symbole erheblich. Die Buchstaben B und R beispielsweise sind in vielen Fällen gleich und so ist es möglich, dass ein B näher dem Prototypen des Buchstaben R ist. Um dies zu verhindern wird ein *k-nearest-neighbours* Algorithmus verwendet. Dieser schreibt fest, dass X als Y klassifiziert wird, wenn eine bestimmte Anzahl der k nächsten Prototypen das Symbol Y sind. Bei dem BrightBoard System wird X als R klassifiziert, wenn drei der fünf nächsten Prototypen das Symbol R sind.

4. Da die Werte für manche Dimensionen weniger zuverlässig sind als andere, wurden Gewichtungen für einzelne Dimensionen empirisch ermittelt. Alle Distanzen einer Dimension werden mit den entsprechenden Gewichten multipliziert.

Die Entwickler von *BrightBoard* wollten nicht jedes auszuführende Kommando statisch implementieren und entschieden sich die Analyse der Symbole mit der logischen Programmiersprache *Prolog* zu realisieren. Für jede gefundene Region wird in einer Prologdatenbank eine neue Regel mit den Informationen der Region angelegt. Diese werden im letzten Schritt mit Hilfe einer "Kommandodatei" in UNIX Befehle umgesetzt. Wie genau diese Regeln aufgebaut sind und die Ausführung funktioniert, findet man unter [Wel93]. Danach wird wieder mit dem ersten Schritt begonnen. Dabei werden die gezeichneten Inhalte und Kommandos des Anwenders abgearbeitet, ohne jedoch diese beispielsweise mit der Hilfe eines Projektors zu erweitern und dadurch dem Anwender ein direktes Feedback auf dem Whiteboard zu geben.

ZombieBoard

Ein ähnliches System ist das *ZombieBoard*, vorgestellt von Saund [Sau99]. Hier wird ein hochauflösendes Bild des Whiteboards durch Zusammensetzen vieler überlappender Bilder erzeugt. Dies geschieht durch die automatische Steuerung des

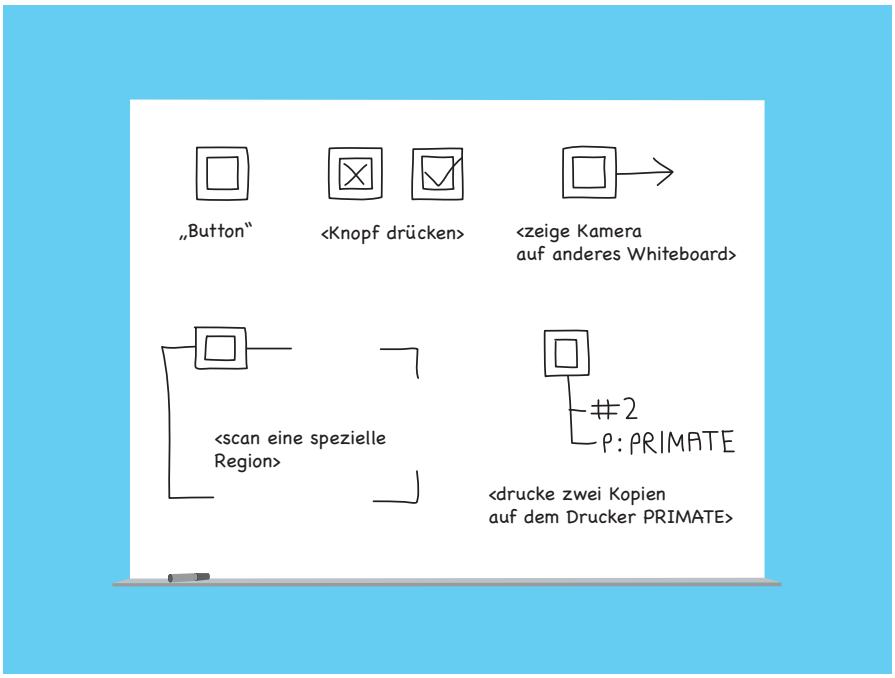


Abbildung 2.1: Schematische Komandos des ZombieBoards nach [Sau99].

Zooms, der Neigung und des Kameraschwenks. Die Interaktion mit dem Board findet über eine schematische Benutzeroberfläche statt. Dabei zeichnet der Benutzer einen Knopf auf das Whiteboard und erlangt somit die Aufmerksamkeit, dass ein Kommando bereit zur Abarbeitung ist. Ein Knopf besteht aus zwei verschachtelten Rechtecken. Der Knopf kann durch Zeichnen eines "X" oder eines Hakens in das innere Reckteck gedrückt bzw. ausgewählt werden. Wird ein Pfeil an das äußere Rechteck gemalt, bekommt ein anderes Whiteboard, welches sich in der Richtung des Pfeils befinden muss, die Aufmerksamkeit der Kamera. Weiter können bestimmte Bereiche des Whiteboards gescannt oder symbolische Informationen wie beispielsweise die Anzahl der gedruckten Kopien, den verwendeten Drucker usw. angegeben werden. Abbildung 2.1 beschreibt ein Anwendungsszenario. Wird ein bestimmter Bereich nicht per Kommando gezoomt betrachtet, findet die Detektion und Interpretation der schematischen Kommandos mit normaler Auflösung statt. Wie auch bei den zuvor beschriebenen Verfahren wird das Whiteboard nur als Eingabesystem und nicht auch als Ausgabesystem verwendet.

2.2.3 Whiteboard als Projektionsfläche

Durch die bisher vorgestellten Systeme werden die Anwendungsmöglichkeiten des Whiteboards durch Computerunterstützung erweitert. Sei es durch einfache Digi-

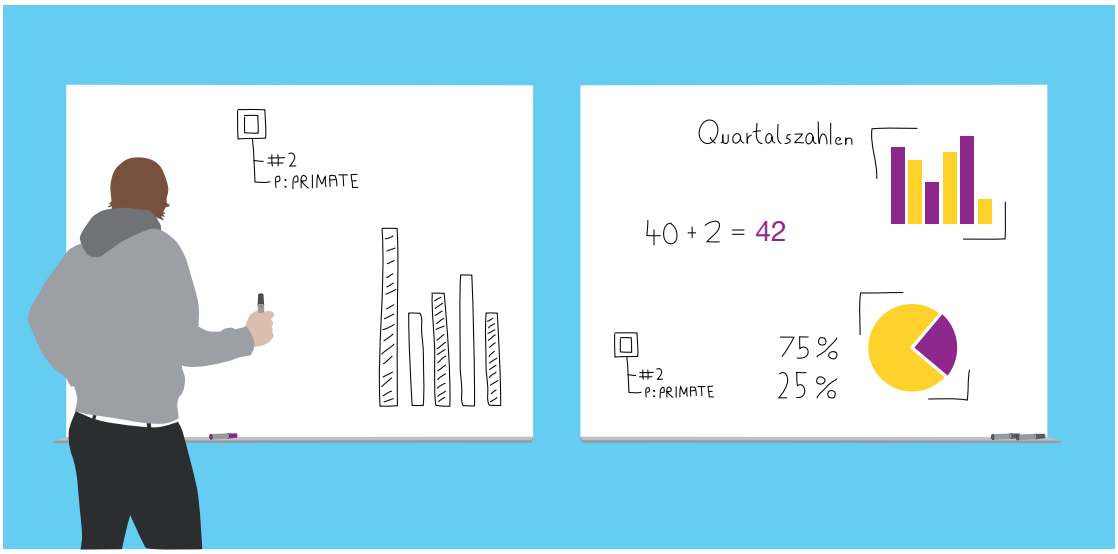


Abbildung 2.2: Links das ZombieBoard und rechts die Erweiterung mit Projektor.

talisierung der Inhalte oder als Eingabegerät zur Steuerung eines Computers. Doch leider hat das Whiteboard dabei nur die Rolle eines einfachen Eingabegerätes und der Anwender bekommt nicht wie bei einem Computermonitor ein direktes Feedback. Dieses Feedback ist nicht immer notwendig. Wird z. B. die Helligkeit des Raumes über das Whiteboard gesteuert, so erkennt man schnell, ob es dunkler oder heller wird. Möchte man aber die Inhalte des Whitboards durch andere Inhalte (z. B. Grafiken, Videos usw.) erweitern, muss man das Whitboard auch als Bildschirm verwenden. Dies eröffnet einem ganz neue Möglichkeiten der Interaktion zwischen Mensch und Computer (s. Abbildung 2.2). So könnte man beispielsweise die am Whiteboard berechneten Statistiken während eines Meetings direkt in einem Diagramm auf dem Whiteboard anzeigen lassen. Damit man Whiteboardsysteme mit der Funktion des Ausgabegerätes bereichern kann, muss man ihre Systemkonfiguration um einen Projektor erweitern. Mit Hilfe des Projektors lässt sich das Bild auf das Whiteboard projizieren.

2.3 Graph-Matching

Graphen sind sehr vielseitig und werden in verschiedensten Bereichen als Repräsentation struktureller Informationen verwendet. Dazu zählen die 2D und 3D Bildanalyse, Mustererkennung, Biomedizinische Anwendungen, Videoanalyse, Netzwerke und viele mehr. Bei der Bildanalyse werden üblicherweise Regionen des Bildes als

Knoten eines Graphen und die strukturellen Beziehungen der einzelnen Regionen als Kanten zwischen den Knoten repräsentiert.

Zur Erkennung der komplexen Objekte auf dem Whiteboard wird ein exaktes Graph-Matching Verfahren verwendet. Bevor dieses Verfahren vorgestellt wird, werden die Grundlagen der Graphentheorie erläutert.

2.3.1 Grundbegriffe der Graphentheorie

Ein ungerichteter Graph $G = (V, E)$ (s. Abbildung 2.3) ist durch ein Tupel bestehend aus einer Knotenmenge V und einer Kantenmenge E gegeben, dabei sind zwei Knoten $u, v \in V$ mit einer Kante $e \in E$ verbunden. Im folgenden wird die Notation nach Tittmann [Tit03] verwendet, bei der eine Kante als $e = \{u, v\}$ beschrieben wird. Die Knoten u und v werden dann als Endknoten der Kante e bezeichnet und sind *adjazent* (benachbart) in G. Ist $v \in e$, so wird v auch als inzident zu e bezeichnet. Von einem gerichteten Graphen spricht man, wenn jeder Kante $e = (u, v)$ eindeutig ein geordnetes Paar (u, v) von Knoten zugeordnet wird. Zur Verdeutlichung der Ordnung des Knotenpaares wird bei einem gerichteten Graphen das Knotenpaar als (u, v) statt $\{u, v\}$ geschrieben. Somit ist also (u, v) unterschiedlich zu (v, u). In diesem Fall ($e = (u, v)$) bezeichnet man den Knoten u als direkten Vorgänger von v und den Knoten v als direkten Nachfolger von u (vgl. [AHU74]).

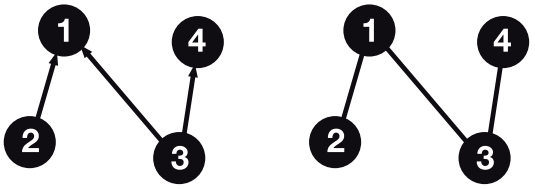


Abbildung 2.3: Gerichteter und ungerichteter Graph

$V(G)$ bzw. $E(V)$ beschreibt die Zugehörigkeit einer Knotenmenge V und einer Kantenmenge E zu einem Graphen G. Der Ausgangsgrad $deg^+(u)$ eines Knotens u ist gleich der Anzahl seiner direkten Nachfolger, d.h.

$$deg^+(u) = |\{v|(u, v) \in E\}|. \tag{2.1}$$

Der Eingangsgrad $deg^-(u)$ eines Knotens u ist gleich der Anzahl seiner direkten Vorgänger, d.h.

$$deg^-(u) = |\{w|(w, u) \in E\}|. \tag{2.2}$$

Ist $deg^-(v) = deg^+(v) = 0$, so spricht man von einem isolierten Knoten (vgl. [Tit03]). Ein isolierter Knoten besitzt somit keine Nachbarknoten. Lässt sich die Menge der Kanten E eines Graphen in zwei disjunkte Teilmengen E_1 und E_2

zerlegen, sodass weder Kanten aus E_1, noch Kanten aus E_2 untereinander adjazent sind, so heißt der Graph *bipartit* (vgl. [Tur96]), d.h.

$$\forall e = (u, v) \in E : u \in E_1; v \in E_2 : E_1 \cap E_2 = \varnothing; E_1 \cup E_2 = E. \tag{2.3}$$

Auf den ersten Blick erscheint schnell identifizierbar zu sein, ob zwei Graphen gleich sind. Doch oft sehen zwei Graphen nicht gleich aus, haben aber die gleiche Struktur. Um diese Gleichheit der Struktur zu beschreiben, verwendet man den Begriff der Isomorphie.

Graphisomorphismus

Zwei Graphen $G_m = (V_m, E_m)$ und $G_d = (V_d, E_d)$ sind isomorph zueinander, wenn eine bijektive (injektive und surjektive) Abbildung

$$M : V_m \to V_d \tag{2.4}$$

existiert, mit

$$(u, v) \in E_m \Leftrightarrow (M(u), M(v)) \in E_d. \tag{2.5}$$

Einfacher gesagt: Geht G_d durch Umnummerierung der Knoten aus G_m hervor, dann sind sie isomorph zueinander. Da es oft nicht leicht ist die Isomorphie zweier Graphen zu erkennen, lohnt sich ein Blick auf die Grapheninvarianten. Grapheninvarianten sind strukturelle Eigenschaften der Graphen, die auch nach dem Übergang zu einem isomorphen Graphen erhalten bleiben. Sind diese nicht gleich, dann sind auch die Graphen nicht gleich. Zu den Grapheninvarianten gehören unter anderem die Anzahl der Knoten und Kanten, sowie der Maximalgrad.

Subgraphisomorphismus

Ein Graph $G_m = (V_m, E_m)$ ist Subgraph eines Graphen $G_d = (V_d, E_d)$, wenn $E_m \subseteq E_d$ und $V_m \subseteq V_m$ gilt (s. Abbildung 2.4). Man spricht von Subgraphisomorphismus, wenn es eine injektive Abbildung

$$M : V_m \to V_d \tag{2.6}$$

existiert, mit

$$(u, v) \in E_m \Rightarrow (M(u), M(v)) \in E_d. \tag{2.7}$$

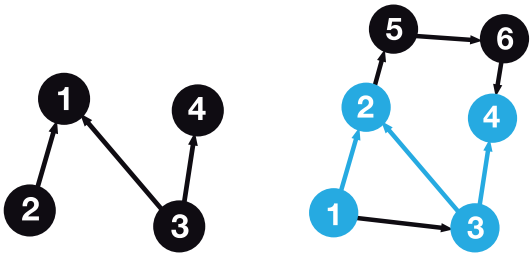


Abbildung 2.4: Graph und Subgraph des Graphen

2.3.2 Graph-Matching Algorithmus

Um das Problem des Graph-Matchings an einem spezielleren Beispiels etwas deutlicher zu machen, betrachtet man zwei verschiedene Graphen. Der erste Graph enthält die Informationen des Bildes und wird im Folgenden als Datengraph (G_d) bezeichnet. Beim zweiten Graphen handelt es sich um den Modellgraphen (G_m), dieser beschreibt das Modell, mit dessen Hilfe Bereiche im Bild erkannt werden sollen (vgl. [Ben02]). Eines der Hauptprobleme dieser Repräsentation fällt unter den Begriff Graph-Matching. Darunter versteht man die Zuweisung von Graphen. Sind zwei Graphen gleich oder ist ein Graph in einem anderen Graphen enthalten? Die Suche nach einer guten Korrespondenz zwischen Modell- und Datengraph ist meistens die Suche nach einer Graphisomorphie. Einer der ersten Ansätze von [Ull76] löst das Graph-Matching Problem mit Subgraphisomorphie. Basierend auf einem *backtracking* Verfahren mit einer vorausschauenden Funktion wird der Suchraum effektiv verkleinert und so Arbeit eingespart. Dabei muss der Modellgraph in exakter struktureller Form dem Datengraph gleichen, bzw. in ihm enthalten sein. Leichte strukturelle Unterschiede vom Modellgraphen zum Datengraphen werden nicht verziehen. Deswegen spricht man auch von einem exakten Graph-Matching verfahren. Da es nicht immer gelingt, Bilder in geeignete Segmente einzuteilen, sodass eine Subgraphisomorphie zwischen dem Bild und dem Modell besteht, existieren neben diesen exakten Methoden noch inexakte Graph-Matching Verfahren. Gerade wenn sich der Modell- und der Datengraph in verschieden Domänen befinden, wird es fast unmöglich exakte Ergebnisse zu finden. So kommen diese Problemarten oft in der modellbasierten Bilderkennung vor. Beispielsweise bei der Erkennung von Gehirnbildern, bei der das Modell (z.B. anatomischer Atlas) üblicher Weise aus Regionen besteht, die in einem gesunden Gehirn vorkommen. Die Bildinformationen der Patienten dagegen kommen aus Magnetresonanzbildern. Ein Ansatz zur Lösung dieses Problems liefert Melnik [MGMR02], der versucht die Ähnlichkeit zweier Graphen mit der Hilfe einer Fixpunktberechnung zu bestimmen. Zwei Elemente eines Modells sind dabei ähnlich, wenn ihre benachbarten Elemente ähnlich sind. Im weiteren Verlauf dieser Arbeit liegt der Fokus auf den exakten Verfahren.

