Karlsruher Institut für Technologie

Differenzbasierte Repräsentation räumlicher Relationen zur probabilistischen Szenenerkennung mittels hierarchischen Constellation Models

Bachelorarbeit
von

Joshua Enrico Link

An der Fakultät für Informatik
Institut für Anthropomatik und Robotik
Lehrstuhl Prof. Dr.-Ing. R. Dillmann

Erstgutachter:	Prof. Dr.-Ing. R. Dillmann
Zweitgutachter:	???
Betreuender Mitarbeiter:	Dipl.-Inform. Pascal Meißner

Bearbeitungszeit: 11. Juni 2017 – 10. September 2017

Hiermit erkläre ich an Eides statt, dass ich die von mir vorgelegte Arbeit selbstständig verfasst habe, dass ich die verwendeten Quellen, Internet-Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen – die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Karlsruhe, den **(Datum)** **ToDo**

Joshua Enrico Link

Inhaltsverzeichnis

Abbildungsverzeichnis

1. Einführung

In der Robotik ist die Servicerobotik wohl der Forschungsbereich, welcher den größten Alltagsbezug für den Menschen hat, da er sich mit der Entwicklung und Weiterentwicklung von autonomen Robotern beschäftigt, welche dem Menschen im Alltag assistieren. Man findet mittlerweile Roboter im Privaten, die das Putzen, Staubsaugen oder Rasenmähen übernehmen, in der Industrie, bei Montage und Fertigung, sowie auch in der Medizin, als Pflegehilfe, Botengänger oder Assistent.

Allerdings müssen die Roboter ihre Umwelt für komplexere Aufgaben so präzise wie möglich wahrnehmen und verstehen. Sie könne Aufgaben übernehmen bei denen sie gezielt Objekte umfahren, suchen und auch aufnehmen und benutzen. Dieser Funktionsumfang kann mit dem Prizip Programmieren durch Vormachen (PdV) ermöglicht werden, bei dem die Roboter Objekte und Tätigkeiten ihrer Umgebung kennen lernen, wieder erkennen und nachahmen können. So lässt sich die hohe Komplexität umgehen, die die manuelle Programmierung vieler Aufgaben mit sich bringen würde.

Um tatsächlich selbstständige Serviceroboter zu schaffen muss man aber noch zu einer Objekterkennung ein Kontextverständnis hinzufügen. Die Roboter müssen erkannte Objekte in einen Zusammenhang bringen, um die dadurch resultierenden Aufgaben zu verstehen. Zum Beispiel hat ein Teelöffel, welcher neben einer Tasse Tee liegt eine andere Aufgabe zu verrichten, als wenn er neben einem Becher Joghurt platziert ist. Nur am Kontext lässt sich dort entscheiden warum im einen Fall umgerührt und im anderen gelöffelt wird. Ebenso wäre ein Stück Butter verschieden zu verwenden, wenn es auf einem Frühstückstisch steht als wenn es mit anderen Zutaten neben einer Rührschüssel vorkommt.

Somit braucht man eine Szenenerkennung, welche zuverlässig die Objekte erkennen und ihren jeweiligen Kontext verstehen und einschätzen kann. Diese Erkennung ist nicht immer eindeutig, da der eben genannte Löffel ebenso zwischen einem Becher Joghurt und einer Tasse Tee liegen könnte, deshalb bietet es sich an mit Wahrscheinlichkeitsabschätzungen des vorliegenden Kontexts zu arbeiten.

2. Motivation und Problemstellung

Das Kapitel geht in Motivation auf die Relevanz des Themas und der vorliegenden Arbeit ein und in Fokus der Arbeit auf die Problemstellung die bearbeitet wird, sowie diverse Einschränkungen und Annahmen die für die Arbeit festgelegt sind.

2.1 Motivation

Wie schon in der Einführung erwähnt ist es elementar wichtig, dass man eine zuverlässige Szenenerkennung und ein gutes Kontextverständnis schafft um den Robotern die Möglichkeit zu bieten, sinnvoll mit ihrer Umwelt zu interagieren. Mit einer präzisen Wahrscheinlichkeitseinschätzung wie die momentane Umgebung beschaffen ist, lässt sich abschätzen welche Aufgaben es zu bewältigen und welche Probleme zu lösen gilt. Um dies zu gewährleisten muss man in das System möglichst viele Referenzdaten einspeisen, damit es jeden vorhanden Kontext erkennen kann. Szenen werden zu diesem Zweck aufgebaut un der Erkennung als neue Szene vorgestellt. Jede Szene die die Szenenerkennung auf diese Weise lernt, hilft das Chaos der sie umgebenen Objekte mehr und mehr zu interpretieren und einzuordnen.

