Universität Bremen

Fachbereich 3
Diplomarbeit der Informatik

Entwicklung eines sensomotorischen Explorationssystems zur Klassifikation von VR-Umgebungen

Erstgutachterin:
Prof. Dr. Schill

Zweitgutachter:
Dr. Barkowsky

Johannes Wolter
wolter@inutil.org
Matrikelnr. 1455351

Bremen, 13. April 2006

Inhaltsverzeichnis

1 Einleitung

Menschen und Tiere sind sehr erfolgreich bei der Exploration von unbekannten Umgebungen und der Orientierung darin. Diese Arbeit geht Hinweisen nach, wie sie diese Aufgaben lösen und inwieweit sich diese biologischen Prinzipien in ein informatisches System übertragen lassen.

Von besonderem Interesse ist diese Thematik für die Entwicklung von autonom navigierenden Robotern. Systeme, die sich in möglichst vielen, unterschiedlichen Umgebungen orientieren sollen, müssen eine hohe Robustheit aufweisen, wie sie in biologischen Organismen zu finden ist. Der Bedarf an Robotern, die sich nicht nur in einer speziellen Umgebung bewegen können, ist sicherlich groß. Hilfen für Menschen, die durch eine Behinderung in ihrer Mobilität eingeschränkt sind oder die Erkundung von für Menschen unzugänglichen Umgebungen wie z.B. Kathastrophengebieten, stellen zwei interessante Anwendungsfelder dar.

Das ersten Kapitel geht auf Erkenntnisse aus der Biologie und der Psychologie ein, die zu verstehen helfen, aufgrund welcher mentalen Repräsentation von räumlichen Konfigurationen biologische Organismen Orientierungs- und Navigationsaufgaben lösen. Ein besonderer Schwerpunkt liegt zum einen auf der Sensomotorik im Bezug auf Repräsentationen der Umwelt im Gehirn und zum anderen auf den bei Ratten erforschten *Place Cells*, *Grid Cells* und *Head Direction Cells* als neuronales Substrat der Raumrepräsentation. Abschließend werden diese beiden Aspekte mit der klassischen Sicht einer *Cognitive Map* verglichen.

Anschließend werden im zweiten Kapitel einige aktuelle Explorationssysteme und ihre Raumrepräsentationsmodelle vorgestellt und mit den vorher diskutierten Eigenschaften biologischer Modelle verglichen. Im Vordergrund stehen Systeme, welche zumindest ansatzweise die Sensomotorik berücksichtigen.

Aufbauend auf diesen Erkenntnissen wird das Explorationssystem *Sensorimotor Explorer (SMX)* entwickelt, dessen Ziel es ist Navigationsaufgaben aufgrund einer sensomotorischen Repräsentation der Umgebung zu lösen. Die Entwicklung soll dazu beitragen ein besseres Verständnis für sensomotorische Repräsentationen zu erlangen und Ideen auf ihre Umsetzbarkeit zu testen. Die hier beschriebene erste Implementierung konzentriert sich auf die Aufgabe der Lokalisation und orientiert sich noch stark an *Okusys,* ein System das Bilder mittels Imitierung menschlicher Augenbewegungen klassifiziert. *Okusys* basiert auf ähnlichen Annahmen bzgl. der Sensomotorik und führt in gewisser Weise eine räumliche Exploration auf einer höheren Granularitätsstufe aus.

Den Abschluss bildet eine Bewertung des *SMX* und beschreibt, welche Möglichkeiten der Weiterentwicklung besonders interessant erscheinen.

2 Physiologische und psychologische Evidenz

> *Wenn das Gehirn so einfach wäre, dass wir es verstehen könnten, dann wären wir so dumm, dass wir es doch nicht verstehen würden.*
>
> *Jostein Gaarder*

Dieses Kapitel befasst sich mit Ergebnisen aus der Neurobiologie und der Psychologie, welche Hinweise geben, wie sich Menschen und Säugetiere im Allgemeinen im Raum orientieren bzw. aufgrund welcher mentalen Repräsentation ihrer Umgebung sie das tun.

Einleitend beleuchtet das Kapitel *Sensomotorik* neue Erkenntnisse zum Verhältnis von Sensorik und Motorik im menschlichen Gehirn und daraus folgende mögliche Konsequenzen für die interne Repräsentation von räumlichen Konfigurationen.

Darauf folgend werden Erkenntnisse über das neuronale Substrat für räumliche Orientierung vorgestellt. Neben den gut erforschten *Place cells* werden auch *Head direction cells* und die noch wenig erforschten *Grid cells* beschrieben. Hierbei handelt es sich um im Besonderen bei Ratten erforschte Neuronen im Hippocampus und dem entorhinalen Kortex, welche abhängig von der räumlichen Lage bzw. Orientierung aktiv sind.

Abschließend wird die klassische Sicht auf Raumrepräsentation im Gehirn in Form der *Cognitive Map* vorgestellt und mit den Erkenntnissen aus den ersten beiden Kapiteln in Beziehung gesetzt.

2.1 Sensomotorik

Der Begriff Sensomotorik umfasst allgemein das *»Zusammenspiel von Sinnesorganen und Muskeln«*[1], welches in diesem Kapitel näher betrachtet werden soll.

Die klassische Sicht, z.B. nach Sanders[2], geht von einer stufenweisen Verarbeitung der sensorischen Eingaben (Reize) aus, dessen Ergebnisse in »höheren« kognitiven Prozessen zum Aufruf von motorischen Programmen (Reaktionen) führen können (siehe Abb. 1). Die Felder Sensorik und Motorik sind hierbei einer konzeptuellen Trennung unterzogen. In den letzten Jahren wendet sich die Forschung jedoch immer mehr von dieser strikten Trennung ab. Es mehren sich Hinweise, dass die Sensorik und die Motorik im Gehirn relativ stark miteinander verwoben sind, welches u.a. Konsequenzen für allein auf Sensorik basierende Repräsentationsmodelle hat. Im Folgenden werden einige dieser Hinweise genauer betrachtet.

1 Brockhaus 1998
2 Sanders 1980 nach Schubö 1998

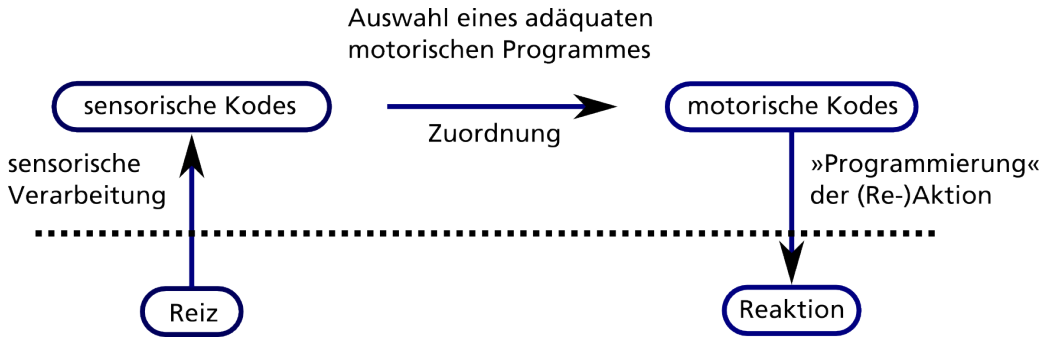

Abb. 1: Klassische Sicht von Wahrnehmung (Sensorik) und Handlungssteuerung (Motorik) nach Sanders (Grafik nach Schubö 1998)

2.1.1 Common Coding

Verhaltensexperimente mit Menschen haben gezeigt, dass die (visuelle) Wahrnehmung von Aktionen anderer Individuen, eigene Aktionen auslösen (Induktion) oder beeinflussen (Inferenz) können. Prinz und Hommel haben versucht, dieses Phänomen mit dem Prinzip des *Common Coding*[3] (bzw. *Event Coding*[4]) zu erklären. Hierbei wird angenommen, dass sich Wahrnehmungsprozesse (Sensorik) und Prozesse zur Planung von Aktionen (Motorik), zumindest teilweise, eine gemeinsame Kodierungs- bzw Repräsentationsebene teilen. D.h. dass die visuelle Wahrnehmung einer Aktion »ähnlich« kodiert wird, wie die Ausführung der selbigen.

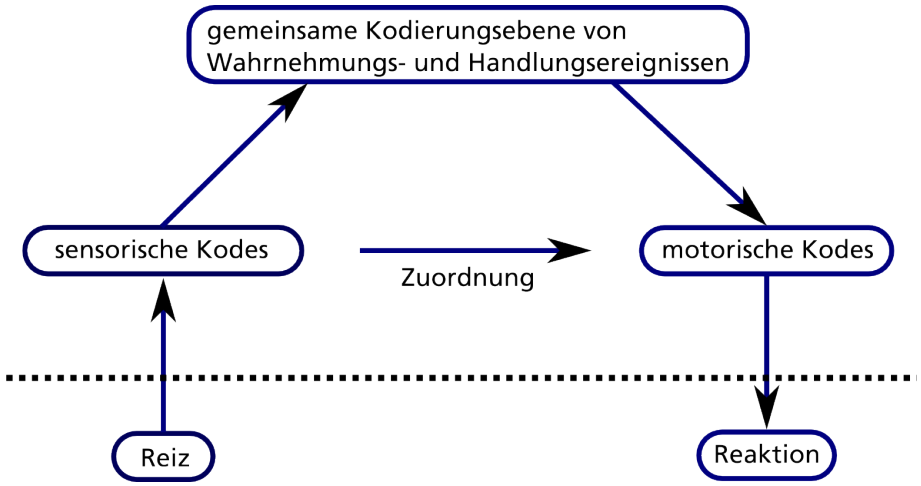

Abb. 2: Wahrnehmung (Sensorik) und Handlungssteuerung (Motorik) nach dem Common Coding Modell (Grafik nach Schubö 1998)

Neben den Verhaltensexperimenten stützt die bei Menschen und Tieren vorhandene Fähigkeit, Aktionen über Imitation zu erlernen, die *Common Coding* Theorie. Sie erklärt nämlich auf direkte Weise, wie die komplizierte Aufgabe, eine visuell wahrgenommene Aktion eines anderen Individuums in eine eigene Aktion umzusetzen, gelöst werden könnte.

3 Prinz 1990, Prinz 1997, Schubö 1998
4 Hommel et al 2001

Decety und Grèzes haben auf neurophysiologischer Ebene nach Hinweisen für das *Common Coding* Modell gesucht[5]. Bei Versuchen haben sie Hirnregeionen gefunden, welche sowohl bei der Wahrnehmung von Aktionen als auch bei der Ausführung selbiger aktiv waren. Einschränkend muss jedoch gesagt werden, dass es diese deutlichen Ergebnisse nur unter der Voraussetzung gab, dass die Versuchspersonen die konkrete Aufgabe hatten, die beobachtete Aktion zu imitieren. Es überrascht auf der anderen Seite allerdings auch nicht, dass diese gemeinsame Kodierungsebene mit einfachen bildgebenden Verfahren auf neuronaler Ebene nicht völlig widerspruchsfrei bestimmt werden kann.

2.1.2 Mental Imagery und Motorik

Über die Rolle der Motorik bei der Repräsentation von der Umwelt im Gehirn geben auch Experimente im Bereich der mentalen Bilder (*Mental Imagery)* Aufschluss[6].

Zunächst einmal ergaben Experimente, dass bei Versuchspersonen, welche Aufgaben mit bildlicher Vorstellung zu lösen hatten, Gehirnareale aktiv waren, welche der höheren visuellen Wahrnehmung zugesprochen werden[7], d.h. dass bildliche Vorstellung und visuelle Wahrnehmung sehr wahrscheinlich auf gemeinsamen neuronalen Systemen basieren. Für die Frage inwiefern Motorik hier hineinspielt, geben zwei im Folgenden näher beschriebene Versuchsreihen Hinweise.

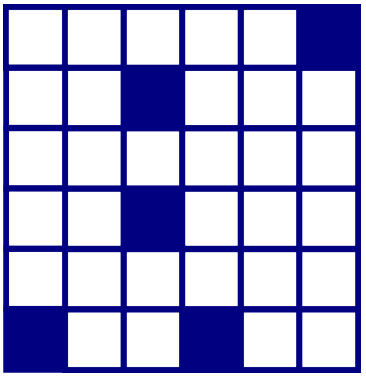

Abb. 3: Beispiel eines Schachbrettbildes mit fünf ausgefüllten Feldern (Grafik nach Laeng, Teodorescu 2001)

1	2	3	4	5	6
7	8	9	10	11	12
13	14	15	16	17	18
19	20	21	22	23	24
25	26	27	28	29	30
31	32	33	34	35	36

Abb. 4: Schachbrett mit nummerierten Feldern (Grafik nach Laeng, Teodorescu 2001)

Laeng und Teodorescu haben Experimente gemacht, in denen sie versuchten, die Rolle von Augenbewegungen in Verbindung mit bildlicher Vorstellung zu bestimmen[8].

In einem ersten Experiment mussten sich die Versuchpersonen fünf Quadrate in einem 6x6 Schachbrettmuster einprägen (siehe Abb. 3). Eine Gruppe war

5 Decety, Grèzes 1999

6 Bei mentalen Bildern handelt es sich um die Vorstellung eines frei erfundenen oder erinnerten visuellen Bildes oder Eindruckes. Auf die Diskussion über die Existenz von mentalen Bildern, auch unter *mental imagery debate* bekannt, wird hier nicht weiter eingegangen. Eine Übersicht aus der Sicht des »unterstützenden Lagers« wird in Kosslyn 1980 und Kosslyn 1994 gegeben.

7 Knauf et al 2000

8 Laeng, Teodorescu 2001

dazu angehalten dabei die Mitte des Schachbretts zu fixieren und die zweite Gruppe durfte in dieser Phase das Bild frei untersuchen. Als zweites Bild sahen die Versuchspersonen ein Schachbrett mit nummerierten Feldern. Die Aufgabe war nun, die Nummer der Felder zu bestimmen, welche im vorigen Bild ausgefüllt waren.

Laeng und Teodorescu haben festgestellt, dass die Versuchspersonen aus der zweiten Gruppe in der Phase der bildlichen Vorstellung (des ersten Schachbretts mit fünf ausgefüllten Feldern) fast die gleichen Augenbewegungen gemacht haben, wie während der Einprägungsphase, während bei Personen aus der ersten Gruppe in der zweiten Phase fast gar keine Augenbewegungen registriert wurden. Des Weiteren war auffällig, dass eine größere Übereinstimmung zwischen den Augenbewegungen in der Einprägungsphase und in der Phase der bildlichen Vorstellung auf eine höhere Trefferquote bei der Bestimmung der Felder hinwies. Einschränkend muss jedoch angemerkt werden, dass die beiden Versuchsgruppen letztendlich die gleiche Trefferquote hatten. Mast und Kosslyn versuchen dies damit zu erklären, dass die Versuchspersonen der ersten Gruppe eine Art »verdeckte Augenbewegungen« ausgeführt haben, indem sie ihre Aufmerksamkeit auf bestimmte Punkte des Schachbretts gelenkt haben, ohne ihre Augen zu bewegen[9].

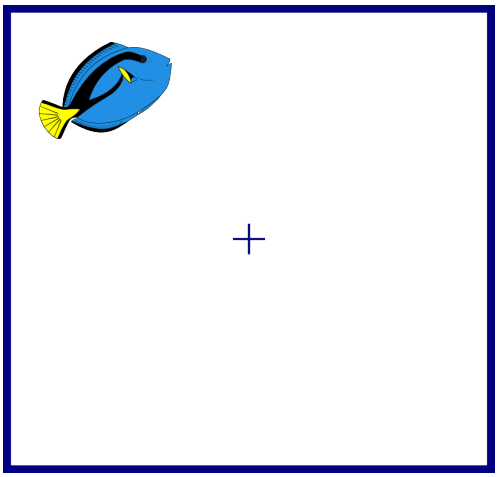

Abb. 5: Experiment bei dem Versuchspersonen sich Eigenschaften der Fische merken mussten. (Grafik nach Laeng, Teodorescu 2001)

In einem zweiten Versuch haben Laeng und Teodorescu versucht zu ermitteln, ob die Augenbewegungen bei der bildlichen Vorstellung eine Funktion haben oder ob es sich lediglich um Epiphänomene handelt. Die Teilnehmer haben dafür ein Bild mit Fixationspunkt in der Mitte zu sehen bekommen. Zusätzlich wurde in einer der vier Ecken das Bild eines Tropenfisches eingeblendet. Nach der Ausblendung wurden die Versuchspersonen zu Eigenschaften der eben angezeigten Fische befragt. Eine Gruppe musste in dieser Phase (bildliche Vorstellung) den Punkt in der Mitte des Bildes fixieren und eine andere Gruppe durfte ihren Blick frei über das Bild bewegen. Die erste Gruppe, welcher es nicht erlaubt war in der Phase der bildlichen Vorstellung Augenbe-

9 Mast, Kosslyn 2002

wegungen (aus der vorigen Einprägungsphase erneut) auszuführen, schnitt bei der Rekonstruktion der Bilder deutlich schlechter ab.

Laeng und Teodorescu kommen zu dem Schluss, dass während der (geistigen) Vorstellung eines erinnerten Bildes die Augenbewegungen, welche bei der ursprünglichen Einprägung des selbigen Bildes ausgeführt wurden, ein wichtiger Zugang zu dieser Erinnerung sind. Allgemeiner ausgedrückt ist die bei der Einprägung eines visuellen Bildes verwendete Motorik auch beim Abruf dieser Erinnerung von Bedeutung.

Eine weitere Versuchsreihe zu bildlicher Vorstellung und Motorik haben Wexler et al. druchgeführt. Sie haben versucht zu ermitteln, in welchem Verhältnis Transformationen von bildlichen Vorstellungen und motorische Prozesse zueinander stehen. Dafür wurde Versuchspersonen die Aufgabe gestellt eine »mentale« Rotation[10] und eine motorische Rotation auszuführen. Die Ergebnisse deuten darauf hin, dass es Inteferenzen zwischen den beiden Rotationen gibt[11].

Die Aufgabe der mentalen Rotation war in Form des *Shepard tasks* gestellt; hierbei wurden den Versuchspersonen nacheinander zwei Bilder eines Objektes gezeigt, welche rotiert und optional gespiegelt waren (s. Abb. 6). Die Versuchsperson musste entscheiden, ob es sich bei den Bildern um Spiegelungen handelte oder ob sie nach einer (mentalen) Rotation des ersten Bildes deckungsgleich sind.

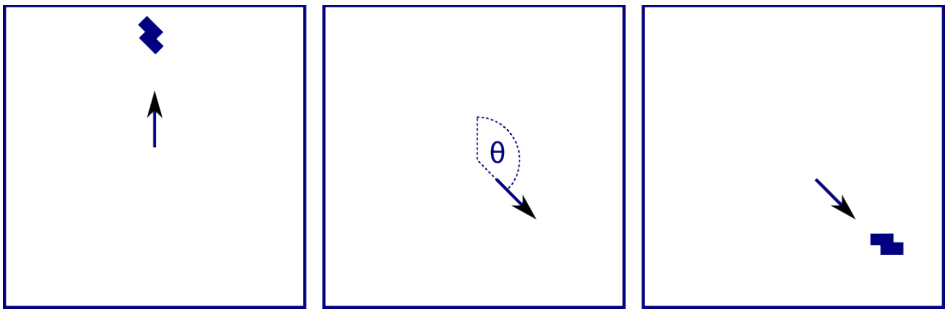

Abb. 6: Abfolge von Bildern, welche den Versuchspersonen gezeigt wurden (1. Objekt in Ausgangsposition, 2. Markierung an der das transformierte Objekt erscheinen wird, 3. Um θ° rotiertes und ggf. gespiegeltes Objekt.) (Grafik nach Wexler et al. 1998)

Gleichzeitig zu dieser mentalen Rotation haben die Versuchspersonen einen Joystick in einer vorher gelernten Geschwindigkeit und in einer vorgegebenen Richtung gedreht.

