Institut für Parallele und Verteilte Systeme

Abteilung Verteilte Systeme

Universität Stuttgart
Universitätsstraße 38
D – 70569 Stuttgart

Diplomarbeit Nr. 3715

Erweiterung und Optimierung von Kommunikation, Koordination und Visualisierung eines Softwareagenten-Demonstrators zum dezentralen Lösen eines Puzzles

Thomas Knödler

Studiengang: Diplom- Informatik
Prüfer: Prof. Dr. K. Rothermel
Dipl.- Ing. Sebastian Abele
begonnen am: 27.01.2015
beendet am: 29.07.2015

CR- Klassifikation: I.2.11

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

CV	**C**omputer **V**ision
GUI	**G**raphical **U**ser **I**nterface
JADE	**J**ava **A**gent **D**evelopment **F**ramework
JPEG/JPG	**J**oint **P**hotographic **E**xperts **G**roup
MVC	**M**odel **V**iew **C**ontroller
PC	**P**ersonal **C**omputer
PNG	**P**ortable **N**etwork **G**raphics

Begriffsverzeichnis

Agentensystem	Mehrere Softwareagenten, die zusammen ein Problem lösen
Benutzeroberfläche	Der auf dem Bildschirm sichtbare Teil einer Software
Bilderkennung / Bildanalyse	Aus einem Foto werden technische Daten des abgebildeten Gegenstands ermittelt
OpenCV	Framework für Bildverarbeitung
Passprüfung	An zwei Puzzlestücken wird jeweils eine Kante auf Zusammengehörigkeit geprüft
Swing	GUI Framework für Java
Softwareagent	Computerprogramm mit selbstständigem Verhalten

Zusammenfassung

Das Gesamtsystem löst ein Legepuzzle mittels Softwareagenten. Den gleiche Softwareagenten wie sie in modernen Automatisierungssystemen Verwendung finden sollen. Ein Puzzle ist ein allgemein verständliches Beispiel das Verhalten dieser Softwareagenten aufzuzeigen, ohne dazu Industrieanlagen genauer kennen zu müssen. Die Aufgabe, einen sich dynamisch anpassenden Lösungsweg in einem komplexen, dezentralen und sich verändernden System mit autonom agierenden, untereinander kommunizierenden und so kooperierenden Elementen, den Softwareagenten, zu finden, ist die Gleiche.

In dieser Diplomarbeit ist ein bestehendes System um Konzepte, Hardware und Software erweitert worden. Hinzugekommen ist ein Kamera und Software zum Betreiben und Auswerten dieser. Ein Monitortisch zum Auflegen der Puzzlestücke und anzeigen der Informationen am Puzzle und um die einzelnen Puzzlestücke herum. Ein Konzept zum zuverlässigeren Lösen des Puzzles mittels Inseln.

Die einzelnen Puzzleteile werden durch Softwareagenten vertreten, welche miteinander kommunizieren, um die zugehörigen Nachbarpuzzlestücke zu finden. Zuerst werden mehrere Fotos von den liegenden Puzzlestücken angefertigt und diese ausgewertet. Dann wird für jedes Puzzlestück ein Agent gestartet, der sein Puzzlestück analysiert und dann mit den anderen Agenten Inseln bildet. Das Einpassen eines neuen Puzzlestücks in eine bestehende Insel oder das Zusammenfügen zweier Inseln geschieht über mehrere Kanten beteiligter Stücke und erhöht somit die Zuverlässigkeit des gewählten Ergebnisses. Während dieser ganzen Vorgänge können Informationen am Monitortisch angezeigt werden.

Abstract

The overall system solves a Jigsaw by software agents. The same software agents as they are to be used in modern automation systems. A jigsaw puzzle is a commonsense example to show the behavior of these software agents, without having to know industrial plants more precisely. The task of finding a dynamically adaptive approach in a complex, decentralized and changing system with acting autonomously, communicating with each other and so cooperating elements, the software agents, is the same.

In this thesis an existing system has been expanded to include concepts, hardware and software. Added is a camera and software for operating and evaluating this. A monitor table for placing the puzzle pieces and show the information about the puzzle and to the individual puzzle pieces. A concept for reliable Solve the puzzle by islands.

The puzzle pieces are represented by software agents which communicate with each other to find the corresponding neighboring puzzle pieces. First, several photos are made by the lying puzzle pieces and evaluated them. Then an agent is started for each puzzle piece, which analyzes his puzzle piece and then forms with the other agents Islands. Fitting a new puzzle piece into an existing island or joining two islands is done over several edges involved pieces, thus increasing the reliability of the selected result. During all operations information can be displayed on the monitor table.

1 Einleitung

Das Gesamtsystem löst ein Legepuzzle mittels Softwareagenten. Den gleiche Softwareagenten wie sie in modernen Automatisierungssystemen Verwendung finden sollen. Ein Puzzle ist ein allgemein verständliches Beispiel das Verhalten dieser Softwareagenten aufzuzeigen, ohne dazu Industrieanlagen genauer kennen zu müssen. Die Aufgabe, einen sich dynamisch anpassenden Lösungsweg in einem komplexen, dezentralen und sich verändernden System mit autonom agierenden, untereinander kommunizierenden und so kooperierenden Elementen, den Softwareagenten, zu finden, ist die Gleiche.

In dieser Diplomarbeit ist ein bestehendes System um Konzepte, Hardware und Software erweitert worden. Hinzugekommen ist ein Kamera und Software zum Betreiben und Auswerten dieser. Ein Monitortisch zum Auflegen der Puzzlestücke und anzeigen der Informationen am Puzzle und um die einzelnen Puzzlestücke herum. Ein Konzept zum zuverlässigeren Lösen des Puzzles mittels Inseln.

Die einzelnen Puzzleteile werden durch Softwareagenten vertreten, welche miteinander kommunizieren, um die zugehörigen Nachbarpuzzlestücke zu finden. Zuerst werden mehrere Fotos von den liegenden Puzzlestücken angefertigt und diese ausgewertet. Dann wird für jedes Puzzlestück ein Agent gestartet, der sein Puzzlestück analysiert und dann mit den anderen Agenten Inseln bildet. Das Einpassen eines neuen Puzzlestücks in eine bestehende Insel oder das Zusammenfügen zweier Inseln geschieht über mehrere Kanten beteiligter Stücke und erhöht somit die Zuverlässigkeit des gewählten Ergebnisses. Während dieser ganzen Vorgänge können Informationen am Monitortisch angezeigt werden.

2 Aufgabenstellung

2.1 Zielbestimmung

In dieser Diplomarbeit soll das bestehende Projekt des Puzzlers erweitert werden. Der Puzzler dient als Demonstrator um das Lösen eines Puzzles mittels Softwareagenten visualisieren zu können. Die einzelnen Puzzleteile sind durch Softwareagenten vertreten, welche miteinander kommunizieren um die zugehörenden angrenzenden Nachbarpuzzleteile zu finden. Dabei geht es weniger um eine optimierte Lösungsstrategie für den Lösevorgang des Puzzles, sondern um ein verständliches Aufzeigen und Darstellen der Einzelschritte in der Agentenkommunikation. Dieses soll auch für interessierte Laien und nicht nur für im System eingearbeitete Experten verständlich nachvollzogen werden können. Es soll zum einen das entstehende Puzzle und seine Einzelteile dargestellt werden, als auch die für das Finden der Nachbarpuzzleteile nötige Unterhaltung der Puzzleagenten.

2.1.1 Musskriterien

Nach dem Einlesen und Erkennen der Puzzleteile soll der Lösevorgang gestartet werden können. Danach soll der zeitliche Verlauf des Lösevorgangs transparent für Experten und interessierte Laien dargestellt werden.

2.1.2 Wunschkriterien

Die zur Lösung des Puzzles stattfindende Kommunikation der Agenten und deren Entscheidungsfindung soll in einem Art Schritt-für-Schritt Modus in Einzelschritten mit beliebiger Verzögerung zwischen den Schritten dargestellt werden. Die Lösungsstrategie des Puzzlers darf zur besseren Visualisierung umgebaut werden.

2.1.3 Abgrenzungskriterien

Es geht um die Darstellung eines Agentensystems. Das Lösen des Puzzles ist zweitrangig.

2.2 Einsatz

2.2.1 Anwendungsbereiche

Der Demonstrator soll dazu dienen, die Fähigkeiten von Agenten, komplexe Probleme zu lösen, anschaulich aufzeigen. Im Besonderen soll der Demonstrator bei Publikumsveranstaltungen am Institut eingesetzt werden. Ein Beispiel hierfür ist der Tag der Wissenschaft.

2.2.2 Zielgruppen

Mitarbeiter und Studenten des Instituts sollen den Demonstrator Besuchern vorführen.

2.2.3 Betriebsbedingungen

Der Demonstrator soll auf der zur Verfügung gestellten Hardware ausgeführt werden.

2.3 Umgebung

2.3.1 Software

Der Demonstrator soll auf Computern mit Java Laufzeitumgebung 1.7 ausgeführt werden.

2.3.2 Hardware

Der Demonstrator soll auf aktuellen, üblichen Computern, wie z.B. im PC- Pool des Instituts stehen, ausgeführt werden können. Außerdem ist darauf zu achten, dass die Software auf der zur Verfügung gestellten Hardware funktioniert, dies ist zurzeit ein Notebook, ein daran angeschlossener Beamer und eine USB- Kamera.

2.3.3 System-Schnittstellen

Schnittstellen sind zusätzlich zu denen bei aktuellen Computern üblichen wie Monitor, Tastatur und Maus noch ein optisches Gerät zu Bilderfassung wie eine Kamera und ein geeignetes Gerät zu Unterstützung der Visualisierung direkt an den Puzzlestücken wie einem Beamer.

2.4 Funktionale Anforderungen

Die jeweiligen Punkte sind mit ihrer dortigen Reihenfolge an das Lastenheft angelehnt und wurden teilweise ergänzt.

/PFA10/ Der am IAS schon vorhandene Puzzler soll weiter entwickelt werden.

/PFA20/ Mit geeigneten Mitteln wie zum Beispiel das Bilden von Gruppen soll der Brute-Force Ansatz zum Lösen verbessert werden. Dadurch soll der Kommunikationsaufwand und damit auch die Zeit zum Lösen verringert werden.

/PFA30/ Es sollen die bisher verwendeten Parameter überprüft und falls nötig verbessert werden. Damit ist nicht nur die prinzipielle Eignung der Parameter als Hilfreich zur Lösungserstellung beitragend gemeint sondern auch die Ermittlung dieser Parameter aus den Puzzlestücken zu überprüfen.

/PFA31/ Es sollen zusätzlich Parameter wie zum Beispiel Farbe eingefügt werden.

/PFA40/ Ein vom Puzzler lösbares Puzzle sollte bei Verwendung der gleicher Aufnahme immer gelöst werden. Ein Unterschied in der Reihenfolge der Kantenvergleiche und Inselbildung, wie sie durch die Nebenläufigkeit der Agenten entstehen, soll keinen Einfluss auf die Lösbarkeit haben.

/PFA50/ Die Visualisierung soll um hinzugefügten Parameter ergänzt werden.

/PFA51/ Es soll eine Visualisierung entstehen, welche direkt auf den Puzzlestücken arbeitet, um eine Zuordnung der Kommunikation zu den Puzzlestücken zu erleichtern.

2.5 Nichtfunktionale Anforderungen

/PNA10/ Die Implementierung soll in JAVA durchgeführt werden.

/PNA11/ Als Framework soll JADE und OpenCV Unterstützung bringen.

/PNA20/ Der Quellcode soll gut strukturiert und dokumentiert werden, damit zukünftige Änderungen und Erweiterungen im Code möglich bleiben.

/PNA30/ Es soll eine ausführliche Benutzeranleitung erstellt werden, auch um ein schnelles erlernen der Bedienung zu erleichtern. Dafür soll eine Schritt-für-Schritt Anleitung enthalten sein.

2.6 Anforderungen an die Benutzungsschnittstelle

/PBA10/ Um den nicht in das System eingearbeiteten Zuschauer nicht zu verwirren, soll das System unnötige Bedien- und Anzeigeelemente vermeiden.

/PBA20/ Um den nicht in das System eingearbeiteten Zuschauer nicht zu verwirren, soll das System zusätzlich Bedien- und Anzeigeelemente ausblendbar gestalten.

/PBA30/ Anzeigeelemente sollen nicht zu klein sein, um ein Zusehen aus etwas Entfernung, zum Beispiel in Gruppen, zu ermöglichen.

2.7 Qualitäts-Zielbestimmung

Produktqualität	sehr hoch	hoch	normal	nicht relevant
Funktionalität			X	
Richtigkeit			X	
Sicherheit (Security)				X
Interoperabilität			X	
Zuverlässigkeit			X	
Reife			X	

Produktqualität	sehr hoch	hoch	normal	nicht relevant
Fehlertoleranz			X	
Wiederherstellbarkeit			X	
Sicherheit (Safety)			X	
Benutzbarkeit		X		
Verständlichkeit		X		
Erlernbarkeit			X	
Bedienbarkeit		X		
Effizienz			X	
Zeitverhalten			X	
Verbrauchsverhalten			X	
Änderbarkeit		X		
Analysierbarkeit		X		
Modifizierbarkeit		X		
Übertragbarkeit		X		

Tabelle 2.1: Qualitäts- Zielbestimmung

2.8 Globale Testszenarien/Testfälle

2.8.1 Puzzle lösen

Bei einem ausgesuchten Testpuzzle soll der Ablauf des Lösevorgangs nachvollzogen werden können.

2.8.2 Programmbedienung

Innerhalb der Programmoberfläche soll der Lösevorgang gestartet und abgebrochen werden können. Das Hauptprogramm muss mit allen Tasks / Threads beendet werden können.