Die Szenenerkennung nutzt also die Daten die sie zur Verfügung gestellt bekommt, bereits gelernte Szenen wiederzuerkennen. In dem bereits vorhandenen PSM(Probabilistic Scene Model)-System werden die Daten pro Szene zu einem Modell zusammengefasst, bei dem Auffällige Zusammenhänge berücksichtigt werden und scheinbar nicht miteinander in Verbindung stehende Objekte voneinander gelöst betrachtet werden. Zum Beispiel findet man eine Computermaus signifikant häufig vor dem Computerbildschirm und nie dahinter, allerdings kann dabei das räumliche Verhältnis der Maus zur Tastatur stark variieren. Die Maus wäre in diesem Beispiel manchmal direkt neben der Tastatur, manchmal dichter beim Bildschirm eben so oft weiter entfernt. In diesem Fall würde möglicherweise die Relation zwischen Maus und Tastatur wegfallen und nur jeweils das räumliche Verhältnis zum Bildschirm betrachtet werden. Dadurch gibt es Vorteile in der

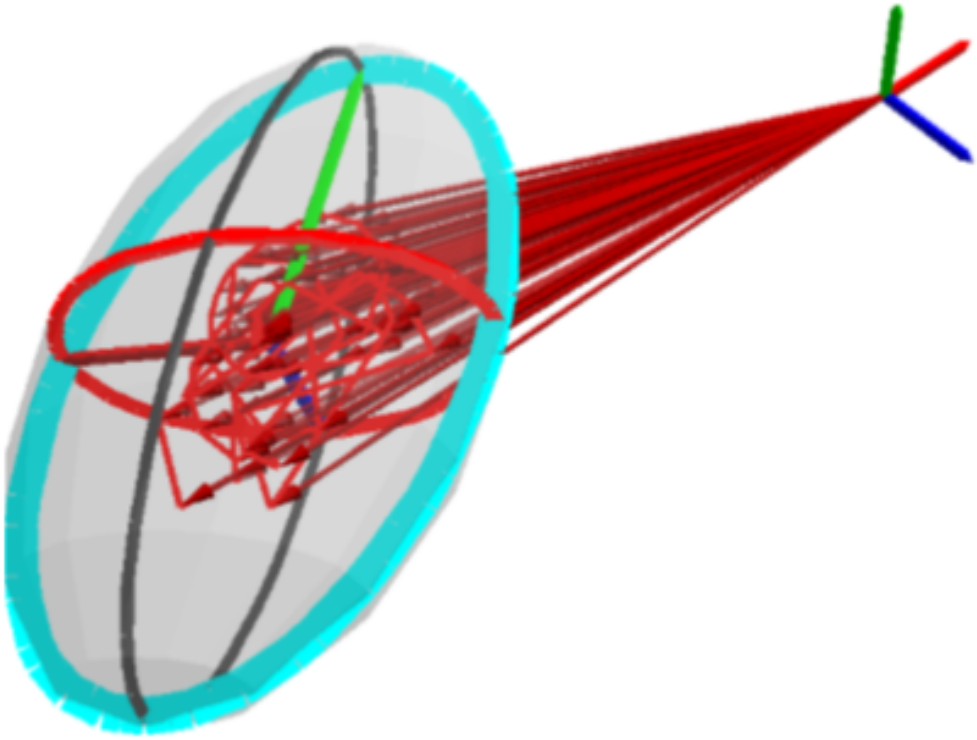

Abbildung 2.1: Beispiel: Relative Position eines Objekts zu einem anderen

Laufzeit und möglicherweise auch eine signifikantere Erkennung, allerdings findet natürlich auch ein Informationsverlust statt, der zu Fehlern führen kann. In dieser Arbeit wird ein Ansatz getestet, der dichter an den erhaltenen Daten arbeitet.