Wexler et al. haben beobachtet, dass wenn die mentale Rotation und die motorische Rotation kompatibel[12] waren, die Versuche schneller und mit weniger Fehlern absolviert wurden. Des Weiteren hatte die Geschwindigkeit bzw. der Winkel der motorischen Rotation Auswirkungen auf die Geschwindigkeit und

10 Rotation einer bildlichen Vorstellung (eines *Mental Images*)

11 Wexler et al. 1998

12 Ein um 45°, 90° und 135° im Uhrzeigersinn rotiertes Bild ist kompatibel zu einer motorischen Rotation im Uhrzeigersinn und ein um 225°, 270° und 315° rotiertes Bild zu einer Rotation gegen den Uhrzeigersinn.

Korrektheit in der die Aufgabe gelöst wurde bzw. in der die mentale Rotation ausgeführt wurde.

Diese Versuchsergebnisse führen Wexler et al letztendlich zu dem Schluss, dass (quasi-)motorische Prozesse verwendet werden, selbst wenn abstrakte Objekte in der Vorstellung, d.h. »mental«, rotiert werden.

2.1.3 Visual Guidance

Zweifel an der beschriebenen konzeptuelle Trennung der motorischen und sensorischen Verarbeitung im Gehirn haben u.a. auch Beobachtungen am gut erforschten visuellen System von Primaten auftreten lassen.

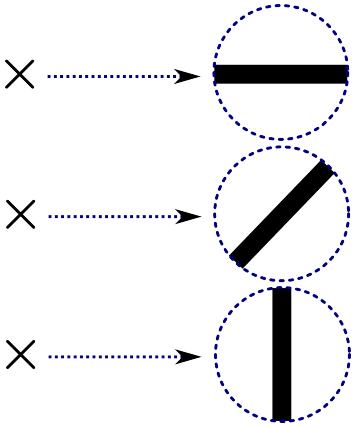

Abb. 7: Fixationspunkt (links) und visueller Reiz (rechts) zu dem eine Augenbewegung gemacht wird. (Grafik nach Moore 1999)

Eine Studie von Moore[13] ergab, dass farb- und orientierungsselektive Neuronen, welchen bislang ausschließlich passive, sensorische Funktionen zugeschrieben waren, direkten Einfluss auf die Augenbewegungen (*Sakkaden*) haben. Vor einer *Sakkade* von einem Fixationspunkt zu einem Balken war bei den betrachteten Neuronen eine Aktivität zu beobachten. Die Orientierung des (Ziel-)Balkens hatte dabei Einfluss auf die Aktivität der Neuronen und auf die Ausführung (Motorik) der *Sakkade*.

Moore interpretiert die präsakkadische Aktivität der Neuronen nicht als direkten motorischen Impuls, kommt jedoch zu dem Ergebnis, dass sie direkten Einfluss auf die motorische Aktion haben, so dass eine strikte Trennung der visuellen (d.h. sensorischen) und motorischen Funktion dieser Neuronen fragwürdig erscheint.

2.1.4 Affordance

Im Zusammenhang mit Sensomotorik sollte auch die *Ecological Psychology* von J.J. Gibson erwähnt werden. Zwar lehnt Gibson eine interne, mentale Repräsentation der Umwelt generell ab, führt jedoch auch ein, Objekte über ihre Interaktionsmöglichkeiten zu klassifizieren. Diese unter dem Begriff *Affordance* bekannte Sicht betont die (Inter-)Aktionen, welche ein Gegenstand erlaubt; d.h. ein »Stuhl« ist nicht darüber definiert, dass er eine Sitzfläche, Lehne und vier Beine hat, sondern, dass man u.a. darauf sitzen kann.

Duchon et al. beschreiben, wie die Ideen von Gibson z.B. im Feld der moderne Robotik wiederzufinden sind und wie die *Ecological Pyschology* diese neuen Entwicklungen bereichern kann[14]. Ein Beispiel sind die *Control Laws*, wie sie in der *Spatial Semantic Hierachy* von Kuipers verwendet werden (siehe nächstes Kapitel).

13 Moore 1999
14 Duchon et al 1998

Auch wenn Repräsentationen der Umwelt wie erwähnt abgelehnt werden, ist doch zu bemerken wie die Frage nach der Auswahl von Aktionen, z.B. *wie* man vom Ort A zum Ort B kommt, im Mittelpunkt steht:

> *[...] this work points to [...] a tighter binding between action and perception. [...] knowledge of the affordances of an environment provides a basis of a choice of action [...] (Duchon et al 1998, S. 26)*

Möller hat die Ideen von Gibson von einem reaktiven sensomotorischen System ohne interne Repräsentation zu einem antizipatorischen Agenten weiterentwickelt, dessen Repräsentation die Bewertung von sensomotorischen Sequenzen enthält[15]. Dadurch kann es vor der Ausführung einer der möglichen motorischen Aktionen, dessen Folgen antizipieren und danach entscheiden.

Zusammenfassend lässt sich sagen, dass Gibson Aktionen und ihre enge Kopplung an Wahrnehmungsprozessen betont.

15 Möller 1999

2.2 Neuronales Substrat der Raumrepräsentation

Im Folgenden werden Erkenntnisse zur neuronalen Ebene der Raumrepräsentation beschrieben. Dieses Feld ist besonders bei Ratten gut erforscht und umfasst sog. *Place, Head-Direction* und *Grid Cells*, welche sich im *Hippocampus* und anliegenden Hirnregionen befinden.

2.2.1 Place Cells

Bei *Place Cells* handelt es sich um bereits 1978 bei Ratten im *Hippocampus*[16] nachgewiesene Neuronen, welche abhängig von der räumlichen Lage der Ratte aktiv sind[17]. Eine Zelle repräsentiert in etwa 10 bis 20 Prozent einer Umgebung, das sog. *Place Field*, in der sich die Ratte momentan befindet.

Die Eigenschaften der Zellen wurden ermittelt, indem man Ratten in einem Labyrinth nach Futter hat suchen lassen. Bestimmte Neuronen(-gruppen) waren nur dann aktiv, wenn sich eine Ratte an einem bestimmten Punkt im Labyrinth befand.

Die ursprüngliche Annahme, die Aktivität von *Place Cells* sei ausschließlich an visuelle Markierungen (z.B. bunte Karten im Labyrinth) gebunden, ist mittlerweile zumindest teilweise in Frage gestellt worden. Beobachtet wurde, dass ihre Aktivität auch von der räumlichen Orientierung, welche die Ratte von ihrer Eigenbewegung ableitet (*Dead Reckoning*), beeinflusst wird[18]. Eine Hypothese ist, dass Ratten mittels *Dead Reckoning* die Zuverlässigkeit von visuellen Hinweisen bestimmen.

Generell ist die Aktivität einer *Place Cell* kontextspezifisch; d.h. in unterschiedlichen Umgebungen repräsentiert eine Zelle völlig unterschiedliche Orte. Experimente von Leutgeb et al.[19] haben gezeigt, dass es im Hippocampus zwei Arten von *Place Cells* gibt. Die Zellen im *CA3* zeichen sich dadurch aus, dass sie eine stärker Kontextabhängigkeit haben als *Place Cells* im *CA1*.

2.2.2 Head Direction Cells

Neben den *Place Cells* wurden bei Ratten in Hirnregionen[20] mit direkten und indirekten Verbindungen zum Hippocampus sog. *Head-Direction Cells* gefunden[21]. Diese Zellen sind unabhängig von der Position einer Ratte, aber selektiv bzgl. der Orientierung des Kopfes. Sie zeigen nur bei *einem* bestimmten Winkel eine besondere Aktivität; je weiter die Orientierung des Rattenkopfes von diesem Winkel abweicht, desto schwächer ist diese Aktivität. Die Orientierungen, welche beobachtete Zellen präferieren, decken komplett 360° ab und sind über Tage stabil. Als Referenz für diese Orientierungen dienen sehr warscheinlich visuelle Markierungen, welche auch schon bei *Place Cells* eine Rolle spielten.

16 Bestandteil des Gehirns
17 O'Keefe, Conway 1978
18 Knierim et al 1995
19 Leutgeb et al. 2004 zit. n. Bilkey 2004
20 Subiculum, Präsubikulum, Postsubikulum, Parasubikulum, entorhinaler und perirhinaler Kortex
21 Taube et al 1990a, Taube et al 1990b

Es wird angenommen, dass die von diesen Zellen kodierten Informationen u.a. für die Abschätzung der eigenen Position über die beobachtete Eigenbewegung (*Dead Reckoning*) verwendet werden.

2.2.3 Grid Cells

Neben den *Place Cells* im *Hippocampus* wurden kürzlich weitere räumlich selektive Neuronen im *dorsocaudal Medial Entorhinal Cortex* (*dMEC*), eine Nachbarregion des *Hippocampus*, gefunden[22]. Diese Neuronen sind, im Gegensatz zu den *Place cells*, nicht nur dann aktiv, wenn sich eine Ratte an *einem* bestimmten Ort befindet, sondern sie »reagieren« auf mehrere Orte. Diese Orte unterteilen gewissermaßen den von den Ratten explorierten Raum in ein Gatter (*Grid*) aus gleichseitigen Dreiecken (siehe Abb. 8).

Die Größe, Phase und Orientierung der Dreiecke, welche die Grids bilden, variiert zwischen den einzelnen Neuronen (siehe Abb. 9); zwischen benachbarten Neuronen variieren Größe und Orientierung jedoch weniger als zwischen weit voneinander entfernten.

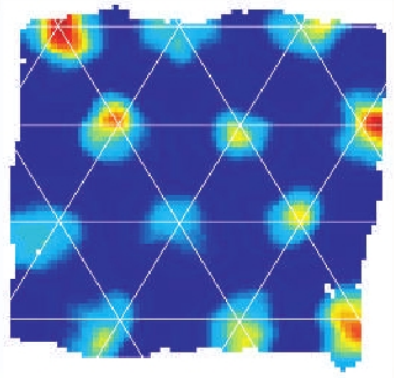

Abb. 8: Aktivitätsdiagramm eines Neurons aus dem dMEC (Grafik nach Hafting et al 2005)

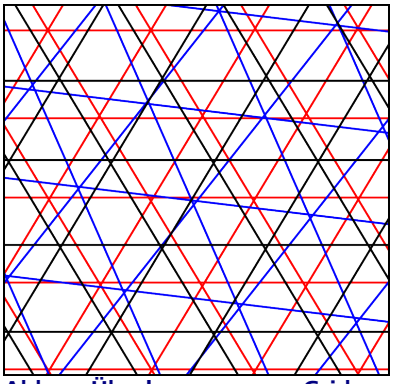

Abb. 9: Überlagerung von Grids, welche die Neuronen »aufbauen« (Grafik nach Hafting et al 2005)

Die Entfernung zwischen den Orten, an denen eine *Grid Cell* aktiv ist – und damit die Größe der gridbildenden Dreiecke –, variiert von Zelle zu Zelle zwischen 40 und 70cm. Eine *Grid Cell* feuert jedoch erstaunlich regelmäßig; die Standardabweichung liegt hier bei lediglich 3,2 cm.

Die Phase und Orientierung der Grids ist an Orientierungspunkte in der Umgebung gebunden und wird nicht von der Eigenbewegung abgeleitet (D*ead Reckoning*). Hat sich das Grid aber einmal stabilisiert, ist es anschließend auch ohne diese Orientierungspunkte, z.B. in völliger Dunkelheit, aktiv.

Es gibt Hinweise, dass *Grid Cells* in unterschiedlichen Umgebungen in etwa die gleiche Aktivität vorweisen und damit die gleiche Funktion erfüllen. Damit grenzen sie sich von *Place Cells* ab, deren Aktivität in unterschiedlichen Kontexten stark variiert.

Eine ungelöste Frage ist noch, wie aus der Aktivität der *Grid Cells* eine Position abgeleitet wird. Die Aktivität der Zellen ist an unterschiedlichen Positio-

22 Hafting et al 2005, Fyhn 2004

nen doch recht ähnlich, und es bedarf schon einiger Zellen, welche sich in Phase, Orientierung und Abstand ausreichend unterscheiden, um zu einem zuverlässigen Ergebnis zu kommen. Und ob diese potentielle »Berechnung« im Kortex (*dMEC*) oder im Hippocampus stattfindet, ist völlig offen.

2.3 Raumrepräsentation im Gehirn

Was lässt sich nun aus den zusammengetragenen Erkenntnissen für die Repräsentation von Raum im Gehirn schließen?

2.3.1 Cognitive Map

Zur Beantwortung dieser Frage müssen vielleicht zunächst einmal die Anfänge der Forschung zur Raumrepräsentation im Gehirn erläutert werden.

Tolman machte Versuche mit Ratten, in welchen sie in einem Labyrinth nach Nahrung suchen[23]. Das Navigationsverhalten der Ratten führt Tolman zu dem Schluss, dass selbige eine mentale Karte von ihrer Umgebung aufbauen, anhand der sie sich orientieren. Für diese Art der Raumrepräsentation prägt er den Begriff *Cognitive Map*.

Siegel und White[24] entwickelten zu dieser Theorie ein Modell, wie diese Karte aufgebaut, d.h. gelernt wird. Sie gehen davon aus, dass dies ein Prozess ist, der sich in drei Phasen unterteilen lässt. Am Anfang stehen die *Landmarks* – eindeutige Objekte in einer Umgebung mit einer festen Position, gewissermaßen Orientierungspunkte. In der nächsten Stufe wird das *Route Knowledge* aufgebaut, indem Sequenzen von *Landmarks,* wie sie während einer Exploration wahrgenommen werden, durch Routen repräsentiert werden. Erst in der letzten Stufe, der z.T. metrischen *Survey Knowledge*, werden Routen und *Landmarks* zu einer Repräsentation integriert und in einen globalen Bezugsrahmen gesetzt.

Nachdem nun anfangs davon ausgegangen wurde, dass bei Menschen und Tieren Raum mittels einer mentalen Karte – im Sinne eines zweidimensionalen geometrisch korrekten Abbilds der Umwelt – repräsentiert wird, kamen Zweifel auf, wie kartenähnlich diese *Cognitive Map* wirklich ist.

Tversky führte den Begriff der *Cognitive Collage* und des *Spatial Mental Models* ein, um Erkenntnissen aus der Psychologie über die mentale Repräsentation von Raum Rechnung zu tragen, welche sich mit einer mentalen Karte nicht vereinen lassen[25]. *Cognitive Collage* soll dabei zum Ausdruck bringen, dass räumliches Wissen häufig aus sehr unterschiedlichen Komponenten besteht, von denen einige keineswegs kartenähnlich sind. Und *Spatial Mental Models* umfassen räumliche, mehr oder weniger akkurate, dreidimensionale Repräsentationen von einfachen oder gut bekannten Umgebungen, welche vor allem räumliche Relationen bewahren und weniger exakte metrische Informationen. In dieser Repräsentation ist es Menschen z.B. möglich, die eigene Perspektive in der Umgebung zu ändern.

23 Tolman 1949
24 Siegel, White 1975 zit. n. Werner et al. 2000
25 Tversky 1993

Gemein haben jedoch die *Cognitive Map*, die *Cognitive Collage* und das *Spatial Mental Model*, dass sie allein auf während der Exploration gesammelten sensorischen Informationen beruhen.

2.3.2 Sensomotorik und Raumrepräsentation

Die Ergebnisse aus den Experimenten zu *Visual Guidance*[26] und *Common Coding*[27] deuten an, dass sich Sensorik und Motorik im Gehirn nicht strikt trennen lassen und sie evtl. eine gemeinsame Repräsentationsebene haben.

Neben dieser generellen Sicht auf Sensorik und Motorik ergaben Experimente im Bereich der *Mental Imagery*[28], dass es einen direkten Zusammenhang zwischen der Rotation von mentalen Bildern, welche auch eine Form einer Repräsentation darstellen, und der »echten« motorischen Rotation gibt. Und ein weiteres Experiment ergab, dass für den Zugriff auf Erinnerungen in Form von mentalen Bildern die motorischen Aktionen eine Rolle spielen, welche bei der Wahrnehmung ausgeführt wurden.

Auf der Ebene der Repräsentation von Raum könnte die Integration der Motorik den Vorteil bieten, dass für die Aufgabe, mit Hilfe der Repräsentation einer bekannten Umgebung zu navigieren, bereits Information verfügbar ist *wie* ein bestimmter Ort erreicht werden kann und nicht nur *wo* dieser ungefähr liegt.

Diese Indizien unterstützen, dass es wahrscheinlich ein lohnenswertes Unterfangen ist, in räumlichen Repräsentationen die für die Exploration nötige Motorik zu berücksichtigen, anstatt allein sensorische Informationen in Betracht zu ziehen.

2.3.3 Place Cells & Grid Cells

Welche Hinweise geben nun die Erkenntnisse über *Place Cells* und *Grid Cells* zur Repräsentation von Raum im Gehirn?

Die Frage, ob es sich bei der Repräsentation im Kortex (*dMEC*) um eine *Cognitive Map* im klassischen Sinne handelt, kann wohl noch nicht beantwortet werden, da es momentan recht wenig Versuchsergebnisse gibt. Fyhn et al. haben gezeigt, dass Läsionen im Hippocampus die *Grid Cells* im Kortex nicht wirklich beeinträchtigt haben und damit diese Repräsentation wahrscheinlich nicht auf den in den *Place Cells* kodierten Informationen aufbaut. Ob andersherum die in den *Place Cells* kodierten Informationen (ausschließlich) auf den *Grid Cells* basieren, ist noch offen.

Bzgl. einer sensomotorischen Repräsentation ist ein neuronales Modell des Kortex und Hippocampus von Banquet et al. von besonderem Interesse[29]. Es ist in drei Ebenen untereilt: Das *Object Location Level* identifiziert Orientierungspunkte (*Landmarks*) anhand der zur Verfügung stehenden Sensorik, und das *Subject Location Level* berechnet die *Place Fields*[30] mit Hilfe der

26 siehe Kapitel 2.1.3, S. 14
27 siehe Kapitel 2.1.1, S. 10
28 siehe Kapitel 2.1.2, S. 11
29 Banquet et al. 2005
30 siehe Kapitel 2.2.1, S. 16

Landmarks und der von den *Grid Cells* im Kortex kodierten Information über die Eigenbewegung (*Dead Reckoning*). Die dritte Ebene, das *Spatiotemporal Level*, kodiert die räumlichen und zeitlichen Übergänge (*Temporospatial Transitions*) zwischen *Place Fields* über *Transition Cells*. D.h. sie ermöglichen das Kodieren der *Route Knowledge*[31] und lösen nach der Ansicht von Banqeut et al. das Problem, aus einer *Cognitive Map* für eine Navigationsaufgabe konkrete motorische Aktionen abzuleiten:

> *[...] a dynamical spatiotemporal (and not just spatial) representation of the space and task environment through the computation and encoding of transitions in the CA field (transition cells) [...] provided for a straightforward solution to the theoretical difficulty to switch from a spatial cognitive map to its motor implementation [...] (Banquet et al. 2005, S. 10)*

Dieses Modell bezieht damit durch die *Transition Cells*, wenn auch indirekt, eine motorische Komponente mit ein.