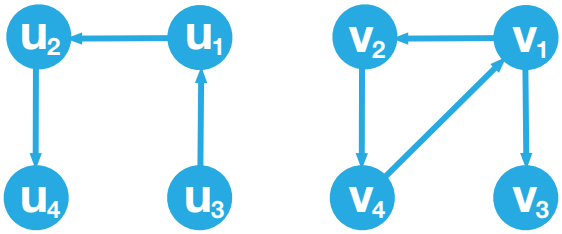
Wie schon in Kapitel 2.3.1 beschrieben, wird mit der Graph- bzw. Subgraphisomorphie versucht die Gleichheit zwischen Modellgraph $G_m = (V_m, E_m)$ und Datengraph $G_d = (V_d, E_d)$ mit der Hilfe einer Abbildungsfunktion M zu bestimmen, welche korrespondierende Knoten aus dem Modellgraphen auf dem Datengraphen und umgekehrt abbildet. M wird als Menge geordneter Paare (u, v) beschrieben, mit $u \in G_m$ und $v \in G_d$, wobei jedes Paar die Abbildung eines Knoten u aus G_m auf einem Knoten v aus G_d repräsentiert (vgl. [CFSV99]).

$$M = \{(u, v) \in V_m \times V_d \mid u \text{ wird auf } v \text{ abgebildet}\} \tag{2.8}$$

Des weiteren wird eine *State Space Representation* (kurz SSR) eingeführt, um den Machtingprozess besser beschreiben zu können. Dabei steht jeder Zustand s für eine partielle Abbildung $M(s)$ mit $M(s) \subseteq M$ und enthält beispielsweise nur wenige Komponenten von M (vgl. Abbildung 2.5). Eine partielle Abbildung identifiziert zwei Subgraphen aus dem Modellgraph und dem Datengraph, genannt $G_m(s)$ und $G_d(s)$ nur durch Auswahl der Knoten, die in den Komponenten von $M(s)$ beinhaltet sind und den Kanten, die diese Knoten mit einander verbinden. Die Projektion von $M(s)$ auf V_m wird mit $M_m(s)$ und entsprechend von $M(s)$ auf V_d mit $M_d(s)$ bezeichnet, während die Menge der Kanten von $G_m(s)$ und $G_d(s)$ mit $E_m(s)$ bzw. $E_d(s)$ bezeichnet wird.

Ein Übergang zwischen zwei Zuständen in der gewählten SSR findet statt, wenn ein neues Paar zugewiesener Knoten hinzugefügt wird. Im Prinzip ist der Ansatz alle partiellen Abbildungen, die existieren, mittels Brute-Force-Methode auszuprobieren und alle mit der erfolgreichen Zuweisung auszusuchen. Es existieren dabei jedoch viele Wege vom initialen Zustand s_0 zum gewünschten Ergebniszustand, die durch geschickte Voraussicht hätten eingespart werden können. Um Wege, die nicht zum Ziel führen, zu umgehen, werden für korrespondierende partielle Abbildungen Bedingungen der Kohärenz aufgestellt. Diese Kohärenzbedingungen sind abhängig von der gewählten Abbildungsvariante. Das bedeutet, wenn man z. B. nach Subgraphisomorphie sucht, dann ist es notwendig, dass auch die partiellen Abbildung der korrespondierenden Teilgraphen isomorph zueinander sind. Wird jetzt die partielle Abbildung um ein Knotenpaar erweitert und der dadurch resultierende Zustand erfüllt die Zugehörigkeitsbedingung nicht mehr, können alle weiteren Pfade des Weges weggelassen werden, da dieser Weg ab dieser Stelle nicht mehr zum Ziel führen wird.

Um Wege die nicht zum Ziel führen frühzeitig erkennen zu können bzw. um den Suchraum entschieden zu verkleinern, wird eine Vorhersagekriterium eingeführt, das prüft, ob der Zustand s nach einer bestimmten Anzahl an Schritten einen kohärenten Nachfolger besitzt. Diese Kriterium wird von Cordella et. al [CFSV04] als eine bool'sche Ausführbarkeitsfunktion $F(s, u, v)$ beschrieben. Da diese Ausführbarkeitsfunktion nur abhängig von der Struktur der Graphen ist, die



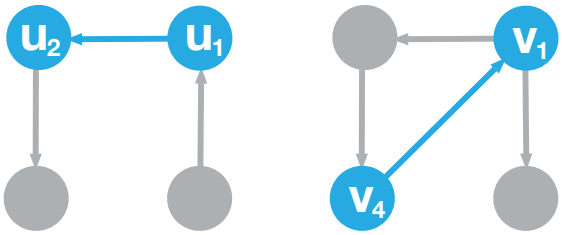
(a)

$$M = \{(u_1, v_4), (u_2, v_1), (u_3, v_2), (u_4, v_3)\}$$

(b)

$$M(s) = \{(u_1, v_4), (u_2, v_1)\}\ M_m(s) = \{u_1, u_2\}\ M_d(s) = \{v_4, v_1\}$$

(c)



(d)

Abbildung 2.5: (a) Model- und Datengraph, (b) sowie die einzig mögliche Abbildung der beiden Graphen, (c) eine partielle Abbildungslösung der Graphen $M(s)$ und (d) die korrespondierenden Subgraphen $G_m(s)$ und $G_d(s)$

Kanten und Knoten des Graphen aber auch Attribute besitzen können, wird die Ausführbarkeitsfunktion wie folgt definiert:

$$F(s, u, v) = F_{\text{syn}}(s, u, v) \wedge F_{\text{sem}}(s, u, v), \tag{2.9}$$

wobei F_{syn} (syntaktische Ausführbarkeit) abhängig von der Struktur des Graphen ist und F_{sem} (semantische Ausführbarkeit) abhängig von den Attributen.

Der Algorithmus

Algorithmus 2.1 beschreibt das Verfahren. Im initiale Zustand s_0 besitzt die Abbildungsfunktion keine Komponente, so ist $M(s_0) = \emptyset$. Für jeden zwischenzeitlichen Zustand s berechnet der Algorithmus die Menge $P(s)$ der Knotenpaare die als Kandidaten zur Aufnahme in den aktuellen Zustand s infrage kommen. Die Berechnung von $P(s)$ wird später erklärt. Für jedes Paar $(u, v) \in P(s)$ werden die Ausführbarkeitsregeln ausgewertet. Ist $F(s, u, v)$ wahr, wird der nachfolgende Zustand $s' = s \cup (u, v)$ berechnet. Dieser Vorgang wird rekursiv mit s' fortgeführt.

```
Algorithmus :Match(s)
Input : an intermediate state s; the initial state s_0 has M(s_0) = ∅
Output : the mappings between the two graphs
if M(s) covers all the nodes of G_d then
    return M(s)
else
    Compute the set P(s) of the pairs candidate for inclusion in M(s);
    foreach (u, v) ∈ P(s) do
        if F(s, u, v) then
            Compute the state s' obtained by adding (u, v) to M(s);
            Match(s');
        end
    end
    Restore data structures;
end
```

Algorithmus 2.1 : Der VF Graph-Matching Algorithmus nach Cordella et. al [CFSV04].

Das Erforschen des Graphen der SSR gleicht einer Tiefensuche, dadurch kann ein Zustand über verschiedene Wege erreicht werden. Um zu verhindern, dass der Algorithmus nutzlose und schon erstellte Zustände generiert, wird eine spezielle Prozedur zur Erstellung eines nachfolgenden Knotens verwendet. Eine beliebige totale Ordnungsrelation $\prec$ ist auf den Knoten von G_d definiert, die zugehörig zu $P(s)$ sind. Da die Reihenfolge in der Knoten zur partiellen Abbildung $M(s)$ hinzugefügt werden nicht den resultierenden Zustand beeinträchtigt, wird jedes Paar (u_j, v_j) aus $P(s)$ ignoriert, wenn die Menge schon einen Knoten $v_k \prec v_j$ enthält. Diese einfache Strategie erlaubt es, dass jeder Zustand nur einmal erstellt wird (vgl. [CFSV04]).

Berechnung der möglichen Kandidaten für $M(s)$

Als $P(s)$ wird die Menge der möglichen Kandidaten für den Einschluss in $M(s)$ bezeichnet. Die Menge $P(s)$ erhält man unter Berücksichtigung der Menge der Knoten, die direkt mit $G_m(s)$ und $G_d(s)$ verbunden sind. Gegeben sei ein Graph G und einer seiner Knoten u. Die Vorgänger von u ($Pred(G, u)$) wird als Menge der Startknoten bezeichnet, von denen aus eine Kante beginnt und in dem Knoten u endet. Ähnlich wird die Menge der Nachfolger von u ($Succ(G, u)$) der Endknoten bezeichnet, deren Kante bei dem Knoten u startet. Weiter wird von Cordella et. al [CFSV04] die Menge der Knoten des Modellgraphen als *out-terminal* Menge $T_m^{out}(s)$ definiert, die nicht in $M_m(s)$ enthalten sind, aber zu der Menge der Nachfolger

eines Knotens aus $M_m(s)$ gehören. Die Menge der Knoten des Modellgraphen, die nicht in $M_m(s)$ enthalten sind, aber Vorgänger von einem Knoten aus $M_m(s)$ sind, werden als *in-terminal* Menge $T_m^{in}(s)$ definiert. $T_d^{out}(s)$ und $T_d^{in}(s)$ sind analog dazu. Die Menge $P(s)$ wird nach [CFSV04] auf folgende Weise berechnet:

$$P(s) = \begin{cases} T_m^{out}(s) \times T_d^{out}(s), & T_m^{out}(s) \neq \varnothing \quad \wedge \quad T_d^{out}(s) \neq \varnothing \\ T_m^{in}(s) \times T_d^{in}(s), & \begin{array}{l} T_m^{out}(s) = \varnothing \quad \wedge \quad T_d^{out}(s) = \varnothing \\ \wedge \quad T_m^{in}(s) \neq \varnothing \quad \wedge \quad T_d^{in}(s) \neq \varnothing \end{array} \\ (V_m - M_m(s)) \times (V_d - M_d(s)), & \begin{array}{l} T_m^{out}(s) = \varnothing \quad \wedge \quad T_d^{out}(s) = \varnothing \\ \wedge \quad T_m^{in}(s) = \varnothing \quad \wedge \quad T_d^{in}(s) = \varnothing \end{array} \end{cases} \tag{2.10}$$

In den übrigen Fällen, bei denen nur eine der beiden Mengen entweder der *out-terminal* Mengen oder der *in-terminal* leer ist, kann demonstriert werden, dass der Zustand s nicht Teil des Matchings seien kann und wird nicht weiter untersucht (vgl. [CFSV04]).

Die fünf Ausführbarkeitsregeln

Die fünf Ausführbarkeitsregeln sind: R_{pred}, R_{succ}, R_{in}, R_{out} und R_{new}. Die ersten beiden Regeln überprüfen die Stimmigkeit der partiellen Abbildung $M(s')$, die aus der Erweiterung der aktuellen partiellen Abbildung $M(s)$ mit dem auserwählten Kandidatenpaar (u, v) besteht. Die restlichen drei Regeln sind zum Verkleinern des Suchraums gedacht. Genauer gesagt führen R_{in} und R_{out} im Suchprozess einen Lookahead mit dem Wert 1 durch und R_{new} einen Lookahead mit dem Wert 2. Weiter wird die Ausführbarkeitsfunktion wie folgt beschrieben:

$$F_{\text{syn}}(s, u, v) = R_{\text{pred}} \wedge R_{\text{succ}} \wedge R_{\text{in}} \wedge R_{\text{out}} \wedge R_{\text{new}}. \tag{2.11}$$

Gegeben ist ein Graph $G = (V, E)$, ein Knoten $u \in V$ und die Mengen der Vorgänger $Pred(G, u)$ und Nachfolger $Succ(G, u)$. Des weiteren werden die Mengen $T_m(s) = T_m^{in}(s) \cup T_m^{out}(s)$ und $\tilde{N}_m(s) = N_m - M_m(s) - T_m(s)$ verwendet. Die Mengen T_d und $\tilde{N}_d$ sind analog dazu. Die formalen Definitionen der fünf Regeln im Falle der Subgraphisomorphie nach [CFSV04] lauten:

$$\begin{array}{l} R_{\text{pred}}(s, u, v) \Leftrightarrow \\ (\forall u' \in M_m(s) \cap Pred(G_m, u) \qquad \exists v' \in Pred(G_d, v) | (u', v') \in M(s)) \wedge \\ (\forall v' \in M_d(s) \cap Pred(G_d, v) \qquad \exists u' \in Pred(G_m, u) | (u', v') \in M(s)), \end{array} \tag{2.12}$$

$$
\begin{aligned}
&R_{\text{succ}}(s,u,v) \Leftrightarrow \\
&(\forall u' \in M_m(s) \cap Succ(G_m,u) \quad \exists v' \in Pred(G_d,v) | (u',v') \in M(s)) \wedge \\
&(\forall v' \in M_d(s) \cap Succ(G_d,v) \quad \exists u' \in Pred(G_m,u) | (u',v') \in M(s)),
\end{aligned} \tag{2.13}
$$

$$
\begin{aligned}
&R_{\text{in}}(s,u,v) \Leftrightarrow \\
&|(Succ(G_m,u) \cap T_m^{in}(s))| \quad \geq \quad |(Succ(G_d,v) \cap T_d^{in}(s))| \wedge \\
&|(Pred(G_m,u) \cap T_m^{in}(s))| \quad \geq \quad |(Pred(G_d,v) \cap T_d^{in}(s))|,
\end{aligned} \tag{2.14}
$$

$$
\begin{aligned}
&R_{\text{out}}(s,u,v) \Leftrightarrow \\
&|(Succ(G_m,u) \cap T_m^{out}(s))| \quad \geq \quad |(Succ(G_d,v) \cap T_d^{out}(s))| \wedge \\
&|(Pred(G_m,u) \cap T_m^{out}(s))| \quad \geq \quad |(Pred(G_d,v) \cap T_d^{out}(s))|,
\end{aligned} \tag{2.15}
$$

$$
\begin{aligned}
&R_{\text{new}}(s,u,v) \Leftrightarrow \\
&|(\tilde{N}_m(s) \cap Pred(G_m,u))| \quad \geq \quad |(\tilde{N}_d(s) \cap Pred(G_d,u))| \wedge \\
&|(\tilde{N}_m(s) \cap Succ(G_m,u))| \quad \geq \quad |(\tilde{N}_d(s) \cap Succ(G_d,u))|.
\end{aligned} \tag{2.16}
$$

Unter Berücksichtigung des Graphisomorphismus anstelle des Subgraphisomorphismus bleiben die Regeln R_{succ} und R_{pred} gleich, lediglich bei den anderen Regeln R_{in}, R_{out}, R_{new} muss der Operator $\geq$ mit $=$ ersetzt werden.

Semantische Ausführbarkeit

Mit Hilfe der semantischen Ausführbarkeitsregel werden die Attribute der Knoten und Kanten überprüft. Abhängig von symbolischen oder numerischen Attributen geschieht dies auf zwei Wegen. Für symbolische Attribute, die sich in gewisser Fällen bedingt durch die Quantisierung von numerischen Werten ableiten lassen, wird eine Kompatibilitätsrelation $\approx$ definiert, da für manche Programme eine Gleichheit nicht tolerant genug wäre. Jedes mal, wenn die Ausführbarkeit eines neuen Paares getestet wird, werden Knoten und Kanten die hinzugefügt werden sollen auf semantische Kompatibilität getestet. Formal definiert bedeutet das:

$$
\begin{aligned}
F_{\text{sem}}(s,u,v) \Leftrightarrow \quad & u \approx v \\
& \wedge \forall (u',v') \in M(s), (u,u') \in E_m \Rightarrow (u,u') \approx (v,v') \\
& \wedge \forall (u',v') \in M(s), (u',u) \in E_m \Rightarrow (u',u) \approx (v',v).
\end{aligned} \tag{2.17}
$$

Diese Informationen können für die numerischen Attribute auf zwei Wegen ausgenutzt werden. Zum einen wird eine Kompatibilitätsrelation der zu vergleichenden Attribute auf Basis eines Schwellwertes der absoluten Distanz definiert,

welche zu einer semantischen Ausführbarkeitsfunktion analog zu Abbildung 2.17 führt. Zum anderen wird eine Kostenfunktion eingeführt die eine quantitative Evaluation der Unähnlichkeit zwischen zwei Knoten bzw. Kanten erstellt. Der Algorithmus speichert während seiner Erforschung des Suchraumes nur das Matching mit den minimalen Gesamtkosten. Die Kosten werden aktuell für jeden Zustand s berechnet und bestehen aus der Summe der Kosten der vorherigen Zustände und der neuhinzugefügten Knoten und Kanten. Da die Kosten nicht negative sein können, sind die Gesamtkosten eines Zustands größer oder gleich seiner Vorgänger. Dadurch können alle Zustände aussortiert werden, deren Kosten größer als der bisher erreichte beste finale Zustand sind. Dies führt zu einer weiteren Verkleinerung des Suchraums.