In Abbildung 2.1 sieht man wie das Modell eine Relation zwischen zwei Objekten aufgrund der erhaltenen Daten erstellt. Die roten Pfeile beschreiben hier die jeweiligen Relationen, die aus den Daten berechnet wurden und die Ellipsen zeigen, welchen Bereich das Modell als Basis für die Wahrscheinlichkeitsabschätzung benutzt. [Geh14]

2.2 Fokus der Arbeit

Ziel der vorliegenden Arbeit ist es das PSM-System zu überarbeiten und einen neuen Modus der Szenenerkennung im PSM-System zu entwickeln. Dieser Modus soll alternativ zu den bereits vorhandenen Modi auswählbar sein und das System erweitern. Außerdem soll sowohl die Positionierung als auch die Rotation der erkannten Objekten mit den bekannten Szenen verglichen werden um eine Wahrscheinlichkeitsabschätzung auszugeben, welche die Wahrscheinlichkeit aller möglichen Szenen ausgibt sowie auch die Wahrscheinlichkeit, dass es sich um keine der Szenen handelt.

Um eine interaktive Szenenerkennung zu schaffen ist es sinnvoll das System datengetrieben zu programmieren. Datengetriebene Entwicklung zeichnet sich dadurch aus, dass sich der Programmfluss ändert aufgrund der Daten die das System während der Laufzeit als Eingabe erhält. Das PSM-System wurde bereits datengetrieben programmiert und in der zu dieser Arbeit gehörigen Implementierung wurde auch am datengetriebenen Entwicklungsmodell festgehalten.

Da die Erkennung innerhalb des PSM-Systems nutzbar sein soll sind Eingabe- und Ausgabeschnittstellen sowie die Visualisierung vorgeschrieben und sollen für gute Vergleichbarkeit denen des vorhandenen Systems entsprechen.

Die Arbeit ist wie folgend strukturiert. In 3 Grundlagen wird auf die Grundkenntnisse eingegangen die man braucht um den Rest der Arbeit gut zu verstehen. Vorallem wird in diesem Kapitel das bestehende PSM-System erklärt. In 4 Konzept wird das theoretische Konzept thematisiert, welches zum Algorithmus geführt hat, dass den neuen Modus vom alten System unterscheidet. Außerdem wird der Algorithmus selbst erläutert. In 5 Implementierung wird die Software beschrieben und dokumentiert die im Zuge dieser Arbeit entstanden ist sowie die Änderungen die am bestehenden PSM System vorgenommen wurden. 6 Evaluation beschreibt die durchgeführten Experimente und interpretiert sie. In 7 Zusammenfassung und Ausblick wird die Arbeit noch einmal zusammen gefasst und ein Ausblick darauf gegeben in welche Richtung man das bestehende System weiter entwickeln und auf welche Weise man möglicherweise die Szenenerkennung noch weiter verbessern kann. Am Schluss stehen die Quellen welche zur Recherche für diese Arbeit genutzt wurden.

3. Grundlagen

In diesem Kapitel wird das bestehende Probabilistic Scene Model - System vorgestellt und erklärt. Dabei wird auf das Grundprinzip, die Struktur und die einzelnen Komponenten eingegangen. Außerdem wird das Modell der datengetriebenen Entwicklung erklärt und an einem Beispiel verständlich gemacht.

3.1 PSM (Probabilistic Scene Model)

Bei der Entwicklung des Systems war das Ziel, ein System zur Erkennung von Szenen zu entwickeln. Dabei sollten viele Informationen umfasst werden. Dies umfasste die beteiligten Objekte bzw. deren Erscheinung, welche von den zur Erkennung eingesetzten Werkzeugen abhängig ist. Die Auftrittshäufigkeit der Objekte sollte ebenfalls mit einfließen. Der Hauptinformationsträger sind die Relationen der Objekte untereinander, welche in Form von relativen Objektlagen berücksichtigt wurden. Es wurde berücksichtigt, dass Relationen nicht statisch, sondern dynamisch sind. Da eine Szene auch unabhängig von ihrem Ort der Demonstration erkannt werden soll, musste die Lagebeschreibung invariant gegenüber Rotation und Translation sein.