31 siehe S. 18

3 Existierende Explorationssysteme und Raumrepräsentationen

In diesem Kapitel werden aktuelle Modelle zur Raumrepräsentation und ihre Implementierung beschrieben und mit den neurophysiologischen und psychologischen Erkenntnissen über Raumrepräsentation aus dem letzten Kapitel ins Verhältnis gesetzt. Es handelt sich hier um keine erschöpfende Übersicht dieser Thematik; abgesehen von den *Occupancy Grids* werden vornehmlich Modelle vorgestellt, welche einer sensomotorischen Repräsentation nahe kommen.

3.1 Occupancy Grids

Eine dominierende Repräsentationsform aus dem Bereich der Robotik ist das von Moravec und Elfes eingeführte *Occupancy Grid*, welches die zu repräsentierende Umgebung in ein Gatter aus meist quadratischen Zellen unterteilet[32] (siehe Abb. 10). Für jede dieser Zellen wird gespeichert, ob der von ihr repräsentierte Bereich belegt ist , d.h. ein Hindernis enthält, oder nicht. Aufgebaut werden diese *Grids* meist mit Hilfe von Reichweitensensoren[33], welche Hindernisse in einem bestimmten Radius erkennen und dessen Entfernung bestimmen können.

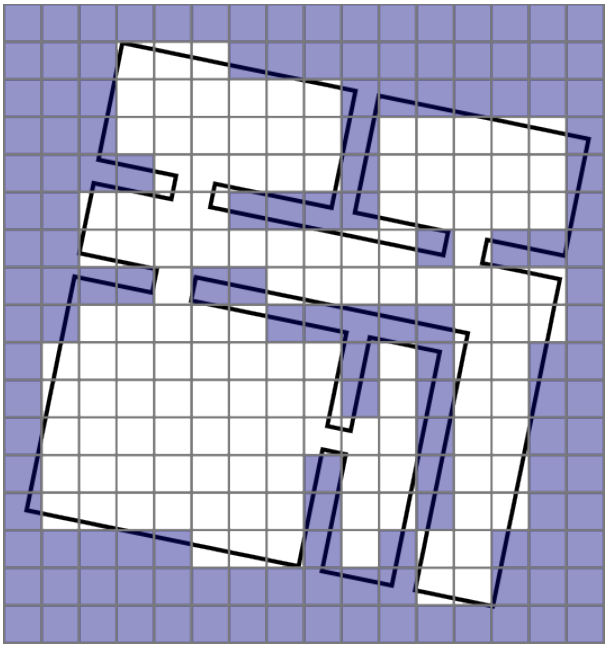

Abb. 10: Binäres Occupancy Grid einer »Innenraumumgebung« mit grober Auflösung. Skizze der repräsentierten Umgebung (schwarz), belegte Zelle (blau), »freie« Zelle (weiß)

Da verrauschte Sensordaten kein eindeutiges *Occupancy Grid* ergeben, werden den Zellen anstatt binären Belegungsindikatoren meist Wahrscheinlichkeitswerte für ihren Zustand zugewiesen. Die sich ergebenden Belegungen aus den einzelnen Sensorschnappschüssen werden dann gemäß der Bayes'schen Regel zu einem *Occupancy Grid* integriert.

Seine Position im Occupancy Grid bestimmt ein Roboter fortlaufend anhand der Eigenbewegung (*Dead Reckoning*). Um den sich mit der Zeit aufsummierenden Fehler durch z.B. durchdrehende Räder auszugleichen, wird jeder Sensorschnappschuss mit dem aktuellen *Occupancy Grid* verglichen, um

32 Moravec 1988, Thrun 1998, Elfes 1987 zit. n. Thrun 1998
33 Laser Range Scanner, Sonar u.ä.

einen evtl. Fehler seit dem letzten Schnappschuss abzuschätzen. Die dadurch erreichte Fehlerkorrektur in der Positionsbestimmung wird z.T. noch dadurch verfeinert, dass der Roboter sich an in Innenräumen häufig vorkommenden geraden Wänden ausrichtet.

Insgesamt lässt sich sagen, dass es sich bei *Occupancy Grids* um eine Raumrepräsentationsform handelt, die sich relativ einfach aufbauen und verwalten lässt und bzgl. Sensordaten ähnliche Orte gut unterscheidbar macht. Auf der anderen Seite ist die strategische Wegplanung kompliziert und auch das Erreichen eines naheliegenden in der Repräsentation enthaltenen Ortes ist eine alles andere als triviale Aufgabe, da in einem *Occupancy Grid* noch nicht enthalten ist, *wie* man zu einem bestimmten Ort gelangt. Allein das Wissen über die relative räumliche Lage des Ortes ergibt noch keine einfache Berechnung einer Aktion, die einen Agenten zuverlässig zu diesem Ort bringt.

3.2 Topological Maps & Route Maps

Topologische Karten (*Topological Maps*) speichern »spezielle« Orte (*Landmarks*) der Umgebung und in welchen Nachbarschaftsbeziehungen sich diese befinden. Zusätzlich zu den Nachbarschaftsbeziehungen ist in den im Folgenden beschriebenen Repräsentationen auch noch vertreten *ob* bzw. *wie* man von einem Ort zu einem anderen Ort gelangt. Sie lassen sich gut als Graphen visualisieren, dessen Knoten *Landmarks* repräsentieren und dessen Kanten Verbindungen bzw. Übergänge zwischen zwei *Landmarks* darstellen. Kritische Punkte bei der Konstruktion von Topologischen Karten ist die eindeutige Identifizierung und Wiedererkennung von *Landmarks*, sowie die Kodierung der Übergänge.

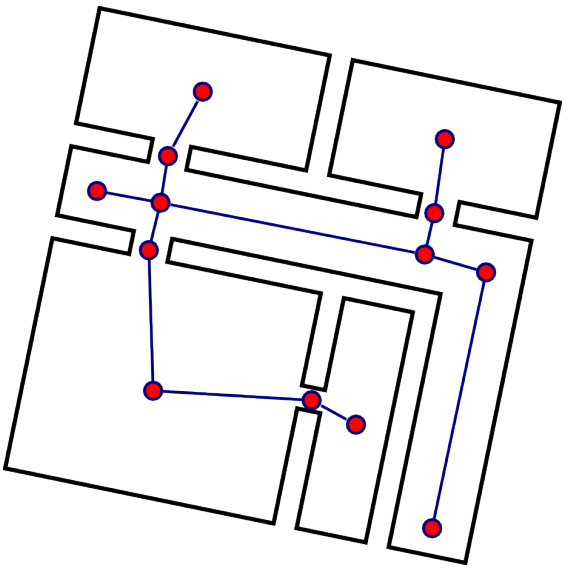

Abb. 11: Mögliche topologische Karte der Umgebung aus .

Thrun hat versucht aus dem Feld der *Occupancy Grids* eine Brücke zu topologischen Karten zu bauen[34]; ein grundsätzlicherer Ansatz findet sich jedoch in der *Spatial Semantic Hierarchy (SSH)* von Kuipers[35], dem *View Graph* von Schölkopf und Mallot[36] oder dem *Route Graph* von Werner et al.[37].

34 Thrun 1998
35 Kuipers 2000, Remolina et al 2004
36 Schölkopf, Mallot 1995
37 Werner et al. 2000

3.2.1 Topologische Karte auf Basis eines Occupancy Grids

Thrun, welcher maßgeblich an der Entwicklung von *Occupancy Grids* arbeitet, hat ein System entwickelt, welches auf Basis einer bestehenden Repräsentation in Form eines *Grids* eine topologische Karte aufbaut, um effizientere Wegplanung zu ermöglichen[32].

Im ersten Schritt wird mit Hilfe des *Occupancy Grid* ein *Voronoi Diagramm*[38] aufgebaut. Dieses Diagramm besteht aus Punkten, welche einen minimalen Abstand zu mindestens zwei Hindernissen im *Grid* haben. Aus der Menge dieser Punkte werden die *Critical Points* bestimmt, deren minimaler Abstand geringer ist als der aller Punkte in einer ε-Umgebung. Die *Critical Points* bilden in dem System von Thrun Knoten der topologischen Karte anhand derer eine effizientere Wegplanung möglich ist.

3.2.2 Spatial Semantic Hierarchy

Die *Spatial Semantic Hierarchy (SSH)*[39] von Kuipers ist, wie der Name schon andeutet, in mehrere hierarchisch angeordnete Verarbeitungs- und Repräsentationsebenen unterteilt:

Sensory & Control Level

Die unterste Schicht behandelt die Sensorik und die Motorik des Systems. Die Bewegungen des Agenten sind über *Control laws* definiert, welche jeweils die Beziehung zwischen eingehenden Sensordaten und den ausgehenden Motorsignalen spezifizieren. D.h. Aktionen zur Exploration der Umgebung werden ausschließlich aufgrund der Sensorik und nicht mit Hilfe metrischer Karten ausgeführt. Erst wenn eine Umgebung genügend erkundet ist und eine zuverlässige topologische Karte aufgebaut wurde, werden metrische Informationen z.B. zur Berechnung von Abkürzungen herangezogen.

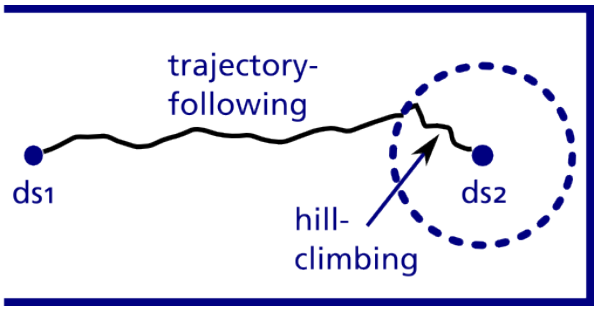

Abb. 12: Bewegung vom *Distinctive State ds1* zu *ds2* mit einem *Trajectory-Following* und einem *Hill-Climbing Control Law.* (Grafik nach Kuipers 2000)

38 für weitere Informationen siehe Aurenhammer 1991
39 Kuipers 2000 und Remolina et al 2004

Kuipers unterscheidet zwei Klassen von *Control Laws* (siehe Abb. 12). *Trajectory-Following Control Laws* bringen den Agent typischerweise von einem *Distinctive State* zu einer neuen Umgebung; z.B. »folge dieser Wand im Abstand von einem Meter« oder »Gehe in der Mitte dieses Ganges«. Die zweite Klasse sind die *Hill-Climbing Control Laws,* welche den Agenten in einer neuen Umgebung zu einem *Distinctive State* bringen. Unter einem *Distinctive State* versteht Kuipers einen festen und eindeutigen Zustand, den der Agent durch das Ausführen eines C*ontrol Laws* erreichen kann. Sie sind gewissermaßen die Fixpunkte an der die Raumrepräsentation aufgehängt ist. Und da sie allein durch eine Sequenz von *Control Laws* definiert sind, benötigen sie keine Abschätzung der Position in einem metrischen Koordinatensystem, welche durch fehlerhafte odometrische Daten viele Probleme nach sich zieht (siehe *Occupancy Grids*).

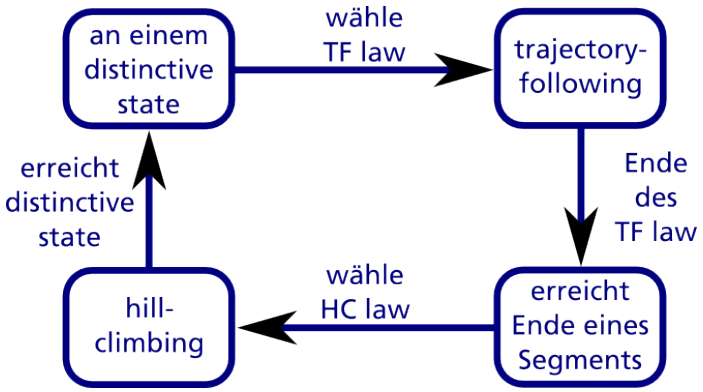

Abb. 13: Auffinden von *distinctive states* durch abwechselnde *trajectory-following* und *hill-climbing Control Laws* (Grafik nach Kuipers 2000).

Um eine Umgebung explorieren zu können wird also eine Menge von *Control Laws,* eine Methode zur Selektion selbiger sowie Bedingungen, welche bestimmen wann ein *Control Law* das Ende eines Segments erreicht, benötigt.

Causal Level

Auf dem *Causal Level* wird zunächst die Sequenz *(Distinctive State1, Control Laws, Distinctive State2)* zu *(View1, Action, View2)* abstrahiert. Ein *View* entspricht den Sensordaten an einem *Distinctive State,* und *Action* fasst mehrere *Control Laws*, welche den Agenten zu einem nächsten *Distinctive State* bringen, zusammen. Es wird dabei noch zwischen *Turn* und *Travel Actions* unterschieden: Bei Ersteren ist der Agent nach der Aktion noch am selben Ort, da er lediglich seine Orientierung ändert, und *Travel Actions* bringen den Agenten an einen neuen Ort.

Der (auch in Kapitel 3.2.4 kurz beschriebene) *View Graph,* der aus den gesammelten *Views* und *Actions* aufgebaut werden kann, hat Probleme mit Ambiguitäten umzugehen; wenn an zwei Orten die gleichen Sensordaten gemessen werden, sind diese beiden Orte bzgl. des *Views* nicht unterscheidbar. In Abb. 14 ermittelt der Agent an den drei Orten *a, b* und *c* den gleichen *View*, nämlich eine Kreuzung, welches einen wenig Aussagekräftigen *View Graph* ergibt.

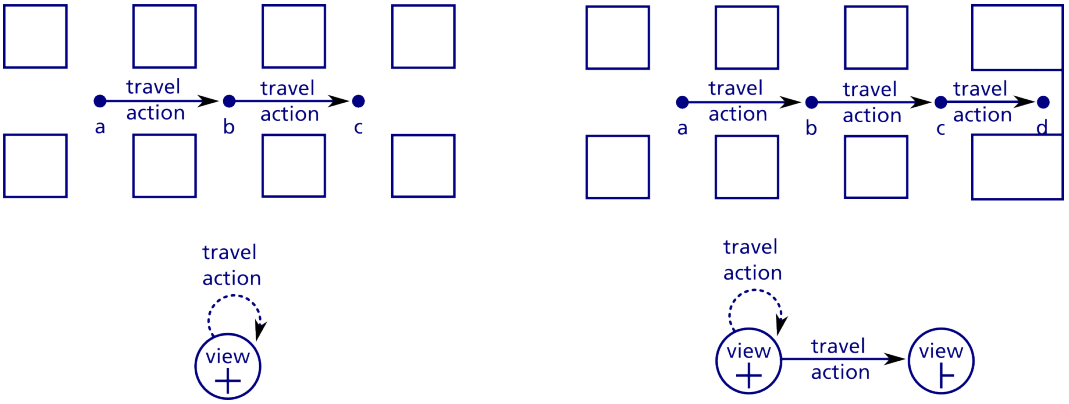

Abb. 14: Skizze einer Umgebung und inwieweit sie exploriert wurde (links) mit dazugehörigen *View Graph* (rechts). An den drei Orten *a, b* und *c* hat die Sensorik immer eine Kreuzung »erkannt«, nur *d* unterscheidet sich bzgl. der Sensordaten (Grafik nach Remolina et al 2004).

Kuipers unterscheidet die Zustände, in denen sich der Agent befinden kann, auf dem *Causal Level* auch bzgl. der Aktionen bzw., wie der Name schon andeutet, die kausalen Zusammenhänge zwischen *Views* und *Actions*: »Wenn ich *Action a* ausführe komme ich zu *View b*«. Hierbei werden jedoch (noch) nicht die räumlichen Qualitäten der Aktionen berücksichtigt. Zwei Zustände die auf dem *Causal Level* nicht unterscheidbar sind, beschreibt Kuipers mit dem Prädikat *Causally Equal (ceq)*.

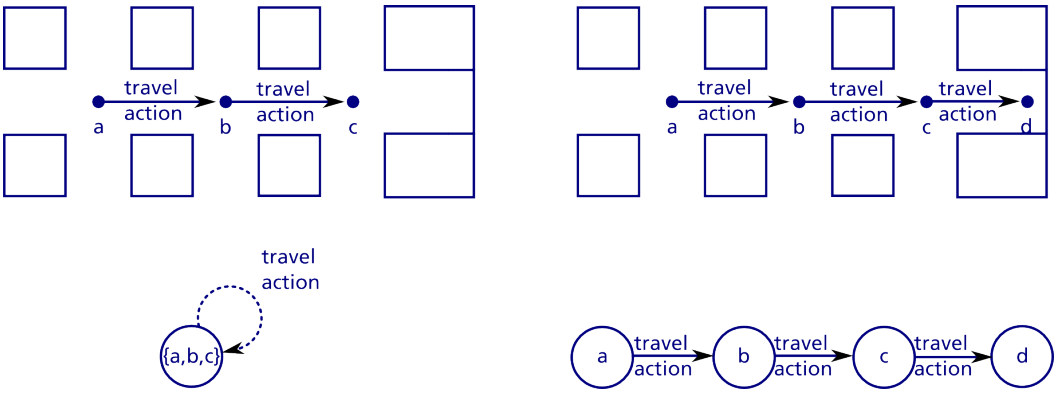

Abb. 15: Die Skizze der Umgebung und inwieweit sie exploriert wurde (oben) und der dazugehörige *causal graph* unten (Grafik nach Remolina et al 2004).

Für die linke Umgebung aus Abb. 15 sind der *Causal Graph* und der *View Graph* der gleiche. In der rechten Umgebung werden die Orte *a, b* und *c* jedoch im *Causal Graph* unterscheidbar, d.h. *ceq(a,b), ceq(a,c)* etc. gilt nicht mehr. Dies folgt aus der Bedingung, dass bei nicht unterscheidbaren Zustän-

den eine beliebige Sequenz von Aktionen die gleich Sequenz an *Views* hervorbringen muss. Damit z.b. *ceq(b,c)* gilt, müsste *b→Travel-Action* den gleichen *View* wie *c→Travel-Action* ergeben, was aber offensichtlich nicht der Fall ist.

Topological Level

Auf dem *Topological Level* wird nun nicht nur Abfolge von *Views* und *Actions* berücksichtigt, sondern die räumlichen Qualitäten der *Actions*, welche wie bereits erwähnt in *Turn* und *Travel Actions* unterteilt werden.

Die Umgebung wird nun durch Orte (*Places)* und Pfade (P*aths)* repräsentiert. Aufgebaut wird diese Repräsentation aufgrund der *View-Action-View*-Sequenzen aus dem *Causal Level.* Genauer gesagt wird mittels Abduktion versucht das Wissen aus dem *Causal Level* mit einer minimalen Anzahl an *Places* und *Paths* zu erklären.

Unter einem *Place* fasst Kuipers nun alle *Distincitve States* zusammen, welche lediglich durch eine *Travel Action* verknüpft sind, und es ist die Möglichkeit vorgesehen, dass ein *Place* im Zuge einer Abstraktion eine komplette *Region* repräsentiert.

Ein *Path* repräsentiert die Verknüpfung von *Places* und durch *Travel Actions*, ohne *Turn Actions,* und stellt damit eine Ordnungsrelation zwischen *Places* auf. Ein *Path* hat immer eine Richtung und entspricht in etwa einer Straße oder einem Gang in der Umgebung.