2.9 Entwicklungs-Umgebung

2.9.1 Software

Als integrierte Entwicklungsumgebung wird Eclipse verwendet. Die Programmiersprache ist Java, die Frameworks sind JADE und OpenCV.

2.9.2 Hardware

Die Entwicklung geschieht an Computern im PC- Pool oder vergleichbaren Computern.

2.9.3 Orgware

Als Orgware wird Microsoft Office benutzt.

2.9.4 Entwicklungs-Schnittstellen

Es gibt keine besonderen Hardware- Entwicklungs- Schnittstellen.

2.10 Durchführung

Die Arbeit wird nach dem IAS-Vorgehensmodell (Modell zur Softwareentwicklung) durchgeführt werden.

Der Stand der Arbeit und die Ergebnisse werden in regelmäßigen Abständen (ca. alle 2 Wochen) mit den Betreuern diskutiert.

Bei der Durchführung der Arbeit und der Anfertigung der Ausarbeitung werden die Richtlinien des IAS beachtet.

3 Projektplan

3.1 Projektstrukturplan

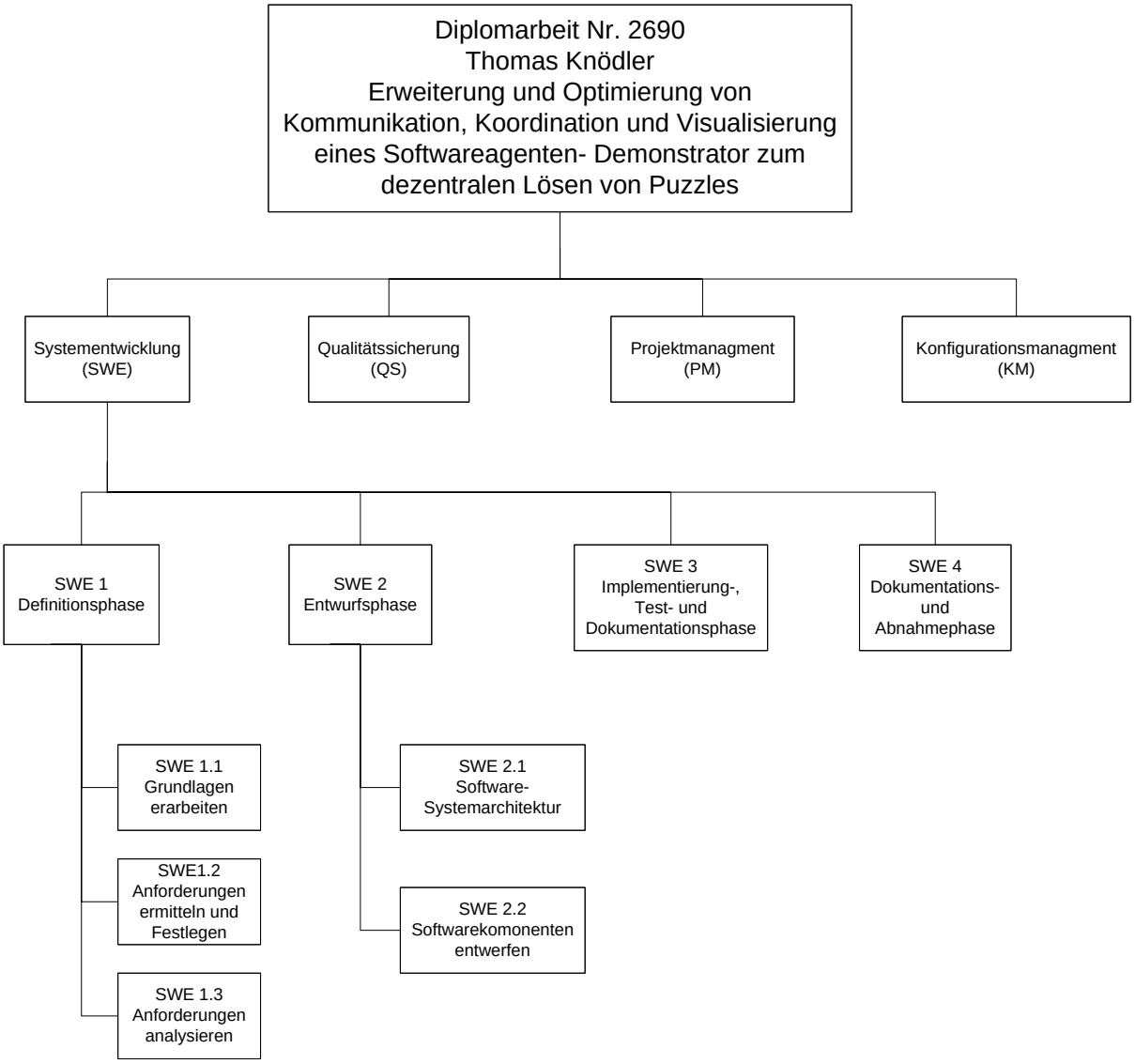

Abbildung 3.1: Projektstrukturplan

3.2 Arbeitspakete

3.3 Definitionsphase

3.3.1 Grundlagen erarbeiten

Es soll ein bestehendes System erweitert werden. Dazu wird eine Istzustandsanalyse durchgeführt. Das bestehende System benutzt das Agenten- Framework JADE. Es ist notwendig, sich in JADE

einzuarbeiten. Der Hauptzweck dieser Arbeit ist das Betreiben der Hardware und die Erstellung einer Visualisierung für dieses System. Dazu ist es notwendig, sich in die Programmierung des grafischen Benutzerinterfaces und die Programmierung der Nebenläufigkeit (Multithreading) von Java einzuarbeiten und für die Bilderfassung ein Konzept zu entwickeln.

3.3.2 Anforderungen ermitteln und festlegen

Die aus dem Lastenheft und durch Gespräche mit dem Betreuer entstandenen Anforderungen werden im Pflichtenheft festgehalten.

3.3.3 Anforderungen analysieren

Mit Hilfe der Analysemethode Use Cases werden die Anforderungen analysiert. Dadurch kann eine Benutzeroberfläche entworfen werden. Das Ergebnis dieses Arbeitsschrittes fließt in das Software- Systemmodell.

3.4 Entwurfsphase

3.4.1 Software- Systemarchitektur

Aus dem Systemmodell wird durch Verfeinerung und Konkretisierung die Systemarchitektur entwickelt in der auch festgelegt wird, wie die Anforderungen umgesetzt werden sollen. Auch gehört in diesem Arbeitsschritt das Festlegen der Schnittstellen der einzelnen Komponenten untereinander.

3.4.2 Software- Komponenten entwerfen

Die Systemarchitektur wird in Komponenten zerlegt und es wird festgelegt, wie diese später unabhängig voneinander umgesetzt werden können. Dabei sind auch die Rand- und Sonderfälle zu beachten und zu notieren, mit denen die späteren Testfälle ergänzt werden.

3.5 Implementierungs- Test- und Dokumentationsphase

Es werden die Komponenten aus der Systemarchitektur implementiert, durch die Rand- und Sonderfälle ergänzt, werden die Testfälle festgelegt und anschließend durchgeführt. Dabei wird immer auf eine vollständige Inline- Dokumentation geachtet, auch eventuelle Merkhilfen für die Hauptdokumentation werden notiert.

3.6 Dokumentations- und Abnahmephase

Die Hauptdokumentation wird überarbeitet und um Installations- und Benutzerhilfen ergänzt. Die Installations- und Benutzerhilfen werden an der laufenden Software überprüft. Dieses Gesamtwerk ist das Endprodukt.

3.7 Meilensteine

Nr.	Meilensteine	Produkte / Dokumente	Soll- Termin	Ist- Termin
1	Projektstart		Mo 12.01.2015	Mo 12.01.2015

2	Definitions-Review	Projektplan, Pflichtenheft, Software- Systemmodell, Definitions- Review-Protokoll	Fr 27.02.2015	
3	Entwurfs- Review	Software- Systemarchitektur, Spezifikation der Softwarekomponenten, Prüfspezifikation, Entwurfs-Review- Protokoll	Sa 28.03.2015	
4	Implementierungs-Review	Quellprogramme mit integrierter Dokumentation, Prüfprotokolle der Softwarekomponenten, Implementierungs- Review-Protokoll	Fr 08.05.2015	
5	Abnahme- Review	Installiertes System, Dokumentation, Prüfprotokoll, Abnahme-Review- Protokoll	Di 09.06.2015	
6	Puffer	4 Wochen Puffer	Fr 10.07.2015	
7	Projektabschluss	Abschlussbericht, Ausarbeitung	Fr 20.07.2015	

Tabelle 3.1: Meilensteine

4 Grundlegende Funktionen

4.1 Übersicht über die Anwendungsfälle

- Bild aufnehmen
- Puzzlestücke erkennen
- Puzzle lösen starten
- Programm neu starten
- Puzzlestücke erneut erkennen
- Puzzle erneut lösen
- Beenden
- Bild speichern
- abgespeichertes Bild einlesen

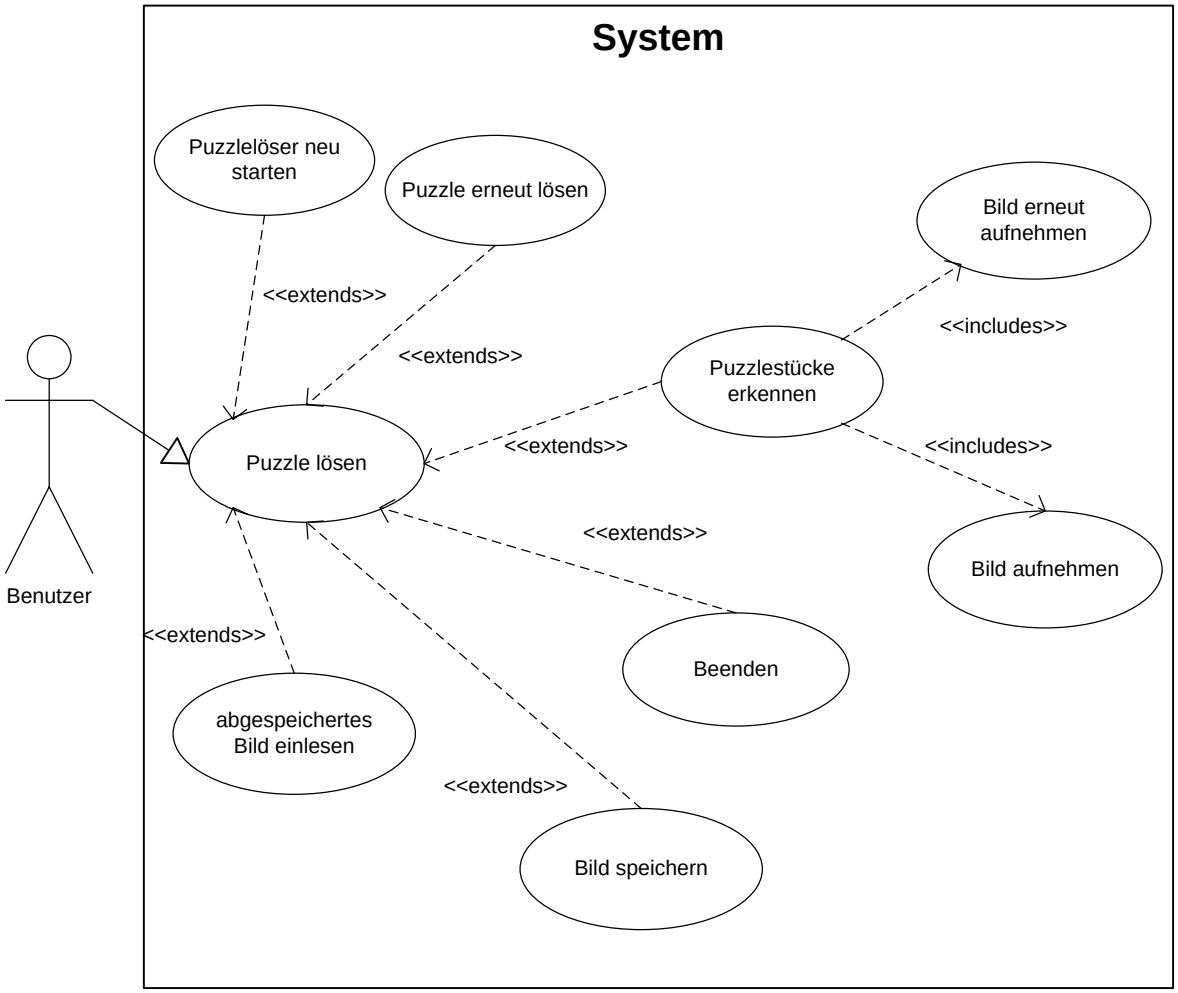

Abbildung 4.1: System

4.2 Beschreibung der Anwendungsfälle

4.2.1 Beschreibung des Anwendungsfalls „Bild aufnehmen“

Anwendungsfall	Das Bild wird aufgenommen.
Ziel	Die Kamera nimmt ein Bild auf.
Kategorie	**primär**
Externe Akteure	Benutzer
Vorbedingung	1. Das Programm muss ordnungsgemäß gestartet sein. 2. Optional: „Programm neu starten“ wurde gewählt.
Nachbedingung Erfolg	Ein auswertbares Bild wurde mit der Kamera erzeugt.
Nachbedingung Fehlschlag	Es wird eine Fehlermeldung angezeigt.
Auslösendes Ereignis	Der „Bild“- Button wurde gedrückt.
Beschreibung	Mit der Kamera wird ein Bild mit Puzzlestücken darauf zu Auswertung gemacht.
Erweiterungen	Keine
Alternativen	Eine abgespeicherte Bilddatei wird eingelesen.