Kapitel 3

Realisierung eines interaktiven Whiteboardsystem

In diesem Kapitel wird die Realisierung eines interaktiven Whiteboardsystem beschrieben. Ähnlich wie in Kapitel 2.2.2 dargestellt, soll die Interaktion zwischen Anwender und Computer durch das Zeichnen von Symbolen auf dem Whiteboard stattfinden. Dabei wird das Whiteboard jedoch nicht nur als einfache Eingabequelle dienen, sondern zeitgleich auch als Bildschirm um so die Interaktion visuell zu erweitern. Die von Hand gezeichneten Symbole werden von einer Kamera aufgenommen, vom Computer verstanden und in passender Form (z. B. als Animation) wieder auf das Whiteboard projiziert. Anders als in den beiden zuvor beschriebenen Systemen sollen diese Symbole aus einfachen Primitiven bestehen, die zu komplexeren Objekten zusammen gesetzt werden. Diese komplexeren Objekte sollen durch Abgleichen mit einer Wissensbasis mittels Graph-Matching Verfahren wiedergefunden werden.

Als erstes wird der Aufbau des Systems, sowie die verwendete Hardware beschrieben, gefolgt von der Kalibrierung der Kamera mit dem Projektor. Danach werden alle notwendigen Vorverarbeitungsschritte für die drauf folgende Objekterkennung erläutert. Wie aus den einzelnen Objekten ein Graph generiert wird, der mit den Modellgraphen aus der Wissensbasis verglichen werden kann, steht in den letzen beiden Abschnitten.

3.1 Systemaufbau

Der Aufbau ist ohne großen Aufwand (s. Abbildung 3.1) zu realisieren. Benötigt wird ein Computer, ein Projektor, eine Kamera und ein handelsübliches Whiteboard. Der Projektor muss von Hand auf das Whiteboard ausgerichtet sein, so dass ein sauberes Bild direkt auf das Whiteboard projiziert wird. Das Bild des

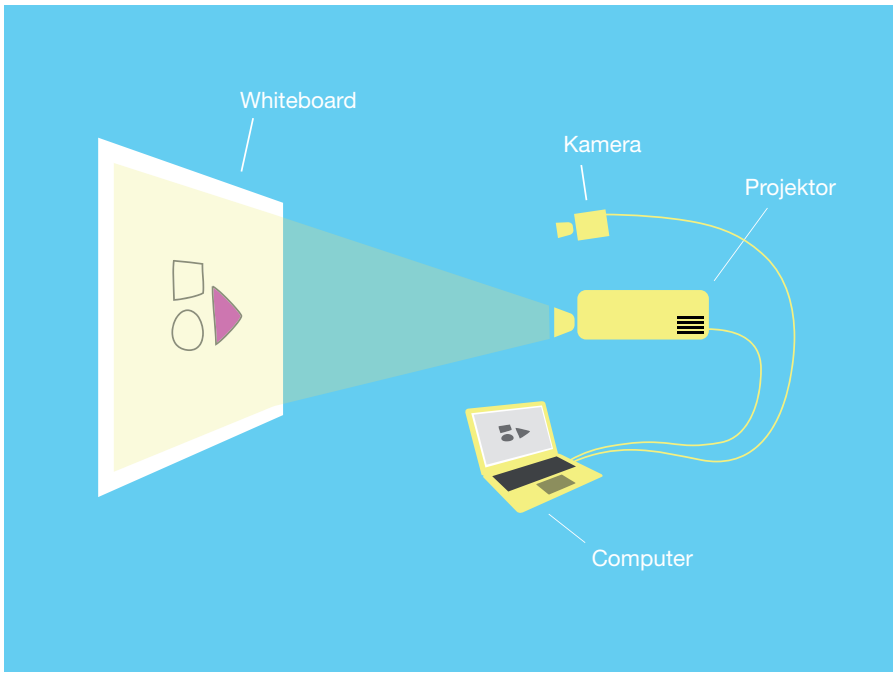


Abbildung 3.1: Systemaufbau: Projektor und Kamera sind mit dem Computer verbunden und auf das Whiteboard gerichtet.

Projektors sollte am Besten nicht über die Kanten des Whiteboards hinausragen. Danach wird die Kamera so ausgerichtet, dass sie das ganze Bild des Projektors einfängt. Generell sollte man bei der Platzierung des Projektors und der Kamera darauf achten, dass keine Reflexionen des Projektors auf dem Whiteboard im Bild der Kamera sichtbar sind, denn diese könnten zu einer fehlerhaften Objekterkennung führen. Projektor und Kamera sind mit dem Computer verbunden.

Die Software wurde mit *C++* geschrieben. Die Benutzeroberfläche wurde mit *Qt* [4] erstellt. Für einfache bildverarbeitende Arbeitsschritte, sowie das Einlesen der Kamerabilder, wurden neben eigenen Implementierungen die *OpenCV* Bibliothek [5] verwendet. Die Zuweisung der Graphen wurde mittels *vf2lib* [1] realisiert. Die Animationen der fertigen Applikation sind mit *OpenGL* verwirklicht und durch die Physikengine *Box2D* [2] unterstützt worden. Animationsunterstützende Soundeffekte werden mit der Hilfe von *Qt* eingespielt.

3.2 Pipeline des Whiteboardsystems

Abbildung 3.2 beschreibt die nötigen Schritte zur Realisierung des Whiteboardsystems. Der erste Schritt ist das Einlesen der Bilder. Dies erledigt ein *Framegrabber Modul*, welches mittels OpenCV die Bilder einer USB oder FireWire Kamera

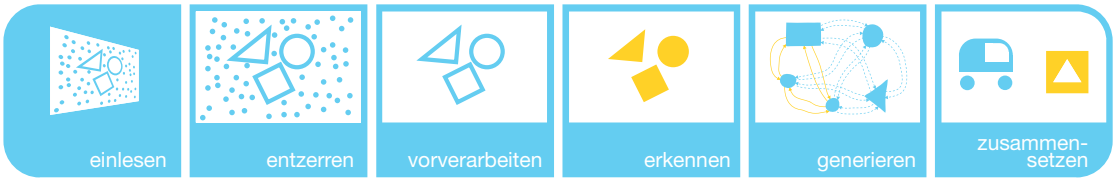


Abbildung 3.2: Die Pipeline des Whiteboardsystems.

einliest. Danach wird zur Weiterverarbeitung das Eingangsbild entzerrt. Anschließend wird das entzerrte Bild vorverarbeitet, damit danach die Primitiverkennung stattfinden kann. Aufbauend auf den Ergebnissen der Primitiverkennung wird ein Datengraph generiert. Dieser wird im letzten Schritt mit der Wissensbasis abgeglichen, damit aus den primitiven Objekten komplexere Objekte zusammengesetzt werden können.

Bis auf das Einlesen der Bilder, werden in den Abschnitten 3.4 bis 3.8 alle Schritte näher erläutert.

3.3 Autobeispiel

Zur besseren Verständlichkeit der nächsten Abschnitte, wird ein kleines Beispiel eingeführt. Das Modell eines Autos besteht aus einem Rechteck mit zwei darunterliegenden Kreisen. Ein auf das Whiteboard gezeichnetes Auto soll dem Modellauto zugewiesen werden (s. Abbildung 3.3).

3.4 Kalibrierung

Der erste Schritt, nachdem Kamera und Projektor richtig ausgerichtet sind, ist die Kalibrierung der beiden, damit die später folgende Objekterkennung auf unverzerrten Bildern stattfinden kann. Dabei wird das Kamerabild mittels einer Homographie auf ein ideales Bild abgebildet. Die radiale Verzerrung der Kamera wurde nicht berücksichtigt, da diese nicht entschieden genug die Segmentierung beeinträchtigt.

Die Homographie wird unter der Zuhilfenahme der OpenCV Funktion *cvFindHomography* berechnet. Diese benötigt als Eingaben die Koordinaten des gewünschten Bereiches des originalen Bildes, sowie die korrespondierenden Zielkoordinaten des neuen Bereiches, indem der gewünschte Teil des originalen Bildes abgebildet wird. Die daraus berechnete Homographiematrix $\boldsymbol{H}$ bildet also jeden Punkt p_i des Ursprungbildes auf einen Punkt p'_i des Zielbildes mit den folgender Gleichung ab:

$$p'_i = \boldsymbol{H} p_i, \tag{3.1}$$

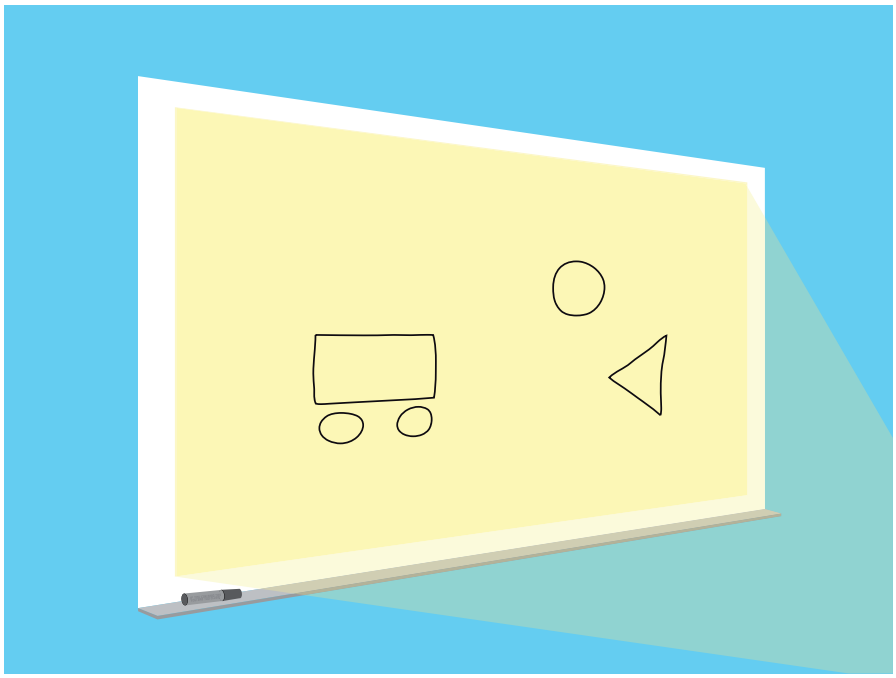


Abbildung 3.3: Noch nicht kalibriertes Bild der Kamera.

Die Berechnung von $\boldsymbol{H}$ wird unter Anhang B beschrieben. OpenCV erstellt die Homographiematrix so, dass der Rückprojektionsfehler minimiert wird. Der abzubildende Teil aus dem originalen Bild besteht aus dem Projektorbild, welches auf das Whiteboard projiziert wird. Durch die Leuchtkraft des Projektors kann das Projektorbild recht einfach über einen Schwellwert freigestellt werden. Diese Einstellung muss vom Benutzer manuell vorgenommen werden. Danach kann die automatische Kalibrierung gestartet werden. Dabei wird mit dem im Abschnitt 3.6.3 beschriebenen Verfahren nach dem viereckigen Bild des Projektors gesucht. Das gesuchte Viereck muss zwei Bedingungen erfüllen:

1. Es muss einen Anteil von 80% des gesamten Kamerabildes einnehmen, um eine ausreichende Größe zur Weiterverarbeitung garantieren zu können und kleiner Vierecke, die beispielsweise in der Form von gezeichneten Vierecken auf dem Whiteboard vorhanden sind, auszuschließen.

2. Des Weiteren müssen die Innenwinkel des Vierecks zwischen 80° und 100° liegen.

War die Segmentierung erfolgreich, werden die Eckpunkte des Viereckes als Eingabekoordinaten verwendet. Jetzt fehlen nur noch die Zielkoordinaten. Diese sind standardmäßig auf eine Bildgröße von 1024×768 Pixel festgelegt, können aber nach belieben geändert werden. Mit Hilfe der errechneten Homographiematrix lässt

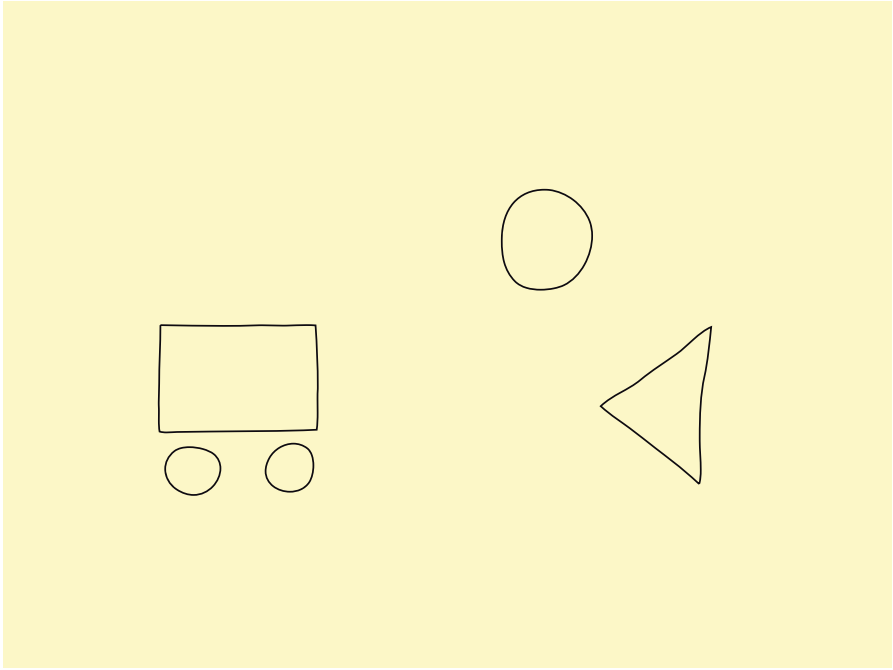


Abbildung 3.4: Entzerrtes Bild der Kamera.

sich also der im Kamerabild erkannte Bereich des Projektors auf ein in bestimmter Größe definiertes ideales Bild abbilden. Abbildung 3.4 zeigt das ideale Bild des Beispiels.

3.5 Vorverarbeitung

Um eine Objekterkennung nach später beschriebenen Verfahren durchführen zu können, ist es sinnvoll zuvor den Vordergrund (gezeichnete Symbole) vom Hintergrund (Whiteboard) zu trennen. Dies geschieht mit Hilfe einer Binarisierung. Dabei soll der Vordergrund schwarz und der Hintergrund weiß sein. Danach wird versucht eventuelle Lücken in den Konturen zu schließen.

3.5.1 Binarisierung

Es wurden verschiedene Verfahren zur Binarisierung getestet. Die Ergebnisse dieser werden im Folgenden beschrieben.

Globales Schwellwertverfahren

Der erste Versuch den Vordergrund vom Hintergrund zu trennen war ein globales Schwellwertverfahren. Dabei wird jeder Pixel des Bildes betrachtet, mit einem zuvor definierten Schwellwert verglichen und dann entweder als Hintergrund (weiß) oder Vordergrund (schwarz) markiert. Damit der Schwellwert nicht von Hand ermittelt werden muss, gibt es verschiedene Ansätze dies zu automatisieren. Eine Variante ist die von Otsu [Ots79], dabei wird der Schwellwert mit Hilfe eines Histogramms der Grauwerte auch dann bestimmt, wenn er im Histogramm nicht klar ersichtlich ist. Die Grenzen des globalen Verfahrens wurden schnell sichtbar. Die Bilder der Kamera haben oft eine Randabschattung, d.h. sie werden nach außen hin dunkler. Dieser Effekt macht sich gerade bei schlechten Kameras mit Weitwinkelobjektiv stärker bemerkbar. Egal welcher Schwellwert gewählt wird, im Randbereich werden Pixel, die zum Hintergrund gehören dem Vordergrund zugeordnet.