Weiterhin sollte Robustheit gegenüber verschiedenen Störfaktoren bestehen. Fehlende Objekte sollen eine Erkennung der Szene erlauben, wenn auch mit entsprechend reduzierter Konfidenz. Da jedes Objekt fehlen kann darf es kein zentrales Referenzobjekt geben, von dem die Relationen zu den anderen Objekten der Szene ausgehen. Überzählige Objekte sollen sich nicht negativ auf das Erkennungsergebnis auswirken. Generell soll sich die Funktionsweise des entwickelten Systems im Rahmen dessen bewegen, was plausibel erscheint.

Es wird im folgenden erst auf die Struktur des Systems, dann auf die Eingabeverarbeitung, das innere Datenmodell und letztendlich auf die eigentliche Erkennung eingegangen. [Geh14]

3.1.1 Aufbau

Das System ist in mehrere Komponenten aufgeteilt die verschiedenen Aufgaben dienen. Für diese Arbeit wichtig sind vorallem die Komponenten Learner und Inference. Der Learner ist für das Anlernen der Daten und das Einspeichern von Szenen zuständig. Die Inference übernimmt die Erkennung der Szene indem sie in Echtzeit Daten empfängt und die Wahrscheinlichkeiten der Szenen ausgibt. Es gibt außerdem eine Visualizer-Komponente, die dafür zuständig ist, dass verschiedene programminterne Prozesse für den Nutzer sichtbar und verständlich gemacht werden.

3.1.2 Learner - Anlernen der Daten

3.1.3 Scene Model - Modellerzeugung

3.1.4 Inference - Szenenerkennung

3.2 Datengetriebene Entwicklung

3.2.1 Allgemein

3.2.2 Am Beispiel: PSM

Länge max. halb so lang wie Konzept + Implementierung

simpel beschreiben
konkret:
bestehendes System : PSM
Relevanz erklären?
Datengetriebene Entwicklung erklären
Viele Bilder benutzen, auch aus Joachims Arbeit
Auch aus Joachims Arbeit

[Geh14]

4. Konzept

Das folgende Kapitel thematisiert das Konzept, dass im Zuge der vorliegenden Arbeit entwickelt wurde, um die vorgestellte Problemstellung zu lösen. Zwischen vielen verschiedenen möglichen Ansätzen einen neuen Modus für das PSM System zu entwickeln, entschied ich mich für einen differenzbasierten Vergleich der Positions- und Rotationsrelationen. Dieser wird im ersten Abschnitt erläutert und anschließend wird der Algorithmus sprachlich, grafisch und in Pseudocode erklärt. Zum Schluss wird die Wahrscheinlichkeitsabschätzung für den Algorithmus erklärt und begründet.

4.1 Ansatz

Im vorhandenen PSM-System werden im Learner die erhaltenen Positions- und Rotationsdaten zu einem Modell zusammengefasst, dass die vorkommen aller Objekte in Relation zueinander zusammenfasst. Dieser Vorgang führt dazu, dass teilweise Zusammenhänge in den Daten betont werden, aber auch zu einem Informationsverlust da Ausreißer und mutmaßliche Fehlmessungen dadurch verloren gehen. Deshalb kann es sinnvoll sein direkt auf den gemessenen Daten zu arbeiten, um ein Ergebnis zu erhalten welches alle Daten berücksichtigt.

Wir betrachten folgendes Szenario. Ein Roboter soll seine Aufgaben aufgrund von einer Szenenerkennung einschätzen und durchführen. In der Szene "Kaffee" gibt es eine volle Kaffeetasse und einen Teelöffel und seine Aufgabe ist es mit dem Löffel den Kaffee umzurühren. In seinen Referenzdaten zu der Szene war der Löffel meist direkt neben der Tasse und nur in einem Fall ein Stück weiter entfernt. Allerdings gilt jede einzelne aufgezeichnete Refernzszene auf äquivalente Weise als Beispiel für die Szene "Kaffee". Das Parametermodell würde diese Ausreißerdaten allerdings glätten und kaum berücksichtigen, sodass der Roboter die Szene selbst mit genau dem Aufbau aus den Refernzdaten möglicherweise nicht erkennen würde. Wenn man allerdings die Erkennung direkt auf den Referenzdaten basiert, erkennt die Szenenerkennung den Ausreißer auch, da sie ja eine Instanz der Szene mit diesem vergleicht, welche diesem entspricht.