Tabelle 4.1: Beschreibung des Anwendungsfalls „Bild aufnehmen“

4.2.2 Beschreibung des Anwendungsfalls „Puzzlestücke erkennen“

Anwendungsfall	Die Puzzlestücke werden erkannt.
Ziel	Die sich im Bild befindenden Puzzlestücke werden erkannt.
Kategorie	**primär**
Externe Akteure	Keine
Vorbedingung	Ein Bild wurde aufgenommen oder geladen.
Nachbedingung Erfolg	Alle Puzzlestücke im Bild und keine anderen Objekte wurden erkannt.

Nachbedingung Fehlschlag	Nicht alle oder falsche Puzzlestücke wurden erkannt.
Auslösendes Ereignis	Ein Bild wurde aufgenommen oder geladen.
Beschreibung	Der Vorgang zum Aufteilen des Gesamtbildes in die Bilder der einzelnen Puzzlestücke.
Erweiterungen	Keine
Alternativen	Keine

Tabelle 4.2: Beschreibung des Anwendungsfalls „Puzzlestücke erkennen“

4.2.3 Beschreibung des Anwendungsfalls „Puzzle lösen starten“

Anwendungsfall	Der Puzzlelösevorgang wird gestartet.
Ziel	Das Puzzle wird gelöst und die Lösung angezeigt.
Kategorie	**primär**
Externe Akteure	Benutzer
Vorbedingung	Die Puzzlestücke im Bild wurden erkannt.
Nachbedingung Erfolg	Die Ergebnisse werden angezeigt.
Nachbedingung Fehlschlag	Es wird eine entsprechende Fehlermeldung ausgegeben.
Auslösendes Ereignis	Der „Puzzle lösen“- Button wurde gedrückt.
Beschreibung	Es wird versucht das Puzzle zu lösen. Die Kommunikation der Puzzleagenten wird angezeigt.
Erweiterungen	Keine
Alternativen	Eine andere Aktion wählen.

Tabelle 4.3: Beschreibung des Anwendungsfalls „Puzzle lösen starten“

4.2.4 Beschreibung des Anwendungsfalls „Programm neu starten“

Anwendungsfall	Die Software wird neu Initialisiert.
Ziel	Neustart der Anwendung.
Kategorie	**sekundär**
Externe Akteure	Benutzer
Vorbedingung	Keine
Nachbedingung Erfolg	Das Programm ist neu initialisiert.
Nachbedingung Fehlschlag	Das Programm wurde nicht neu initialisiert.
Auslösendes Ereignis	Der „Puzzlelöser neu starten“- Button wurde gedrückt.
Beschreibung	Die Software soll in den Status direkt nach dem Start gehen, also noch ohne Benutzerinteraktion. Dies ist z.B. bei einer Fehlbedienung oder eines Fehllaufs des Lösealgorithmus nötig.
Erweiterungen	Keine
Alternativen	Keine

Tabelle 4.4: Beschreibung des Anwendungsfalls „Programm neu starten“

4.2.5 Beschreibung des Anwendungsfalls „Puzzlestücke erneut erkennen“

Anwendungsfall	Die Puzzlestücke werden erneut erkannt.
Ziel	Die sich im Bild befindenden Puzzlestücke werden erkannt.
Kategorie	**sekundär**
Externe Akteure	Benutzer
Vorbedingung	Ein Bild wurde aufgenommen oder geladen.
Nachbedingung Erfolg	Alle Puzzlestücke im Bild und keine anderen Objekte wurden erkannt.

Nachbedingung Fehlschlag	Nicht alle oder falsche Puzzlestücke wurden erkannt.
Auslösendes Ereignis	Der Button „Erneut erkennen“ wurde gedrückt.
Beschreibung	Der Vorgang zum Aufteilen des Gesamtbildes in die Bilder der einzelnen Puzzlestücke wird erneut durchlaufen. Es wird das gleiche Bild verwendet.
Erweiterungen	Keine
Alternativen	Keine

Tabelle 4.5: Beschreibung des Anwendungsfalls „Puzzlestücke erneut erkennen“

4.2.6 Beschreibung des Anwendungsfalls „Puzzle erneut lösen“

Anwendungsfall	Der Puzzlelösevorgang wird erneut gestartet.
Ziel	Das Puzzle wird gelöst und die Lösung angezeigt.
Kategorie	**sekundär**
Externe Akteure	Benutzer
Vorbedingung	Die Puzzlestücke im Bild wurden erkannt.
Nachbedingung Erfolg	Die Ergebnisse werden angezeigt.
Nachbedingung Fehlschlag	Es wird eine entsprechende Fehlermeldung ausgegeben.
Auslösendes Ereignis	Der Button „Puzzle lösen“ wurde gedrückt.
Beschreibung	Es wird erneut versucht das Puzzle mit den gleichen, vorhandenen Daten zu lösen. Die Kommunikation der Puzzleagenten wird angezeigt.
Erweiterungen	Keine
Alternativen	Keine

Tabelle 4.6: Beschreibung des Anwendungsfalls „Puzzle erneut lösen“

4.2.7 Beschreibung des Anwendungsfalls „Beenden“

Anwendungsfall	Die Software soll beendet werden.
Ziel	Die Software mit allen Threads soll beendet werden.
Kategorie	**primär**
Externe Akteure	Benutzer
Vorbedingung	Keine
Nachbedingung Erfolg	Die Software mit allen Threads ist beendet.
Nachbedingung Fehlschlag	Keine
Auslösendes Ereignis	Der „Beenden“- Button wurde gedrückt.
Beschreibung	Die Software soll mit allen Threads beendet werden.
Erweiterungen	Keine
Alternativen	Keine

4.2.8 Beschreibung des Anwendungsfalls „Bild speichern“

Anwendungsfall	Das Bild wird abgespeichert.
Ziel	Die gemachte Aufnahme wird als Datei abgespeichert.
Kategorie	**sekundär**
Externe Akteure	Benutzer
Vorbedingung	Ein Bild wurde aufgenommen oder geladen.
Nachbedingung Erfolg	Das aktuell verwendete Bild wird als Datei abgespeichert.
Nachbedingung Fehlschlag	Es wird eine entsprechende Fehlermeldung ausgegeben.
Auslösendes Ereignis	Der Button „speichern“ wurde gedrückt.

Beschreibung	Das aktuell verwendete Bild wird für spätere Wiederverwendung auf einen Datenträger als Bilddatei abgespeichert.
Erweiterungen	Keine
Alternativen	Keine

Tabelle 4.7: Beschreibung des Anwendungsfalls „Bild speichern“

4.2.9 Beschreibung des Anwendungsfalls „abgespeichertes Bild einlesen“

Anwendungsfall	Eine Bilddatei mit Puzzleteilen auswählen.
Ziel	Das Verzeichnis und die Datei mit dem Bild der Puzzleteile werden ausgewählt.
Kategorie	**sekundär**
Externe Akteure	Benutzer
Vorbedingung	3. Das Programm muss ordnungsgemäß gestartet sein. 4. Optional: „Programm neu starten“ wurde gewählt.
Nachbedingung Erfolg	Eine Datei wurde ausgewählt.
Nachbedingung Fehlschlag	Es wurde keine Datei ausgewählt.
Auslösendes Ereignis	Der „Datei auswählen“- Button wurde gedrückt.
Beschreibung	Ein „Datei öffnen Dialog“ öffnet sich und es wird eine Datei in einem Verzeichnis gesucht und ausgewählt. Die gewählte Datei ersetzt die aktuelle Datei.
Erweiterungen	Keine
Alternativen	Bild mit der Kamera aufnehmen.

Tabelle 4.8: Beschreibung des Anwendungsfalls „abgespeichertes Bild einlesen“

4.3 GUI-Konzept

4.3.1 Prinzipielle Bedienkonzepte

Um auch vor Publikum gezeigt werden zu können, soll der Demonstrator möglichst einfach gehalten werden und auch zu fein auflösende Darstellungen vermeiden. Da aber der interessierte Laie auch nicht gelangweilt werden sollte, ist trotzdem auf eine lückenlos nachvollziehbare Lösung des Puzzles zu achten. Deshalb sind großflächige Symbole dem tatsächlichen Statuscode vorzuziehen. Die Anzeige ist auf den normalen Durchlauf der Lösung zu optimieren, sollte aber bei Sonderfällen nicht an Übersicht und Verständlichkeit verlieren. Im Notfall besser den Vorgang mit „Puzzle nicht lösbar“ abbrechen als Chaos anrichten.

4.3.2 Elemente der Benutzungsschnittstelle

Eine Bildaufnahme der Puzzlestücke als Grundlage der Auswertung kann mit einem Button erstellt werden. Der Lösevorgang wird mit dem „Start“- Button ausgelöst. Läuft der Vorgang sollen weniger Einstellungsmöglichkeiten zur Verfügung stehen als vor Beginn. Zur späteren Wiederverwendung kann das Bild über geeignete Dialogfenster abgespeichert und auch wieder geladen werden. Über Dialogfenster können Parameter eingestellt werden. Der Gesamtvorgang lässt sich durch den „Neustart“- Button reseten und damit neu beginnen. Dabei gehen alle Informationen außer den extra Abgespeicherten verloren. Durch den Log- Level Auswahlschalter kann das Filtern der Log- Meldungen eingestellt werden.

4.4 Fensterstruktur und –gestaltung

Es soll nur ein Hauptfenster mit den Bedienelementen und allen anzuzeigenden Informationen geben. Zusätzlich werden Informationen über den Beamer direkt auf die Arbeitsfläche mit den Puzzlestücken projiziert.

4.5 Dialoggestaltung

Als Dateiauswahldialog wird der Standarddialog der verwendeten Programmieroberfläche benutzt. Die Fehlermeldungen dieses Dialogs werden als Popups angezeigt.

Die Agentenkommunikation erfolgt als Auflistung zeitlich sortiert von oben (älter) nach unten (jünger) in der Form:

4.5.1.1.1.1.1 Puzzleteil 1 < - - - Meldung - - - > Puzzleteil 2

Für die Puzzleteilinseln wird die gleiche Form benutzt. Als mögliche Meldungen sind folgende Texte (Beispiele) denkbar: nehmen Kontakt auf; überprüfen zusammenpassen; passt nicht; beenden Kommunikation; passen zusammen; bilden Insel.

Zusätzlich sollen diese Meldungen auch über den Beamer auf die Arbeitsfläche zwischen die beteiligten Puzzlestücke projiziert werden. Das nötige Verschieben und Drehen der Puzzlestücke soll auch über den Beamer direkt auf die Arbeitsfläche mitgeteilt werden.

Die möglichen Log- Meldungen werden mit einer Priorität versehen (Log- Level). Damit sind eine Filterung und dadurch eine Veränderung des Meldeumfangs zwischen wenig Meldungen zur Übersichtserhöhung oder mehr Meldungen für detailreichere Auskunft möglich.

5 Entwurf

5.1 Umgebungs- und Randbedingungen

Der Demonstrator soll vor Publikum das Lösen eines Puzzles mit Hilfe eines Agentensystems aufzeigen. Das zu entwickelnde System soll wiederverwertbare Teile aus den schon vorhandenen Projekten „Prototyp des Demonstrator“, „Bilderkennung“ und „Visualisierung“ übernehmen. Im entstehenden Demonstrator soll das entstehende Puzzle, aber auch die Agentenkommunikation dargestellt werden.

5.2 Grundlegende Entwurfsentscheidungen

Das Foto der Puzzleteile wird mit der Kamera erstellt. Die Software wird in Java, die grafische Benutzeroberfläche in Java/Swing entwickelt. Dabei werden die Frameworks JADE für die Agenten und OpenCV für die Bilderstellung und Verarbeitung verwendet. Bei dieser Art Software bietet es sich an, die einzelnen Komponenten aus der GUI- Komponente heraus aufzurufen und zu steuern, welche somit eine zentrale Rolle einnimmt. Da der Demonstrator für beobachten durch interessierte Laien entwickelt wird soll das System sich nicht in Details verlieren sondern ein gewisses Maß an Vereinfachung in der Darstellung tolerieren.

5.3 Diagramm der Software-Systemarchitektur

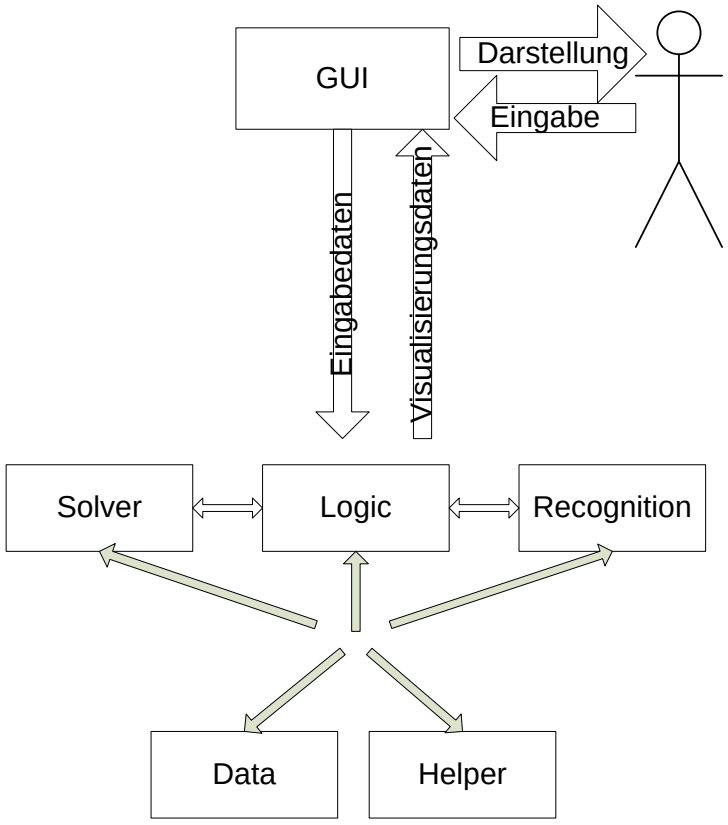

Abbildung 5.1: Systemarchitektur

Abbildung 5.1 zeigt den Datenfluss zwischen den einzelnen Komponenten. Der Benutzer macht seine Eingaben in der GUI. Diese leitet die Wünsche an die Logic Komponente weiter, welche die einzelnen Schritte ausführt. Die Ergebnisse werden wieder an die GUI weiter gegeben und dort angezeigt. Die Logic- Komponente lädt die Fotos, führt die nötigen Teilschritte zum Lösen des Puzzles durch und speichert die entstehenden Daten in der Daten- Komponente zwischen.