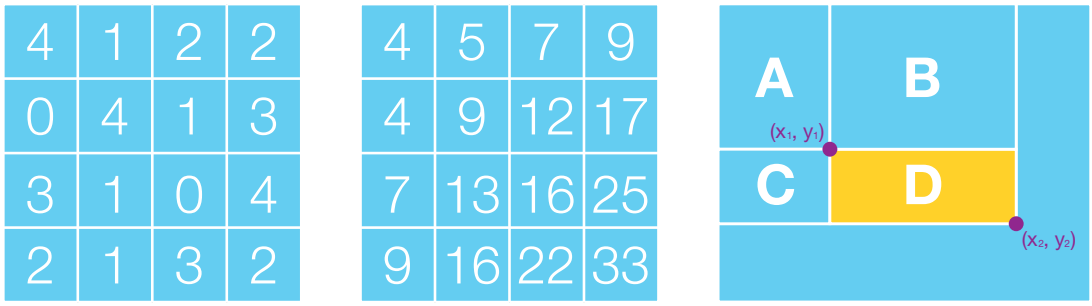
Adaptives Schwellwertverfahren

Da es nicht möglich ist nur einen Schwellwert zu definieren, der für das ganze Bild ausreichende Ergebnisse liefert, wurde versucht das Bild in Segmente zu teilen und für jedes Segment einen eigenen Schwellwert zu bestimmen. Diese Segmente müssen in ausreichend kleiner Größe geteilt sein, damit das eben beschriebene Problem nicht auch bei zu großen Segmenten auftritt. Aber auch dieser Ansatz lieferte keine ausreichenden Ergebnisse, da die meisten Segmente nur aus dem Hintergrund bestehen und ohne bestimmte Einschränkungen des Otsu Verfahrens auch bei diesen Segmenten versucht wird den Hintergrund vom Vordergrund zu trennen. Dies führt zu verstärktem Rauschen in den nicht beschriebenen Bereichen des Whiteboards.

Adaptives Schwellwertverfahren mittels Integralbild

Bessere Ergebnisse ohne störendes Rauschen lieferte ein adaptives Schwellwertverfahren mittels Integralbild nach Bradley und Roth [BR07]. Das Integralbild kann benutzt werden um schnell und effektiv Werte einer Funktion, beispielsweise eines Bildes, aufzusummieren. Ohne das Integralbild könnte die Summe auch in linearer Zeit pro Rechteck im Bild durch Berechnung des Funktionswertes jedes Pixels bestimmt werden. Wird aber die Summe mehrerer überlappender Rechtecke benötigt, ist dies auf diese Weise nicht mehr in linearer Zeit zu berechnen. Mit einem Integralbild allerdings kann dies in einer konstanten Anzahl an Operationen mit nur linearem Aufwand an Vorverarbeitung erreicht werden (vgl. [BR07]).

Das Integralbild (*Summed Area Table*, kurz SAT) ist eine Tabelle der aufsummierten Helligkeitswerte des Bildes. Die Tabelle entspricht den Abmessungen des



(a) (b) (c)

Abbildung 3.5: (a) Bildwerte und (b) das daraus resultierende Integralbild. (c) Berechnung der Summe über Rechteck D mittels Integralbild.

Bildes. Die Werte SAT(x, y) werden durch die Summe aller Helligkeitswerte des Bildes zwischen dem Ursprung und (x, y) berechnet:

$$\mathrm{SAT(x, y)} = \sum_{\mathrm{x'}=0}^{\mathrm{x}} \sum_{\mathrm{y'}=0}^{\mathrm{y}} \mathrm{f(x', y')} \tag{3.2}$$

Die Tabelle kann in einem zeilenweisen Durchlauf durch das Bild von links oben nach rechts unten aufgestellt werden, wobei der Helligkeitswert eines Pixel an der Stelle (x, y) durch den Wert $f(x, y)$ angegeben wird:

$$SAT(x, y) = f(x, y) + SAT(x - 1, y) + SAT(x, y - 1) - SAT(x - 1, y - 1) \tag{3.3}$$

mit

$$SAT(x, y) = 0, \text{wenn } x < 0 \vee y < 0 \tag{3.4}$$

Abbildung 3.5(a) und 3.5(b) zeigt die Berechnung eines Integralbildes. Wurde das Integral bestimmt, kann die Summe der Helligkeitswerte eines jeden Rechtecks mit linken oberen Ecke (x_1, y_2) und rechten unteren Ecke (x_2, y_2) in konstanter Zeit mit folgender Gleichung berechnet werden:

$$\sum_{x=x_1}^{x_2} \sum_{y=y_1}^{y_2} f(x, y) = SAT(x_2, y_2) - SAT(x_2, y_1 - 1) - SAT(x_1 - 1, y_2) + SAT(x_1 - 1, y_1 - 1) \tag{3.5}$$

In Abbildung 3.5(c) wird dargestellt, dass die Berechnung der Summe von $f(x, y)$ über das Rechteck D mit der Hilfe von Gleichung 3.5 äquivalent zur Berechnung der Summe über die Rechtecke $(A + B + C + D) - (A + B) - (A + C) + A$ ist.

Um nun zu entscheiden ob ein Pixel zum Vordergrund oder zum Hintergrund gehört werden zwei Schritte durchgeführt.

1. Berechne das Integralbild
2. Berechne für jeden Pixel den Durchschnittswert eines $s \times s$ großen Fensters und vergleiche den Wert des Pixels mit dem Durchschnittswert. Liegt der Wert des Pixels t Prozent unterhalb des Durchschnittswertes, gehört er zum Vordergrund anderenfalls zum Hintergrund.

Der Algorithmus wird im Anhang A näher erläutert.

3.5.2 Morphologische Operatoren

Um kleine Lücken in den Konturen zu schließen werden wie in [Ser88] beschrieben als letzten Vorverarbeitungsschritt morphologische Operatoren auf das Bild angewandt. Dafür wird eine Funktion *cvMorphologyEx* von OpenCV mit dem Parameter *CV_MOP_CLOSE* verwendet. Diese schließt durch eine Dilatation gefolgt von einer Erosion Lücken der Größe von 3 Pixel. Dieser Wert kann von dem Benutzer verändert werden. Wird dieser Wert allerdings zu stark erhöht, können nahegelegene Kanten verschiedener handgezeichneter Objekte miteinander verschmelzen und werden dadurch nicht mehr richtig bzw. gar nicht erkannt.

3.6 Erkennung primitiver Objekte

Der nächste Schritt ist die Erkennung der gezeichneten Symbole auf dem Whiteboard. Dafür wurde ein einfaches Verfahren gesucht, mit dessen Hilfe es möglich ist aus den binären Ergebnisbildern der Vorverarbeitung die Form, Lage und Größe geometrischer Primitive auszulesen.

3.6.1 Erkennung primitiver Objekte mittels Houghtransformation

Eines der gängigsten Verfahren für die Erkennung von Linien, Kreisen und anderen einfachen Formen ist die Houghtransformation. Vorgestellt in [Hou59] ist sie ursprünglich zur Ermittlung von Linien entwickelt worden. Später wurde das Prinzip von Rosenfeld als Bildverarbeitungsalgorithmus in [Ros69] verwendet. Die Houghtransformation definiert eine Abbildung von Bildpunkten in einen Akkumulatorraum (Houghraum (vgl. [NA02])). Jeder Punkt im Akkumulatorraum entspricht einem geometrischem Objekt im Bildraum. Für eine Gerade kann das z. B. der y-Achsenabschnitt und die Steigung sein und für ein Kreis beispielsweise der Mittelpunkt und der Radius. Danach wird der Akkumulatorraum ausgewertet, indem man nach Häufungen sucht, die der gesuchten Form entsprechen. Die Suche nach Kreisen unterstützt durch die Houghtransformation wird unter anderem in

[KBS75] beschrieben. Yuen et. al [YPIK90] geben einen guten Überblick verschiedener Methoden zur Kreisbestimmung mittels Houghtransformation. Dieses Verfahren kann auch für Ellipsen und andere durch eine parametrische Gleichung darstellbare Formen, angewandt werden. Auch beliebige Formen, die oft nicht durch eine parametrische Repräsentation darstellbar sind, können mit der generalisierten Houghtransformation gefunden werden (siehe [Bal81]).

Probleme der Houghtransformation

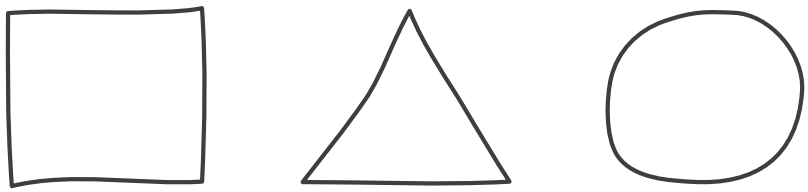
Der erste Versuch zur Erkennung der Primitive wurde mittels Houghtransformation unternommen. Doch es zeigte sich schnell, dass die Houghtransformation ohne große Erweiterungen für die handgezeichneten Symbole ungeeignet ist. Kreise wurden oft nicht erkannt, da sie mehr Ellipsen ähneln als Kreisen. Für die Berechnung einer Ellipse erhöht sich die Anzahl der Dimensionen im Houghraum um ein Vielfaches im Vergleich zum Kreis und wurde deshalb nicht umgesetzt. Auch die Erkennung der Vierecke schlug häufig fehl, da die handgezeichneten Linien oft nicht gerade genug waren. Wegen der schlechten Ergebnisse wurde die Erkennung der Dreiecke gar nicht erst realisiert.

3.6.2 Erkennung primitiver Objekte mittels Quattuorvigintieck

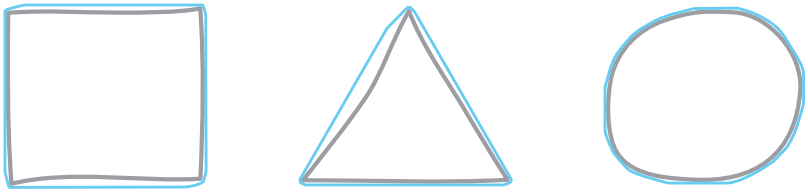
Eine weitere Möglichkeit zur Erkennung von Primitiven ist eine Formanalyse durch ein Quattuorvigintieck (Vierundzwanzigeck). Diese wurde für eine Erkennung von Verkehrszeichen in Echtzeit von Priese et al. [PKL+95] entwickelt. Da die geometrische Form der gesuchten Objekte oft gestört ist, wird nicht direkt nach der Form gesucht, sondern nach der konvexen Hülle der Form. Um die aufwendige Berechnung der konvexen Hülle (vgl. Schian [Sch99]) zu umgehen, wurde eine Approximation der konvexen Hülle entwickelt, das Quattuorvigintieck genannt wurde.

Ein Quattuorvigintieck eines Objektes ist ein Polygon und muss folgende Bedingungen erfüllen (nach Schian [Sch99]):

1. das Polygon hat vierundzwanzig Seiten mit einer Länge ≥ 0,
2. der Winkel zwischen zwei benachbarten Seiten beträgt immer $165^{\circ} = \frac{11}{12}\pi$,
3. zwei Seiten verlaufen parallel zur X-Achse,
4. das Objekt liegt vollständig innerhalb des Polygons,
5. die Summe der Seitenlängen ist minimal.



(a)



(b)

Abbildung 3.6: (a)Primitive und (b) Primitive umgeben vom Quattuorvigintieck. Die Konturen der Quattuorvigintiecke wurden aus Anschauungsgründen nachgezeichnet.

Weiter wird in [Sch99] beschrieben, dass ein Quattuorvigintieck eines Objektes das kleinste 24-Eck ist, welches das Objekt vollständig umschließt und damit eine Approximation der konvexen Hülle des Objektes darstellt.

Der Formtest auf Basis des Quattuorvigintiecks wird durch die Betrachtung der Seitenlängen des Quattuorvigintieck durchgeführt. Dabei wird eine gerade Seite einer konvexen Struktur durch höchstens zwei benachbarte Seiten im Quattuorvigintieck beschrieben. Die Länge der approximierten Seite erhält man durch Addition der beiden Seitenlänge. Eine Gerade lässt sich somit mit nur zwei Seiten eines Quattuorvigintiecks beschreiben, Schian [Sch99] empfiehlt allerdings die Wahl von drei Seiten in der Praxis vorzunehmen, da beispielsweise durch eine leichte Ausbuchtung in der Mitte der Linie diese nicht mehr mit nur zwei Seiten beschrieben werden kann.

Ein rechter Winkel wird vom Quattuorvigintieck mit vier oder fünfe aufeinanderfolgenden Seiten der Länge 0 approximiert, sehr flache Winkel dagegen lassen sich nicht sicher bestimmen. Die Formerkennung ist nicht rotationsinvariant, da die einzelnen Seiten des Quattuorvigintiecks eine feste Richtung haben. Möchte man beispielsweise ein gleichseitiges Dreieck bestimmen, welches beliebig rotiert

ist, müssen sechs Tests durchgeführt werden, deren Seiten sich jeweils um 15^o unterscheiden.

Abbildung 3.6 zeigt Primitive und die dazu erstellten Quattuorvigintiecke. Die genaue Berechnung des Quattuorvigintieck sowie die Bestimmung der Form aus dem Quattuorvigintieck beschreibt Schian in [Sch99].

Probleme des Quattuorvigintiecks

Integriert wurde dieses Verfahren unter zu Hilfenahme der *Koblenzer Image Processing Library* (KIPL). Entwickelt wurde die Bibliothek KIPL ursprünglich von Patrick Sturm und Frank Schmitt und wird heute noch weiterentwickelt von Frank Schmitt und studentischen Mitgliedern der Arbeitsgruppe Labor Bilderkennen[1] der Universität Koblenz. Die damit erzielten Ergebnisse waren wesentlich besser als die der Houghtransformation und viel versprechender. Allerdings anders als von Schian [Sch99] beschrieben, müssen die Primitve folgende Bedingungen erfüllen:

- Alle Konturen müssen geschlossen sein.
- Dreiecke müssen nach oben oder unten zeigen.
- Rechtecke müssen Achsenparallel sein.

Ohne grundlegende Änderungen im Quellcode der Bibliothek können diese Bedingungen nicht außer Kraft gesetzt werden. Somit ist die Verwendung des Quattuorvigintieck der KIPL-Bibliothek zur Erkennung der Primitive auf dem Whiteboard nur bedingt geeignet.

3.6.3 Eigener Ansatz

Die Idee ist anhand der Anzahl der Eckpunkte des Primitiven auf die Form dessen zu schließen. Die Eckpunkte werden dabei über die Distanz der einzelnen Konturpunkte zum Mittelpunkt des Primitven ermittelt. Bei den ausgewählten Primitiven (Viereck, Dreieck, Kreis) bedeutet das:

- bei 3 Maxima in der Distanzfunktion handelt es sich um ein Dreieck,
- bei 4 Maxima in der Distanzfunktion handelt es sich um ein Viereck,
- sind alle Punkte gleich weit vom Mittelpunkt entfernt, handelt es sich um einen Kreis.

[1] www.uni-koblenz-landau.de/koblenz/fb4/institute/icv/agpriese

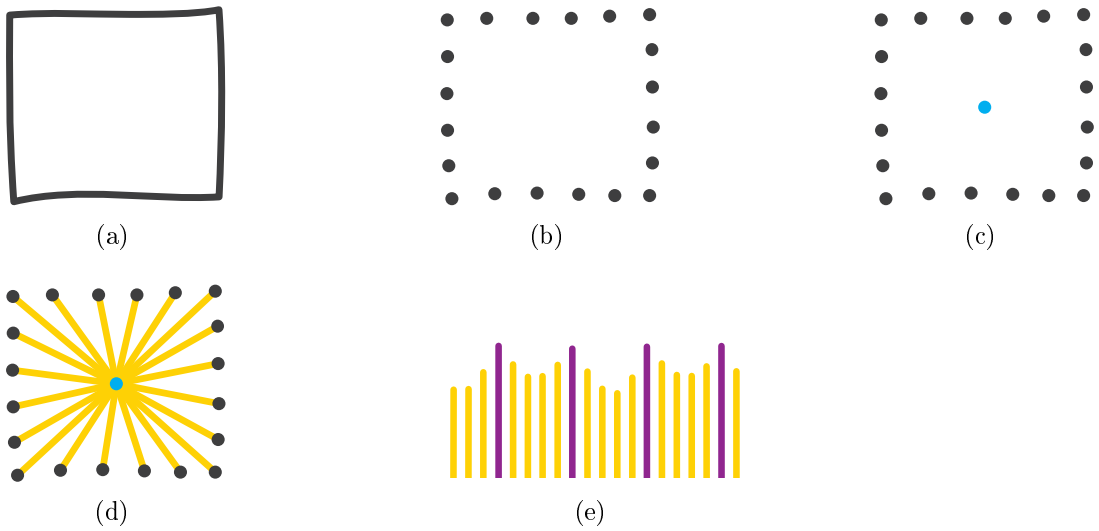


Abbildung 3.7: Erkennung eines Vierecks.(a) das Symbol, (b) die Kontur des Symbols, (c) der Mittelpunkt, (d) die Abstände der einzelnen Punkte zum Mittelpunkt und (d) die vier gefundenen Maxima (lila).

Dafür müssen zuerst die Konturen der möglichen Primitive in dem Binärbild bestimmt werden (s. Abbildung 3.7(a)). Dies wurde mit der OpenCV Funktion *cvFindContours* realisiert. Die Konturen werden in einer Hierarchie mit zwei Ebenen zurückgeliefert, wobei die obere Hierarchie die äußere Kontur und die innere Hierarchie das Loch der Kontur ist. Zur Formbestimmung ist nur die äußere Kontur notwendig (vgl. Abbildung 3.7(b)).