Abbildung 4.1: Beispiel: Kaffeetasse - Ausreißer in den Daten

Abbildung 4.1 verdeutlicht das genannte Szenario. Die roten Punkte stehen für die gemessenen Positionen und die grünen Pfeile stehen für die räumliche Relation zwischen den Objekten. Links sieht man, dass die meisten Messungen einen kleinen Abstand zwischen Löffel und Tasse haben, rechts ist der Ausreißer dargestellt, der möglicherweise vom alten System nicht als die gelernte Szene erkannt wird.

Im differenzbasierten Modus sollen also gemessene Objekte direkt mit den Referenzdaten verglichen werden, die das System bereits gelernt hat. Der Algorithmus betrachtet alle Objekte vollvermascht, sodass er die Szenenreferenz findet, die die maximale Ähnlichkeit zu den gemessenen Objekten hat. Darauf basierend wird die Wahrscheinlichkeit abgeschätzt, dass die gemessenen Objekte die Referenzszene enthalten oder repräsentieren. Dabei stören zusätzliche Objekte die Erkennung nicht und eine Unvollständigkeit der Szene führt zu einer kleineren Wahrscheinlichkeit aber nicht zu direkter Ablehnung, da die Szene noch durch weitere Objekterkennungen vervollständigt werden könnte.

4.2 Erkennungsalgorithmus

Der Algorithmus wurde mit den in Ansatz genannten Annahmen und Einschränkungen entwickelt und hat zur Aufgabe zu jeder Szene die auf Vorkommen geprüft wird die Instanz der Szene in den Daten zu finden, die am dichtesten an den gemessenen Daten liegt und so die höchste Wahrscheinlichkeit aufzeigt, dass die gemessenen Daten die Szene enthalten. Nachdem die Wahrscheinlichkeit einer Szene bestimmt ist wird diese mit den anderen Szenen genau wie im bestehenden PSM-System verrechnet, sodass am Ende die Relative Wahrscheinlichkeit für alle zutestenden Szenen angegeben wird.

4.2.1 Algorithmus: Beschreibung

Der Algorithmus läuft wie folgt ab. Für jedes Objekt, der zu testenden Szene, wird überprüft ob es sich um ein Objekt handelt, welches gerade von der Objekterkennung erkannt