5.4 Softwarekomponenten

5.4.1 GUI

Diese Komponente beinhaltet die Kommunikation mit dem Benutzer. Er kann hier den Ablauf steuern und bekommt die Ergebnisse angezeigt. Da diese Komponente so zentral ist übernimmt sie auch den Rest der prinzipiellen Ablaufsteuerung, verteilt aber dazu Details an Unterkomponenten im Block Logic.

5.4.2 Logic

Hier befindet sich die nicht in andere Komponenten ausgelagerte Logik des Programms.

5.4.3 Helper

Hilfskomponenten, welche nichts mit dem Extrahieren der Puzzleteile und dem Extrahieren deren Charakteristika zu tun haben sondern für den Ablauf und die GUI gebraucht werden, aber größeren Umfang haben, sodass sich eine Auslagerung in den Helper angeboten hat.

5.4.4 Recognition

Komponenten, welche mit dem Extrahieren der Puzzleteile und dem Extrahieren deren Charakteristika zu tun haben. Es werden Teile aus den Vorprojekten übernommen.

5.4.5 Solver

Komponenten, welche mit dem eigentlichen Lösen des Puzzles zu tun haben. Es werden Teile aus den Vorprojekten übernommen.

5.4.6 Data

Durch Setter und Getter gekapselte Datenhaltung, es sind teilweise auch Wrapper zu bestehenden Komponenten. Entgegen der klassischen Vorgehensweise in der objektorientierten Programmierung, die Daten und Methoden zu deren Bearbeitung in der gleichen Klasse zu implementieren, werden die Methoden zur Bearbeitung der Daten, welche das Extrahieren der Puzzleteile und das Extrahieren deren Charakteristika und das Lösen des Puzzles betreffen in die Komponenten Logic, Recognition und Solver abgetrennt. Aufgrund der Deklaration als „static“ der Data- Klassen können diese dort aber wie in der Klasse befindend benutzt werden. Diese Vorgehensweise wurde aus den Vorprojekten so übernommen und hat den Vorteil, die Hauptkomponenten des Puzzlelösens hervorzuheben und austauschbarer zu machen.

5.5 Schnittstellendefinitionen

5.5.1 Schnittstelle „Foto laden“

Das Foto wird aus dem Dateisystem des Computers geladen.

5.5.2 Schnittstelle „Foto speichern“

Das Foto wird in das Dateisystem des Computers gespeichert.

5.5.3 Schnittstelle „Foto erstellen“

Die Kamera wird über das Framework OpenCV angesprochen.

5.5.4 Schnittstelle „Beamer“

Der Beamer hat eine für Monitore übliche Schnittstelle und wird deshalb als Zweitbildschirm am Computer angeschlossen. Er soll nur ein Fenster ausgeben, welches nötigenfalls bei Programmstart manuell mit der Maus dorthin gezogen wird.

5.5.5 Schnittstelle zu den Vorprojekten

Die benötigten Module der Vorprojekte werden, falls nicht direkt einbindbar, durch Wrapper gekapselt um die Schnittstellen anzugleichen.

5.5.6 Schnittstelle zum Benutzer

Die, bei auf dem Markt üblichen Computern, also Tastatur, Maus und Monitor- Kombination.

5.6 Bildverarbeitung

Zur Bildverarbeitung wird in dieser Arbeit das Framework OpenCV benutzt. Zusätzlich zur nativen C Bibliothek gibt es davon auch einen Wrapper nach Java.

Die Einzelschritte der automatischen Bildaufbereitung bis zur Übergabe an die schon vorhandene Puzzlestückextraktion werden aufgezeigt.

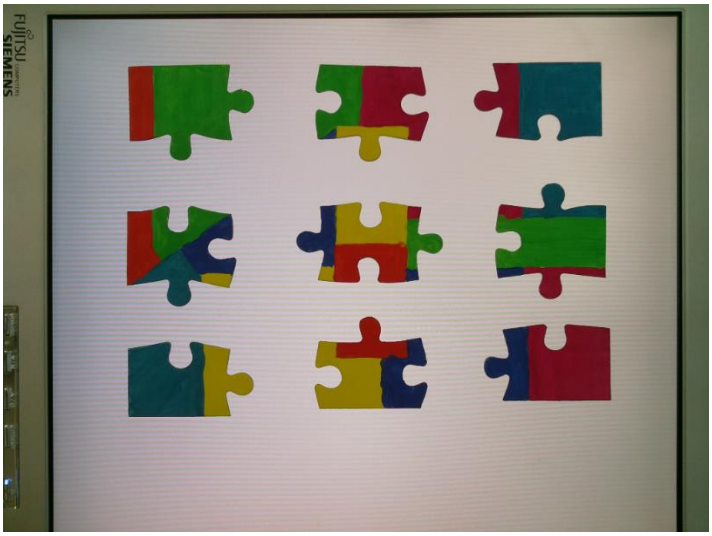

Abbildung 5.2: Originalbild der Kamera

Abbildung 5.3: Initbild

Aus den beiden Bildern mit und ohne Puzzlestücke kann der unveränderte Bereich durch Subtraktion der beiden Bilder ermittelt werden. Etwas minus das gleiche ergibt Null, in additiver Farbdarstellung entsprechend schwarz.

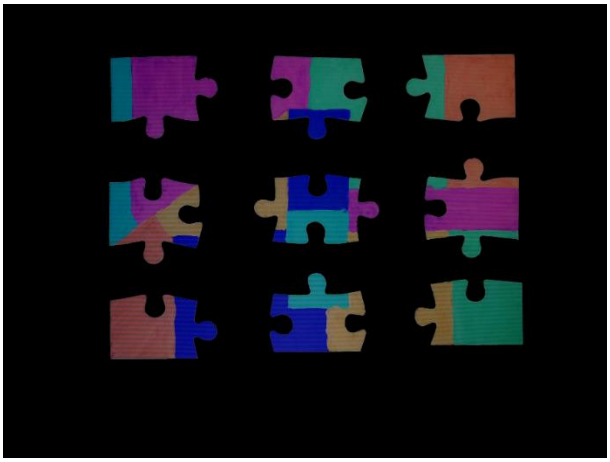

Abbildung 5.4: nach Subtraktion

Die nicht schwarzen Farben im Subtraktionsbild werden durch einen Grenzwertfilter zu weiß.

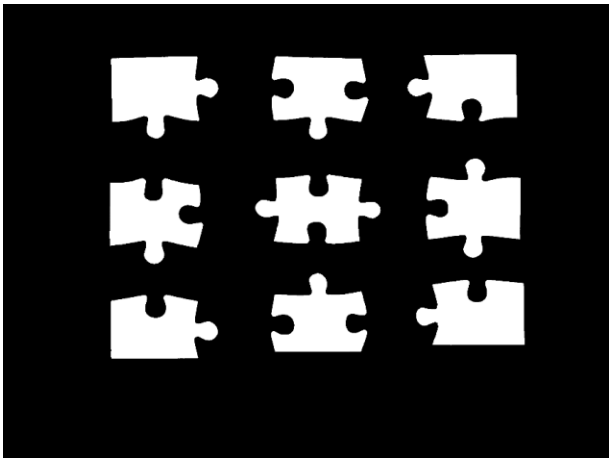

Abbildung 5.5: Maske

Damit ergibt sich zum einen das Bild für die Extraktion der einzelnen Puzzlestücke und eine Maske zum Entfernen des Hintergrundes.

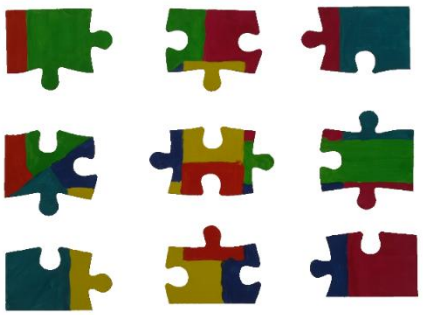

Abbildung 5.6: maskiertes Originalbild

Durch Addition eins komplett schwarzen Bildes (Farbwert 0) mit dem Originalbild an nur nicht schwarzen Stellen (Maske) in ein komplett weißes Bild (Vorbelegung) ergibt sich ein Abtrennen des störenden Hintergrundes.

Durch einrasten in RGB Farbwerte kann daraus die Farbgruppe für das Clustering gewonnen werden.

Filter für Störungen entfernen und Zwischenbilder schärfen wurden in dieser Darstellung nicht aufgeführt.

5.7 Clustering

Durch das Clustering soll der Kommunikationsaufwand verringert werden. Bisher wurde von jedem zu jedem Puzzlestück getestet, auch wenn die Antwort von vornherein schon fest stand. Durch das Einsortieren der Puzzlestücke während der Charakterisierung in Gruppen ist es möglich Nachrichten nur an in Frage kommende Gruppen zu senden. Zum Beispiel kann an ein Eckstück nur ein Randstück angebaut werden, die Anfrage an Mittelstücke ist Sinnlos und so wird diese Gruppe erst gar nicht angefragt.

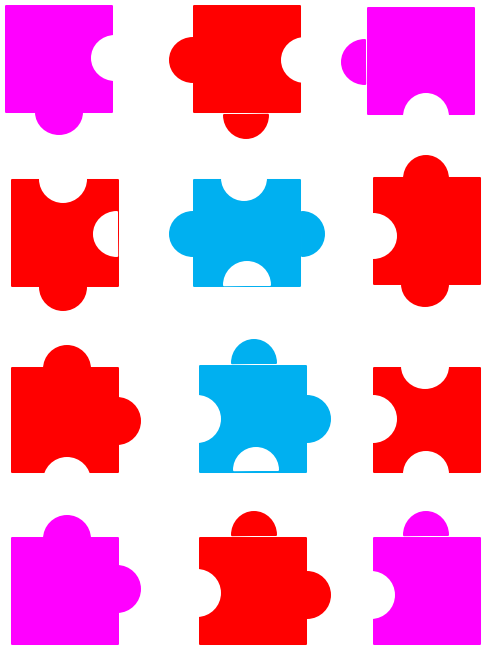

Abbildung 5.7: Beispiel Clustering

Mögliche Gruppen für das Clustering sind

- Randstück, nicht Randstück
- Zapfen, Buchse
- Farbe der Kante

5.8 Das Konzept der Insel

Es gibt keine Möglichkeit mit Informationen, welche aus nur einem Puzzlestück gewonnen werden können, herauszufinden, in welche Zeile / Spalte dieses Puzzlestück im Gesamtpuzzle gehört. Nur eine lokale Überprüfung des Zusammenpassens zweier beteiligter Puzzlestücke ist durch möglich. Passen die beiden Stücke zusammen ist dies der Anfang einer Insel. Die Inselbildung kann benutzt werden um Vorteile zu erreichen.

5.8.1 Inselbildung

Durch das Konzept der Inselbildung soll die Lösegenauigkeit verbessert werden. Es werden so nicht nur pro neues Puzzlestück ein Kantenpaar sondern zwei Kantenpaare verglichen welche

zueinander passen müssen. Für Inseln neu hinzugekommene Aufgaben übernehmen spezielle Inselagenden. Die auftretenden Fälle werden an Beispielen erklärt.

Abbildung 5.8: Beginn einer Insel

Theoretisch kann mit jedem Puzzlestück eine Insel gestartet werden. Die Auswahl von Eckstücken hat allerdings Vorteile. Die Darstellung muss nicht mehr verschoben werden da eine Ecke am Rand liegt. Bei einer Ecke kann nur mit Randstücken weiter gemacht werden. Dadurch werden beim neuen Puzzlestück zwei Kanten überprüft. Die Kontaktkante zum Eckstück und das Weiterführen der Puzzlekante durch das neue Stück.

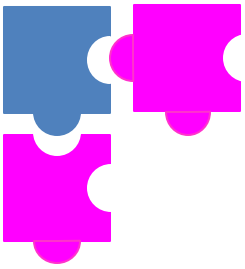

Abbildung 5.9: weiter mit Randstücken

Im nächsten Schritt wird die Insel zum Rechteck ergänzt in dem nicht Randstücke angebaut werden. Auch hier sind wieder zwei Kanten des neuen Stücks zur Insel zu überprüfen. Es ergibt sich wieder eine erhöhte Sicherheit durch diese zwei überprüften Kanten das richtige Puzzlestück gefunden zu haben.

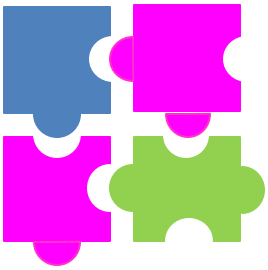

Abbildung 5.10: zum Viereck ergänzen

Diese beiden Schritte wiederholen sich so lange bis das Puzzle fertig ist. Immer nach dem Schritt „zum Rechteck ergänzen“ versucht der Inselagent sich mit anderen Inselagenten zu verschmelzen.