Im Folgenden wird mit den Ortsvektoren der Punkte gerechnet, wobei im Weiteren der Einfachheit halber von Punkten gesprochen wird. Der Mittelpunkt $\boldsymbol{p}_\mathrm{c}$ einer Kontur mit den Punkten $\boldsymbol{p}_i; i \in [0; n]$ wird wie folgt bestimmt (s. Abbildung 3.7(c)):

$$\boldsymbol{p}_\mathrm{c} = \frac{1}{n} \sum_{i=0}^{n} \boldsymbol{p}_i \tag{3.6}$$

Danach wird der Abstand d_i der einzelnen Punkte zum Mittelpunkt berechnet (s. Abbildung 3.7(d)).

$$d_i = \|\boldsymbol{p}_\mathrm{c} - \boldsymbol{p}_i\|_2 \tag{3.7}$$

Sind alle Abstände bestimmt, werden diese mittels Medianfilter behandelt, um eventuelle Ausreißer zu eliminieren. Danach kann die Untersuchung der Form beginnen. Für einen Kreis müssen alle Abstände gleich sein. Da es sich um handgezeichnete Symbole handelt, wird das nur in den seltensten Fällen zu treffen. Des-

halb wurde empirisch ein Schwellwert ermittelt. Es handelt sich um einen Kreis, wenn für die Abstände aller Punkte gilt:

$$\sqrt{(d_i - d_\mathrm{m})^2} < \frac{1}{4} d_\mathrm{m}, \tag{3.8}$$

wobei d_m der Mittelwert aller Abstände einer Kontur ist.

Trifft die Bedingung nicht zu, wird als nächstes auf Drei- bzw. Viereck untersucht. Wie Abbildung 3.7(e) zeigt, wird der Abstand zum Mittelpunkt in den Ecken größer. Das bedeutet, für jede Ecke existiert ein Maximum. Die Nulldurchgänge der ersten Ableitung werden mit Hilfe eines Ableitungs-Gaussfilters ermittelt. Die optimalen Werte des Filters wurden empirisch ermittelt.

Theoretisch lassen sich auf diese Weise auch Formen mit mehr als vier Ecken ermitteln, jedoch muss man beachten, dass Formen mit einer hohen Eckenanzahl Kreisen immer ähnlicher werden können und es so zu fehlerhaften Zuweisungen kommt.

Probleme mit der richtigen Erkennung treten auf, wenn die gezeichneten Symbole nicht ganz geschlossen sind und diese Lücken während der Vorverarbeitung nicht richtig überbrückt werden. Bei Kreisen wirkt sich das Problem nicht so groß wie bei Drei- und Vierecken aus. Ist mehr als eine Lücke in der Kontur vorhanden, werden mehrere Konturen anstelle der einen Kontur gefunden.

3.6.4 Offene Konturen

Eine Lösung des Problems der offenen Konturen ist die Bestimmung der Form über eine Approximation seiner konvexen Hülle. Dafür wird die konvexe Hülle des Objekts mittels der OpenCV Funktion *cvConvexHull2* bestimmt. Dann wird, wie in den Gleichungen (3.6) und (3.7) beschrieben, der Kreistest durchgeführt. Trifft die Bedingung nicht zu, handelt es sich nicht um einen Kreis. Um nun zu überprüfen, ob das Primitive ein Viereck oder Dreieck ist, muss eine Approximation der Hülle durchgeführt werden.

Die Approximation wurde mit dem Verfahren von Douglas und Peucker [DP73] durchgeführt. Dabei wird eine Kontur bzw. Kurve (Abbildung 3.8(a)) mit einer geordneten Menge an Punkten durch das Weglassen einzelner Punkte vereinfacht, ohne dabei grundlegende Änderung an der Gestalt der Kurve vorzunehmen. Dies geschieht mit der Definition eines Schwellwertes. Dieser Schwellwert beschreibt den maximalen Abstand zwischen den Punkten der Ausgangskurve und der Zielkurve.

Eine Kontur k besteht aus einer Folge an Punkten $\boldsymbol{p}_1, ..., \boldsymbol{p}_n$ die in der Form $\boldsymbol{p}_{i+1} - \boldsymbol{p}_i; i = 1, ..., n-1$ mit einander verbunden sind. Zuerst wird durch Verknüpfung von Startpunkt $\boldsymbol{p}_1$ und Endpunkt $\boldsymbol{p}_n$ die Grundlinie $\boldsymbol{l}$ erzeugt:

$$\boldsymbol{l} = \boldsymbol{p}_n - \boldsymbol{p}_1 \tag{3.9}$$

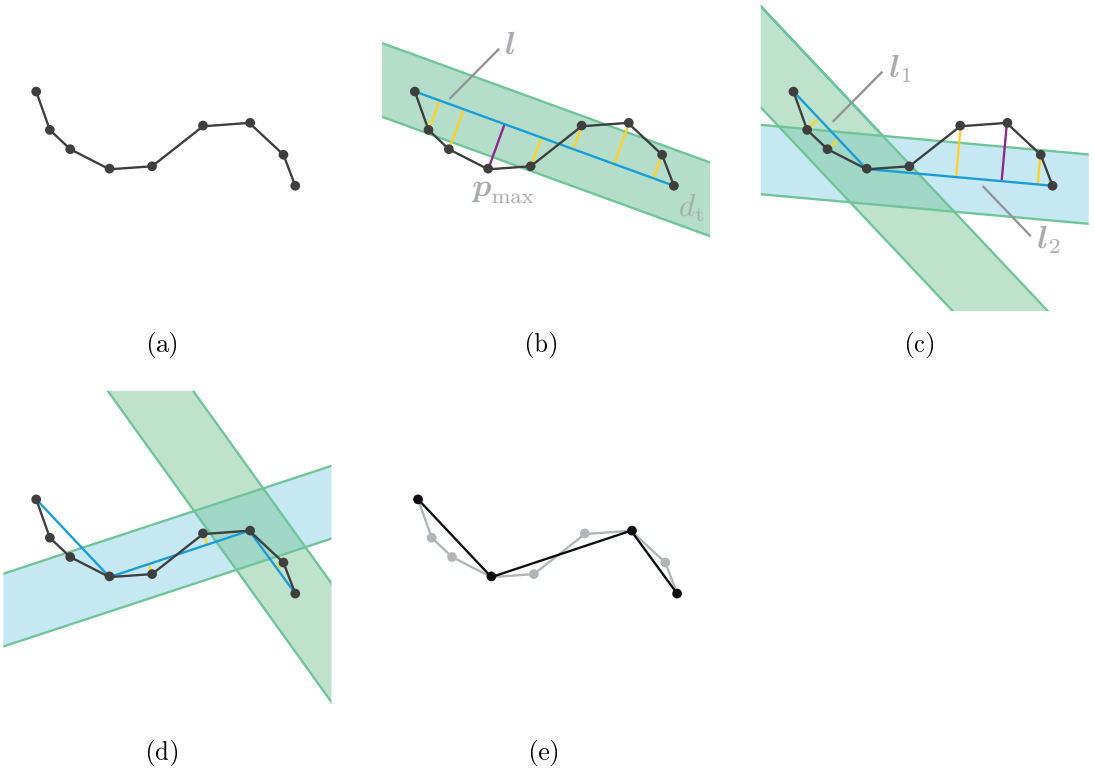


Abbildung 3.8: Punktreduzierung nach Peucker. (a) eine Kontur, (b) Abstände der Punkte zur Grundlinie, (c) und (d) Unterteilung der Grundlinie und (d) die reduzierte Kontur.

Des weiteren wird ein Schwellwert d_t festgelegt. Danach werden die Abstände d_i aller Punkte $\boldsymbol{p}_2, ..., \boldsymbol{p}_{n-1}$, die sich zwischen dem Start- und dem Endpunkt befinden, ermittelt. Liegen die Abstände aller Punkte innerhalb des Schwellwertes ($d_n < d_\mathrm{t}$), werden alle Zwischenpunkte gelöscht. Ist dies nicht der Fall und es befinden sich somit Punkte weiter von der Grundlinie entfernt als der Schwellwert es vorschreibt, wird der Punkt $\boldsymbol{p}_\mathrm{max}$ mit dem maximalen Abstand ermittelt (siehe Abbildung 3.8(b)). Die ursprüngliche Grundlinie wird in zwei neue Segmente

$$\boldsymbol{l}_1 = \boldsymbol{p}_\mathrm{max} - \boldsymbol{p}_1 \tag{3.10}$$

und

$$\boldsymbol{l}_2 = \boldsymbol{p}_n - \boldsymbol{p}_\mathrm{max} \tag{3.11}$$

geteilt. Für diese zwei neuen Grundlinien wird der Vorgang getrennt voneinander solange rekursiv wiederholt (Abbildung 3.8(c) bis 3.8(d)), bis keine überflüssigen Punkte mehr existieren. Die approxmierte Kurve liegt dann in Form der Grundlinien vor (s. Abbildung 3.8(e)).

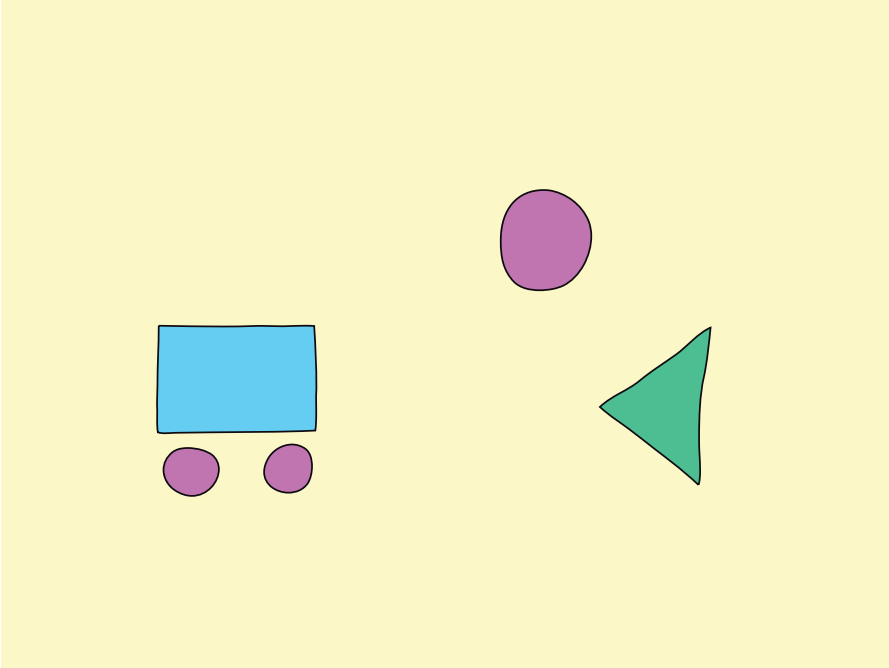


Abbildung 3.9: Bild mit erkannten Primitiven. Alle erkannten Kreise sind lila, das Rechteck ist blau und das Dreieck ist grün.

Nun kann anhand der Anzahl der verbliebenen Punkte (bei drei Punkten handelt es sich um ein Dreieck, bei vier Punkten um ein Viereck usw.) auf die Form des Primitiven geschlossen werden. Im weiteren Verlauf werden lediglich spezifische Vierecke betrachtet, welche die Bedingungen eines Rechtecks erfüllen, alle anderen werden ignoriert. Abbildung 3.9 illustriert die erkannten Primitive aus dem Autobeispiel. Gleiche Formen haben die gleiche Farbe.

3.7 Generierung des Datengraphen

Wie in Abbildung 3.9 gezeigt, liegen nach der Primitiverkennung alle handgezeichneten Symbole als geometrische Primitive vor. Es ist also der Typ (Kreis, Dreieck, Rechteck), die Größe und die Lage jedes Symbols bekannt. Bevor man erkennen kann, ob sich unter den ganzen Primitiven auch Teile unseres Auto befinden, müssen Beziehungen zwischen diesen aufgestellt und in Form eines Graphen gespeichert werden. In dieser Arbeit werden die Primitive auf dem Whiteboard als Knoten und deren Beziehung zueinander als Kanten repräsentiert. Der dazugehörige Graph wird als Datengraph bezeichnet. Des Weiteren existieren verschiedene Modellgraphen. Diese Modellgraphen repräsentieren komplexere Objekte, die aus primitiven Objekten zusammengesetzt sind. Die Aufgabe ist es, diese Modellgra-

phen im Datengraphen wiederzufinden. Da das im nächsten Abschnitt verwendete Graph-Matching Verfahren auf direkten attributierten Graphen basiert, muss dies bei der Erstellung des Graphen berücksichtigt werden.

3.7.1 Erstellung der Knoten

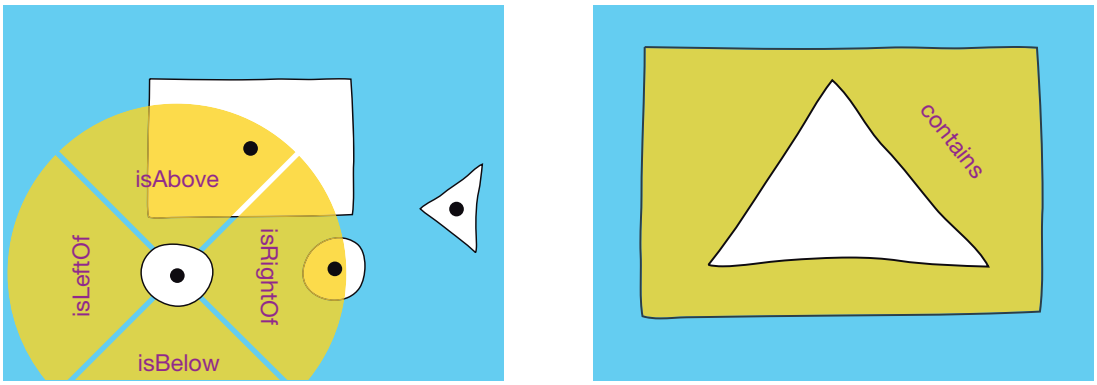
Der erste Schritt zum fertigen Graphen ist die Erzeugung aller Knoten. Dabei wird für jedes erkannte Primitiv ein Knoten erzeugt, welcher als Attribut den Typ des Primitiven erhält. Das bedeutet, es existieren nur Knoten, die entweder *Circle*, *Rectangle* oder *Triangle* als Attribut haben und für jedes primitive Objekt existiert ein korrespondierender Knoten.

3.7.2 Erstellung der Kanten

Als nächstes werden die Kanten zwischen den Knoten erzeugt. Bei dem Datengraph handelt es sich um einen vollständigen Graph, das bedeutet, das alle Knoten miteinander verbunden sind. Genauer betrachtet, existieren sogar zwei Kanten zwischen den einzelnen Knoten, wobei die Attribute der Kanten die jeweilige geometrischen Relationen zwischen den beiden Knoten beschreiben. Es sind folgende Relationen möglich:

- *isLeftOf*
- *isRightOf*
- *isBelow*
- *isAbove*
- *isIn*
- *contains*
- *isOutOfBounds*

Die Realation “isOutOfBounds” bedeutet, dass die Distanz zwischen zwei Primitiven über einem bestimmten Schwellwert liegt. Abbildung 3.10 beschreibt die Zuweisung der Relationen. Ob ein Primitiv links, rechts, über, unter oder außerhalb der Reichweite eines anderen Primitiven liegt, wird anhand der Lage ihrer Mittelpunkte entschieden (vgl. Abbildung 3.10(a)). Anders ist es wenn ein Primitiv sich innerhalb eines anderen Primitiven befindet, dazu müssen alle seine Punkte innerhalb der Fläche des anderen Primitiven liegen (vgl. Abbildung 3.10(b)).



(a) (b)

Abbildung 3.10: (a) Zuweisung der geometrischen Relationen aus der Sicht des linken Rades. Das Dreieck außerhalb des gelben Kreises bekommt die Relation "outOfBounds" zugewiesen. (b) alle Punkte des Dreiecks befinden sich innerhalb des Rechtecks.

3.7.3 Datengraph des Beispiels

Abbildung 3.11 illustriert die Darstellung des Datengraphes aus dem Beispiel. Man sieht deutlich, dass alle Objekte mit zwei Kanten miteinander verbunden sind. Die Attribute der Knoten bestehen immer noch aus der Form der Primitive (Kreis, Dreieck oder Rechteck). Die semantischen Bedeutungen wie beispielsweise *Fahrzeugkabine* oder *linkes Rad* erhalten sie erst im nächsten Schritt.