Eine *Region* repräsentiert einen Teil der Umgebung und ist durch ein gemeinsames Bezugssystem, umgebende Grenzen oder eine Abstraktion von *Places* definiert.

Auf *Topological Level* wird die Unterscheidbarkeit von *Distincitve States* gegenüber dem *Causal Level* weiter verfeinert. Zwei Zustände sind *Topological Equal (teq)*, wenn sie *Causally Equal* sind und darüber hinaus noch am gleichen *Place* in die gleiche Richtung auf einem *path* schauen. Damit sind die *Distincitve States a,b* und *c* aus Abb. 14 und Abb. 15 auch ohne die Exploration zu *d* unterscheidbar, da sie nicht am gleichen *Place* sind.

Metrical Level

Der optionale *Metrical Level* hilft mit mehrdeutigen Konfigurationen umzugehen. Metrische Daten werden an *Places, Paths* und *Regions* geknüpft. Ein *Path* hat einen eindimensionalen Bezugsrahmen, der jedem zugehörigen *Place* eine reelle Zahl zuweist, welche die Position auf dem Pfad bestimmt. Ein *Place* hat einen Bezugsrahmen, der jedem ausgehenden *Path* eine Orientierung in Form eines Winkels zuweist. Mit dieser zusätzlichen metrischen Information ist es möglich Mehrdeutigkeiten wie in Abb. 16 aufzulösen.

Da die durch die Sensorik gewonnenen metrischen Daten, wie bereits erwähnt, nicht fehlerfrei sind, ist es wahrscheinlich, dass der Agent bei einer erneuten Exploration der gleichen Umgebung zu Ergebnissen kommt, die mehr oder weniger große Unterschiede zur ersten Exploration aufweisen. Kuipers sieht vor, diese Unsicherheit in der metrischen Information z.B.

durch *Probability Distribution Functions* abzubilden. Diese können mit jeder Exploration aktualisiert werden, um so die Genauigkeit der metrischen Information zu verbessern oder wengistens die Ungenauigkeit zu erkennen.

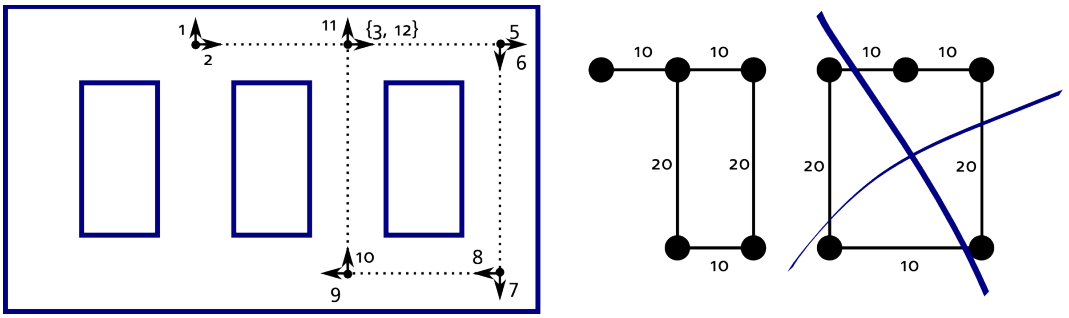

Abb. 16: Für den Fall, dass die *Distincitve States 2, 3* und *12* die gleichen *Views* haben, ist ohne metrische Information nicht zu entscheiden, ob die *Distinctive States 2* und *12* oder *3* und *12* identisch sind. Sind jedoch zuverlässige metrische Daten vorhanden, kann entschieden werden, welche der beiden topologischen Karten die richtige ist. (Grafik nach Remolina et al 2004)

Zu beachten ist, dass im Gegensatz zu *Occupancy Grids* die Nutzung der metrischen Daten optional ist. Zuerst wird die Topologie aufgebaut, und erst anschließend wird sie wenn nötig und möglich durch die metrische Information verfeinert.

3.2.3 Route Graph

Der *Route Graph*[40] weist gewisse Ähnlichkeiten mit der *Topological Map* von Kuipers auf; der Fokus liegt jedoch nicht so stark auf der Robotik. Werner et al. versuchen Erkenntnisse über das Navigationsverhalten von Menschen und biologischen Organismen im Allgemeinen mit Routennavigation aus der Robotik zu vereinen. Ein wichtige Grundlage ist sicherlich die von Siegel und White eingeführte Klassifizierung räumlichen Wissens in *Landmark, Route* und *Survey Knowledge*[41].

Auch wenn basierend auf dem *Route Graph* konkrete Robotiksysteme entwickelt wurden, ist er eher ein grober Rahmen, welcher im Gegensatz zur *SSH* von Kuipers viele Implementationsdetails offen lässt.

Elemente des Modells

Eine Route besteht aus mehreren Routensegmenten, welche wiederum einen Start- und einen Zielort haben, die durch eine Bewegung (*Course*) miteinander verknüpft sind. Diese Bewegung ist dabei nicht immer umkehrbar; d.h. wenn sie den Agenten vom Ort *A* zum Ort *B* bringt, muss nicht gewährleistet sein, dass es möglich ist von *B* nach *A* zu gelangen. Eine Route setzt sich aus mehreren dieser Routensegmente zusammen, wobei der Zielort immer identisch mit dem Startort des folgenden Segments ist.

40 Werner et al. 2000 und Krieg-Brückner 2004
41 siehe Kapitel 1

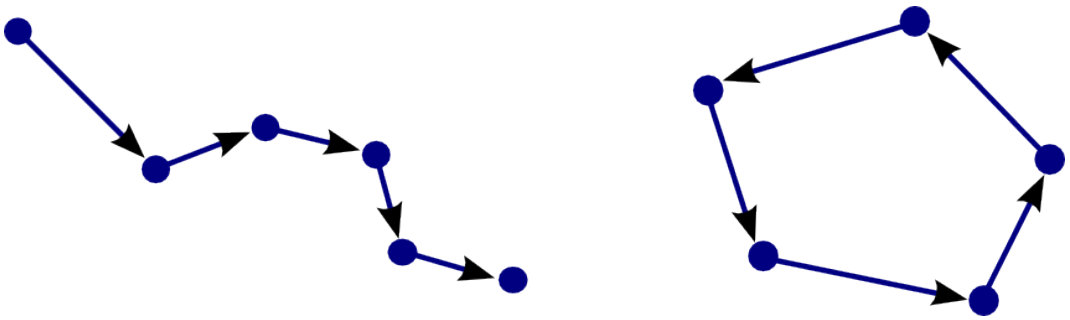

Abb. 17: Eine einfache Route und eine zyklische Route. (Grafik nach Werner et al. 2000)

Neben einer Route gibt es noch Pfade, welche die wirkliche Nutzung einer Route beschreiben. Bei einer zyklischen Route kann ein Pfad mehr Segmente als die Route haben, indem er zwei »Umrundungen« umfasst.

Für den Übergang von einem Routensegment in das nächste muss zumindest der Austritt aus dem einen Segment oder der Eintritt in das nächste Segment klar definiert sein, damit der Agent sich entsprechend orientieren kann. Diese Orientierungen an einem Start- oder Zielort werden in einem lokalen Bezugssystem angegeben. Dieses ist z.B. durch eine *Landmark* im Sichtfeld des Agenten oder durch die Richtung aus der aktuelle Ort erreicht wurde gegeben.

Kombination von mehreren Routen zu einem *Route Graph*

Um aus mehreren Routen einen *Route Graph* zu erhalten, müssen diese kombiniert werden (siehe Abb. 18). Dafür müssen gemeinsame Orte zweier Routen erkannt werden, welches einige Probleme aufwirft.

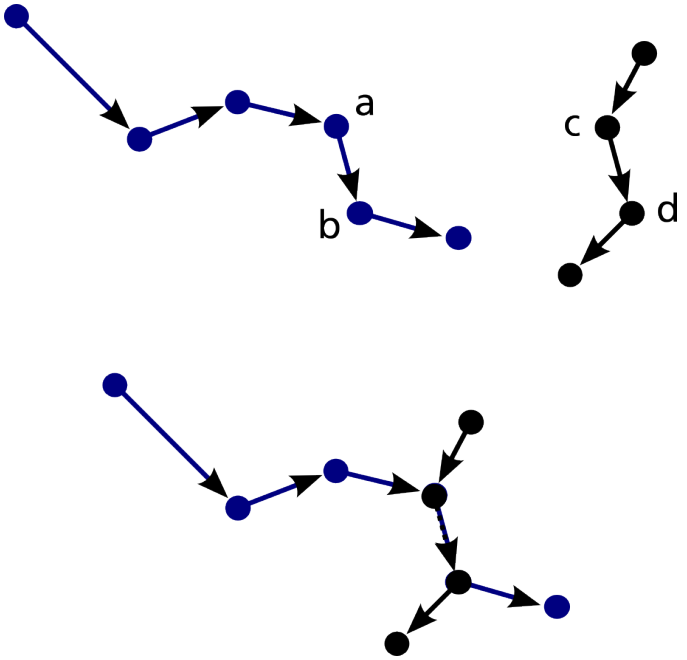

Abb. 18: Kombination von zwei Routen zu einem Route Graphen (Grafik nach Werner et al. 2000)

Es stellt sich die Frage, wie festgestellt wird, ob die Orte *a* und *c* bzw. *b* und *d* identisch sind. Zum einen werden die Orte aus unterschiedlichen Richtungen erreicht, und zum anderen kann es sein, dass in den beiden Routen für die Orte unterschiedliche Bezugssysteme verwendet wurden. Um in dem kombinierten *Route Graph* später navigieren zu können, muss dann bei der Integration zweier Orte auch ein gemeinsames Bezugssystem gefunden werden.

Kombiniert man nun alle Routen und Orte einer Umgebung, so erhält man den O*verview Graph*. Dieser repräsentiert das *Survey Knowledge* und zeichnet sich durch ein gemeinsames, globales Bezugssystem für alle Orte und Routen aus.

3.2.4 View Graph

Der von Schölkopf und Mallot entwickelte *View Graph*[42] wird ausschließlich mit Hilfe einer visuellen Sensorik aufgebaut. Schnappschüsse von der Umgebung bilden die Knoten des Graphen, und die Kanten repräsentieren mögliche Übergänge. Es existieren Systeme, die ein Labyrinth mit Infrarotsensoren[43] explorieren oder in einer »offenen« Umgebung mit einer Panoramakamera[44] navigieren[45].

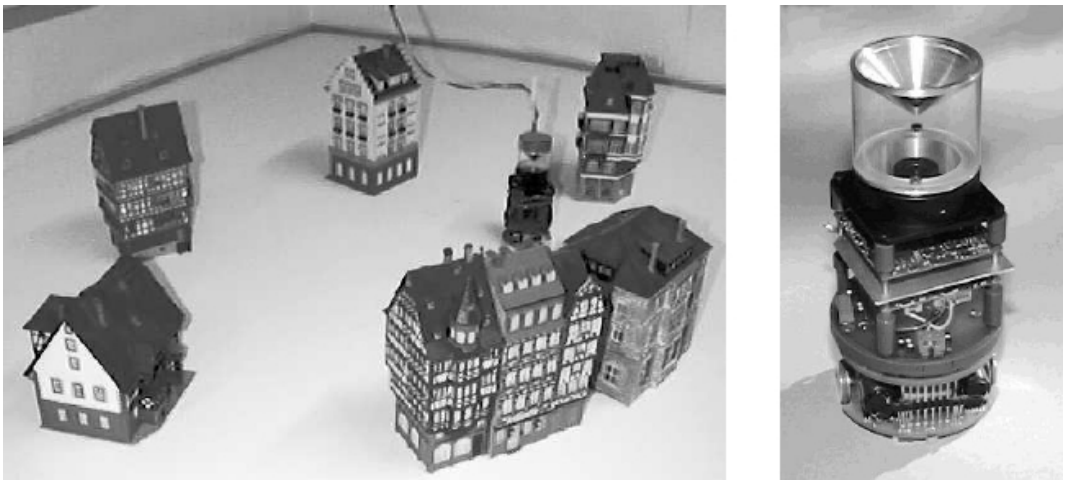

Abb. 19: Roboter mit Kamera, die über einen Panoramaspiegel einen 360°-Rundumblick hat. (Grafik aus Franz et al. 1998)

View Classifier

Ein zentrales Element des vorgestellten Systems ist der *View Classifier*, dessen Aufgabe es ist die visuellen Aufnahmen der Umgebung (*Views*) an kritischen Orten möglichst zuverlässig wiederzuerkennen bzw. zu unterscheiden. Diese Komponente spielt eine entscheidende Rolle, da die ermittelten *Views* in der Repräsentation die Knoten – und damit die Grundbausteine – des Graphen darstellen.

Der verwendete Sensor, eine auf einen Panoramaspiegel gerichtete Kamera, macht 360°-Aufnahmen der Umgebung (siehe Abb. 19 (rechts) und Abb. 20). Die aufgenommen Schnappschüsse der Umgebung sind eindimensional und enthalten Grauwerte.

42 Schölkopf, Mallot 1995
43 Mallot et al. 1995
44 Franz et al. 1998, siehe Abb. 10
45 Da der initiale Artikel über den *View Graphen* (Schölkopf, Mallot 1995) nicht zur Verfügung steht, basiert die folgende Beschreibung auf einem späteren von Franz, Schölkopf und Mallot entwickelten System, welches die Umgebung anhand einer Panoramakamera exploriert (Franz et al. 1998).

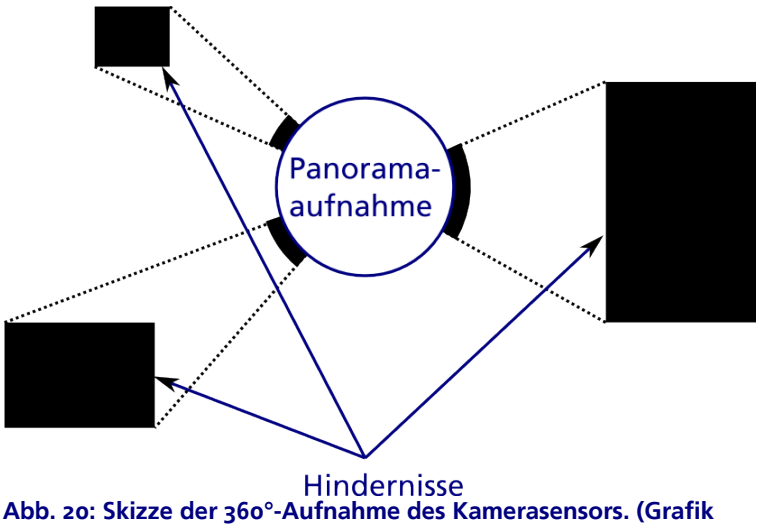

Abb. 20: Skizze der 360°-Aufnahme des Kamerasensors. (Grafik nach Franz et al. 1998)

Navigation (View-Based Homing)

Die Navigation zwischen zwei Knoten des *View Graph* wird allein über eine visuelle Zielsuche (*visual homing*) bewerkstelligt, d.h. allein über die Sensordaten wird das System zu einem naheliegenden Ort gesteuert, dessen *View* bereits bekannt ist[46]. Die Fahrtrichtung wird aus der Differenz des aktuellen *Views* und des Ziel-*Views* abgeleitet. Für die Berechnung dieser Richtung setzen Franz et al. jedoch voraus, dass alle (in den visuellen Schnappschüssen aufgenommenen) Hindernisse in etwa den gleichen Abstand vom Roboter haben.

Vorraussetzung für diese Navigationsstrategie ist, dass angrenzende *Views* keine allzu große Entfernung voneinander haben. Das vorgestellte System hat bei einer Entfernung von 15 cm eine Zuverlässigkeit von 90%, welche bei 20cm jedoch bereits auf 50% sinkt.

Exploration

Die eben beschriebene Navigation ermöglicht noch keine Aufnahme von neuen *Views*, sondern lediglich das Navigieren zwischen bekannten *Views*, welches ggf. neue Kanten im Graphen zu Tage bringt.

Der Explorationsalgorithmus (siehe unten) wählt die nächste Explorationsrichtung; dies kann bedeuten, dass ein bereits bekannter *View* angesteuert wird, weil vom aktuellen *View* bereits genügend Kanten ausgehen bzw. alle Richtungen mehr oder weniger abgedeckt sind oder dass der Roboter eine unerforschte Richtung ansteuert. Für Letztere wird die Richtung zu den benachbarten *Views* abgeschätzt und die Mitte des größten offenen Winkels Φ zwischen zwei *Views* gewählt (siehe Abb. 21).

46 Für eine Übersicht zu *View-Based Homing* Strategien siehe Röfer 1998, S. 131 ff

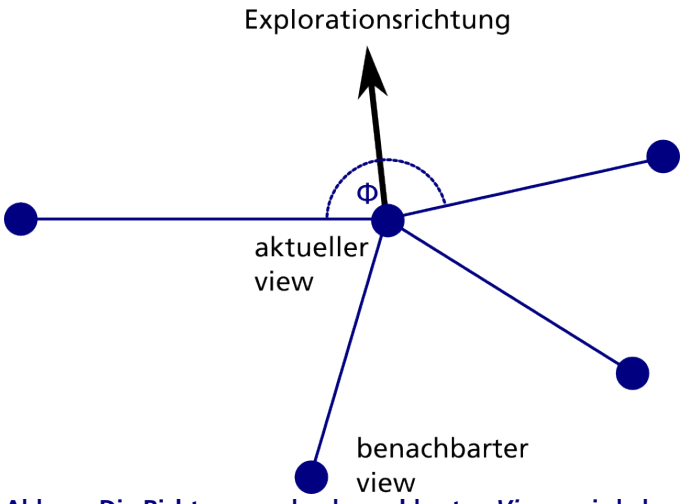

Abb. 21: Die Richtung zu den benachbarten *Views* wird abgeschätzt und die Mitte des größten offenen Winkels Φ wird als Richtung für den nächsten Explorationsschritt gewählt. (Grafik nach Franz et al. 1998)

Der Explorationsalgorithmus im Pseudocode:

```
Wähle nächste Explorationsrichtung:
n := Anzahl der Kanten (am aktuellen Knoten=View)
FOR alle verbundenen views DO
    Berechne Winkel zum verbundenen view
Φ := größter offener Winkel
γ := Vektor, der zur Mitte von Φ zeigt
IF n > max edges OR Φ min angle THEN
    Explorationsrichtung := fixed angle
ELSE
    Explorationsrichtung := γ
                                    (Pseudocode nach Franz et al. 1998)
```

Vergleich zur SSH

Wie im Kapitel 3.2.2 (S. 26) schon angedeutet, ist auf dem *Causal Level* der *Spatial Semantic Hierarchy* eine dem *View Graph* sehr ähnliche Repräsentation vorhanden. Der Hauptunterschied liegt in der verwendeten Sensorik und der Explorationsstrategie.

3.3 Sensomotorische Repräsentation

Nun stellt sich die Frage, inwiefern die vorgestellten Raumrepräsentationen der im ersten Kapitel formulierten Anforderung genügen, d.h. ob sie in irgendeiner Weise auch die bei der Exploration verwendeten motorischen Aktionen repräsentiert.

Das *Occupancy Grid* ist sozusagen das »Negativbeispiel«, da es allein auf Sensorik basiert und die Motorik nicht vertreten ist. Damit entspricht es in vielen Punkten der klassischen Vorstellung einer *Cognitive Map* nach Tolman.