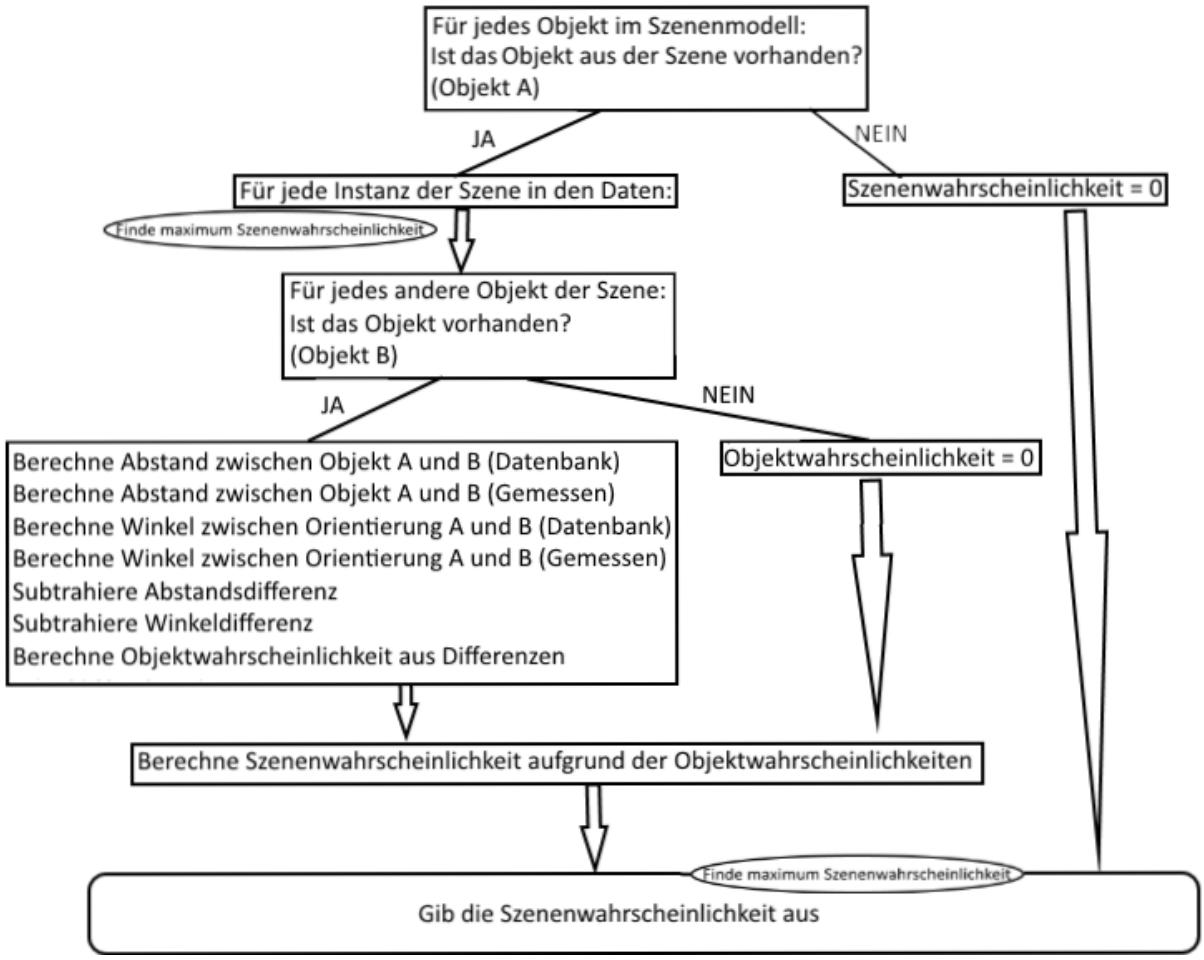

Abbildung 4.2: Algorithmus als vereinfachtes Flussdiagramm

wird. Wenn dies nicht der Fall ist, wird das Objekt übersprungen. Wenn dies allerdings der Fall ist, wird das Objekt zeitweise zu unserem Referenzobjekt und der Algorithmus führt für jede Instanz der zu testenden Szene in den Daten folgendes aus:

Es wird über alle anderen Objekte der Szeneninstanz iteriert und für jedes Objekt abgefragt, ob es ein gemessenes Objekt gibt, welches die Repräsentation für das Datenobjekt sein kann. Falls das Objekt nicht in den momentan wahrgenommenen Objekten vorkommt, ist die Szene allein aus Sicht dieses Objekts betrachtet unwahrscheinlich. Deshalb wird eine Objektwahrscheinlichkeit von 0 eingespeichert, welche aussagt, dass dieses Objekt allein die Szene als nicht auffindbar beschreibt. Wenn allerdings eine Instanz der Objekts in der Iteration gefunden wird, berechnet der Algorithmus die Positions- und Rotationsrelation zwischen den beiden Objekten aus den Daten. Genauso wird die Position und Rotationsrelation der gemessenen Objekte zueinander berechnet, welche mutmaßlich den Objekten aus der Datenbank entsprechen sollten. Nach diesen Berechnungen wird die Differenz verglichen die zwischen den beiden Paaren jeweils besteht. Dabei werden Positionsrelationen und Orientierung beziehungsweise Rotation jeweils getrennt betrachtet. Aus dem Grad der Ähnlichkeit dieser Differenzen lässt sich nun die Wahrscheinlichkeit ableiten, Objekt so vorkommt wie die Szene aufgezeichnet wurde. Somit hat man wieder eine Objektwahrscheinlichkeit.

Nun werden alle Objektwahrscheinlichkeiten zu einer Szenenwahrscheinlichkeit zusammengefasst. Diese wird innerhalb der Iteration über die Szeneninstanzen maximiert. Außerdem wird dieser Maximalwert wiederum innerhalb der äußersten Schleife maximiert, welche über alle Objekte iteriert, die sowohl in der zu testenden Szene sind als auch von der Objekterkennung erkannt wurden. Es wird also der absolute maximale Wert der Szenenwahrscheinlichkeit bestimmt, den man mit einer der Szeneninstanzen mit der beschriebenen Schleife mit jedwedem Referenzobjekt erzeugen kann. Dieser Maximalwert ist die Wahrscheinlichkeit, ob die zu testende Szene in den Gemessenen Objekten und potentiell weiteren unbekannten Objekten enthalten ist, welche von der Objekterkennung noch erkannt werden könnten.