3.8 Erzeugung komplexer Objekte durch Graph-Matching

Da jetzt die Informationen des Bildes in Form des Datengraphen vorliegt, kann schließlich der Abgleich des Datengraphen mit der Wissensbasis stattfinden. Bei der Wissensbasis handelt es sich um verschiedene Modellgraphen, die in einzelnen Textdateien vorliegen. Sie enthalten die Informationen, aus welchen Primitiven eine komplexes Objekt besteht und wie die Relation der verschieden Primitiven zueinander ist. Das bedeutet, dass jedes komplexe Objekt durch einen Modellgraphen beschrieben wird.

Um jetzt einen der Modellgraphen im Datengraphen wieder zu finden, wird das unter Kapitel 2.3.2 beschriebene Verfahren verwendet. Alle Primitive, die zu einem Knoten im Datengraphen gehören, der auf den Modellgraphen abgebildet wurde, werden zu dem zugehörigen komplexen Objekt zusammengesetzt. Dadurch erhalten die primitiven Objekte ihre Semantik. Jedem primitiven Objekt kann nur

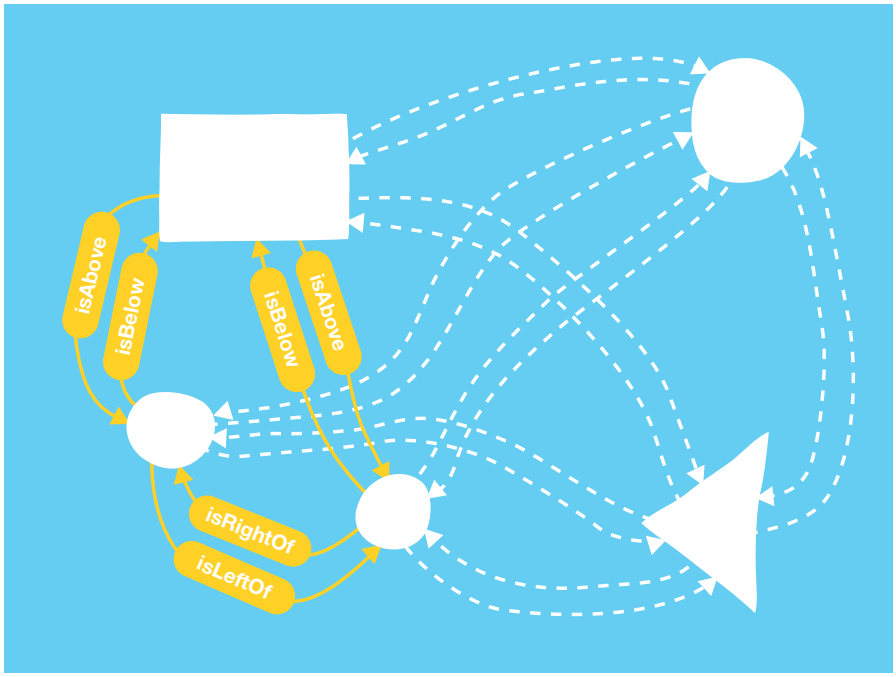


Abbildung 3.11: Datengraph des Beispiels. Aus Gründen der Übersicht wurden alle Kanten mit dem Attribut "outOfBounds" als gestrichelte Linie dargestellt.

einmal eine semantische Bedeutung zugewiesen werden. Deshalb ist es wichtig, dass die Modellgraphen nach ihrer Komplexität sortiert in der Wissensbasis vorliegen und die Suche mit dem komplexesten Objekt beginnt. Dadurch wird es vermieden, dass beispielsweise dem Rad eines Autos, welches aus einem Kreis besteht, die Semantik eines Balls, der auch aus einem Kreis besteht, zugewiesen wird.

3.8.1 Komplexe Objekte des Beispiels

Die zugehörige Wissensbasis des Beispiels besteht aus drei Modellgraphen: Einem Auto, einem Ball und einem Dreieck. Es existieren also drei verschiedene komplexe Objekte, geeignet sortiert noch oben beschriebener Komplexität wird während des Graph-Matchings erst nach dem Auto und dann entweder nach dem Ball oder dem Dreieck gesucht. Abbildung 3.12 stellt die erfolgreiche Zuweisung sowie die Erzeugung der komplexen Objekte dar.

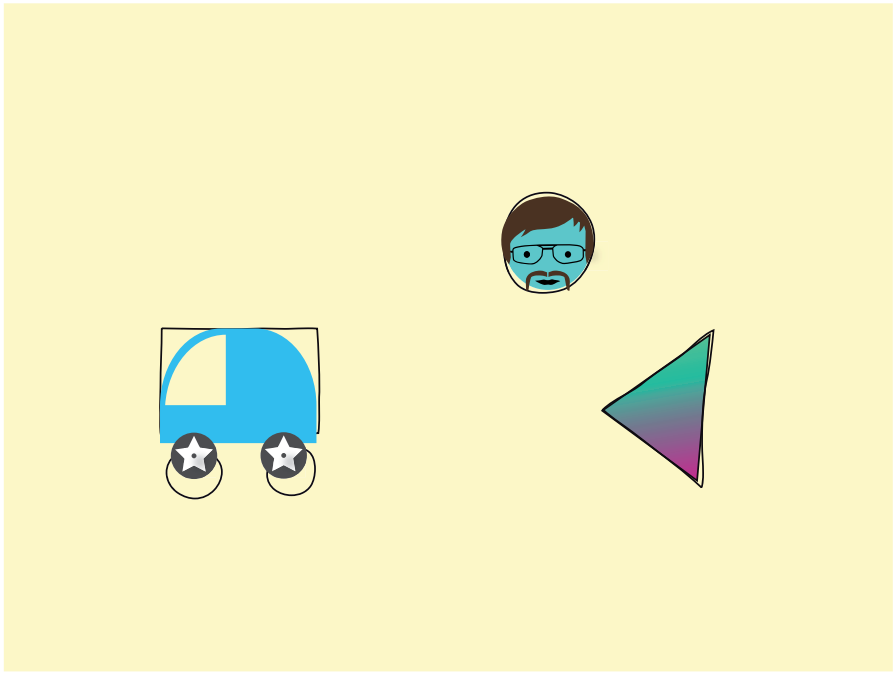


Abbildung 3.12: Bild der erkannten komplexen Objekte mit zugehöriger Textur.

Kapitel 4

Applikation für das interaktive Whiteboardsystem

Dieses Kapitel beschreibt die Realisierung einer Spieleapplikation für das interaktive Whiteboardsystem. Dafür wird als erstes der Ablauf der Applikation und anschließend das Konzept des Spiels vorgestellt. Danach wird beschrieben wie aus dem zugewiesenen Datengraphenen die physikalischen Objekte der Welt werden. Zum Schluss wird die Simulation der ganzen Objekte erklärt.

4.1 Ablauf des Systems

Das System wurde nach Kapitel 3.1 und 3.4 aufgebaut und kalibriert, so kann ein einzelner Durchlauf folgendermaßen beschrieben werden:

- Der Benutzer startet die Applikation,
- daraufhin wird das aktuelle Level des Spiels erklärt.
- Der Benutzer beginnt die gewünschten Symbole auf das Whiteboard zu zeichnen und betätigt anschließend die Starttaste im Hauptfenster.
- Das Bild des Projektors wird gelöscht und ein neues Kamerabild aufgenommen. Wie in Kapitel 3 beschrieben, wird das Bild entzerrt, vorverarbeitet, der Datengraph generiert und mit dem Modelgraphen abgeglichen.
- Die physikalischen Objekte werden erzeugt und
- die Simulation wird gestartet.

Führten die gezeichneten Zeichen nicht zum Ziel, können diese beliebig verändert oder weggewischt werden. Auch können neue Zeichen auf das Whiteboard gemalt werden. Danach wird einfach wieder der Startknopf betätigt und das Spiel beginnt von Neuem.

4.2 Interaktives Whiteboardspiel

Das Konzept des Spiels sollte sich von den gängigen Physiksimulationen, bei denen durch die einfache Erzeugung von physikalischen Objekten durch das Zeichnen von Polygonen, ein Gegenstand zu einem Zweiten bewegt werden muss, abheben. Der erste Schritt dazu ist, dass durch die Kombination einfacher Zeichen nicht nur Polygone, sondern Objekte (z. B. in der Form eines Autos oder einer Kanone) kreiert werden, mit denen man in der Welt des Spiels unterschiedliche Aktionen auslösen kann. Nicht nur der Weg zum Ziel, sondern auch das Ziel selbst soll sich von Level zu Level ändern. Die Ausführung soll nun am Beispiel des ersten Level erklärt werden.

4.2.1 Spielgegenstände

Abbildung 4.1: Spielkugeln

Der wichtigste Gegenstand ist die Spielkugel. Diese wird in Form eines Kreises auf das Whiteboard gemalt. Es gibt sie in verschiedenen Ausführungen, die zufällig ausgewählt werden und rein kosmetischer Natur sind (siehe Abbildung 4.1). In Abbildung 4.2 werden alle restlichen Gegenstände sowie ihre dazugehörigen Symbole, die bis zur Fertigstellung dieser Diplomarbeit in das Spiel aufgenommen wurden, vorgestellt. Darunter befinden sich einfache geometrische Objekte wie Dreieck und Rechteck, eine Kanone und ein Auto. Alle diese Gegenstände bis auf die Kanone haben eine Masse und unterliegen somit der Schwerkraft. Die Kanone bleibt an der Stelle, an der sie gezeichnet wurde. Das Auto bewegt sich in eine Richtung. Trifft es auf einen fest in der Welt positionierten Gegenstand (dazu zählen auch die Wände) ändert es die Richtung. Zu einem späteren Zeitpunkt soll diese Gegenstandsliste erweitert werden.

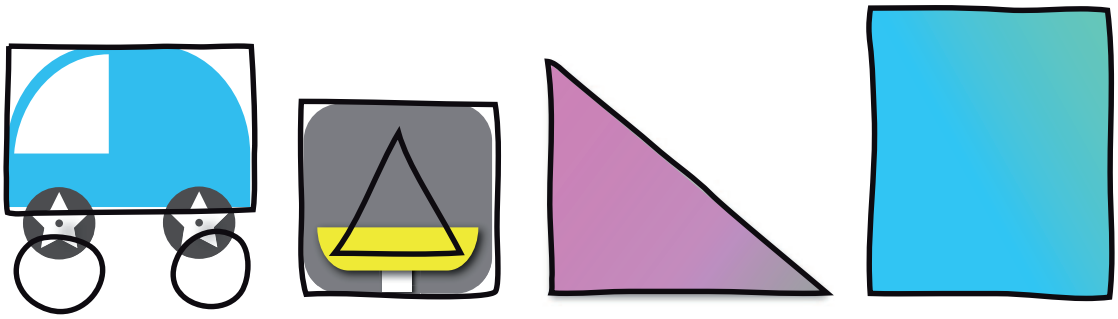


Abbildung 4.2: Restliche Spielgegenstände.

4.2.2 Erstes Level

Im Ersten Level ist das Ziel, den roten Baron, der das friedliche Leben der Bolitas stört, abzuschießen. Die verwendeten Hilfsmittel dabei sind alle zur Verfügung stehenden Spielgegenstände. Jedoch einzig und allein die Spielkugel kann dem roten Baron gefährlich werden. Diese muss in das dafür vorgesehen blaue Feld gezeichnet werden. Die Genaue Position im Feld als auch die Größe sind dabei variabel. Wurde die Kugel (es sind auch mehrere Kugeln möglich) in das blaue Feld auf dem Whiteboard gezeichnet und das Spiel gestartet, gehorcht sie den Gesetzen der Schwerkraft und fällt auf den Boden. Dabei sollt man den Kaktus beachten, der die Kugel zum Platzen bringt. Fällt die Kugel sicher zu Boden, muss durch geschickte Platzierung der anderen Gegenstände die Kugel gegen den roten Baron geschossen werden. Alle Gegenstände, mit Ausnahme der Kugel, dürfen frei positioniert werden. Eine mögliche Lösung wird in Abbildung 4.3 skizziert.

4.3 Physikalische Objekte

Hat der Benutzer das Spiel gestartet, kann er mit dem Zeichnen der Spielgegenstände loslegen. Sind alle Symbole auf das Whiteboard gezeichnet, müssen mit Hilfe dieser die physikalische Objekte in der Welt des Spiels erzeugt werden. Dazu werden die einzelnen Symbole wie in Kapitel 3.6 beschrieben zuerst erkannt, bevor sie danach (s. Kapitel 3.7 und 3.8) komplexeren Objekten zugeordnet werden. Aus diesen komplexen Objekten werden physikalische Objekte anhand ihrer semantischen Bedeutung erzeugt. Betrachtet man sich das Autobeispiel aus Kapitel 3.3. Das beschriebene Auto ist ein komplexes Objekt, welches aus Primitiven mit semantischer Bedeutung zusammengesetzt ist. Das Auto besteht aus einem Rechteck mit der Bedeutung der Fahrzeugkabine und zwei Kreisen mit der Bedeutung des linken und rechten Rades. Aus diesen Primitiven, werden die einzelnen physikalischen Körper erstellt, die zusammen das ganze physikalische Objekt erge-

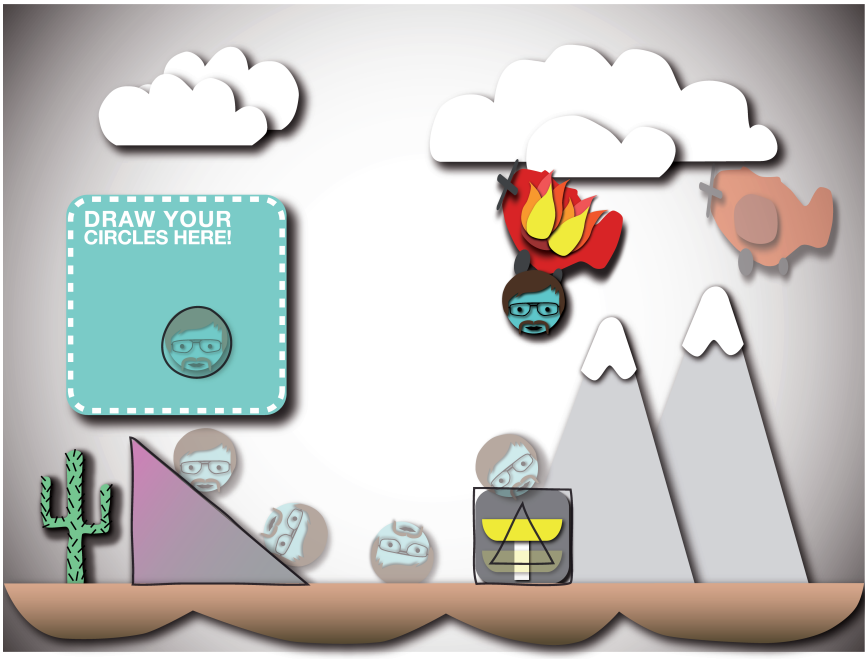


Abbildung 4.3: Der rote Baron wurde getroffen.

ben. Die Fahrzeugkabine und die Räder haben dabei unterschiedliche physikalische Eigenschaften[1] und sind miteinander verbunden. Aber nicht nur die gezeichneten Objekte auf dem Whiteboard werden auf diese Weise erzeugt, auch die einzelnen Levelobjekte liegen als komplexe Objekte vor und werden so erstellt. Das Objekte miteinander kollidieren oder andere Aktivitäten (z. B. als Sensoren fungieren) ausüben können wird mit der Hilfe der Physikengine erledigt.

4.4 Simulation

Die eigentliche Simulation des Spiels findet in einer Spielschleife statt. Damit die Physiksimulation keine falschen Ergebnisse liefert, müssen alle Schritte nacheinander in folgender Reihenfolge ausgeführt werden:

1. Einen Zeitschritt in der Physikwelt voranschreiten.

2. Alle Kontakte zwischen den Objekten ermitteln.

3. Ggf. Objekte löschen.

[1] Dazu zählen Masse, Reibung usw. Alle Eigenschaften können dem Handbuch der Box2D Physiksimulation [3] entnommen werden.

4. Alle vorhanden Objekte zeichnen.

Nach einem Zeitschritt erhält man alle veränderten Informationen der physikalischen Welt. Beispielsweise welcher Körper sich wohin bewegt hat oder auf welchen Körper gestoßen ist. Dabei kann man festlegen, wie oft diese Informationen aktualisiert werden. Ein guter Mittelwert zwischen genauer Simulation und ausreichender Geschwindigkeit ist eine Aktualisierung alle $\frac{1}{60}$ Sekunde, wobei die Anzahl der Iterationen zur Bestimmung der Veränderungen bei 10 liegt. Wird die Anzahl der Iterationen erhöht, wird die physikalische Simulation zu lasten der Geschwindigkeit genauer.

Als nächstes wird anhand der zurückgelieferten Objekte mittels eines Kontaktermittlers festgestellt, ob ein Körper einen anderen berührt, in ihn eingedrungen oder ihn verlassen hat. Mit diesen Informationen ist es möglich z. B. die Kugel bei Berührung des Kaktus platzen zu lassen. Besteht also ein Kontakt, wird je nach Objekt entschieden, was nach dem Kontakt mit beiden Objekten passiert. Im Falle der Kugel wird diese zum Löschen freigegeben und ein passendes Geräusch dazu eingespielt.