In der *Spatial Semantic Hierarchy* von Kuipers stellt sich dies schon anders dar. Seine Kernfrage beim Aufbau einer Repräsentation ist, *wie* man von einem Ort zu einem weiteren navigiert:

[...] the irreducible core of the cognitive spatial description is the »Map in the Head«. This essay argues that the opposite is true: Knowledge of where things are is built on a foundation of knowledge of how to navigate from one place to another. (Kuipers 1982, S. 5)

Die in der *SSH* formulierten Repräsentation fußt auf eindeutigen Zuständen (*Distinctive States*), in denen sich der Agent befinden kann, welche mittels eines oder mehrerer *Control Laws* verknüpft sind. Und diese *Control Laws* sind über eine direkte Relation von Sensorik und Motorik definiert, wie z.B. »fahre so lange wie möglich diese Wand im Abstand von einem Meter entlang«[47]. Dieses Verhalten ist im *Control & Sensory Level,* der untersten Stufe der hierarchisch aufgebauten Repräsentation von Kuipers, definiert, und ist damit die Basis für alle weiteren Operationen.

Der *Route Graph* von Werner et al. ist ein etwas abstrakteres Modell, da es viele Implementierungsdetails offen lässt und vor allem Möglichkeit bieten möchte, den Austausch zwischen Disziplinen wie Psychologie, Kognitionswissenschaften, Künstlicher Intelligenz, Robotik etc. zu fördern. Die Definition eines Routensegments ähnelt in vielerlei Hinsicht den *View-Action-View* Sequenzen aus dem *Causal Level* von Kuipers *SSH*. Wie man zu selbigen gelangt ist jedoch nicht genau ausgeführt; es sind unter dem Begriff *guidance* einige Ideen angerissen, welche z.T. in die Richtung der *Control Laws* von Kuipers und dem *View Graph* gehen. Dem Modell ist anzusehen, dass eine motorische Komponente vorgesehen ist, nur steht diese nicht wirklich im Fokus und ist demzufolge auch nicht detailliert beschrieben.

Der von Schölkopf und Mallot entwickelte *View Graph* zeigt große Ähnlichkeiten zum *Sensory, Control* und *Causal Level* der *SSH* von Kuipers. Der deutlichste Unterschied liegt in der anderen Sensorik. Die Entsprechung zu Kuipers *Control Laws* ist das *View-Based Homing*[48], welches einen Agenten von einem Ort zum nächsten bringt.

Was fehlt noch?

Die hier vorgestellten Systeme haben gemein, dass sie für spezielle Umgebungen konzipiert sind; das gilt besonders für ihre sensomotorischen Eigenschaften. Kuipers geht von Innenraumumgebungen aus, für die sich in Kombination mit Reichweitensensoren relativ einfach *Control Laws* definieren lassen. Das auf dem *View Graph* basierende System ist hingegen ausschließlich für offene Umgebungen konzipiert, da ansonsten die *Views* nicht eindeutig genug sind, welches das *View-Based Homing* unmöglich macht.

Der vielversprechende Ansatz von Kuipers ist in dem Sinne beschränkt, dass es eine sehr begrenzte Menge an Aktionen (*Control Laws*) mit vorgegebenen Abbruchkriterien gibt, welche – so weit aus den betrachteten Artikeln hervorgeht – ausschließlich in einer Kombination aus Innenraumumgebungen und Reichweitensensoren funktionieren. Es ist evtl. interessant *Control Laws* für

47 siehe Kapitel 3.2.2, S. 25
48 Für eine Überblick zu *View-Based Homing* siehe Röfer 1998.

eine andere Sensorik zu entwickeln und zu überprüfen inwieweit das Modell noch trägt, wenn es keine geraden Korridore oder Wände gibt, an denen ein Roboter entlang fahren kann. Ein am Menschen oder an Ratten[49] orientiertes System könnte z.B. auf eine haptische, propriozeptive und (nicht omnidirektionale) visuelle Sensorik aufbauen.

Es stellt sich die Frage *wie* in biologischen Systemen (senso-)motorische Abläufe kodiert und verarbeitet werden. Ohne dieses Wissen ist es schwer eine biologisch motivierte Raumrepräsentation aufzubauen, welche den motorischen Aspekt berücksichtigt.

49 Die Raumrepräsentation auf neuronaler Ebene ist besonders bei Ratten gut erforscht.

4 Sensorimotor Explorer (SMX)

> *The first 90% of the code accounts for the first 90% of the development time. The remaining 10% of the code accounts for the other 90% of the development time.*
>
> *Tom Cargill*

In diesem Kapitel wird das Agentensystem *Sensorimotor Explorer (SMX)* beschrieben, welches basierend auf den Ergebnissen der letzten Kapitel Orientierungsaufgaben in einer VR-Umgebung löst. Der *SMX* dient dazu, eine sensomotorische Raumrepräsentation und Explorationsstrategie zu entwickeln und zu testen.

Zunächst wird das biologisch motivierte Systemdesign grob umrissen, und anschließend werden die einzelnen Komponenten in ihrer Funktion und Implementierung genauer erläutert. Abschließend wird noch eine Bewertung des Systems und ein Ausblick zu vielversprechenden Schritten der Weiterentwicklung gegeben.

4.1 Systemdesign

Das Design des *SMX* ist an biologische Systeme angelehnt, was besonders in der verwendeten Raumrepräsentation, Sensorik und Inferenzstrategie deutlich wird. Die hier beschriebene erste Implementierung befasst sich mit dem Problem der Lokalisierung in einer unbekannten Umgebung. Weitere Aufgaben aus dem Bereich der Orientierung und Navigation werden in dieser Arbeit lediglich angerissen; ihre Lösung ist jedoch keineswegs ausgeklammert, d.h. es handelt sich um keine konzeptuelle und damit dauerhafte Beschränkung des Systems.

4.1.1 Biologische Motivation

Wie im zweiten Kapitel dargelegt spricht einiges dafür, dass bei Menschen und Tieren Sensorik und Motorik im Gehirn nicht strikt getrennt werden und letztere in der Raumrepräsentation vertreten ist. Diese Eigenschaft spiegelt sich auch in der Raumrepräsentation des *SMX* wieder, indem sie auf *sensomotorischen* Grundbausteinen basiert.

Bzgl. der Sensorik wird auch ein anderer Weg eingeschlagen. Aus der Robotik stammende Explorationssysteme nutzen neben Kontaktsensoren und odometrischen Daten hauptsächlich Abstandssensoren wie Sonar und Laserscanner, um eine Raumrepräsentation aufzubauen. Diese beim Menschen und den meisten Tieren nicht vorhandenen Abstandssensoren werden im *Sensorimotor Explorer* nicht verwendet. Stattdessen fußt das System auf einer an die menschliche Wahrnehmung anglehnte visuelle Sensorik.

Neben dieser *bottom-up* Verarbeitung der Sensordaten besitzt der *SMX* noch eine *top-down* Strategie, welche erlernte Erwartungen bzgl. der Umwelt integriert bzw. höhere kognitive Prozesse simuliert. Diese Komponente, welche mit unsicheren Wissen umgeht, bedient sich der *Theory of Evidence* von Dempster und Shafer[50].

4.1.2 Architektur

Der Aufbau des *SMX* lässt sich in mehrere Komponenten unterteilen, die sich um das *SMX* Kernmodul gruppieren (siehe). Unter diesen Komponenten ist die VR-Umgebung mit ihrer sensomotorischen Schnittstelle zur Steuerung des Agenten, ein sakkadisches Bildanalysesystem zur Verarbeitung der

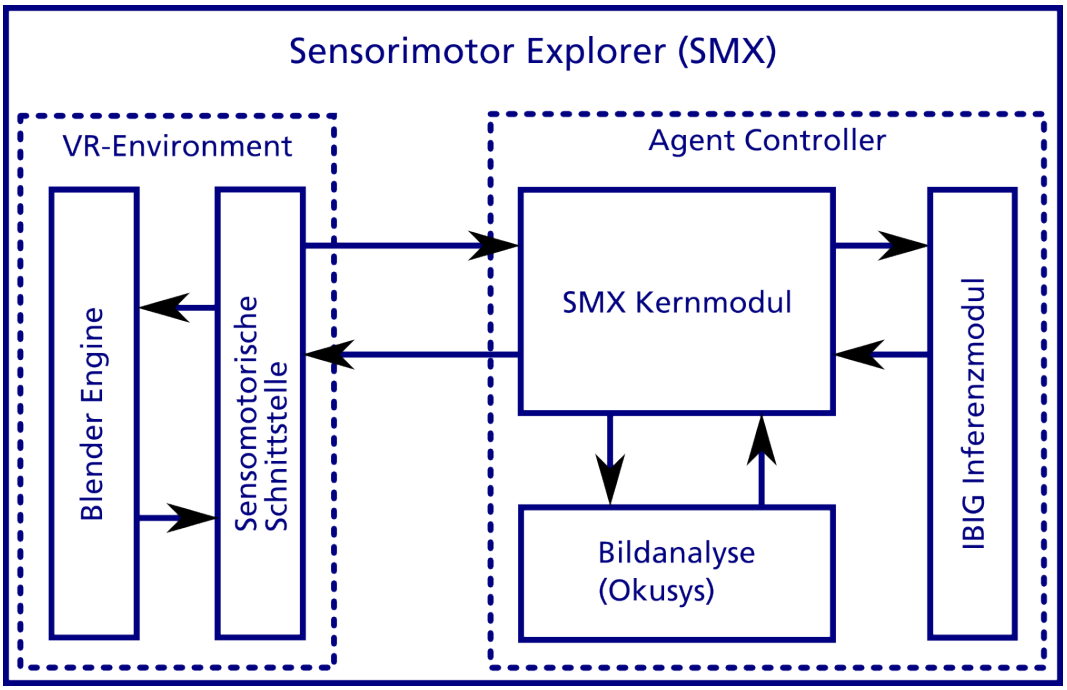

Abb. 22: Aufbau des Sensorimotor Explorers (SMX)

visuellen Information aus der VR-Umgebung, sowie ein Inferenzmodul, welches den jeweils nächsten Explorationsschritt in der VR-Umgebung bestimmt. Das Kernmodul ist die Schaltstelle dieser Komponenten die den Porgrammablauf regelt (siehe Abb. 23):

(1) Abfrage der Sensorik des Agenten aus der VR-Umgebung (visueller Schnapschuss) und anschließende Analyse dieser Daten durch das Szenenanalysemodul.

(2) Übermittlung der Ergebnisse des Bildanalysemoduls an das Inferenzmodul.

(3) Berechnung der aktuellen Hypothese über die Umgebung anhand der erhaltenen Daten.

50 Shafer 1976

(a) Eine Hypothese wird ausreichend unterstützt, so dass der Lokalisationsprozesses als abgeschlossen angesehen wird.

(b) Keine Hypothese ist ausreichend unterstützt, so dass eine weitere Exploration der Umgebung nötig ist.

(4) Ermittlung der interessanten Orte (*Views*) in unmittelbarer Umgebung an das Inferenzmodul. Anschließende Berechnung des nächsten Explorationsschrittes aufgrund der übermittelten Orte und der aktuellen Hypothesen.

(5) Ausführung des berechneten Explorationsschrittes in der VR-Umgebung.

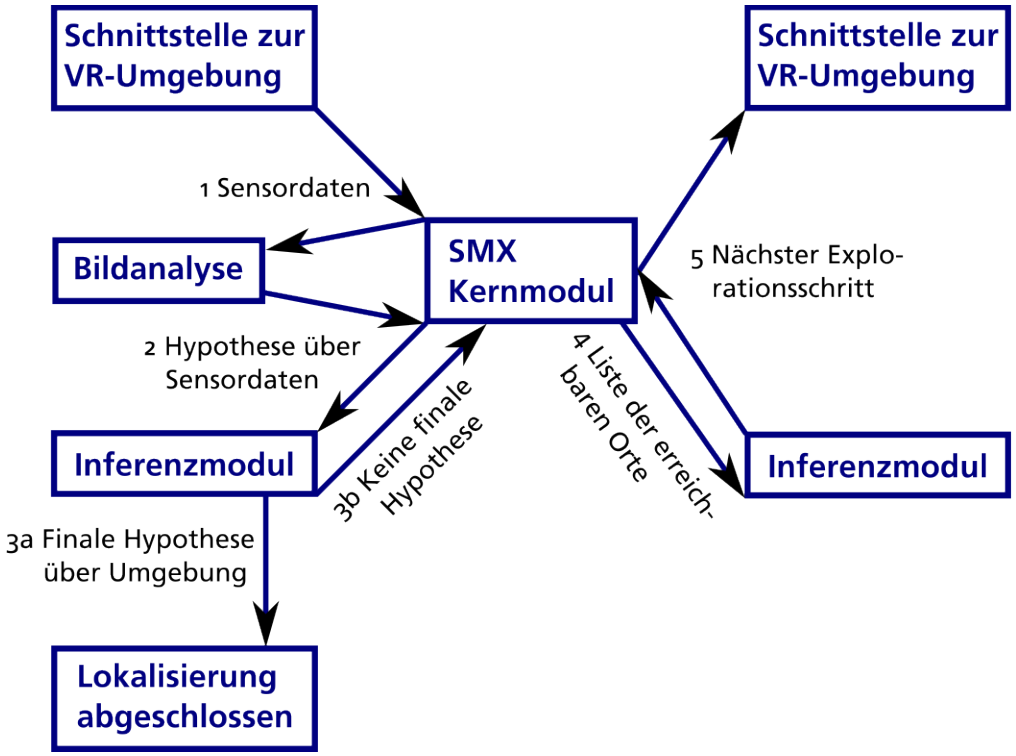

Abb. 23: Ablauf, wie der *SMX* eine Hypothese aufstellt, in welcher Umgebung er sich befindet (Lokalisation).

4.2 Sensomotorische Raumrepräsentation

Die Raumrepräsentation basiert auf sensomotorischen Eigenschaften, welche aus zwei durch eine Aktion verbundene *Views* bestehen: *[View, Action, View]*. Diese Eigenschaften sind die Grundbausteine der sensomotorischen Raumrepräsentation und entsprechen gewissermaßen einem Explorationsschritt in der VR-Umgebung.

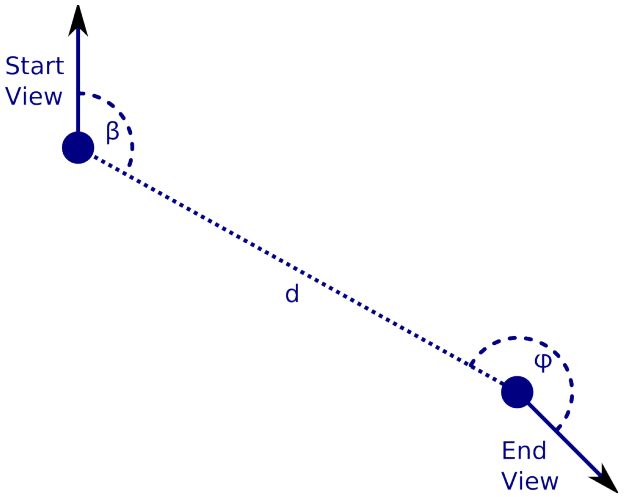

Abb. 24: Schema eines sensomotorischen Grundbausteins der Raumrepräsentation.

Ein *View* repräsentiert den sensorischen Anteil und ist das Ergebnis der Bildanalyse von den »visuellen Schnappschüssen« des Agenten in der VR-Umgebung. Damit ist ein *View* über den *Viewpoint* (der Ort und die Orientierung) des Agenten in der VR-Umgebung definiert, an dem der Schnappschuss gemacht wurde. Die Aktion, welche der Agent in der VR-Umgebung zwischen zwei *Views* ausgeführt hat und welche diese sensorischen Stimuli so miteinander verknüpft, repräsentiert den motorischen Anteil.

Die Aktion, welche zwei *Views* miteinander verknüpft, besteht momentan aus zwei Rotationen und einer Positionsänderung. Die erste Rotation β rotiert den Agenten, so dass er zum Ort des *End Views* blickt und die folgende geradlinige Bewegung mit der Länge d den Agenten zu diesem Ort bringt. Die zweite Rotation φ richtet den Agenten nun noch zu der Orientierung des zweiten *Views* aus (siehe Abb. 24).

Views

Aus neurophysiologischer Sicht lassen sich die *Views* durchaus mit den im zweiten Kapitel beschriebenen *Place Cells* rechtfertigen. Die Aktivität dieser Zellen ist relativ fest an visuelle Markierungen gebunden. Diese visuellen Markierungen, in manchen Zusammenhängen auch als *Landmarks* bezeichnet, sind mehr oder weniger markante Orientierungspunkte wie Weggabelungen oder aus der Umgebung hervorstechende Merkmale. Ein konzeptueller Unterschied der *Place Cells* zu den *Views* (bzw. deren *Viewpoints*) ist ihre Orientierungsinvarianz. Das heißt, sie integrieren alle *Viewpoints*, die sich le-

diglich durch eine Rotation unterscheiden. Dieser Schritt ist in der aktuellen Repräsentation noch nicht vorgesehen.

Abb. 25: »Visueller Schnappschuss« des Agenten in einer VR-Umgebung

Die Frage, wie die *Viewpoints* in einer VR-Umgebung bestimmt werden, ist im *SMX* noch nicht berücksichtigt. Die Positionen, aus denen der Agent visuelle Schnappschüsse machen kann, sind vorgegeben; allerdings hat das System immer nur Zugriff auf die *Viewpoints*, welche sich in seiner unmittelbaren Umgebung befinden, und der *View,* der sich hinter diesen *Viewpoints* verbirgt, ist auch nicht bekannt. Das System weiß durch das zusätzliche Wissen der *Viewpoints* noch nicht, ob sich z.B. rechts hinter einer Weggabelung oder im Nebenzimmer ein *Viewpoint* befindet. Und wenn bekannt ist das drei Meter vor dem Agenten ein *Viewpoint* ist, kann das System nicht ermitteln, welcher *View* daran geknüpft ist, ohne einen Explorationsschritt zu diesem *View* zu machen.

Sensomotorische Eigenschaften in der Exploration

Wie eingangs erwähnt, entspricht eine sensomotorische Eigenschaft *[View, Action, View]* in etwa einem Explorationsschritt. In Abb. 26 ist die Exploration eines Raumes inklusive der *Views* und Aktionen skizziert. Der *SMX* würde bei dieser Exploration folgende sensorischen Eigenschaften sammeln: *[View1, Action1, View2], [View2, Action2, View3], [View3, Action3, View4]*

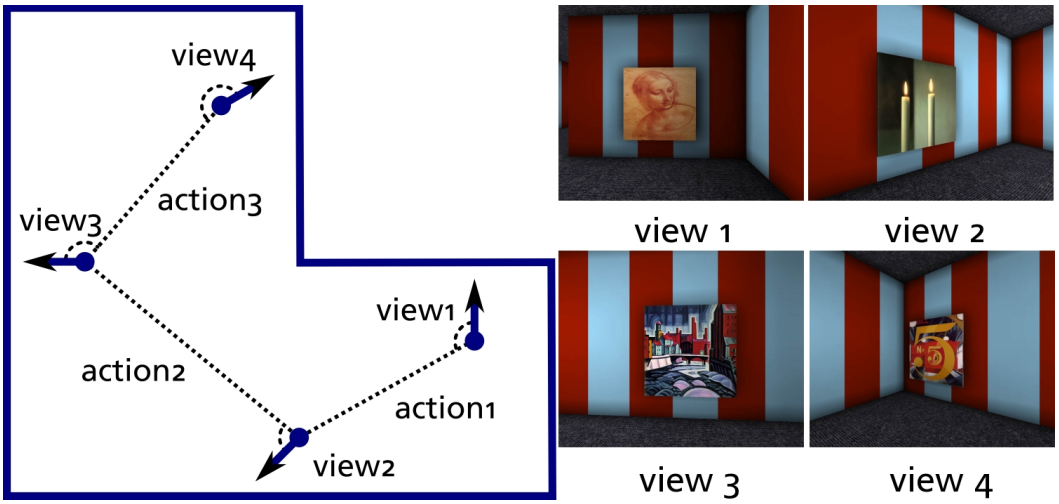

Abb. 26: Exploration eines Raumes mit *Views* und Aktionen.