5.8.2 Insel verschmelzen

Der Grund für die angestrebte Rechteckform einer Insel ist die einfachere Verschmelzung zweier Inseln. So gibt es nur zwei prinzipiell mögliche Fälle.

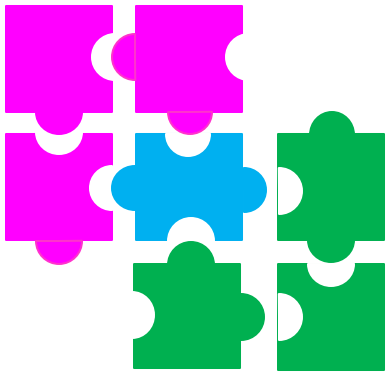

Abbildung 5.11: gemeinsame Ecke

Nur ein Puzzlestück ist in beiden Inseln eingebaut. Dies kann einfach festgestellt werden. Weiter geht es nun mit „zum Rechteck ergänzen“.

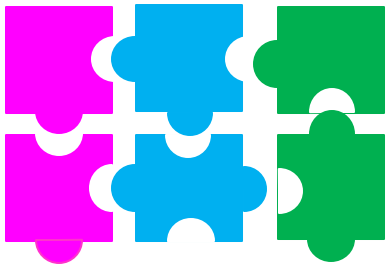

Abbildung 5.12: gemeinsame Kante

Es ist eine ganze Außenreihe von Puzzlestücken (horizontal oder vertikal) gemeinsam. Dies kann auch einfach festgestellt werden. Weiter geht es nun mit „Randstück(e) anfügen.

5.9 Ausgabegestaltung

Der Monitortisch dient nicht nur zum Abfotografieren der Puzzlestücke, sondern auch zum Ablegen des schon zusammengesetzten Puzzles und zum Anzeigen des Lösevorgangs. Dabei soll hauptsächlich über eine Veränderung der Farbe des Hintergrunds um das beteiligte Puzzlestück herum gearbeitet werden und dem Anzeigen des Textes am unteren Rand.

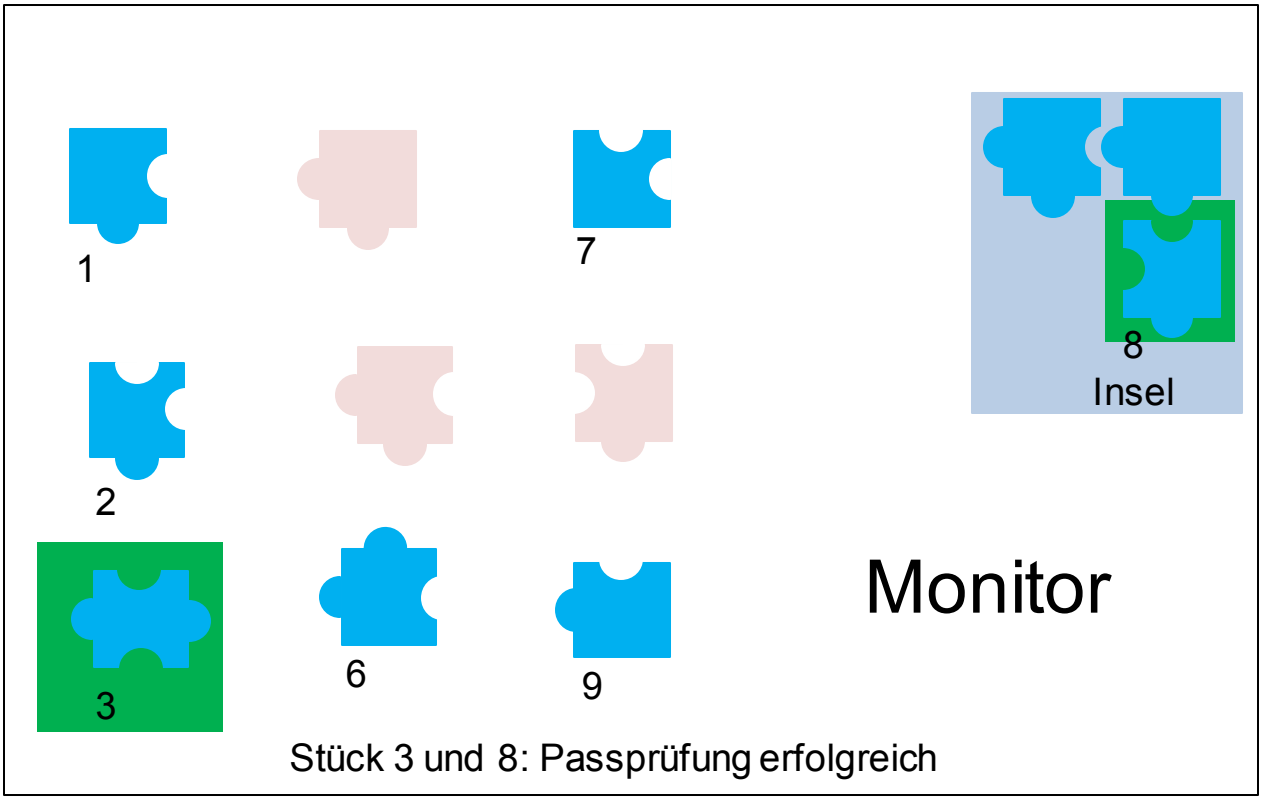

Abbildung 5.13: Ausgabe auf Monitortisch

In diesem Beispiel kommuniziert ein Puzzlestück (Nr. 8) aus der Insel mit einem noch nicht eingebauten Puzzlestück (Nr. 3). Der Text „Passprüfung erfolgreich“ wird unten angezeigt und begründet durch das positive Ergebnis der Hintergrund der beteiligten Puzzlestücke grün eingefärbt.

6 Spezifikation der Systemkomponenten

6.1 PC

6.1.1 Komponente data

Die Datenhaltung des Programms. Der Zugriff erfolgt über entsprechende Setter- und Getter-Methoden. Aufgrund der Deklaration als „static“ ist der Zugriff aus dem gesamten Programm heraus direkt möglich. Hier sind auch die Klassen welche als Vorlage für die Objekte der Datenelemente dienen untergebracht. Teilweise aus dem Vorprojekt da-2516-Emerich entnommen.

6.1.1.1 Klasse Edge.java

Hier wird eine Kante eines Puzzlestücks definiert.
(Quelle: da-2516-Emerich)

6.1.1.2 Klasse ListOfPuzzlePiece.java

In dieser Klasse wird die Liste der Puzzleteile abgespeichert.
(Quelle: da-2516-Emerich)

6.1.1.3 Klasse ProgramDefaults.java

Für das Programm hilfreiche Festlegungen werden in diese Klasse abgespeichert.

Attribut	Typ	Beschreibung
RASPI = 1	static final int	Konstante für RasPi
USBCAM = 2	static final int	Konstante für USBCam
FILESYSTEM = 3	static final int	Konstante für FileSystem
MAKEFOTO = "make Foto"	static final String	Konstante für Nachricht an Raspberry Pi
SHUTDOWN = "shutdown"	static final String	Konstante für Nachricht an Raspberry Pi
GUIEVENTPICRASPI = 1	static final int	Konstante für GUI Ereignis
GUIEVENTRASPISHUTDOWN = 2	static final int	Konstante für GUI Ereignis

Methode	Beschreibung

getImageDir	Das Bildverzeichnis auf dem Datenträger
getNamePicDefault	Defaultname für Bilddatei
getNamePicInit	Der Dateiname der Initdatei
getNamePicPuzzle	Der Dateiname der Puzzledatei
getNamePicChess	Der Dateiname der Schachbilddatei
getNameInitPicture	Der Dateiname für das Initbild
getNamePuzzlePicture	Der Dateiname für das Puzzlebild
getSelectedInput	Anfangsdefault für Input Methode
getWebCamNr	Nummer der USB Kamera
getBGRFilterDefault	Anfangswert für RGB Filter
getScale	Anfangswert für Größenveränderung im ShohWindow

6.1.1.4 Klasse PuzzlePiece.java

Hier wird ein Puzzlestück definiert.
(Quelle: da-2516-Emerich)

6.1.2 Komponente gui

Diese Komponente beinhaltet die Kommunikation mit dem Benutzer. Er kann hier den Ablauf steuern und bekommt die Ergebnisse angezeigt. Da diese Komponente so zentral ist übernimmt sie auch den Rest der prinzipiellen Ablaufsteuerung, verteilt aber dazu Deteils an Unterkomponenten in der Komponente logic.

6.1.2.1 Klasse FileChooser.java

Attribut	Typ	Beschreibung
chooser	JFileChooser	Referenz auf FileChooser
fenster	JDialog	Referenz auf aufrufendes Fenster
returnValue	int	Ergebnis
filter	FileFilter	Dateifilter

Methode	Beschreibung
FileChooser	Konstruktor
chooseFile	Ruft FileChooser auf
getFile	Liefert Ergebnisliste der Dateien zurück

6.1.2.2 Klasse MainWindow.java

Attribut	Typ	Beschreibung
mySystemAgent	SystemAgent	Referenz auf SystemAgent
myUSBWebCam	WebCamControl	Referenz auf USB Kamera
mySelectInput	SelectInput	Referenz auf Eingabewahl
myPicFromFiles	PicFromFiles	Referenz auf PicFromFiles
myPicFromRasPi	PicFromRasPi	Referenz auf PicFromRasPi
myPicFromUSBCam	PicFromUSBCam	Referenz auf PicFromUSBCam
myShowWindow	ShowWindow	Referenz auf ShowWindow
windowsClosing	WindowClosingAdapter	Zum beenden des GUI Teils
iconIAS	ImageIcon	Variable für IAS Icon
labelIAS	JLabel	Variable für IAS Label
panelFotos	JPanel	Variable für IAS Panel
originalFotoMat	Mat	Originalfoto der Kamera
originalFotoMatOnlyPuzzle	Mat	Original Puzzlestücke ohne Hintergrund
fotoMatdiscretBGR	Mat	Puzzlestücke in RGB Farben
fotoMatBW1C	Mat	Als 1 Kanal Farbe
fotoMatBW1Cinv	Mat	Als 1 Kanel Farbe invertiert
fotoArrayBW	byte[][]	Schwarz/Weiß Array
fotoArrayColor	byte[][][]	Farb Array

fotoWhiteBalance	Mat	Initbild
blackPicture	Mat	Schwarzbild
background	byte[]	Hintergrundfarbe
BGRGrenzwert	int[]	Grenzwert für RGB
Scale	int	Skalierung ShowWindow
fotoBereich	JScrollPane	Variable für Scrollpane
buttonStart	JButton	Variable für Button
buttonSearch	JButton	Variable für Button
buttonCharacteristics	JButton	Variable für Button
buttonNewPicture	JButton	Variable für Button
buttonReset	JButton	Variable für Button
buttonRasPiShutdown	JButton	Variable für Button
buttonEnde	JButton	Variable für Button
toolBar	JToolBar	Variable für Toolbar
buttonRGB	JRadioButton	Variable für Radiobutton
buttonBW	JRadioButton	Variable für Radiobutton
groupColor	ButtonGroup	Variable für Buttongruppe
sliderB	JSlider	Variable für Slider
sliderG	JSlider	Variable für Slider
sliderR	JSlider	Variable für Slider
sliderLabelB	JLabel	Label für sliderB
sliderLabelG	JLabel	Label für sliderG
sliderLabelR	JLabel	Label für sliderR
sliderPanel	JPanel	Panel für Slider
sliderPanelB	JSplitPane	Splitpane für Slider
sliderPanelG	JSplitPane	Splitpane für Slider

sliderPanelR	JSplitPane	Splitpane für Slider
sliderShow	JSlider	Variable für Slider
sliderLabelShow	JLabel	Label für Slider
sliderShowPanel	JPanel	Panel für Slider
sliderShowSplitPane	JSplitPane	Splitpane für Slider

Methode	Beschreibung
MainWindow	Konstruktor
setSystemAgent	Zum Setzen der Referenz
initMainWindow	Initialisierung, macht Fenster
work	Zeichnet Inhalte des Fensters neu
actionPerformed	Reaktion auf GUI Aktion
mouseReleased	Reaktion auf Mausereignis
mouseClicked	Reaktion auf Mausereignis
mouseEntered	Reaktion auf Mausereignis
mouseExited	Reaktion auf Mausereignis
mousePressed	Reaktion auf Mausereignis
mouseWheelMoved	Reaktion auf Mausereignis
paintFotos	Erstellt und zeichnet die Bilder in MainWindow
makeAllVersionOfFoto	Bildverarbeitung des Fotos

6.1.2.3 Klasse PicFromFiles.java

Attribut	Typ	Beschreibung
myChooseFiles	FileChooser	Referenz auf Dateiladendialog

fileInit	File	Datei mit Initbild
filePuzzle	File	Datei mit Puzzlebild
buttonOK	JButton	Button ok
buttonInitPic	JButton	Button Initbild erstellen
buttonPuzzlePic	JButton	Button Puzzlebild erstelln
labelInit	JLabel	Variable für Label
labelPuzzle	JLabel	Variable für Label
iconInit	ImageIcon	Variable für Icon
iconPuzzle	ImageIcon	Variable für Icon
labelIconInit	JLabel	Variable für Label
labelIconPuzzle	JLabel	Variable für Label
panelLabels	JPanel	Variable für Panel
panelPuzzles	JPanel	Variable für Panel
panelButtons	JPanel	Variable für Panel

Methode	Beschreibung
PicFromFiles	Konstruktor
initSelectFiles	Initialisierung, macht Fenster
dummy	Dummy
actionPerformed	Reaktion auf GUI Aktion