Existieren Objekte, die gelöscht werden sollen, geschieht dies unmittelbar bevor die restlichen Objekte gezeichnet werden. Danach beginnt die Schleife von vorne.

Kapitel 5

Tests und Ergebnisse

Die Interaktion zwischen Benutzer und Computer findet über handgezeichneten Symbole auf dem Whiteboard statt und es ist zwingend notwendig, dass diese Symbole richtig erkannt werden. Daher wird in den folgenden Tests die Klassifikation der Primitive untersucht.

5.1 Evaluation der Primitiverkennung

Zur Evaluation dieser Klassifikation wurden 10 Personen des Fachbereichs Informatik der Universität Koblenz beauftragt das System zu benutzen. Die Probanden hatten zuvor keine Erfahrung mit dem System, waren zwischen 24 und 39 Jahren und haben ein Whiteboard für Besprechungen oder Ähnliches schon einmal benutzt. Alle Tests wurden im Labor Bilderkennen der Universität Koblenz durchgeführt. Als Basissystem diente ein Macbook mit einem 2 GHz Intel Core 2 Duo Prozessor und 1 GB Arbeitsspeicher sowie das Betriebssytem Mac OS X 10.5.4. Bei der verwendeten Kamera handelt es sich um die *DFK 21F04* der Marke "THE IMAGING SOURCE" mit 640×480 Pixel mit einem Cosmicar TV Lens Objektiv (8 mm, 1:1,4). Als Projektor wurde ein Canon LV 7365 benutzt. Die Evaluation erfolgte in zwei Phasen. In der ersten Phase wurde den Probanden das System vorgestellt und ihre Aufgabe erklärt. Ihnen wurde beschrieben, wie man mit dem System interagiert und was dabei zu beachten ist. Dazu wurden sie gebeten folgende Bedingungen einzuhalten:

- Die Symbole sollten geschlossen sein.
- Symbole dürfen sich nicht überschneiden.

Die eigentliche Aufgabe bestand darin ein beschriebenes Bild (s. Abbildung 5.1) zu zeichnen, welches vom System erkannt werden sollte. Das Bild lässt sich in

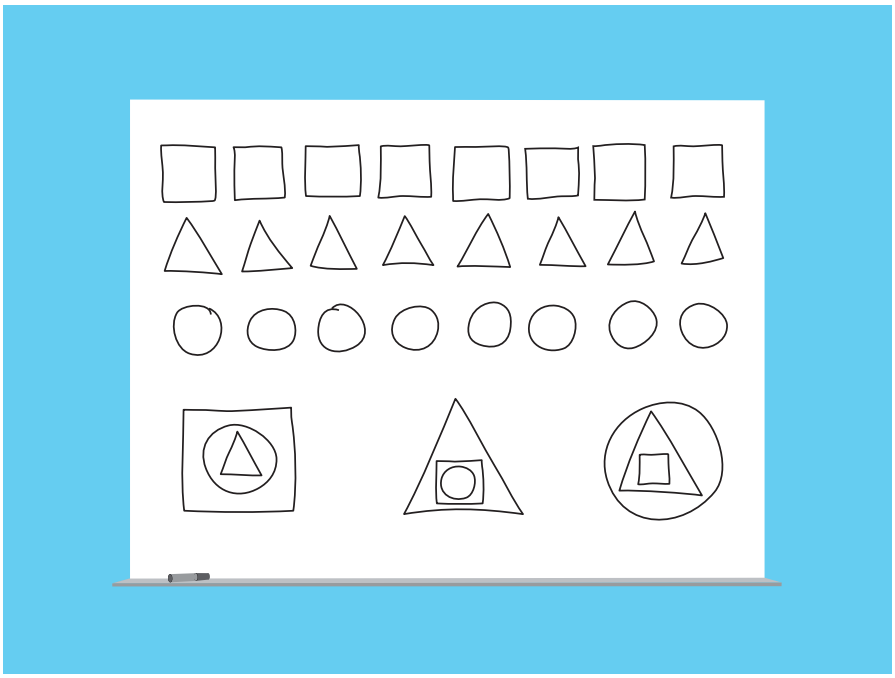


Abbildung 5.1: Dies ist die Skizze der Evaluationsaufgabe. Sie wurde den Probanden nicht gezeigt, um diese nicht zu beeinflussen.

zwei Bereiche teilen. Der obere Bereich des Bildes testet die Eingabe der einzelnen Primitiven. Der untere Bereich testet die Eingabe der Primitive als verschachtelte Objekte und somit unter erschwerten Bedingungen. Den Probanden wurde erklärt, dass alleine das Ergebnis und nicht die Zeit ausschlaggebend ist. Außerdem durften Symbole auch korrigiert werden.

Die zweite Phase war die Durchführung der Aufgabe. Diese dauerte zwischen einer und drei Minuten. Nachdem die Probanden mit dem Zeichnen fertig waren, wurde das Originalbild der Kamera gespeichert.

5.1.1 Auswertung

Um die Genauigkeit der Klassifikation zu bestimmen, wurden die Ergebnisse der Tests in eine *Confusion Matrix* geschrieben. Die Matrix aus Tabelle 5.1 zeigt die Ergebnisse der Klassifikation einzelner Primitive. Es existieren die drei Klassen der Primitiven und eine Rückweisungsklasse Ω_0, in die alle nicht definierten Klassifikationen fallen. Des weiteren werden in der Matrix segmentierungsbedingte Fehlklassifikationen Ψ_0 dargestellt.

Die Erkennung der Dreiecke lieferte sehr gute Ergebnisse und es konnte nur zweimal ein Dreieck nicht zugewiesen werden. Ähnlich gut war es bei der Klassifi-

	vorhergesagte Klasse				
korrekte Klasse	□	△	○	Ω_0	Ψ_0
□	85%	1,25%			13,75%
△		97,5%		2,5 %	
○	15%		61,25%	20%	3,75%

Tabelle 5.1: Konfusionsmatrix der Primitiverkennung mit einer Auflösung von 640×480 Pixel.

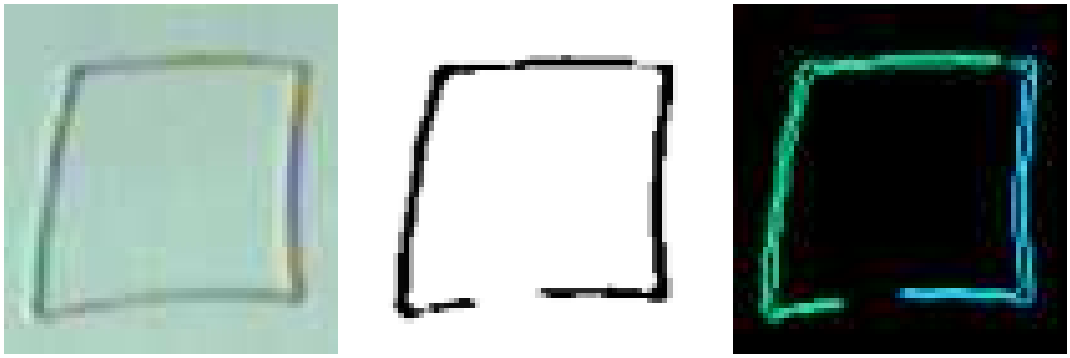
kation der Rechtecke, wobei die Zahl der Fehlentscheidung höher ist. So wurde in manchen Fällen durch Segmentierungsfehler ein Rechteck wegen offener Konturen als zwei Dreiecke oder als ein Dreieck und Rechteck erkannt. Die Ergebnisse der Kreise dagegen sind zufrieden stellend. Man erkennt deutlich, dass der Kreis oft nicht erkannt oder fälschlicherweise dem Rechteck zugewiesen wurde.

Die Ursache der falschen Zuweisung liegt zum Teil an der Kombination einer niedrigen Auflösungung von nur 640×480 Pixel und sehr feingezeichneten Linien. Dadurch entstanden bei manchen Symbolen während der Binarisierung mehr als eine Öffnung. Diese konnten teilweise nicht mehr durch die morphologischen Operatoren geschlossen werden und hatten somit zur Folge, dass beispielsweise anstelle eines Rechtecks zwei Dreiecke gefunden wurden. Auch die Wahl und Verwendung des Stiftes kann dieses Problem verstärken. Ein Teil der Probanden übte beim Zeichnen einen niedrigeren Druck auf den Stift aus, wodurch die Linien weniger deckend waren und somit zur Lückenbildung beitrugen.

Tabelle 5.2 beinhaltet die durchschnittlich Ergebnisse der zusammengesetzten Objekte. Nur gut die Hälfte bis Zweidrittel wurden richtig zugewiesen. Manchmal wurden nur zwei der drei bzw. eines der drei Primitiven oder gar kein Primitiv erkannt. Dies liegt zum einen an den zuvor beschriebenen Problemen, zum andern aber daran, dass die Symbole einiger Probanden sich überschnitten. In diesen Fällen wurde nur das äußere der beiden verbundenen Primitiven erkannt (vgl. Abbildung 5.3). Wurde eines der Primitiven falsch klassifiziert, wurde das ganze Objekt als nicht erkannt gewertet.

5.1.2 Erneuter Test mit höherer Auflösung

Aufgrund der nur durchschnittlichen Ergebnisse des ersten Tests mit der Kamera DFK 21F04 der Marke THE IMAGING SOURCE, wurde der Test mit einer höher auflösenden Kamera der gleichen Firma durchgeführt. Es handelte sich um das Modell *DFK 31BF03* mit einer Auflösung von 1024×768 Pixel. Dabei wurde versucht



(a) (b) (c)

Abbildung 5.2: Ausschnitte des (a) entzerrten Kamerabildes, (b) des Schwellwertbildes und (c) des Ergebnisbildes. Dabei kann man deutlich erkennen, dass dem Rechteck ein Rechteck (grün) und ein Dreieck (blau) zugewiesen wurde

	richtig erkannt	
komplexes Objekt	ja	nein
	60%	40%
	60%	40%
	50%	50%

Tabelle 5.2: Konfusionsmatrix der komplexen Objekte-Erkennung mit einer Auflösung von 640×480 Pixel.

die Ausgangsbedingungen so wenig wie möglich zu ändern. Das Labor inklusive Projektor und Whiteboard war das selbe. Weiterhin wurde auf ähnliche Lichtverhältnisse geachtet, das selbe Objektiv verwendet und die gleiche Kameraposition eingehalten. Es waren nur zum Teil die gleichen Probanden.

Die Ergebnisse des zweiten Durchgangs sind erheblich besser als die des Ersten. Es wurden alle Recht- und Dreiecke richtig zugewiesen. Auch die Anzahl der erkannten Kreise hat sich stark erhöht. Trotzdem bleiben noch wenige Fehlentscheidungen, wobei der Anteil der segmentierungsbedingten Falschzuweisungen geringer ist als der Anteil der Zuweisungen der Rückweisungsklasse. Die nicht zugewiesenen Kreise ähnelten oft Ellipsen, bei denen sich bestimmte Teile der Kontur außerhalb des gültigen Schwellwertes befinden und so der Kreistest fehlschlägt (s. Kapitel 3.6.3). Auch bei den komplexen Objekten zeigte sich durch die höhere Auflösung eine wesentliche Steigerung der Erkennungsrate. Das Problem der über-

(a) (b) (c)

Abbildung 5.3: Ausschnitte des (a) entzerrten Kamerabildes, (b) des Schwellwertbildes und (c) des Ergebnisbildes. Bei den beiden rechten komplexen Symbolen wurde nur zwei der drei Primitive wegen Überschneidung der Kanten erkannt.

	vorhergesagte Klasse				
korrekte Klasse	□	△	○	Ω_0	Ψ_0
□	100%				
△		100%			
○	2,5%		91,25%	6,25%	

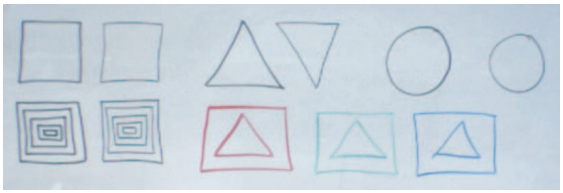
Tabelle 5.3: Konfusionsmatrix der Primitiverkennung. Der Test wurde mit einer Auflösung von 1024×768 Pixel durchgeführt.

schneidenden Kanten existiert zwar weiterhin, jedoch wurde keines der Objekte wegen Lücken in den Konturen falsch zugewiesen.

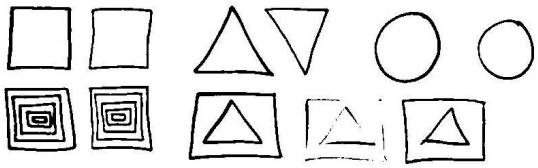
	richtig erkannt	
komplexes Objekt	ja	nein
	80%	20%
	70%	30%
	70%	30%

Tabelle 5.4: Konfusionsmatrix der komplexen Objekte-Erkennung des zweiten Tests mit einer Auflösung von 1024×768 Pixel.

5.2 Test mit unterschiedlichen Stiften



(a)



(b)

(c)

Abbildung 5.4: Ausschnitte des (a) entzerrten Kamerabildes, (b) des Schwellwertbildes und (c) des Ergebnisbildes. Der grüne Stift hebt sich zu wenig vom Hintergrund ab. Die zwei unterschiedlichen schwarzen Stifte zeigen nur bei sehr kleinen Symbolen Unterschiede.

In diesem Test wurden schwarze Stifte mit zwei unterschiedlichen Stärken, sowie Stifte mit unterschiedlichen Farben verwendet. Dabei zeigte sich, dass der grüne Stift sich nicht entschieden genug vom Hintergrund abhebt und es dadurch zu falschen Zuweisungen kommt (Abbildung 5.4). Die anderen Farben wurden erkannt. Unterschiedliche Strichstärken wirken sich nur bei sehr kleingezeichneten Symbolen aus. So können sehr kleingezeichnete Symbole, die mit dickerer Stiftstärke gezeichnet, nur sehr schlecht erkannt werden.

Kapitel 6

Zusammenfassung und Ausblick

Innerhalb dieses Kapitels werden zusammenfassend die erreichten Ergebnisse dargestellt. Im drauf folgenden Abschnitt wird ein kurzer Ausblick über nützliche Erweiterungen gegeben, die aufgrund des zeitlich vorgeschriebenen Rahmens keinen Platz in dieser Diplomarbeit fanden, es aber dennoch wert sind hier erwähnt zu werden.

6.1 Zusammenfassung

Zusammenfassend kann gesagt werden, dass alle Ziele, die zu Beginn der Arbeit gesteckt wurden, erreicht werden konnten. Es ist ein interaktives System bestehend aus einem handelsüblichen Whiteboard, Kamera, Projektor und Computer entwickelt worden. Dazu mussten Kamera und Projektor aufeinander kalibriert werden. Dies wurde mit der Hilfe einer Homographiematrix erledigt, die den Bereich des Projektors im Kamerabild auf ein ideales Bild abbildete. Weiterhin mussten die handgezeichneten Symbole erkannt werden. Dafür wurden die entzerrten Bilder zuerst vorverarbeitet und somit der Hintergrund vom Vordergrund getrennt. Dabei ist ein adaptives Schwellwertverfahren auf der Basis eines Integralbildes verwendet worden um die auftretenden Probleme der Randabschattung in den Bilder zu lösen. Die Primitive des Vordergrundes wurden mit Hilfe ihrer konvexen Hülle bestimmt. Zur Erkennung der zusammengesetzten Zeichen wurden die erkannten Primitive und ihrer geometrischen Relationen in einem topologischen Graphen gespeichert. Die Objekte des neuerzeugten Graphen konnten mittels exaktem Graph-Matching Verfahren den zuvor erstellten Modellen aus der Wissensbasis zugeordnet werden. Als Applikation wurde ein Spiel entwickelt, welches auf physikalischen Objekten basiert.

6.2 Ausblick

Um die Verwendung des interaktiven Whiteboardsystems noch einfacher und besser zu gestalten, werden im Folgenden nützliche Erweiterungen beschrieben.

6.2.1 Personenerkennung

Im derzeitigen Whiteboardsystem wird nicht erkannt, ob sich eine Person vor dem Whiteboard befindet und somit das Whiteboard im Kamerabild verdeckt. Dadurch wird eine automatische Erkennung von Veränderungen auf dem Whiteboard erschwert. Das in Kapitel 2.2.2 beschriebene Verfahren, bei dem bestimmte Bereiche im Kamerabild auf Änderungen untersucht und bei Überschreiten eines Schwellwerts als Person im Bild behandelt werden, ist bei einem System mit Projektion auf dem Whiteboard nur bedingt geeignet. Sind beispielsweise Animationen im Hintergrund müssten diese bei der Überprüfung des Schwellwertes mit in Betracht gezogen werden, da sonst das System bei animationsbedingten Änderungen des Hintergrunds von einer Person ausgehen würde.