Die sensomotorischen Tripel lassen sich durchaus mit Routensegmenten aus dem *Route Graph*, der Repräsentation im *Causal Level* in der *SSH* oder dem *View Graph* vergleichen. Eine eingehendere Bewertung wird im Kapitel 4.8.1 (S. 80) gegeben.

4.3 Inferenzstrategie

Wird der Agent in einer ihm unbekannten VR-Umgebung »ausgesetzt« benötigt er eine Strategie, nach der er seine Umwelt exploriert, um schließlich eine Vermutung anstellen zu können, wo er sich befindet.

Die im *SMX* verwendete Inferenzstrategie zur Bestimmung der Explorationsschritte basiert auf der *Theory of Evidence* von Dempster und Shafer, welche eine Möglichkeit bietet, mit unsicherem Wissen umzugehen. Darauf aufbauend berechnet der nach Informationszuwachs optimierende Algorithmus *Inference by Information Gain (IBIG)* Explorationsschritte. Die vorgestellte Architektur ist an das im *SMX* verwendete Bildanalysesystem von Schill et al.[51] angelehnt, welches in gewisser Weise eine räumliche Klassifikation auf anderer Granularitätsstufe realisiert.

4.3.1 Dempster Shafer

Verrauschte Sensordaten und unvollständiges Wissen über die Umgebung erfordern eine Möglichkeit, mit dieser Form der Unsicherheit umgehen zu können. Der *Sensorimotor Explorer* bedient sich der *Theory of Evidence* von Dempster und Shafer. Im Vergleich zur bekannteren *Bayes'schen* Theorie[52] bietet sie Vorteile in der Ausdrucksfähigkeit von »Nichtwissen«, im Vergeben von Evidenz an *Mengen* von Propositionen sowie der Möglichkeit, Evidenz ohne fundierte statistische Grundlage (durch sog. Expertenwissen) verteilen zu können[53].

Frame of Discernment & Basic Probability Assignments / Numbers

Ausgangspunkt der *Theory of Evidence* ist die Menge der Propositionen bzw. (Einzel-)Hypothesen[54] Θ, auch *Frame of Discernment* genannt, welche eine Domäne möglichst komplett beschreiben. Ein Beispiel für eine Hypothese im *SMX* ist wäre: »Der Agent steht momentan vor einer Tür«, oder »Der Agent befindet sich in einer Küche«. Neben dem *Frame of Discernment* sind die sog. *Basic Probability Assignments (bpa)* die Grundbausteine der Theorie. Sie unterstützen eine oder mehrere Hypothesen zu einem gewissen Grad, indem sie beliebigen Teilmengen von Θ, d.h. Elementen aus der Potenzmenge $P(\Theta)$, einen Wert zwischen 0 (keine Unterstützung) und 1 (volle Unterstützung) zuweisen; dieser Wert wird auch mit *Basic Probability Number (bpn)* bezeichnet. Alle *Basic Probability Assignments* müssen folgende Bedingungen erfüllen:

$$(I) \quad m : 2^{\Theta} \rightarrow [0,1]$$
$$(II) \quad m(\emptyset) = 0$$
$$(III) \quad \sum_{A \in P(\Theta)} m(A) = 1$$

51 Schill 2001
52 Jensen 1996
53 Für einen detaillierten Vergleich siehe Shafer 1976.
54 Die Begriffe *Proposition* und *Einzelhypothese* werden hier äquivalent verwendet. Eine *Hypothese* kann sowohl eine einzelne Proposition als auch eine Menge von Propositionen repräsentieren.

Das *Basic Probability Assignment* $m(\{Büro, Küche\})=0.3$ drückt z.B. die Evidenz aus, dass der Agent sich mit 30-prozentiger Sicherheit in einem Büro oder einer Küche aufhält[55].

Das *bpa* $m(\Theta)=0.2$ stellt einen Sonderfall dar und drückt gewissermaßen 20-prozentige Unwissenheit aus. Alle Evidenz, die nicht einzelne Hypothesen unterstüzt, d.h. $1-\sum_{A\in P(\Theta)\setminus\Theta} m(A)$, wird automatisch Θ zugeschlagen.

Das *Basic Probability Assignment* ist jedoch nicht mit dem aktuellen »Glauben« in eine Hypothese zu verwechseln. Diese ergibt sich, zusätzlich zu der *bpn* für die der Hypothese zugehörige Teilmenge, aus den *bpn's* aller Untermengen dieser Teilmenge. Dies wird über die sog. *Belief Function* ausgedrückt:

$$Bel(A)=\sum_{B\subseteq A} m(B)$$

Dempster's Rule of Combination

Gibt es nun mehrere Evidenzen, ist es wünschenswert, die daran geknüpften *bpa's* kombinieren zu können. Definiert ist diese Kombination über *Dempster's Rule of Combination*. Zwei *bpa's,* welche auf dem selben *Frame of Discernment* Θ operieren, werden kombiniert, indem die *Basic Probability Numbers*, welche die erste Funktion den Teilmengen zuweist, mit allen *Basic Probability Numbers* der zweiten Funktion multipliziert werden. Das Ergebnis dieser Multiplikationen ist die *Basic Probability Number*, welche das neue »kombinierte« *bpa* der Schnittmenge aus den beiden jeweils betrachteten Teilmengen zuweist.

$$m_{1,2}(A)=m_1\circ m_2=\sum_{\substack{i,j\\ B_i\cap C_i=A}} m_1(B_i)m_2(C_i)$$

Die Abb. 27 ist eine Veranschaulichung dieser Operation: Die beiden Achsen haben einen Wertebereich von 0 bis 1 und bilden die Verteilung der Evidenz durch das *bpa* m_1 (horizontal) und m_2 (vertikal) ab; die Länge eines Abschnittes entspricht der *Basic Probabilty Number* für die angegebene Teilmenge. Die durch Achsensegmente aufgespannten Flächen entsprechen den *Basic Probabilty Numbers* der neuen Funktion $m_{1,2}$ jeweils für die Schnittmenge der Teilmengen an den Achsen. Das markierte Feld steht z.B. für die Kombination $m_1(B_k)\circ m_2(C_j)=m_{1,2}(C_j\cap B_k)$.

55 Im *SMX* werden diese Funktionen über einen Prozess des überwachten Lernens aufgebaut.

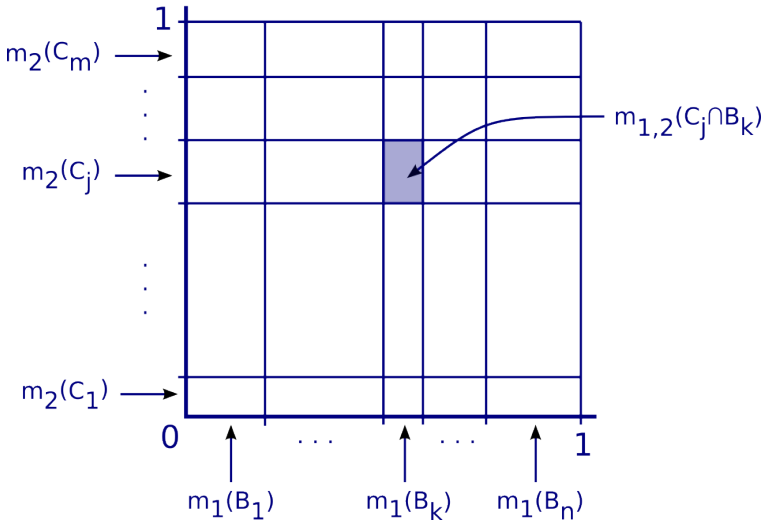

Abb. 27: Verknüpfung zweier *Basic Probability Assignments* (m1 und m2) nach *Dempster's Rule of Combination*.

Normalisierung

Berücksichtigt werden müssen nun noch die Produkte, die der leeren Menge zugewiesen werden. Sie treten auf, wenn zwei kombinierte Teilmengen eine leere Schnittmenge haben. Da diese Evidenz κ sozusagen »verloren« geht, wird die zweite Bedingung für *Basic Probability Assignments* (II) nicht mehr erfüllt. Eine Normalisierung, bei der die *Basic Probability Numbers* durch $1-\kappa$ geteilt werden, erzeugt wieder ein gültiges *bpa*.

$$m_{1,2}(A) = m_1 \circ m_2 = \frac{1}{1-\kappa} \sum_{\substack{i,j \\ B_i \cap C_i = A}} m_1(B_i)\, m_2(C_i)$$

$$\kappa = \sum_{\substack{i,j \\ B_i \cap C_i = \emptyset}} m_1(B_i)\, m_2(C_i)$$

Simple Support Functions

Eine Sonderform der *bpa's* sind die *Simple Support Functions*. Diese zeichnen sich dadurch aus, dass sie lediglich *eine* Hypothese, d.h. *ein* Element aus 2^{Θ}, unterstützen. Ein Vorteil dieser Funktionen ist ihre einfache Kombinationsmöglichkeit. Zwei *Simple Support Functions*, welche die gleiche Hypothese unterstützen, lassen sich durch eine Vereinfachung von *Dempster's Rule of Combination,* auch als *Bernoulli's Rule of Combination* bekannt, kombinieren:

$$m_1(A) = s_1$$
$$m_2(A) = s_2$$
$$m_{1,2}(A) = 1 - (1 - s_1)(1 - s_2)$$

4.3.2 Hierarchischer Hypothesenraum

Ein entscheidender Nachteil der *Theory of Evidence* ist ihre Komplexität bzgl. der Berechnung, welche exponentiell zu der Anzahl der Propositionen bzw.

(Einzel-)Hypothesen Θ wächst. Die Ursache liegt in einem der großen Vorteile dieser Theorie, indem sie erlaubt, nicht nur an Einzelhypothesen, sondern an beliebigen Teilmengen von Θ »Evidenz zu verteilen«. Das heißt, es wird auf der Potenzmenge $P(\Theta)$ operiert, welche nun exponentiell mit den Elmenten in Θ wächst (2^{Θ}). Einen möglichen Ausweg aus diesem Problem haben Gordon und Shortliffe gefunden, indem sie nicht alle Teilmengen von Θ berücksichtigen, ohne das Modell jedoch grundsätzlich abzuschwächen[56].

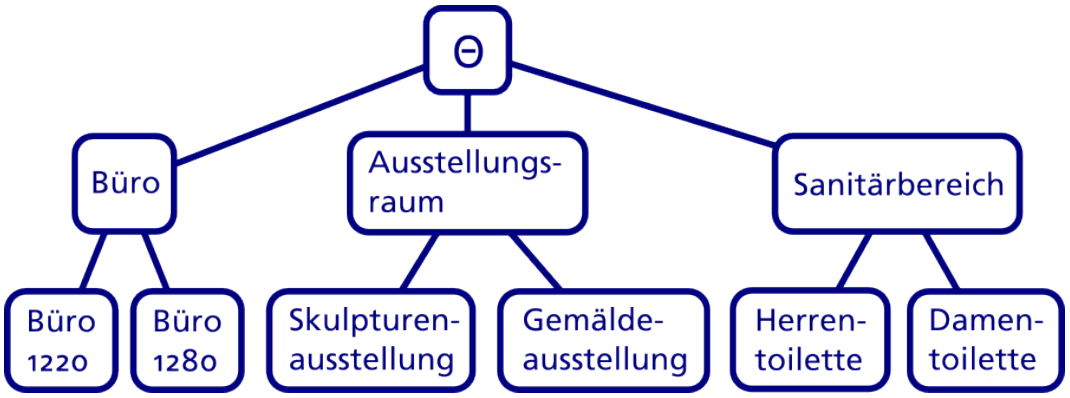

Abb. 28: Beispielhafte Hypothesenhierarchie für Räume

Ihr Ausgangspunkt ist die Annahme, dass nicht alle Teilmengen von Θ eine sinnvolle Hypothese darstellen und der »Hypothesenraum« sich hierarchisch organisieren lässt. Sie gehen davon aus, dass zwei Einzelhypothesen, welche keine Gemeinsamkeiten haben bzw. von ihrer Semantik her keiner gemeinsamen Kategorie zugeordnet werden können, auch nicht durch eine Teilmenge repräsentiert werden müssen. Stattdessen wird eine Hierarchie aus sinnvollen Teilmengen mit Einzelhypothesen an den Blättern aufgebaut. Zur Veranschaulichung ist in Abb. 28 eine Hypothesenhierarchie für Räume (in denen sich der Agent befinden könnte) abgebildet. Die Blätter dieser Hierarchie (»Büro 1220«, »Büro 1280«, etc.) sind Einzelhypothesen und die Knoten über den Blättern repräsentieren die Teilmengen aus den darunterliegenden Knoten; der Knoten »Büro« steht somit für die Teilmenge »{Büro 1220; Büro 1280}«. Durch diese hierarchische Aufteilung des Hypothesenraumes, fallen z.B. Teilmengen wie {»Büro 1220«; »Damentoilette«} weg, da sie nicht als sinnvolle (semantische) Kategorie angesehen werden. Es wird angenommen, dass es kein *Basic Probability Assignment* gibt, welches gerade dieser Teilmenge Evidenz zuweist, da ein Büro und eine Toilette meist eher geringe Gemeinsamkeiten aufweisen. Gewonnen wird dadurch ein wesentlich kleinerer Hypothesenraum, eine Untermenge von $P(\Theta)$, der im Folgenden durch die Menge T bestimmt ist. Die einzige Bedingung an diese Menge ist, dass ihre Elemente eine strikte Hierarchie bilden.

56 Shortliffe, Gordon 1985

Diese hierarchische Organisation des Hypothesenraumes lässt sich durchaus aus biologischer Sicht begründen, wie psychologische Experimente von Collins und Quillian zeigen[57].

Eine Frage lässt diese Form der Vereinfachung jedoch noch offen. Bei der Kombination von *Basic Porbabilty Assignments* treten bei der Schnittmengenbildung u.U. neue Teilmengen auf, sofern die Funktionen nicht genau die selben Teilmengen unterstützen. Und diese Schnittmengen sind nun nicht zwangsläufig nicht in der Hierarchie vertreten, so dass ein Weg gefunden werden muss, die an diese Menge geknüpfte Evidenz zu verteilen.

Evidenzberechnung in der Hierarchie

Gordon und Shortliffe gehen in drei Schritten vor, von denen hier nur die ersten beiden, für bestätigende Evidenz relevanten, Schritte beschrieben werden:

Für jede Hypothese, d.h. für jede Teilmenge $X_i \in T$, werden die Evidenzen miteinander kombiniert. Hierfür werden die vorhanden *bpa's* zu *Simple Support Functions* »aufgeteilt«, sofern dies nicht bereits gewährleistet ist[58]. Diese Funktionen lassen sich mittels der Regel von *Bernoulli* einfach kombinieren, so dass man schließlich für jede Teilmenge X_i ein *bpa* m_{X_i} erhält.

Gewünscht ist jedoch *ein* »kombiniertes« *Basic Probability Assignment* m_T, welches *allen* Hypothesen Evidenz zuweist, d.h. die *bpa's* m_{X_i} müssen zu einer Funktion m_T kombiniert werden:

Am einfachsten zu berechnen ist die *Basic Probability Number*, welche m_T Θ zuweist[59]. Dafür müssen lediglich alle »*Theta-Anteile*«[60] der eben konstruierten *bpa's* m_{X_i} verknüpft werden:

$$m_T(\Theta) = \prod_{x \in T} m_X(\Theta)$$

Bei der Berechnung der *bpn's*, welche m_T den restlichen Hypothesen $X_i \in T \neq \Theta$ zuweist, kommt die geforderte hierarchische Struktur der Hypothesen zum Tragen. Die relative »Lage« der Hypothesen im Baum bzw. ihre Schnittmengen bestimmen die gegenseitige Beeinflussung durch die eben berechneten *bpa's* für Einzelhypothesen (m_{X_i}). Es werden grundsätzlich drei Fälle Unterschieden; die Evidenz der Hypothese *A* ergibt sich wie folgt aus der Summe der m_{X_i} :

(a) $m_A(A)$

(b) $\prod_{\substack{X \in T \\ X \not\supseteq A}} m_X(\Theta)$

57 zit. n. Anderson 1988, S. 118 ff

58 Eine Strategie, wie diese »Aufteilung« möglich ist, wird im folgenden Kapitel beschrieben.

59 $m_T(\Theta)$ ist die Evidenz, die keiner der Hypothesen zugewiesen wird und kann als Gradmesser für die Unwissenheit gesehen werden.

60 Eine *Simple Support Function* unterstützt immer *eine* Hypothese zu einem gewissen Grad (m(A)=s), und die restliche Evidenz (1-s) wird Θ zugewiesen. Dieser Anteil wird im folgenden mit *Theta-Anteil* bezeichnet.

(c) $\prod_{\substack{X \in T \\ X \supset A}} m_X(X)$ und $\prod_{\substack{X \in T \\ X \supset A}} m_X(\Theta)$

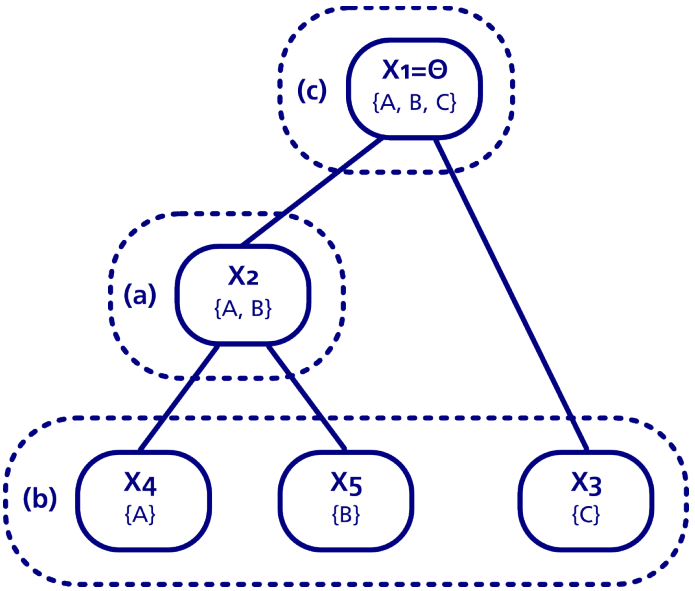

Abb. 29: Hypothesenhierarchie

Damit fließen in die Evidenz $m_T(A)$ (der Hypothese $A \in T \neq \Theta$) neben der offensichtlichen Einzelevidenz für *A* (a) noch weitere Komponenten ein: Alle Hypothesen, welche keine Schnittmenge mit *A* haben oder eine Untermenge von A sind, unterstützen *A* mit ihrem *Theta-Anteil* (b). Und Hypothesen, die eine Obermenge von *A* sind (außer *A* selber), unterstützen *A* mit beiden Anteilen, der bei *Simple Support Functions* in der Summe eins ergibt und im Produkt somit wegfällt. Dies führt zu folgendem »kombinierten« *Basic Probability Assignment*[61]:

$$m_T(A) = m_A(A) \prod_{\substack{X \in T \\ X \not\supset A}} m_X(\Theta) \prod_{\substack{X \in T \\ X \supset A}} (m_X(X) + m_X(\Theta))$$

Und nach der Kürzung:

$$m_T(A) = m_A(A) \prod_{\substack{X \in T \\ X \not\supset A}} m_X(\Theta)$$

4.3.3 Integration der sensomotorischen Repräsentation

Wie schon angedeutet werden mit Hilfe der *Theory of Evidence* von Dempster und Shafer Hypothesen über die aktuelle Umgebung des Agenten aufgestellt. Es geht z.B. um die Beantwortung der Frage, in welchem Raum in der VR-Umgebung sich der *SMX* momentan befindet.