6.1.2.4 Klasse PicFromRasPi.java

Attribut	Typ	Beschreibung
mySystemAgent	SystemAgent	Referenz auf SystemAgent
fileInit	File	Referenz für Initdatei
filtePuzzle	File	Referenz für Puzzledatei

fileFotoDefault	File	Defaultname für Foto
buttonOK	JButton	Button ok
buttonInitPic	JButton	Button Initbild erstellen
buttonPuzzlePic	JButton	Button Puzzlebild erstelln
labelInit	JLabel	Variable für Label
labelPuzzle	JLabel	Variable für Label
iconInit	ImageIcon	Variable für Icon
iconPuzzle	ImageIcon	Variable für Icon
labelIconInit	JLabel	Variable für Label
labelIconPuzzle	JLabel	Variable für Label
panelLabels	JPanel	Variable für Panel
panelPuzzles	JPanel	Variable für Panel
panelButtons	JPanel	Variable für Panel

Methode	Beschreibung
PicFromRasPi	Konstruktor
initPicFromRasPi	Initialisierung, macht Fenster
dummy	Dummy
actionPerformed	Reaktion auf GUI Aktion

6.1.2.5 Klasse PicFromUSBCam.java

Attribut	Typ	Beschreibung
mySystemAgent	SystemAgent	Referenz auf SystemAgent
myWebCamControl	WebCamControl	Referenc auf USB Kamera
fileInit	File	Referenz für Initdatei
filtePuzzle	File	Referenz für Puzzledatei

picInit	Mut	Bild Init
picPuzzle	Mut	Bild Puzzle
buttonOK	JButton	Button ok
buttonInitPic	JButton	Button Initbild erstellen
buttonPuzzlePic	JButton	Button Puzzlebild erstelln
labelInit	JLabel	Variable für Label
labelPuzzle	JLabel	Variable für Label
iconInit	ImageIcon	Variable für Icon
iconPuzzle	ImageIcon	Variable für Icon
labelIconInit	JLabel	Variable für Label
labelIconPuzzle	JLabel	Variable für Label
panelLabels	JPanel	Variable für Panel
panelPuzzles	JPanel	Variable für Panel
panelButtons	JPanel	Variable für Panel

Methode	Beschreibung
PicFromUSBCam	Konstruktor
initPicFromUSBCam	Initialisierung, macht Fenster
dummy	Dummy
actionPerformed	Reaktion auf GUI Aktion

6.1.2.6 Klasse SelectInput.java

Attribut	Typ	Beschreibung
radioButtonRasPi	JRadioButton	Radiobutton für RasPi
radioButtonUSBCam	JRadioButton	Radiobutton für USBCam
radioButtonFileSystem	JRadioButton	Radiobutton für Dateisystem

groupRadioButton	ButtonGroup	Gruppe zum zusammen fassen
radioButtonPanel	JPanel	Variable für Panel
label	JLabel	Variable für Label

Methode	Beschreibung
SelectInput	Konstruktor
dummy	Dummy
initSelectInput	Initialisierung, macht Fenster
actionPerformed	Reaktion auf GUI Aktion

6.1.2.7 Klasse ShowWindow.java

Attribut	Typ	Beschreibung
image	BufferedImage	Bild zum Anzeigen
panelFoto	JPanel	Panelbereich für Foto
panelZusammen	JPanel	Panelbereich für zusammengesetztes Puzzle
panelMitte	JPanel	Panelbereich
icon	ImageIcon	Variable für Icon
labelIcon	JLabel	Variable für Label

Methode	Beschreibung
ShowWindow	Konstruktor
white	Hintergrund wird weiß zum Foto erstellen
calib	Malt openCV Schachbrett
setImage	Bild für panelFoto
renew	Andere Größe in panelFoto

together	Bild für panelZusammen

6.1.2.8 Klasse WindowClosingAdapter.java

Attribut	Typ	Beschreibung
exitSystem	boolean	nur Fenster weg oder auch ProgrammEnde

Methode	Beschreibung
WindowClosingAdapter	Konstruktor mit Parameter für exitSystem
WindowClosingAdapter	Konstruktor
windowClosing	Aufräumen (z.B. JADE) und exit

6.1.3 Komponente helper

Hilfskomponenten, welche nichts mit dem Finden der Puzzleteile im Bild, dem Extrahieren deren Charakteristika oder dem Lösen und Anzeigen des Puzzles zu tun haben sondern für den Ablauf und die GUI gebraucht werden, aber größeren Umfang haben, sodass sich ein Auslagern in eine Komponente helper angeboten hat.

6.1.3.1 Klasse ArrayCopy.java

Attribut	Typ	Beschreibung
target	byte[][]	Zielarray

Methode	Beschreibung
copy	Kopiert einen 2 dimensionalen Array ohne Referenzen

6.1.3.2 Klasse CopyFile.java

Methode	Beschreibung
copy	Kopiert eine Datei auf einem Datenträger

6.1.3.3 Klasse FlipFlop.java

Attribut	Typ	Beschreibung
Status	boolean	Beinhaltet den Status des Flipflips

Methode	Beschreibung
FlipFlop()	Konstruktor
getStatus()	Gibt aktuellen Status zurück
setStatus()	Setzt den Status neu
changeStatus()	Wechselt den Status

6.1.3.4 Klasse FotoLoader.java

Methode	Beschreibung
loadPuzzleFoto()	Gibt das geladene Foto zurück.

6.1.3.5 Klasse PictureTools.java

Methode	Beschreibung
static	Konstruktor für static Klasse Lädt openCV Lib
init	Initialisierung
convertBItoMatColor	Konvertiert von BufferedImage nach Mat
convertMatcolortoBI	Konvertiert von Mat nach BufferedImage
convertMatgraytoBI	Konvertiert von Mat nach BufferedImage
convertMatcolortoArray	Konvertiert von Mat nach Array
convertMatgraytoArray	Konvertiert von Mat nach Array

discretBGR	Rastet Bild auf BGR Farben ein
convertArrayColortoMatColor	Konvertiert von Array nach Mat
discretBW	Macht binäres schwarz/weiß Bild
filterBadPixel	Filtert Störungen
scalImage	Ändert Bildgröße
makeShowWindowImage	Erstellt Anzeigebild für zweiten Monitor
writeNumber	Schreibt Puzzlestück Nummer in ShowWindow unter das Puzzlestück
getPuzzlePieceBW	Gibt ein Puzzlestück zurück
convertArrayBWtoArrayBW	Gibt ein Puzzlestück zurück
getMaxDim	Gibt die Größe des Bildbereichs des größten Teilbildes zurück
eliminateDots	Löscht Störungen

6.1.3.6 Klasse SelectBackground.java

Attribut	Typ	Beschreibung
value	Byte	Hintergrundfarbe
valid	boolean	Farbwert ist gültig

Methode	Beschreibung
identifyBackground	Ermittelt Farbwert
getBackgroundColor	Gibt Farbwert zurück

6.1.3.7 Klasse WebCamControl.java

Attribut	Typ	Beschreibung
kameraDevice	VideoCapture	Referenz auf Hardware
matImage	Mat	Aufgenommenes Bild

Methode	Beschreibung
WebCamControl	Konstruktor
init	Zum Initialisieren der Hardware
getMatImage	Bild erstellen
initOK	Testet ob init erfolgreich
close	Hardware freigeben

6.1.4 Komponente libs

Externe Programmbibliotheken, zur Zeit nur das Agentenframework jade.jar.

6.1.5 Komponente logic

Diese Komponente beinhaltet Teile der Steuerung des Programmablaufs, welche nichts direkt mit der Bilderkennung oder dem Lösen des Puzzles zu tun haben, aber groß genug sind um nicht in der Komponente gui gelassen werden zu können. Enthält „die“ Main- Routine zum Starten des Programms.

6.1.5.1 Klasse JadeStartStop.java

Attribut	Typ	Beschreibung
myMainContainer	AgentContainer	Agentencontainer
myAgentControl	AgentController	Controller des Agentencontainers
sysAgentNeo	SystemAgent	Systemagent

Methode	Beschreibung
jadeStart	Startet JADE
setMainWindow	Zum Setzen der Referenz
jadeStop	Stoppt JADE

6.1.5.2 Klasse Main.java

Attribut	Typ	Beschreibung
myMainWindow	MainWindow	Referenz auf MainWindow
mySystemAgent	SystemAgent	Referenz auf SystemAgent

Methode	Beschreibung
static main	Die Main Methode

6.1.5.3 Klasse ProgramStatuses.java

Hier sind zentral abgelegte Programmstatuse.

Attribut	Typ	Beschreibung
runSearchStatus	FlipFlop	Ist search schon gestartet worden?
runCharacteristicsStatus	FlipFlop	Ist Characteristics schon gestartet worden?
runSolveStatus	FlipFlop	Ist Solve schon gestartet worden?
pictureSelected	FlipFlop	Wurde schon ein Bild ausgewählt?
currentInput	int	Aktuelle Nummer des Eingangskanal
currentInitPicture	File	Datei des aktuelles Initbild
currentPuzzlePicture	File	Datei des aktuelles Puzzlebild

6.1.5.4 Klasse SystemAgent.java

Attribut	Typ	Beschreibung
myMainWindow	MainWindow	Referenz auf MainWindow
msg	SCLMessage	Agentennachricht

Methode	Beschreibung
setup	Agentenaktionen
setMainWindow	Zum Setzen der Referenz
onGuiEvent	Reaktion auf GUI Ereignis
takeDown	Beenden des Agenten

6.1.6 Komponente recognition

Komponente der Bildverarbeitung zum Finden und Ausschneiden der Puzzleteile aus dem Bild und dem Extrahieren deren Charakteristika. Aus dem Vorprojekt ba-2569-Jung entnommen.

6.1.6.1 Klasse CharacteristicsRecognition.java

Extern (Quelle: [Jung14])

6.1.6.2 Klasse ImageRecognition.java

Extern (Quelle: [Jung14])

6.1.7 Komponente solver

Komponenten zum eigentlichen Lösen des Puzzles.
Extern (Quelle: Sebastian Abele)

6.1.8 Komponente tests

Hier sind Programme zum Testen von Teilaufgaben.

6.1.8.1 Klasse CalibCamera.java

Attribut	Typ	Beschreibung
windowsClosing	WindowClosingAdapter	Zum beenden des GUI Teils
mySystemAgent	SystemAgent	Referenz auf SystemAgent
myShowWindow	ShowWindow	Referenz auf ShowWindow
foto	Mat	Variable für Foto
icon	ImageIcon	Variable für Icon
label	JLabel	Variable für Label
panelFotos	JPanel	Variable für Panel

fotoBereich	JScrollPane	Variable für Scrollpane
buttonPicFromRasPi	JButton	Variable für Button
buttonChess	JButton	Variable für Button
buttonRasPiShutdown	JButton	Variable für Button
buttonEnde	JButton	Variable für Button
toolBar	JToolBar	Variable für Toolbar
fileChess	File	Variable für Bilddatei Schach
fileFotoDefault	File	Variable für Ergebnisdatei

Methode	Beschreibung
static main	Die Main Methode
CalibCamera	Konstruktor
dummy	dummy
getFoto	Foto erstellen
actionPerformed	Nach GUI Ereignis auszuführen

6.2 Raspberry Pi

6.2.1 data

Die Datenhaltung des Programms. Der Zugriff erfolgt über entsprechende Setter- und Getter-Methoden. Aufgrund der Deklaration als „static“ ist der Zugriff aus dem gesamten Programm heraus direkt möglich.

6.2.1.1 Klasse ProgramDefaultsRasPi.java

Hier sind zentral abgelegte Programmdefaults und Konstanten.

Attribut	Typ	Beschreibung
MAKEFOTO = “make Foto”;	static final String	Konstante für empfangbare Nachricht an Raspberry Pi

SHUTDOWN = “shutdown”	static final String	Konstante für empfangbare Nachricht an Raspberry Pi

Methode	Beschreibung
static String getMainHostname()	Liefert den Hostnamen des Maincontainers
static String getMainPortNr()	Liefert die Portnummer des Maincontainers
static String getLocalHostname()	Liefert den lokalen Hostnamen
static String getLocalPortNr()	Liefert die lokale Portnummer
static String getContainerName()	Liefert den lokalen Containernamen

6.2.2 jade_agents

Sammlung der Agenten und eine Routine zum Starten und Beenden der JADE Umgebung auf dem Raspberry Pi.

6.2.2.1 Klasse SystemAgentRasPi.java

Attribut	Typ	Beschreibung
msg	SCLMessage	Agentennachricht

Methode	Beschreibung
setup	Agentenaktionen
takeDown	Beenden des Agenten

6.2.2.2 Klasse JadeStartStopRasPi.java

Attribut	Typ	Beschreibung
myMainContainer	AgentContainer	Agentencontainer
myAgentControl	AgentController	Controller des Agentencontainers
sysAgentNeo	SystemAgentRasPi	Systemagent

Methode	Beschreibung
jadeStart	Startet JADE
jadeStop	Stoppt JADE

6.2.3 logic

Die Steuerung des Programmablaufs außerhalb der Agenten. Enthält „die“ Main- Routine zum Starten des Programms.

6.2.3.1 Klasse MainRasPi.java

Attribut	Typ	Beschreibung
mySystemAgent	SystemAgentRasPi	Referenz auf Systemagent

Methode	Beschreibung
static main	Die Main Methode
MainRasPi	Konstruktor

6.2.4 libs

Externe Programmbibliotheken, zur Zeit nur das Agentenframework jade.jar.

7 Prüfung der Software

7.1 Prüfanforderungen

Die Aufgabe dieser Diplomarbeit ist das Erweitern eines bereits bestehenden Systems um Hardware und das Weiterentwickeln einiger der bestehenden Komponenten. Die veränderten Komponenten und die hinzugefügten Komponenten sind einzeln durch Testprogramme zu Überprüfen. Ebenso ist der Gesamtvorgang durch Testläufe zu Überprüfen.