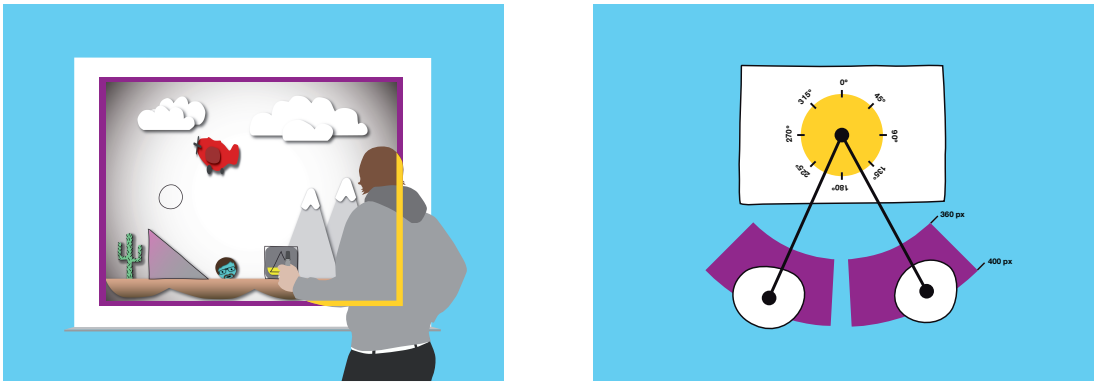
Ein Ansatz wäre es die Projektion des Bildes um einen gewissen Betrag zu verkleinern und durch einen weißen Rahmen zu ersetzen (s. Abbildung 6.1(a)). In diesem Rahmen finden keine systembedingten Veränderungen statt. Bewegt sich eine Person vor die Projektion, lässt sich dies durch die Veränderung im Rahmen feststellen. Bewegt sich die Person wieder aus dem Bild, kann das System das Whiteboard automatisch auf neugezeichnete Symbole untersuchen und ggf. darauf reagieren.

6.2.2 Objekterkennung

Die Objekterkennung liefert sehr gute Ergebnisse (s. Kapitel 5). Die Zusammensetzung komplexer Objekte durch einfache Primitive ist eine gute Eingabe für das entwickelte Spiel. Jedoch sollte man auch die Weiterentwicklung des Systems zu einem arbeitsunterstützenden Werkzeug betrachten. Dafür ist mehr als nur die Klassifikation von Primitiven notwendig und so könnte das System durch eine Schrifterkennung erweitert werden. Dadurch wäre der Benutzer in der Lage zusätzlich über geschriebene Worte mit dem System zu interagieren. Auch eine Weiterentwicklung der Symbolerkennung, die nicht nur auf Primitiven basiert, wäre sinnvoll.

6.2.3 Graphgenerierung

Die unter Kapitel 3.7.2 vorgestellten Attribute der Kanten beschreiben die Relationen zwischen den einzelnen Primitiven. Diese Attribute sind recht einfach gehal-



(a) (b)

Abbildung 6.1: Erweiterungen des Systems. (a) Die Projektionsfläche wird von einem Rahmen umgeben. Dieser wurde zur Verdeutlichung lila gefärbt. Ein Teil des Rahmens (gelber Bereich) wird von dem Benutzer verdeckt. (b) Die Relationen des Rechtecks zu den Kreisen. Beide Kreise befinden sich innerhalb des definierten Bereiches (lila).

ten, indem sie aus "isLeftOf, isRightOf, isBelow, isAbove, isIn, contains, isOutOfBounds" bestehen. Sie sind vollkommen ausreichend für das entwickelte Spiel und viele andere Anwendungsfälle. Trotzdem könnte für manche Programme diese Art von Attributen zu ungenau sein. Eine Alternative der bestehenden Methode ist die geometrischen Relationen in Form von Winkel und Distanz anzugeben. Dabei wird auch die Relation "isOutOfBounds" für zu weit entfernte Objekte nicht mehr benötigt, da diese Filterung über einen Schwellwert der Distanz erreicht werden kann. Abbildung 6.1(b) skizziert das Verfahren am Beispiel des Autos.

Anhang A

Adaptives Schwellwertverfahren mittels Integralbild

```
Algorithmus :AdaptivThreshold(in, out, width, height)
Input : in image, out image, width, height
Output : binary out image
for i ← 0 to widht do
    sum to 0;  for j ← 0 to height do
        if i = 0 then
            intImg[i, j] ← sum
        else
            intImg[i, j] ← intImg[i − 1, j] + sum
        end
    end
end
for i ← 0 to widht do
    for j ← 0 to height do
        x1 ← i − s/2 {border checking is not shown};
        x2 ← i + s/2;
        y1 ← j − s/2;
        y2 ← j + s/2;
        count ← (x2 − x1) × (y2 − y1);
        sum to intImg[x2, y2] − intImg[x2, y1 − 1] − intImg[x1 − 1, y2] + intImg[x1 − 1, y1 − 1];
        if (in[i, j] × count) ≤ (sum × (100 − t)/100) then
            out[i, j] ← 0
        else
            out[i, j] ← 255
        end
    end
end
```

Algorithmus A.1 : Adaptiver Thresholdalgorithmus mittels Integralbild nach Bradley und Roth [BR07]

Anhang B

Berechnung der Homographie durch Punktkorrespondenz

Mit Hilfe einer Homographie $\boldsymbol{H}$ wird ein Punkt p eines Bildes auf einen Punkt p' eines anderen Bildes abgebildet. Für jede Punktkorrespondenz gilt:

$$p'_i = \boldsymbol{H} p_i, \tag{B.1}$$

$$\begin{pmatrix} x' \\ y' \\ k' \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{B.2}$$

ergibt

$$\begin{aligned} x' &= h_{11}x + h_{12}y + h_{13} \\ y' &= h_{21}x + h_{22}y + h_{23} \\ k' &= h_{31}x + h_{32}y + h_{33} \end{aligned} \tag{B.3}$$

Da $k' = 1$ gewährleistet werden muss gilt:

$$\begin{aligned} \frac{x'}{k'} &= \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{33}} \\ \frac{y'}{k'} &= \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{33}} \end{aligned} \tag{B.4}$$

Daraus ergibt sich folgendes Gleichungssystem:

$$\begin{aligned} h_{11}x + h_{12}y + h_{13} - x'(h_{31}x + h_{32}y + h_{33}) = 0 \\ h_{21}x + h_{22}y + h_{23} - y'(h_{31}x + h_{32}y + h_{33}) = 0 \end{aligned} \tag{B.5}$$

und dies ergibt:

$$\begin{aligned} h_{11}x + h_{12}y + h_{13} - h_{31}x'x + h_{32}x'y + h_{33}x' = 0 \\ h_{21}x + h_{22}y + h_{23} - h_{31}y'x + h_{32}y'y + h_{33}y' = 0 \end{aligned} \tag{B.6}$$

Mit $\boldsymbol{h} = (\boldsymbol{H}_{11}, \boldsymbol{H}_{12}, \ldots, \boldsymbol{H}_{33})^{\mathrm{T}}$ als Spaltenvektor $\boldsymbol{h}$ ergibt sich daraus:

$$\begin{pmatrix} x & y & 1 & 0 & 0 & 0 & -x'x & -x'y & -x' \\ 0 & 0 & 0 & x & y & 1 & -y'x & -y'y & -y' \end{pmatrix} \boldsymbol{h} = 0 \tag{B.7}$$

Es müssen 4 linear unabhängige Punktkorrespondenzen sein und so ergibt sich:

$$\begin{pmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1x_1 & -x'_1y_1 & -x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1x_1 & -y'_1y_1 & -y'_1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x'_2x_2 & -x'_2y_2 & -x'_2 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -y'_2x_2 & -y'_2y_2 & -y'_2 \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -x'_3x_3 & -x'_3y_3 & -x'_3 \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -y'_3x_3 & -y'_3y_3 & -y'_3 \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -x'_4x_4 & -x'_4y_4 & -x'_4 \\ 0 & 0 & 0 & x_4 & y_4 & 1 & -y'_4x_4 & -y'_4y_4 & -y'_4 \end{pmatrix} \boldsymbol{h} = 0 \tag{B.8}$$

ergibt

$$\boldsymbol{A}\boldsymbol{h} = 0 \tag{B.9}$$

Somit kann $\boldsymbol{h}$ als Nullraum von $\boldsymbol{M}$ bestimmt werden.

Literaturverzeichnis

[AHU74] Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. *The Design and Analysis of Computer Algorithms.* Addison-Wesley, 1974.

[Bal81] D. H. Ballard. Generalizing the hough transform to detect arbitrary shapes. *Pattern Recognition*, 13(2):111–122, 1981.

[Ben02] E. Bengoetxea. *Inexact Graph Matching Using Estimation of Distribution Algorithms.* PhD thesis, École Nationale Supérieure des Télécommunications, Paris, France, 12 2002.

[BJ98] Michael J. Black and Allan D. Jepson. A probabilistic framework for matching temporal trajectories: Condensation-based recognition of gestures and expressions. In *ECCV '98: Proceedings of the 5th European Conference on Computer Vision-Volume I*, pages 909–924, London, UK, 1998. Springer-Verlag.

[BR07] Derek Bradley and Gerhard Roth. Adaptive thresholding using the integral image. *Journal of Graphics Tools*, 12(2):13–21, 2007.

[CFSV99] L.P. Cordella, P. Foggia, C. Sansone, and M. Vento. Performance evaluation of the vf graph matching algorithm. *Image Analysis and Processing, International Conference on*, pages 1172–1177, 1999.

[CFSV04] L.P. Cordella, P. Foggia, C. Sansone, and M. Vento. A (sub)graph isomorphism algorithm for matching large graphs. *IEEETPAMI: IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26, 2004.

[Chh98] Atul K. Chhabra. Graphic symbol recognition: An overview. In *GREC '97: Selected Papers from the Second International Workshop on Graphics Recognition, Algorithms and Systems*, pages 68–79, London, UK, 1998. Springer-Verlag.

[DP73] D. Douglas and T. Peuker. Algorithms for the Reduction of the Number of Points required to represent a digitised Line or its Caricature. *The Canadian Cartographer*, (10):112–122, 1973.

[Hou59] P. V. C. Hough. Machine analysis of bubble chamber pictures. *Proceeding of the International Conference on High Energy Accelerators and Instrumentation*, pages 554–556, 1959.

[JKW07] Wojciech Jaskowski, Krzysztof Krawiec, and Bartosz Wieloch. Learning and recognition of hand-drawn shapes using generative genetic programming. In *Applications of Evolutionary Computing, EvoWorkshops2007: EvoCOMNET, EvoFIN, EvoIASP, EvoInteraction, EvoMUSART, EvoSTOC, EvoTransLog*, volume 4448, pages 278–287. Springer Verlag, 2007. EvoWorkshops2007.

[KBS75] Carolyn Kimme, Dana Ballard, and Jack Sklansky. Finding circles by an array of accumulators. *Communications of the ACM*, 18(2):120–122, 2 1975.

[KT96] Rangachar Kasturi and Karl Tombre, editors. *Graphics Recognition, Methods and Applications, First International Workshop, University Park, PA, USA, August 10- 11, 1995, Selected Papers*, volume 1072 of *Lecture Notes in Computer Science*. Springer, 1996.

[LKML97] J. Lopez-Krahe, Enric Martí, and Josep Lladós. A system to understand hand-drawn floor plans using subgraph isomorphism and hough transform. *Mach. Vis. Appl.*, 10(3):150–158, 1997.

[Mah36] P. C. Mahalanobis. On generalized distance in statistics. *Proceedings of the National Inst. Sci. (India)*, 12:49–55, 1936.

[MB95] B. T. Messmer and H. Bunke. Automatic learning and recognition of graphical symbols in engineering drawings. In *Proc. Int. Workshop on Graphics Recognition, University Park, PA*, pages 33–43, 1995.

[MGMR02] S. Melnik, H. Garcia-Molina, and E. Rahm. Similarity flooding: a versatile graph matching algorithm and its application to schema matching. In *Data Engineering, 2002. Proceedings. 18th International Conference on*, pages 117–128, 2002.

[NA02] Mark Nixon and Alberto Aguado. *Feature Extraction and Image Processing*. Newnes, Oxford, 2002.

[OKM+88] A. Okazaki, T. Kondo, K. Mori, S. Tsunekawa, and E. Kawamoto. An automatic circuit diagram reader with loop- structure-based symbol recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(3):331–341, 1988.

[Ots79] N. Otsu. A threshold selection method from gray-level histogram. *Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, 1 1979.

[PKL+95] Lutz Priese, Jens Klieber, Raimund Lakmann, Volker Rehrmann, and Rainer Schian. Echtzeit-verkehrszeichenerkennung mit dem color structure code - ein projektbericht. Technical Report 4, Universität Koblenz-Landau, Institut für Computervisualistik, 1995.

[PM96] P. Perona and M. E. Munich. Visual input for pen-based computers. In *ICIP*, pages II: 173–176, 1996.

[Ros69] A Rosenfeld. Picture processing by computer. *ACM Comput. Surv.*, 1(3):147–176, 1969.

[Sau99] Eric Saund. Bringing the marks on a whiteboard to electronic life. In Norbert Streitz, Jane Siegel, Volker Hartkopf, and Shin íchi Konomi, editors, *CoBuild*, volume 1670 of *Lecture Notes in Computer Science*, pages 69–78. Springer, 1999.

[Sch99] Rainer Schian. *Automatische Bildauswertung zur dynamischen Schielwinkelmessung bei Kleinkindern und Säuglingen.* PhD thesis, Universität Koblenz-Landau, 1999. Fölbach Verlag, Koblenz, 161 Seiten.

[Ser88] J. Serra. *Image Analysis and Mathematical Morphology.* Academic Press, London, 1988.

[SFR96] Quentin Stafford-Fraser and Peter Robinson. Brightboard: A video-augmented environment. In *Proceedings of ACM CHI 96 Conference on Human Factors in Computing Systems*, volume 1 of *PAPERS: Virtual and Computer-Augmented Environment*, pages 134–141, 1996.

[SS94] Hanan Samet and Aya Soffer. A legend-driven geographic symbol recognition system. In *In Proceedings of the 12th International Conference on Pattern Recognition, volume II*, pages 350–355, 1994.

[ST01] C. Ah Soon and Karl Tombre. Architectural symbol recognition using a network of constraints. *Pattern Recognition Letters*, 22(2):231–248, 2 2001.

[Tit03] Peter Tittmann. *Graphentheorie: Eine anwendungsorientierte Einführung.* Fachbuchverlag Leipzig im Carl Hanser Verlag, 2003.

[Tur96] Volker Turau. *Algorithmische Graphentheorie.* Addison-Wesley, 1996.

[Ull76] J.R. Ullmann. An algorithm for subgraph isomorphism. *Journal of the Association for Computing Machinery*, 23:31–42, 1976.

[Wel93] Pierre Wellner. Interacting with paper on the digital desk. *Commun. ACM*, 36(7):86–96, 1993.

[WFS05] M. Wienecke, G. A. Fink, and G. Sagerer. Toward automatic video-based whiteboard reading. *International Journal on Document Analysis and Recognition*, 7(2-3):188–200, 7 2005.

[wHLZ02] Li wei He, Z. Liu, and Zhengyou Zhang. Why take notes? use the whiteboard capture system. Technical Report MSR-TR-2002-89, Microsoft Research (MSR), 9 2002.

[YPGD+96] Orly Yadid-Pecht, Moty Gerner, Lior Dvir, Eliyahu Brutman, and Uri Shimony. Recognition of handwritten musical notes by a modified neocognitron. *Mach. Vis. Appl.*, 9(2):65–72, 1996.

[YPIK90] H. K. Yuen, J. Princen, J. Illingworth, and J. Kittler. Comparative study of hough transform methods for circle finding. *Image and Vision Computing*, pages 71 – 77, 1990.

Internetquellen

[1] Dipartimento di Informatica e Sistemistica UniversitÃ degli studi di Napoli "Federico II" [online]. 2009. A direct graph matching tool. Available from: `http://amalfi.dis.unina.it/graph/db/vflib-2.0/` [cited 03 April 2009].

[2] Erin Catto. Electronic references [online]. 2009. Box2D is a feature rich 2d rigid body physics engine, written in C++. Available from: `http://www.box2d.org/` [cited 03 April 2009].

[3] Erin Catto. Electronic references [online]. 2009. The manual of the box2d physic engine. Available from: `http://www.box2d.org/manual.html` [cited 10 April 2009].

[4] Nokia Corporation [online]. 2009. Qt is a cross-platform application and UI framework. Available from: `http://www.qtsoftware.com/` [cited 03 April 2009].

[5] Willow Garage [online]. 2009. OpenCV (Open Source Computer Vision) is a library of programming functions mainly aimed at real time computer vision. Available from: `http://sourceforge.net/projects/opencvlibrary/` [cited 03 April 2009].