Konstruktion von Basic Probability Assignments

Das System assoziiert in der Lernphase Explorationsschritte bzw. die daran geknüpften sensomotorischen Eigenschaften $S_k \in S$ (siehe Kapitel 4.2) mit

61 Die Normalisierung der Funktion wird hier nicht berücksichtigt.

Umgebungen (bzw. Klassen von Umgebungen) $X_i \in T$. S ist die Menge aller bekannten sensomotorischen Tripel *[View, Action, View]*.

Wenn der Agent nach einer Exploration eine Rückmeldung bekommt, in welcher Umgebung er sich befindet, werden die gesammelten sensomotorischen Eigenschaften $S_{1,} S_{2,} \ldots, S_n \in S$ als unterstützende Evidenz für diese Umgebung gewertet. Befindet sich der *SMX* danach in einer unbekannten Umgebung, erhöht ein erneutes Aufkommen eines dieser sensomotorischen Tripel die Evidenz für die eben gelernte Umgebung. Ein Beispiel:

Der Agent erkennt eine Garderobe (*View*), er dreht sich um und läuft drei Meter (*Action*) und sieht eine große Tür. Bekommt er nun die Rückmeldung, dass er sich in einem Hausflur befunden hat, so wird der *SMX* in einer nächsten Umgebung, in der er die gleiche Beobachtung macht, die Evidenz für die Hypothese »Die momentane Umgebung ist ein Hausflur« erhöht.

Der Grad, mit dem eine sensomotorische Eigenschaft $S_k \in S$ eine Umgebung (oder eine Menge von Umgebungen) $X_i \in T$ unterstützt, ergibt sich durch die relative Häufigkeit, mit der für S_k die Rückmeldung X_i gegeben wurde:

$$r_k(X_i) = \frac{t_{ik}}{\sum_{H_l \in T} t_{lk}}$$

Die folgende Matrix ist ein Beispiel für die Anzahl, mit der die sensomotorischen Eigenschaften $S_{1,} \ldots, S_7 \in S$ in Umgebungen $X_{1,} \ldots, X5 \in T$ aufgetreten sind.

	S1	S2	S3	S4	S5	S6	S7
X1	0	0	0	0	0	0	0
X2	0	1	4	0	5	0	1
X3	1	0	0	0	0	1	0
X4	2	0	2	0	9	0	1
X5	0	0	0	0	1	0	0

Das heißt, wenn z.B. für das sensomotorische Tripel *[View2, Action15, View8]* (S_3) vier mal die Rückmeldung »Büro 1280« (X_2) und zwei mal die Rückmeldung »Ausstellungsraum« (X_4) gegeben wurde, bekommt die Umgebungshypothese »Büro 1280« eine Evidenz von

$$r_3(X_2) = \frac{4}{4+2} = 0.\overline{6}$$

und die Umgebung »Ausstellungsraum« eine Evidenz von

$$r_3(X_4) = \frac{2}{4+2} = 0.\overline{3}$$

Die relative Häufigkeit $r_k(X_i)$ entspricht also der im letzten Kapitel beschriebenen Funktion m_{X_i}.

Damit gibt es einen Weg, über sensomotorische Eigenschaften und einem Prozess des überwachten Lernens ein *Basic Probability Assignment* zu konstruieren, welches Hypothesen über Umgebungen Evidenz zuweist.

Dekomposition in Simple Support Functions

Für die Berechnung der Evidenz in einem hierarchischen Hypothesenraum müssen die *bpa's*, wie bereits erwähnt, in *Simple Support Functions* aufgeteilt werden. Dies wird über eine Teilung durch die sog. *commonality function* bewerkstelligt,

$$m_{k,X_i}(X_i)=\frac{r_k(X_i)}{q_k(X_i)} \quad , X_i \in T, S_k \in S$$
$$q_k(X_i)=\sum_{(Y \supseteq X_i)} r_k(Y)$$

die außerdem vorraussetzt, dass für ein $S_k \in S$ nicht alle Evidenz auf die Raumhypothesen verteilt wird, sondern ein ε -Anteil Θ zugewiesen bekommt

ε : Minimale Evidenz für Θ

$$r_k(X_i)=\begin{cases} \frac{t_{ik}}{\sum_{X_l \in T} t_{lk}} \cdot (1-\varepsilon) & \text{falls } h \neq \Theta \\ \varepsilon & \text{falls } h = \Theta \end{cases} \quad , X_i \in T, S_k \in S$$

Berechnung der Evidenzverteilung

Diese *Simple Support Functions* lassen sich jetzt mit der Kombinationsregel von *Bernoulli* kombinieren:

$$m_{X_i}(X_i)=1-\prod(1-m_{k,X_i})$$
$$m_{X_i}(\theta)=\prod m_{k,X_i}(\Theta)$$

Sind nun während einer Exploration mehrere sensomotorische Eigenschaften gesammelt worden und seien diese Eigenschaften durch die Menge S_e definiert, so ergibt sich folgende Kombination dieser gesammelten Evidenzen:

$$m_{X_i}(X_i)=1-\prod_{S_k \in S_e}(1-m_{k,X_i})$$
$$m_{X_i}(\theta)=\prod_{S_k \in S_e} m_{k,X_i}(\Theta)$$

Um aus diesen einzelnen *Basic Probability Assignments* eine Funktion m_T zu konstruieren, welche dann eine generelle Verteilung der Evidenz widerspiegelt, werden die *bpa's*, wie im letzten Kapitel beschrieben, kombiniert:

K : Normalisierungsfaktor ($\emptyset$ zugewiesene Evidenz aufteilen)

$$m_T(X_i) = \begin{cases} K \cdot \prod_{X_i \in T} m_{X_i}(\Theta) & \text{falls } h = \Theta \\ K \cdot m_{X_i}(X_i) \cdot \prod_{\substack{Y \in T \\ Y \not\supseteq h}} m_Y(\Theta) & \text{sonst} \end{cases}$$

4.3.4 Inference By Information Gain

Nun, wo Hypothesen über die momentane Umgebung angestellt werden können, stellt sich noch die Frage, wie eine Umgebung weiter exploriert werden soll, um genügend Evidenz für eine Hypothese zu finden. Dieses Kapitel befasst sich mit der Fragestellung, vor der das System steht, wenn der Agent mehrere Hypothesen über seinen Aufenthaltsort hat und über eine weitere Exploration der Umgebung entscheiden muss. Wie bereits angedeutet orientiert sich das System am Optimierungskriterium Informationszuwachs; konkret wird die Inferenzstrategie *Inference by Information Gain (IBIG)* verwendet[62].

IBIG nutzt keine Heuristiken, sondern ist eher mit einem axiomatischen System zu vergleichen. Das heißt, der Algorithmus benötigt keine Annahmen oder Regelwerke bzgl. der Explorationsstrategie, sondern er passt sich dynamisch der aktuellen Evidenzverteilung an, indem er vor jedem Schritt berechnet, welcher den höchsten Informationszuwachs verspricht.

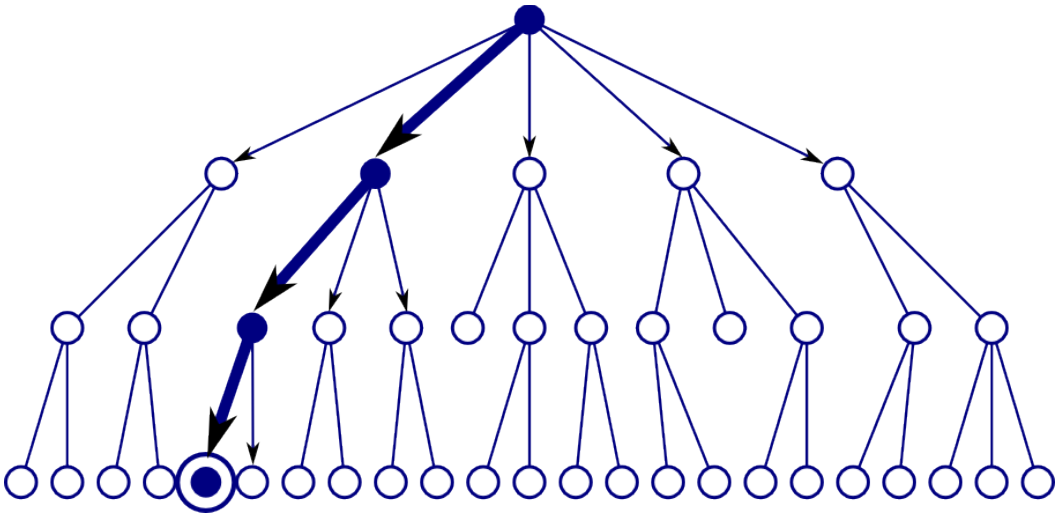

Abb. 30: Erfolgreiches Partitionieren in einem eindeutigen Fall. (Grafik nach Schill 1997)

Die Vorgehensweise gewöhnlicher Inferenzstrategien ist häufig eine Partitionierung. Das heißt, es wird typischerweise eine einzige Hypothese oder eine kleine Menge an Hypothesen aufgrund initialer Evidenz verfolgt. Die ausgewählte Hypothese bestimmt nun den Prozess, welche Daten gesammelt werden. In einer hierarchischen Wissensbasis führt eine Partitionierung dazu, dass auf jeder Hierarchieebene die jeweils wahrscheinlichste Hypothese verfolgt wird, bis man die unterste Ebene erreicht hat (siehe Abb. 30). Für die Hierarchie in Abb. 28 (S. 48) bedeutet das z.B., dass im ersten Schritt Daten

62 Schill 1997

gesammelt werden, welche eine der Hypothesen »Büro«, »Ausstellungsraum« oder »Sanitärbereich« unterstützen. Wenn die Hypothese »Ausstellungsraum« durch diese Daten (über Explorationsschritte gesammelte sensomotorische Eigenschaften) die meiste Evidenz erhält, wird alleine sie weiterverfolgt. Das heißt, im nächsten Schritt wird evtl. geschaut, ob sich in dem Raum Skulpturen oder Gemälde befinden, so dass eine der beiden Hypothesen, die sich in der Hierarchie unter »Ausstellungsraum« befinden, bestätigt werden können.

Zu Problemen führt die Partitionierung, wenn die gesammelten Daten eine mehr oder weniger gleichmäßige Evidenzverteilung ergeben, so dass keine Hypothese besonders hervorsticht und der Algorithmus eine relativ Unsichere weiterverfolgen muss, welche evtl. in einer Sackgasse endet (siehe Abb. 31). In diesem Fall muss die Suche erneut von der Wurzel begonnen werden.

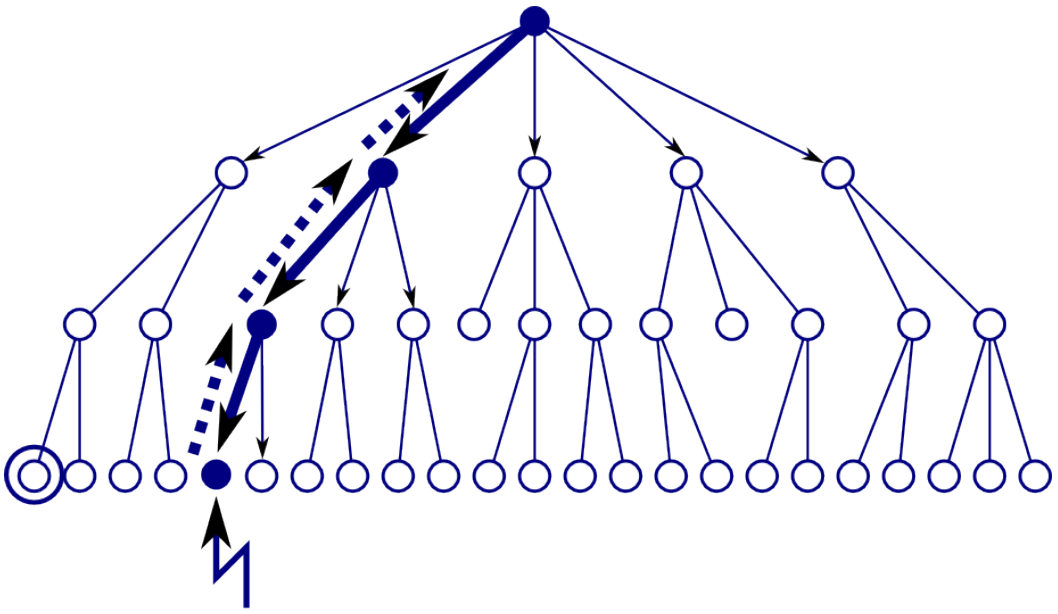

Abb. 31: Partitionierung führt in eine Sackgasse in einem mehrdeutigem Fall. (Grafik nach Schill 1997)

IBIG bietet den Vorteil, dass es gewissermaßen parallel alle Hypothesen verfolgt. Es wird nicht eine wahrscheinliche Hypothese verfolgt, die dann evtl. in einer Sackgasse enden kann, sondern es wird für alle möglichen nächsten Schritte geschaut, welcher den höchsten Informationszuwachs verspricht, d.h. welcher die Evidenzverteilung am stärksten verändert. Das Maß dieser Veränderung I wird in der Differenz zwischen der Unterstützung für die einzelnen Hypothesen vor und nach der Integration der erwarteten Evidenz gemessen.

$$I = |m_T - \hat{m}_T|$$

$\hat{m}_T$ ist dabei die Evidenzverteilung, die das System nach einem möglichen Explorationsschritt erwartet. Für den *SMX* wäre dies z.B. die Evidenzverteilung, wenn der nächste Explorationsschritt die sensomotorische Eigenschaft *[View18, Action2, View3]* ergeben würde. Zu bedenken ist noch, dass das System vor dem Ausführen eines Explorationsschrittes noch nicht weiß, welche sensomotorische Eigenschaft und welche daran geknüpfte Evidenz sich da-

hinter verbirgt, da der *Target View* noch nicht bekannt ist. Daher optimiert der Algorithmus nach dem *vermuteten* Informationszuwachs.

4.4 Sakkadische Bildanalyse

Der *SMX* basiert auf einer visuellen Sensorik. Das Modul zur Analyse dieser sensorischen Daten ist an das visuelle System des Menschen angelehnt[63]. Das Design dieses Moduls diente in vielen Bereichen dem *SMX* als Vorlage und ist in gewisser Weise ein sensomotorisches Raumklassifikationssystem auf einer höheren Granularitätsstufe. Bilder bzw. visuelle Schnappschüsse (siehe Abb. 25, S. 43) werden anhand ihrer sensomotorischen Eigenschaften analysiert, welche sich aus einer Imitation menschlicher Augenbewegungen (*Sakkaden*) ergeben.

Abb. 32: Voreverarbeitung der Szene mit neuronalen Filtern. (Grafik nach Schill 2005)

Die Analyse beginnt mit einer Vorverarbeitung eines Bildes mit neuronalen Filtern, welche »interessante« Bereiche im Bild identifiziert (siehe Abb. 32). Aus diesen Bereichen werden anschließend mit weiteren Filtern Eigenschaften extrahiert, welche mit den verknüpfenden *Sakkaden* eine sensomotori-

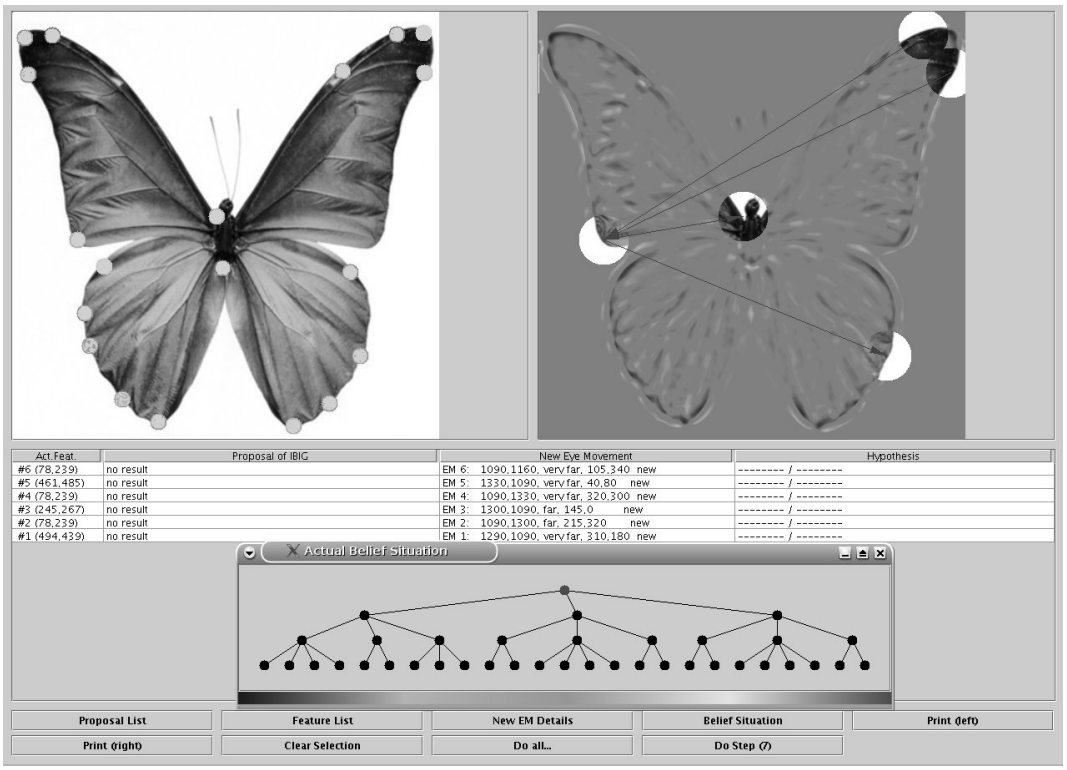

Abb. 33: Screenshot des Bildanalysesystems: Extraktion der »interessanten« Bildbereiche (links); *Sakkaden* auf dem Bild (rechts) (Grafik nach Schill 2005).

63 Schill 2001

sche Repräsentation des visuellen Schnappschusses bilden, nach der es klassifiziert werden kann (siehe Abb. 33).

4.5 VR-Umgebung

Um die Explorationsfähigkeiten des *SMX* bzw. der zugrundeliegenden sensomotorischen Raumrepräsentation unkompliziert testen zu können, wurde zunächst darauf verzichtet das System in einen Roboter zu integrieren. Stattdessen muss sich das System in einer VR-Umgebung, d.h. einer Simulation einer dreidimensionalen Umgebung, bewähren.

Die Vorteile dieser Simulation sind eine sehr hohe Flexibilität. Es kann frei konfiguriert werden, ob bzw. welche physikalischen Gesetze gelten. Damit ist es möglich das System schrittweise zu entwickeln, so dass es anfangs unter vereinfachten Bedingungen arbeitet und dann der »Realitätsgrad« z.B. mit verrauschten Sensordaten steigt.