7.2 Methoden der Prüfung

Die in den Prüffällen aufgeführten möglichen Durchläufe und deren Sollverhalten werden durch Funktiondabdeckung überprüft. Dabei wird folgendes notiert:

- Testfall
- Datum
- Version der getesteten Software
- Sollverhalten erreicht
- Abweichungen (falls vorhanden)

Die Prüffälle sollen nicht nur den Normalfall beinhalten, sondern auch mögliche Fehlbedienungen und Fehlbedingungen beachten. Dazu sind vom gleichen Grundvorgang (wie Programm lässt sich beenden) auch Variationen (gleich nach Start, nach dem Finden der Puzzlestücke aber ohne Lösevorgang, …) zu bestimmen. Es werden in solchen Fällen allerdings nicht alle aus der Kombinatorik dadurch entstehenden Prüffälle einzeln aufgelistet sondern nur die Besonderheit, welche intuitiv um restliche Abläufe ergänzt wird.

7.3 Prüfkriterien

Falls das beobachtete Verhalten dem Sollverhalten entspricht ist die Prüfung bestanden.

8 Komplexitätsanalyse

Der Gesamtvorgang unterteilt sich in die Abschnitte Bildanalyse und Lösevorgang.

8.1 Bildanalyse

Vor dem eigentlichen Lösevorgang steht die Bildanalyse. Es müssen alle Puzzlestücke in dem Ausgangsfoto gefunden und danach jedes einzelne Puzzlestück noch für sich analysiert werden. Für den ersten Vorgang ergibt sich ein Aufwand von O(n). Der zweite Vorgang wird durch die Agenten parallelisiert und beläuft sich dadurch nur auf O(k). Zusammen genommen ergibt sich aber immer noch ein Aufwand von O(n).

8.2 Lösevorgang

Im Prinzip muss zum Lösen eines Puzzles jede Kante eines Puzzlestücks mit jeder Kante aller anderen Puzzlestücke verglichen werden. Also ohne Sonderfälle zu berücksichtigen jedes Puzzlestück mit jedem anderen Puzzlestück multipliziert mit vier Kanten. Aufgrund der mechanischen Randbedingung eines starren Puzzlestücks erübrigt sich die Überprüfung mit den eigenen Kanten. Dies kann bei der Aufwandsanalyse noch berücksichtigt werden. Bei einem flachen Puzzle, welches eben auf einem Tisch zusammen gesetzt werden kann, hat nicht jedes Puzzlestück vier Kontaktkanten. Randstücke und Eckstücke haben ein oder zwei Kontaktkanten weniger. Wie viele Außenkanten bei einem realen Puzzle vorhanden sind hängt allerdings von dessen Geometrie ab. Ein Beachten oder nicht Beachten der Außenkanten hat keine Änderung der Anzahl der Puzzlestücke zur Folge und somit keinen Einfluss auf die Komplexitätsklasse und bleibt deshalb unberücksichtigt. Es wird also vereinfacht die Anzahl der Vergleiche Puzzlestück zu Puzzlestück ermittelt. In Abbildung 8.1: Komplexität des Lösevorgangs

sind die nötigen Vergleiche bildlich angedeutet. Damit lässt sich das Ergebnis $O(n^2)$ schon erahnen.

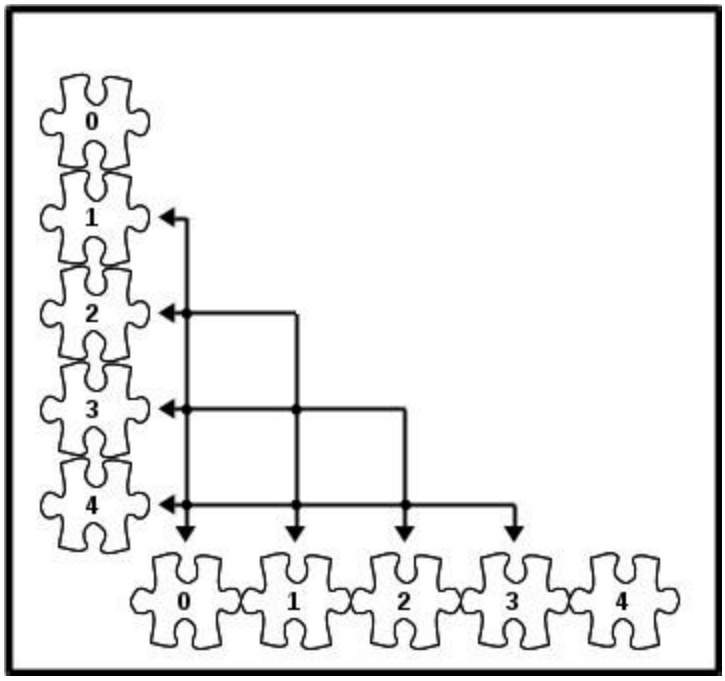

Abbildung 8.1: Komplexität des Lösevorgangs

Ein Vergleich jedes mit jedem, wobei der Vergleich eines Puzzlestücks mit demselben Puzzlestück ungezählt bleibt (Reflexivität) und ein Vergleich dieses Puzzlestücks mit einem Anderen das gleiche Ergebnis wie ein Vergleich des Anderen mit diesem Puzzlestück ergibt (Symmetrie) und somit nur ein mal ermittelt werden muss, ergibt bei N Puzzlestücken folgende Aufsummierung: Das Erste mit allen Restlichen ergibt N-1 Vergleiche, das Zweite mit den nun übrig gebliebenen Restlichen ergibt N-2, und mit diesem Muster weiter bis der übrig gebliebene Rest auf Nichts schrumpft, also das Vorletzte mit dem Letzten N – (N - 1) und das Letzte mit keinem mehr ergibt N – N. Zusammengefasst ist dies N/2 * (N – 1), der n-1-ten Dreieckszahl, was sich auch aus der Kombinatorik als Ergebnis des Binomialkoeffizienten N über 2 ergibt.

Dies liegt in $O(n^2)$. Die Art der Ermittlung dieser Vergleiche erinnert sehr stark an einfache Sortierverfahren. Dort wie hier lässt sich durch keinen einfachen Trick die Komplexitätsklasse zur nächst besseren.

Eine Änderung dieses Wertes ist leider auch nicht bei der Verwendung nur einer Insel zu erreichen. Dort muss ein Puzzlestück bei zwei Kanten anderer Puzzlestücke überprüft werden und nur die beim Ersten erfolgreichen Teststücke auch beim Zweiten geprüft werden. Falls beim Ersten alle Passprüfungen erfolgreich sind ergeben sich für das Zweite ebenso viele Passprüfungen (Worst Case Fall), also nicht weniger als ohne Insel. Nur im Fall weniger erfolgreicher Passprüfungen zum Ersten müssen zum Zweiten nur diese geprüft werden.

Erst der Einsatz mehrerer Inseln ergibt eine Parallelisierung und somit eine Beschleunigung durch die Verwendung von Inseln. Im Idealfall werden dies immer halb so viele Inseln wie im vorhergehenden Lauf sein, es ergibt sich damit ein Binärbaum dessen Blätter die einzelnen Puzzlestücke (N) sind, eine Baumebene darüber sind aus jeweils zwei Puzzlestücken eine Insel gebildet worden, somit N/2 Inseln, und so fort. Die Baumhöhe eines Binärbaums und damit die

Laufzeit ist ld (N). Dies ist zu optimistische für unsere Lösungsstrategie, daher für unsere Inseln andere Kriterien zum Entstehen herangezogen werden als in der obigen Annahme.

8.3 Gesamtvorgang

Da der Lösevorgang nur mit kompletter und erfolgreicher Bildanalyse starten kann addieren sich die beiden Laufzeiten $O(n) + O(n^2)$. Wegen $O(n) \in O(n^2)$ ergibt sich damit $O(n^2)$ für den Gesamtvorgang, der bestimmende Teil ist somit der Lösevorgang. Dieser kann nur unter optimalen Bedingungen verbessert werden, welche in der Praxis nicht zur Verfügung stehen.

9 Installation

9.1 Hardware

Als Hardware Voraussetzung wird ein handelsüblicher PC, ein Raspberry Pi und dessen Standard-Kamera benötigt. Des Weiteren noch ein Monitor für den Monitortisch.

9.2 Software

Das Betriebssystem auf dem Computer ist ein Ubuntu, auf dem Raspberry Pi ein Raspbian. Auf beiden ist die Java Plattform nötig, auf dem Computer auch noch Bibliothek openCV.

9.3 Installation

9.3.1 Vorbereitungen

Computer und Raspberry Pi müssen eine feste IP- Nummer haben. Auf beiden müssen beide IP-Nummern mit den entsprechenden Hostnamen in die /etc/hosts eingetragen werden.

sudo apt-get install vim

sudo vim /etc/hosts

127.0.0.1 localhost

9.3.1.1.1.1.1.1 127.0.1.1 raspberrypi

```
192.168.111.39  raspberrypi
192.168.111.42  ias1056
```

sudo vim /etc/network/interfaces
# iface eth0 inet dhcp
auto eth0
iface eth0 inet static

address 192.168.111.39
netmask 255.255.255.0
network 192.168.111.0
broadcast 192.168.111.255
gateway 192.168.111.130
dns-nameservers 192.168.111.130
dns-search home.lan

Hier am Beispiel des Raspberry Pi.
Für die Bildverarbeitung muss openCV installiert werden.

sudo apt-get install -t jessie libopencv2.4-java

Damit der Zugriff von Computer auf den Raspberry Pi und von diesem auf dem Computer geschehen kann wird SSH mit Public Key ohne Passwort benutzt. Zusätzlich mounted der Raspberry Pi das Zielverzeichnis der Bilder auf dem Computer automatisch beim Booten mittels SSHFS.

Auf Computer und Raspberry Pi:

sh-keygen -t rsa -b 4096

Enter file in which to save the key (/home/"user"/.ssh/id_rsa):

"so lassen"

Enter passphrase (empty for no passphrase): "nichts"

Übertragung des öffentlichen Schlüssels auf den jeweils anderen Rechner.

ssh-copy-id -i ~/.ssh/id_rsa.pub "user"@"server"

Nur auf dem Raspberry Pi:

sudo apt-get install sshfs
sudo vim /etc/fstab
sshfs#"user"@ias1056:/home/"user"/Puzzler_02/livePic

```
/home/pi/Puzzler_02_raspi/livePic fuse

uid=1000,gid=100,umask=0,allow_other,_netdev,
```

IdentityFile=/home/pi/.ssh/id_rsa 0 0

In /etc/fstab diese Zeile ergänzen.

9.3.2 Programme

Das Verzeichnis mit dem Quellcode „Puzzler_02" muss auf den Computer in das Home-Verzeichnis des Benutzers. Dort wird der Code kompiliert.

cd ~/Puzzler_02/

javac –classpath ./src/libs/jade.jar:/usr/share/java/opencv-249.jar

–sourcepath ./src/ ./src/logic/Main.java

Das Verzeichnis mit dem Quellcode „Puzzler_02_raspi" muss auf den Raspberry Pi in das Home-Verzeichnis des Benutzers. Dort wird der Code kompiliert.

cd ~/Puzzler_02_raspi/
javac –classpath ./src/libs/jade.jar

–sourcepath ./src/ ./src/logic/MainRasPi.java

Des Weiteren ist darauf zu achten, dass das Startskript im Homeverzeichnis des Raspberry Pi Ausführungsrechte hat (z.B. mit chmod 755).

10 Benutzeranleitung

10.1 Prinzipielles Vorgehen

Es werden zwei Bilder benötigt: ein Init-Bild (ohne Puzzlestücke) und ein Puzzlestücke-Bild (mit Puzzlestücken). Diese Bilder dürfen sich außer bei den Puzzlestücken (ohne/mit) nicht unterscheiden. Daraus werden die Puzzlestücke ermittelt und das Puzzle gelöst. Falls nicht anders Vorgesehen sollten die Puzzlestücke auf dem Monitortisch aufgelegt werden, dort wird auch die Information über den Lösevorgang angezeigt.

10.2 Inbetriebnahme

Zuerst muss das Notebook inkl. externer Monitor angeschlossen werden. Der Raspberry Pi darf jetzt noch nicht angeschlossen werden. Nach dem Booten und Einloggen in das Notebook kann jetzt der Raspberry Pi verkabelt werden. Zuerst das Netzwerkkabel, danach das USB- Kabel für die Stromversorgung. Der Raspberry Pi bootet automatisch und nimmt dann Kontakt mit dem Notebook auf, die Anwendungen auf ihm starten bei Bedarf vom Notebook aus gesteuert auch automatisch. Er sollte vor dem Beenden der Anwendung auf dem Notebook mit Hilfe dieser wieder herunter gefahren werden (siehe 10.6 Beenden und Ausschalten).

10.3 Monitortisch ausrichten

Es gibt eine extra Anwendung für das Ausrichten des Monitortisches. Diese wird so gestartet:

cd ~/Puzzler_02/
java –classpath

./src/libs/jade.jar:/usr/share/java/opencv-249.jar:./src/

test.CalibCamera

Damit kann mit dem Button „Foto“ beliebig oft ein aktuelles Bild empfangen werden und somit die änderbaren Freiheitsgrade des Monitortischs und der Kamera angepasst werden bis alle Kanten des Schachbrettes parallel laufen.

10.4 Anwendung starten

Der Start der Anwendung erfolgt so:

cd ~/Puzzler_02/
java –classpath

./src/libs/jade.jar:/usr/share/java/opencv-249.jar:./src/

logic.Main

Als erstes muss aus den angebotenen Möglichkeiten der Bildherkunft ausgewählt werden.