Abbildung 4.2 beschreibt den Algorithmus als vereinfachtes Flussdiagramm. Einfache Linien verdeutlichen den Programmfluss in den verschiedenen Fällen und die Flussrichtung ist stehts nach unten. Die Schleifen sind zusammengefasst damit das Diagramm übersichtlich bleibt.

4.2.2 Algorithmus: Pseudocode

Um das Algorithmuskonzept weiter zu verdeutlichen beschreibt Abbildung 4.3 den Algorithmus nochmals mit Pseudocode. Damit der Code verständlich wird hier eine kurze Erklärung zu der Abbildung. Die Einrückungen wurden statt den geschweiften Klammern verwendet, die in den meisten höheren Programmiersprachen vorkommen. Die Parameter sind wie folgt definiert. Szenenmodell beschreibt das Szenenmodell, dass die vorkommenden Objekte in der zu testenden Szene beinhaltet. Der Parameter Gemessen beschreibt die Objekte die momentan erkannt werden und real oder in einer Simulation vorhanden sind. Der Parameter Daten steht für die Instanzen die zur zu testenden Szene als Referenzen gespeichert sind.

```
BerechneSzenenWahrscheinlichkeit ( Objekt[] Szenenmodell, Objekt[] Gemessen, Szeneninstanz[] Daten )
    SzenenWahrscheinlichkeit = 0;

    foreach - Objekt A in Szenenmodell

        if  - Gemessen->EnthaeltObjektMitName ( A->Name )
            ReferenzObjekt = Gemessen->FindeObjektMitName ( A->Name );

            foreach - Szeneninstanz S in Daten
                SzenenWahrscheinlichkeitsSumme = 1;

                foreach - Objekt B in S
                    Objektwahrscheinlichkeit = 0;

                    if - Gemessen->EnthaeltObjektMitName ( B->Name )
                        ZweitesObjekt = Gemessen->FindeObjektMitNamen ( B->Name );
                        Objektwahrscheinlichkeit = BerechneObjektWahrscheinlichkeit( ReferenzObjekt, ZweitesObjekt, A, B);

                    SzenenWahrscheinlichkeitsSumme += Objektwahrscheinlichkeit;

                NeueSzenenWahrscheinlichkeit = SzenenWahrscheinlichkeitsSumme / Szenenmodell->Laenge;

                if - NeueSzenenWahrscheinlichkeit > SzenenWahrscheinlichkeit
                    SzenenWahrscheinlichkeit = NeueSzenenWahrscheinlichkeit;
```

Abbildung 4.3: Algorithmus als PseudoCode

Die Variable Name, die jedes Objekt gesetzt hat ist eine Identifizierung um Objekte ihren mutmaßlichen Vorkommen in der Messung zuzuordnen. Die Funktion Enthaelt-ObjektMitName(string Name) gibt true aus, falls die aufrufende Liste von Objekten ein Objekt mit dem gegebenen Namen enthält. Ansonsten wird false ausgegeben.

Die Funktion FindeObjektMitName(string Name) gibt das Objekt aus der Liste zurück, welches den gegebenen Namen trägt. Falls kein Objekt mit dem gegebenen Namen existiert wird NULL zurückgegeben. BerechneObjektWahrscheinlichkeit(Objekt C, Objekt D, Objekt A, Objekt B) nimmt vier Objekte und berechnet die Positions- und Orientierungsunterschiede zwischen den Parametern C und D sowie zwischen A und B. Anschließend werden die Differenzen abgeglichen und basierend auf den Unterschieden eine Wahrscheinlichkeitsabschätzung zwischen 0 und 1 abgegeben, wobei 1 für "mit der Szene übereinstimmend" und 0 für "weit entfernt" steht. Auch die Komplette Funktion speichert am Ende eine Wahrscheinlichkeit zwischen 0 und 1. Sie wird nicht ausgegeben, da auf die eingespeicherte Wahrscheinlichkeit über eine andere Schnittstelle zugegriffen wird und der Algorithmus nur den Zweck erfüllt die Wahrscheinlichkeit zu berechnen.