- Raspberry Pi: dem angeschlossenen Raspberry Pi und dessen Kamera.
- USB- Kamera: eine über USB direkt an das Notebook angeschlossene Kamera.
- Dateisystem: Das Laden auf anderen Wegen erzeugten Bildern von einem Datenträger.

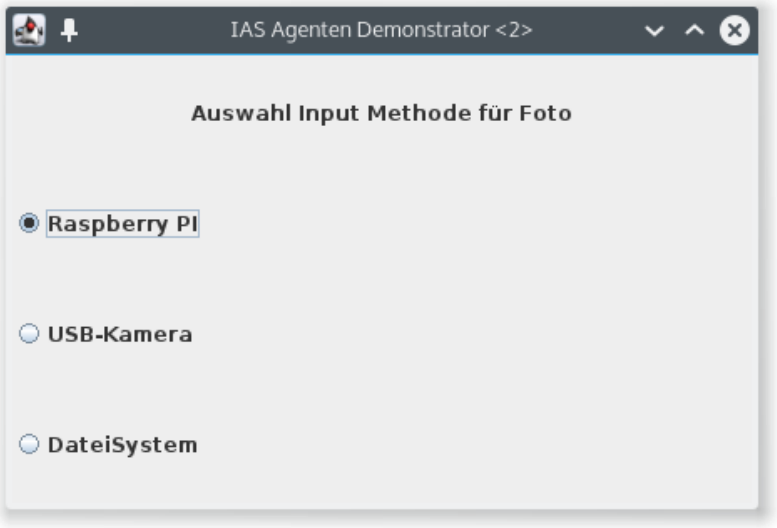

Abbildung 10.1: Bildherkunft auswählen

Dieser Dialog erscheint nur beim Programmstart.

Der nächste Dialog bittet um die Auswahl des Init-Bildes (ohne aufgelegte Puzzlestücke) und des Puzzlestücke-Bildes (mit aufgelegten Puzzlestücken). Die Reihenfolge und Anzahl der Versuche ist beliebig. Weiter geht es mit dem Button „OK“.

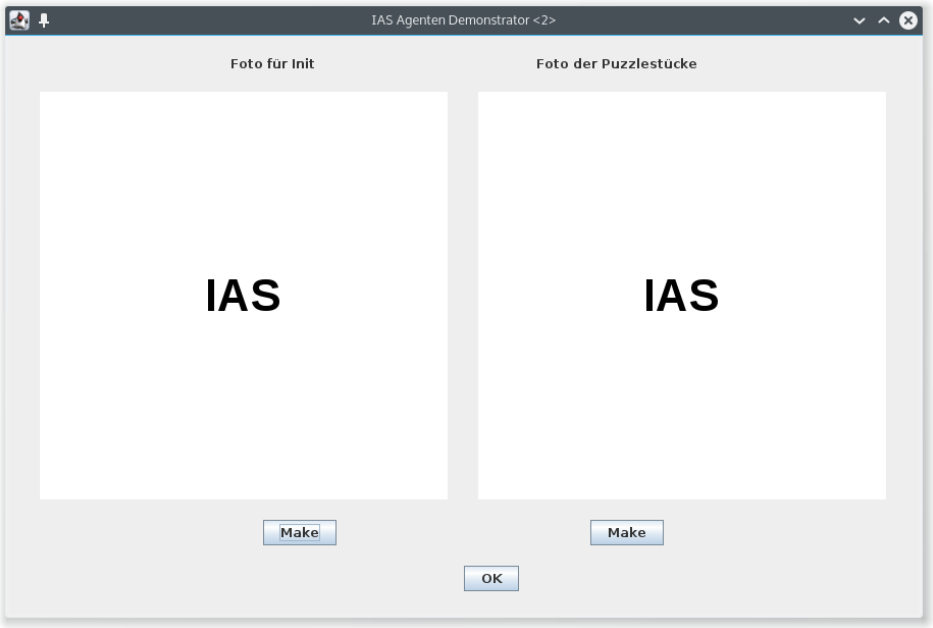

Abbildung 10.2: Init- und Puzzlestück- Bild Auswahl

10.5 Bedienung

Die eigentliche Anwendung hat zwei Fenster. Ein Ausgabefenster, welches mit der Maus auf den zweiten Bildschirm gezogen werden muss und dann dort mit einem Doppelklick auf der Fensterleiste maximiert werden muss. Das Bedienerfenster dient zum Steuern des Programmablaufs und dem Anzeigen von Kontrollinformationen während des Laufs.

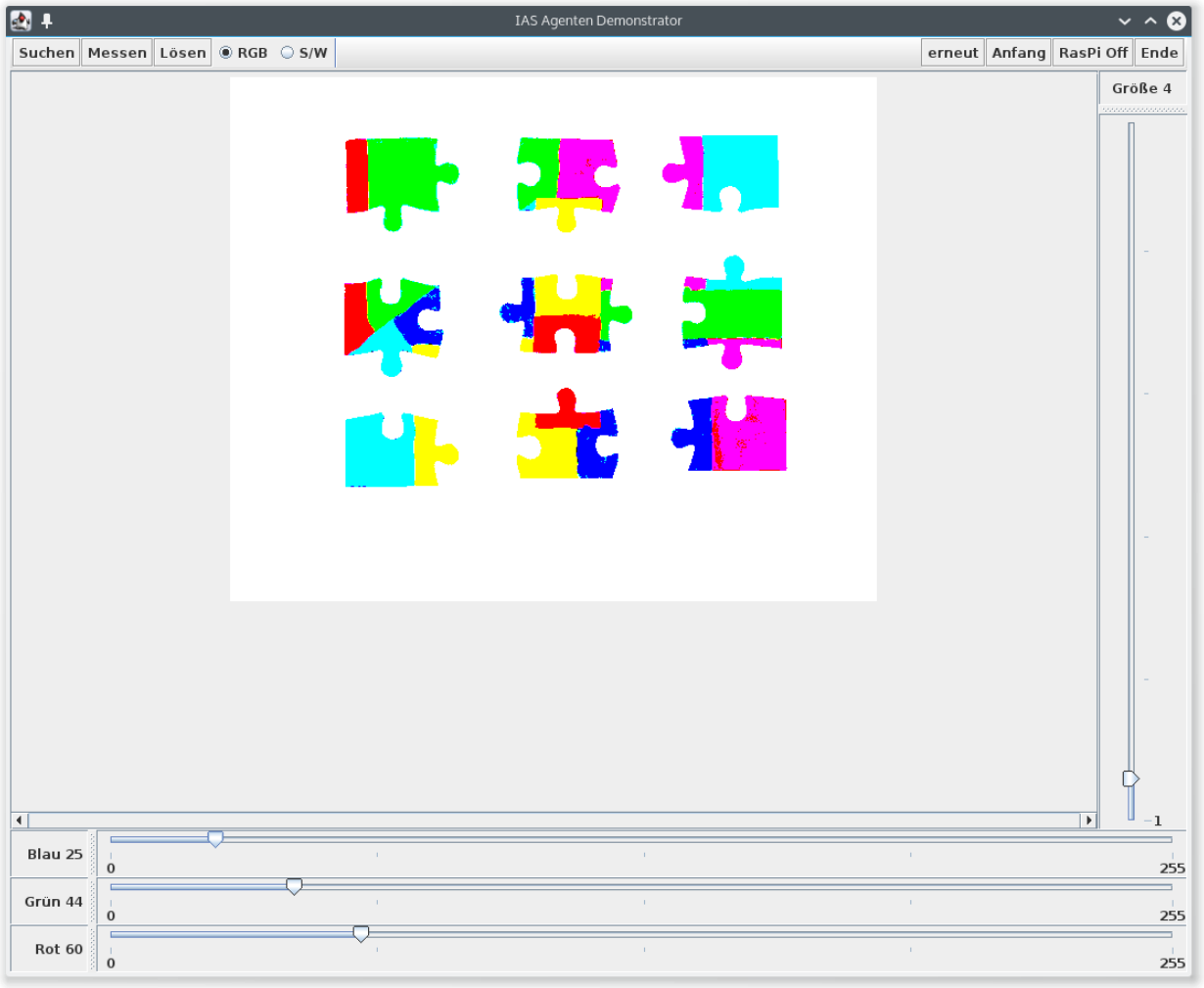

Abbildung 10.3: Bediener Fenster

Hier gezeigt direkt nach der Bilderfassung.

Mit den Schieberegler „Blau“, „Grün“ und „Rot“ kann der Grenzwert zur RGB Einrastung verändert werden.

Mit dem Button „Suchen“ veranlasst man das Suchen der Puzzlestücke in diesem Bild. Die gefundenen werden dann im Ausgabefenster an gleicher Stelle wie die originalen Stücke angezeigt. Der Schieberegler „Größe“ verändert die Darstellung im Ausgabefenster um die echten Puzzlestücke darauf mit den angezeigten Stücken in der richtigen Größe zu hinterlegen.

Der Button „Messen“ startet das Vermessen der gefundenen Puzzlestücke.

Mit diesen Daten kann nun der Vorgang „Lösen“ mit diesem Button ausgelöst werden.

Es gibt mit der Auswahl „RGB“ und „S/W“ die Möglichkeit die Anzeige des erfassten Bildes von Rot-Grün-Blau auf Schwarz/Weiß umzuschalten.

Ein erneuter Durchlauf mit dem gleichen Bild ist durch den Button „erneut“ möglich.

Ganz von vorn beginnt man mit dem Button „Anfang“.

10.6 Beenden und Ausschalten

Vor dem Programmende mit dem Button „Ende“ muss der Raspberry Pi mit dem Button „RasPi Off“ herunter gefahren werden.

11 Ausblick

Die gewählte Methode der Filterung und damit Hintergrundunterdrückung hat sich als funktional aber wenig praxistauglich erwiesen. Der Aufwand bei mehreren Durchläufen jedes Mal alle Puzzlestücke vom Monitortisch zu nehmen und wieder aufzulegen ist wenig Praktikabel. Es wird empfohlen die etwas aufwendiger zu Implementierende Methode der wechselnden Hintergründe zu benutzen, bei der die Puzzlestücke liegen bleiben können und mehrere Aufnahmen mit wechselnden Hintergrundfarben erstellt werden.

Ein Problem sind andere Objekte als Puzzlestücke auf dem Monitor. Diese werden z.Z. auch als Puzzlestücke gewertet. Mit einer Objekterkennung über z.B. ein Neuronale Netz könnten die nicht Puzzlestückobjekte herausgefiltert werden. Die Fähigkeit Werkstücke von anderen Objekten zu unterscheiden wird auch im Rahmen von Industrie 4.0 immer wichtiger.

Verschiebungen und Umplatzierungen von Puzzlestücken (ob gewollte oder ungewollt) können momentan nicht kontrolliert werden. Dies könnte mit einer Bewegungsverfolgung vermieden werden. Auch diese Fähigkeit ist in heutigen Industrieanlagen sehr begehrt.

12 Schlusswort

Der am IAS vorhandene Puzzler wurde erfolgreich um eine Bilderfassung und eine Möglichkeit Informationen am Puzzle und um einzelne Puzzlestücke herum anzuzeigen erweitert. Das Verwenden von Unterlicht durch den Monitortisch hat den willkommenen Nebeneffekt geringerer Schattenbildung und führt so zu einem verbessern der Toleranzen bei der Vermessung der Puzzlestücke.

Das ausgearbeitete Konzept die ermittelten Lösungen mittels Einsatz von Inseln zu verbessern ist gelungen. Die Erhoffte Beschleunigung durch das frühere erzeugen der Agenten und dem Einsatz von Clustering kann aufgrund der geringen Anzahl an verwendeten Puzzlestücken kein praktisches Ergebnis nachgereicht werden.

Literaturverzeichnis

[BuBu06] Burge, M.; Burger, W.: Digitale Bildverarbeitung, Springer, 2006

[Emri13] Emrich, E.: Ausarbeitung der Diplomarbeit, IAS, 2013

[Göhn13] Göhner, P.: Skript zur Vorlesung Softwaretechnik I WS 13/14, IAS, 2013

[Jung14] Jung, T.: Ausarbeitung der Bachelorarbeit, IAS, 2014

[Knöd14] Knödler, T.: Ausarbeitung der Studienarbeit, IAS, 2014

[KrHa11] Krüger, G.; Hansen H.: Handbuch der Java-Programmierung, Addison-Wesley, 2011

[Poli10] Politze, M.: OpenCV in a Nutshell, Seminararbeit, FH Aachen, 2010

[Ulle10] Ullenboom, Ch.: Java ist auch eine Insel, Galileo Press, 2011

[Inte01] Internet001: http://docs.oracle.com/javase/7/docs/index.html, Stand 16.07.2015

[Inte02] Internet002: http://www.dpunkt.de/java/index.html, Stand 16.07.2015

[Inte03] Internet003: http://jade.tilab.com/, Stand 16.07.2015

[Inte04] Internet004: http://opencv.org/, Stand 16.07.2015

12.1 Erklärung

Ich erkläre, die Arbeit selbständig verfasst und bei der Erstellung dieser Arbeit die einschlägigen Bestimmungen, insbesondere zum Urheberrechtsschutz fremder Beiträge, eingehalten zu haben. Soweit meine Arbeit fremde Beiträge (z. B. Bilder, Zeichnungen, Textpassagen) enthält, erkläre ich, dass diese Beiträge als solche gekennzeichnet sind (z. B. Zitat, Quellenangabe) und ich eventuell erforderlich gewordene Zustimmungen der Urheber zur Nutzung dieser Beiträge in meiner Arbeit eingeholt habe.

Unterschrift:

Stuttgart, den 20.07.2015