4.3 Wahrscheinlichkeitsabschätzung

komplexer mathematischer formulieren
Vergleichsbasierte Erkennung erklären
Stochastische Richtigkeit beweisen

[DSS93]

5. Implementierung

Im Kapitel Implementierung wird alles beschrieben und erklärt, was am bestehenden PSM-Projekt verändert und hinzugefügt wurde. Außerdem werden ausgewählte hinzugefügte und veränderte Klassen sowie launch-Dateien dokumentiert, sodass das Kapitel das Verständnis und die Nutzung der Neuheiten im System vereinfacht. Nachdem der Umbauprozess beschrieben wird, bei dem eine Klasse komplett aus dem PSM-Projekt ausgetauscht wurde, widmet sich das Kapitel der Umsetzung des Algorithmuskonzepts und der Einbettung in das vorhandene System.

5.1 Umbau PSM

Da das Paket "pbd_msgs" nicht konstenlos zur Verfügung gestellt wird, mussten alle Vorkommen der Klassen aus diesem Paket zu alternativen Ersatzklassen geändert werden. Teilweise konnte man dies durch simple Ersetzung erreichen, allerdings gab es nicht für jede Klasse eine Ersatzklasse mit dem selben Funktionsumfang. In den Fällen, in denen Funktionen fehlten oder geringfügig anders funktionierten, konnte man den Umbau durch kleine Anpassungen erreichen oder musste eigene Funktionen schreiben, welche die nötigen Operationen verichten konnten. Alle auf diese Weise programmierten Funktionen wurden hinreichend auf Gleichheit mit ihren Ursprungsfunktionen in ihrer Funktionsweise getestet, indem die Ergebnisse bei gleichen Eingangsparametern abgeglichen wurden. Außerdem habe ich eine Datenbankschnittstelle für das PSM-System hinzugefügt.

5.1.1 Klassenaustausch

5.1.2 Ersatzfunktionen

5.1.3 Datenbankeinbindung

Im vergleichbaren ISM Projekt, welches eine ähnliche Zielsetzung wie das ISM Projekt hat, allerdings einen nicht stochastischen Ansatz verfolgt, kann man die Szenen, die

das System lernen soll aus einer Datenbank auslesen. Aus Gründen der Vergleichbarkeit wie auch dem Komfort ist es sinnig, diese Datenbankschnittstelle für das PSM System nachzurüsten. Um dies zu erreichen wurde ein Datenbankpfad in die launch Datei des Learners(learner.launch) hinzugefügt und der veraltete rosbagfiles Parameter damit ersetzt. Die Datenbank wird innerhalb der SceneLearningEngine Klasse ausgelesen und die erhaltenen Daten, welche jeweils Vorkommen und von verschiedenen Objekten in diversen Szenen repräsentieren, werden anschließend konvertiert, sodass sie für das System sinnvolle AsrObject-Instanzen werden.

5.2 Differenzbasierter Erkennungsalgorithmus

Einerseits beschreibt dieses Kapitel, wie der im Konzept vorgestellte Algorithmus programmiertechnisch umgesetzt, andererseits wie dieser in das bestehende System eingebunden wurde.

5.2.1 Algorithmus

5.2.2 Einbindung PSM

Da davon abgesehen wird die Parametisierung des Szenenmodells zu nutzen, braucht man den Learner und das Szenenmodell nur noch zu dem Zweck, dass überliefert wird, welche Szenen potentiell erkannt werden sollen. Die einzigen Informationen die der algorithmus nutzt sind der Name der Szene und Anzahl der enthaltenen Objekte. In der Launch Datei der Erkennung(inference.launch) wurde ein neuer Parameter hinzugefügt um den Pfad der Datenbank anzugeben. ...

alle Klassen die umgebaut wurden
neuer differencebased modus

6. Evaluation

6.1 Experiment 1: Frühstück

6.2 Experiment 2: Office

Viele Bilder, beschreiben Daten
Text interpretiert
Fazit am Ende

[DSS93]

7. Zusammenfassung und Ausblick

Zwei Sätze zu jedem größeren Kapitel

[DSS93]

Literaturverzeichnis

[DSS93] Randall Davis, Howard Shrobe und Peter Szolovits: *What is a Knowledge Representation?* AI Magazine, 14(1):17–33, 1993. `http://www.aaai.org/ojs/index.php/aimagazine/article/view/1029`.

[Geh14] Joachim Gehrung: *Probabilistische Szenenerkennung durch hierarchische Constellation Models über räumliche Relationen aus demonstrierten Objekttrajektorien.* Seiten 5–63, 2014. `http://www.aaai.org/ojs/index.php/aimagazine/article/view/1029`.