Ein weiterer Vorteil ist die Möglichkeit, ohne größeren Aufwand viele VR-Umgebungen kreieren zu können, mit denen dann das System getestet werden kann. Hier sind einer Implementierung, welche in einen »echten« Roboter integriert ist, deutliche Grenzen gesetzt.

Blender

Für die konkrete Simulation der Umgebungen wurde *Blender*[64] verwendet. Der Hauptvorteil dieses Systems ist die Integration von Modellierungsumgebung und Simulationsengine in einer Applikation. Da es sich bei *Blender* im

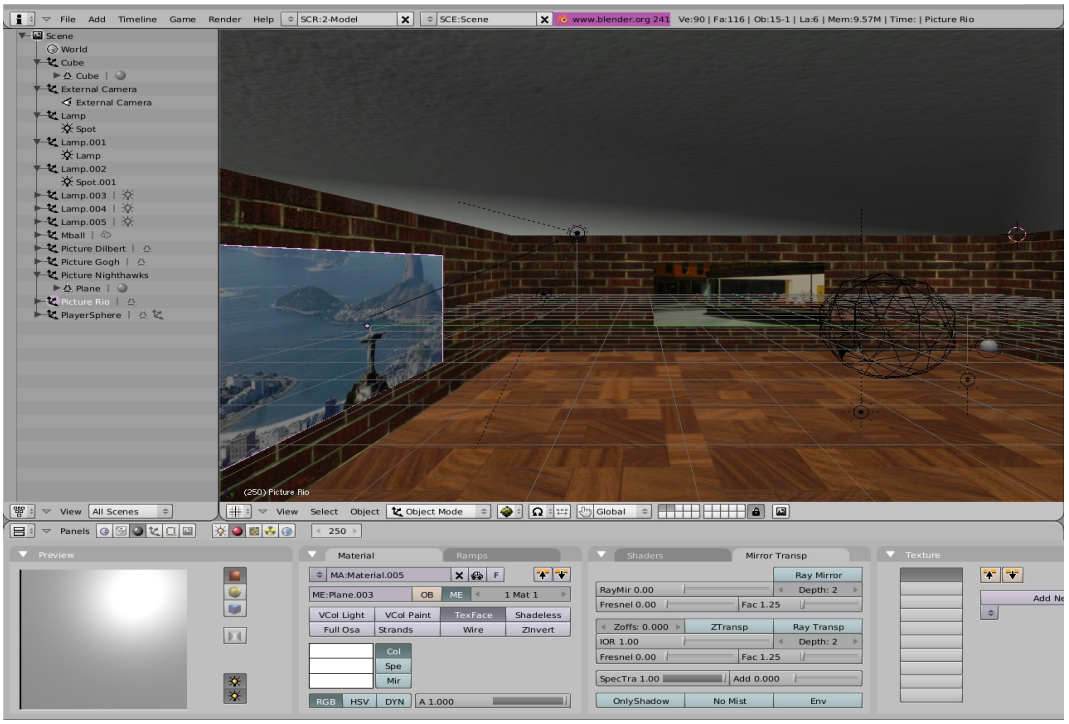

Abb. 34: Modellierungsumgebung von Blender

64 Roosendaal 2004, http://www.blender.org

Kern um ein 3D-Modellierungsprogramm handelt, sind die daran geknüpften Werkzeuge sehr ausgereift und ermöglichen eine bequeme Erzeugung von VR-Umgebungen. Die in *Blender* integrierte sog. *Gameengine*, d.h. Echtzeit Simulationsumgebung, erfüllt keine ausgefallenen Wünsche bzgl. der visuellen Qualität, ist für die benötigten Zwecke jedoch vollkommen ausreichend.

Ein weiterer nicht zu verachtender Vorteil ist die *Python*-Schnittstelle der *Gameengine*, welche eine unkomplizierte Steuerung des Agenten durch die Umgebung erlaubt; besonders, da der Großteil des restlichen Systems gleichfalls auf der Programmiersprache *Python* basiert.

4.5.1 Umgebungen

Für erste Tests wurden Museumsräume als Umgebungen verwendet. Die vordefinierten *Views* waren Gemälde, anhand welcher der Agent die Räume identifizieren musste.

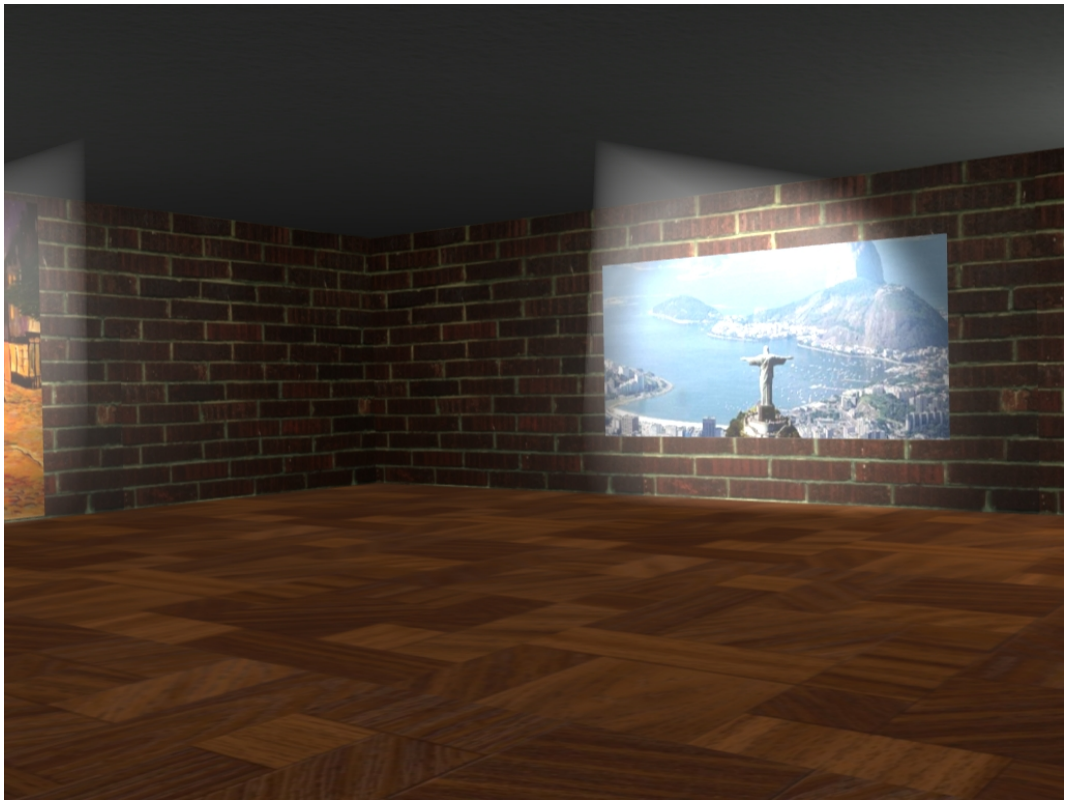

Abb. 35: Perspektive des Agenten in einem Raum der VR-Umgebung

4.5.2 Sensomotorische Schnittstelle

Die sensomotorische Schnittstelle zu *Blender* bietet die Möglichkeit den Agenten durch die Umgebung zu steuern und visuelle Schnappschüsse von der Umgebung (siehe Abb. 35) zu machen. Eine Besonderheit der Schnittstelle ist, dass sie über ein ASCII-basiertes TCP-Protokoll angesprochen wird, so

dass es möglich ist, die VR-Umgebung auf einen separatem Rechner auszulagern. Dies kann von Bedeutung werden, wenn aufwendige VR-Umgebungen simuliert werden sollen, welche ein hohes Maß an Rechenkapazität benötigen.

4.6 SMX Kernmodul

Dieser Abschnitt beschreibt die Kernkomponente des *SMX*, die mit Hilfe des visuellen Analysesystems und des Inferenzmoduls über die sensomotorischen Schnittstelle zur VR-Umgebung die Exploration der VR-Umgebung steuert und Hypothesen anstellt, wo der Agent sich momentan befindet. Zur Veranschaulichung wird die Funktion des Moduls in einem zeitlichen Ablauf geschildert.

4.6.1 Initialisierung

In der Initialisierungsphase wird die Wissensbasis des *SMX* geladen, d.h. das vom System erlernte Wissen und die zusätzlichen Informationen, die es bekommt.

Die Wissensbasis beinhaltet:

- die *Viewpoints* der Umgebungen und wie sie miteinander verknüpft sind[50]
- die erlernten *Views* (Sensorik) und Aktionen (Motorik) sowie die sich daraus ergebenden sensomotorischen Eigenschaften[65]
- die Hierarchie an bekannten Umgebungen bzw. Klassen von Umgebungen[66]
- die Statistik, wie stark eine der sensomotorischen Eigenschaften eine Umgebung unterstützt, d.h. die erlernten *Basic Probability Assignments*[67]

4.6.2 Verbindung mit der VR-Umgebung

Nach der Initialisierung wartet das System auf eine Verbindung mit der VR-Umgebung. Die Schnittstelle zum Agenten in der VR-Umgebung ist ein ASCII-basiertes TCP-Protokoll, so dass der rechenaufwendige Simulationsteil auf einen separaten Rechner ausgelagert werden kann.

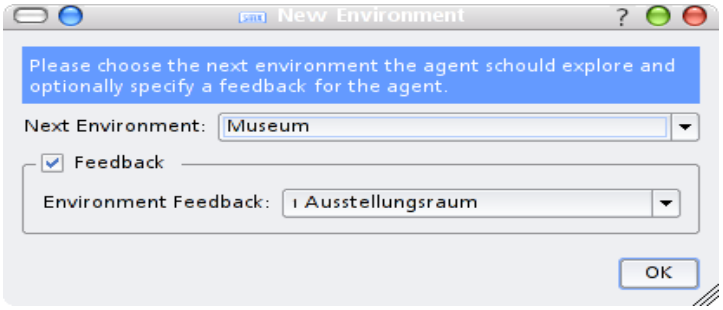

Abb. 36: Auswahl einer zu explorierenden Umgebung und eines Feedbacks für den *SMX*.

Ist eine Verbindung aufgebaut, muss die zu explorierende Umgebung ausgewählt werden, um die zugehörigen *Viewpoints* laden zu können. Optional

65 siehe Kapitel 4.2, S. 42
66 siehe Kapitel 4.3.2, S. 47 und 4.5.1, S. 58
67 siehe Kapitel 4.3.3, S. 50

kann noch ein Feedback für die Umgebung angegeben werden, so dass der *SMX* mit Hilfe während der Exploration gemachten Erfahrungen lernen und seine Fähigkeit seine Umgebung zu erkennen verbessern kann (siehe Abb. 36).

4.6.3 Bestimmung des nächsten Explorationsschrittes

Nach der Auswahl der Umgebung wird der Agent an einem zufälligen Punkt (*Viewpoint*) dieser Umgebung »ausgesetzt«. Das System analysiert den aktuellen *View* und bekommt anschließend die direkt erreichbaren *Viewpoints* in der unmittelbaren Umgebung mit der Information, mit welcher Aktion sie erreicht werden können[68] (s. Abb. 37). Nun ermittelt das System alle sensomotorischen Eigenschaften, welche zum aktuellen *View* und den möglichen Aktionen passen; in dem Beispiel wären das alle Tripel der Form *[View6, Action5, View?]* und *[View6, Action18, View?]*.

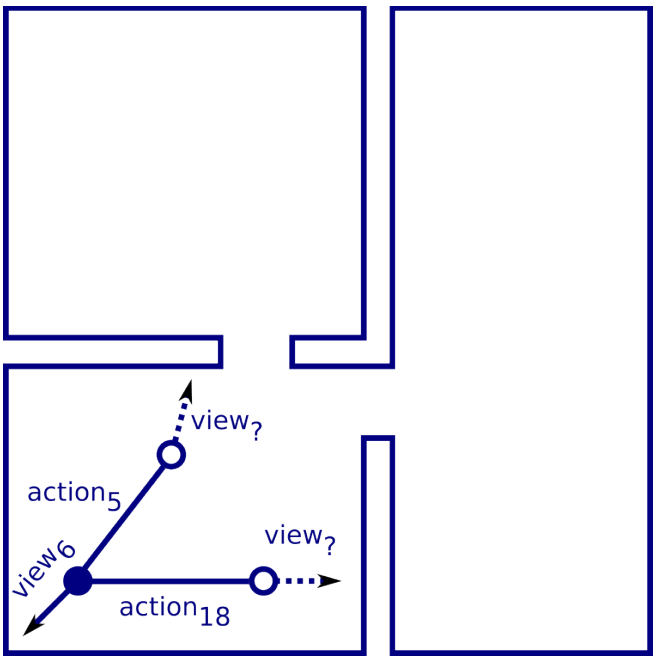

Abb. 37: In der Nähe des aktuellen *view*$_6$ befinden sich zwei weitere *View(points)*, welche sich durch die Aktionen *action*$_5$ und *action*$_{18}$ erreichen lassen.

Für all diese sensomotorischen Eigenschaften wird nun ihr potenzieller Informationszuwachs berechnet. Ausgenommen sind lediglich die Eigenschaften, die in der Umgebung gerade schon ausgeführt wurden. Das heißt, wenn bereits ermittelt wurde, dass die jetztige Umgebung die sensomotorische Eigenschaft *[View6, Action5, View7]* hat, wird dies nicht ein weiteres Mal getestet.

Wurde nun die sensomotorische Eigenschaft mit dem höchsten potenziellen Informationszuwachs bestimmt, führt der Agent die an sie geknüpfte Aktion aus und bestimmt den noch unbekannten *Target View*. Damit hat der *SMX* das Tripel komplettiert und eine neue sensomotorische Eigenschaft ermittelt.

68 siehe Kapitel 4.2, S. 42

Bei dieser Eigenschaft kann es sich um die erwartete handeln; es ist jedoch auch möglich, dass eine andere sensomotorische Eigenschaft ermittelt wird, die dazu u.U. noch völlig unbekannt ist und damit auch keine neue Evidenz mit sich bringt. Für den Fall, dass sie jedoch schon bekannt ist, wird die an sie geknüpfte Evidenz integriert, und der Prozess beginnt von vorne.

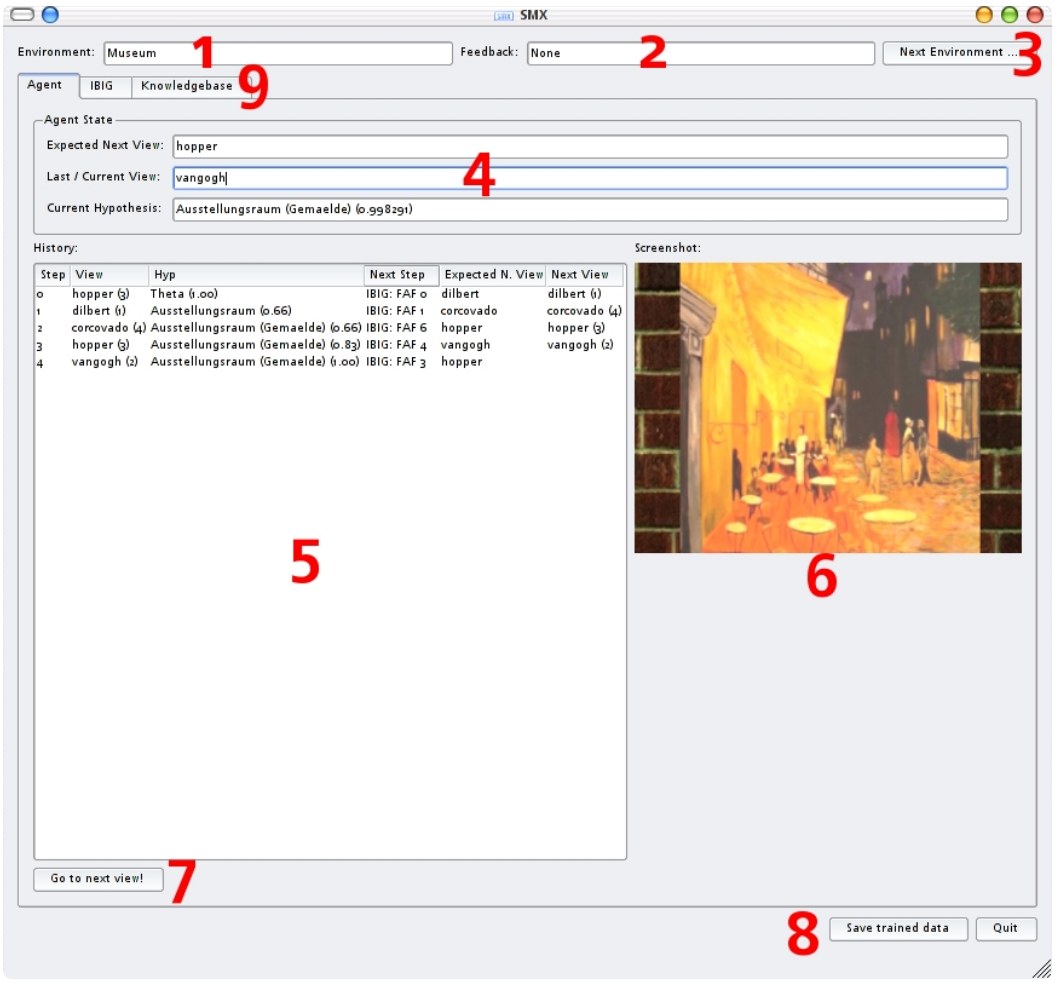

Abb. 38: Screenshot von der Nutzerschnittstelle (GUI) des *SMX*.

Die Nutzerschnittstelle (GUI) für das Kernmodul ermöglicht es (nach einer initialen Auswahl der Umgebung, siehe Abb. 36) die Exploration Schritt für Schritt nachzuverfolgen (siehe Abb. 38). Oben im Hauptfenster ist die aktuelle Umgebung (1) und das Feedback, welches der Agent nach der Exploration bekommen wird, angezeigt. Direkt darunter (4) steht jeweils die Hypothese, über den aktuellen *View*, den erwarteten nächsten *View* und über den Raum. Die letzten Explorationsschritte lassen sich über die *History* (5) nachverfolgen:

- »View« ist das Ergebnis der Anaylse des aktuellen Schnappschusses (rechts)
- »Hyp« ist die Raumhypothese mit der größten Unterstützung.

- »Next Step« zeigt an, ob der nächste Explorationsschritt aufgrund einer sensomotorischen Eigenschaft (*FAF*) oder zufällig ausgewählt wurde.
- »Expected N. View« ist der »View« der gemäßt dem ausgewählten sensomotorischen Tripel erwartet wird. Bei zufällig ausgewählten Schritten wird lediglich »Unknown“ angegeben.
- »Next View« wird erst nach dem Ausführen der Aktion angezeigt. Er ist das Ergebnis der Bildanalyse am *Target View*.

Rechts neben der *History* (6) wird immer der aktuelle visuelle Schnappschuss, oder der in der *History* ausgewählte, angezeigt, und mit dem *Button* unter der *History* (7) kann der nächste Explorationsschritt angestoßen werden. Mit »Next Environment...« (3) wird dem Agenten das anfangs gewählte Feedback übermittelt und eine neue Umgebung kann ausgewählt werden. Wenn die gesammelten Informationen einer Exploration in der Wissensbasis gespeichert werden sollen, kann dies mit dem *Button* rechts unten getan werden (8). Und direkt daneben ist der Button zum beenden des Programms.

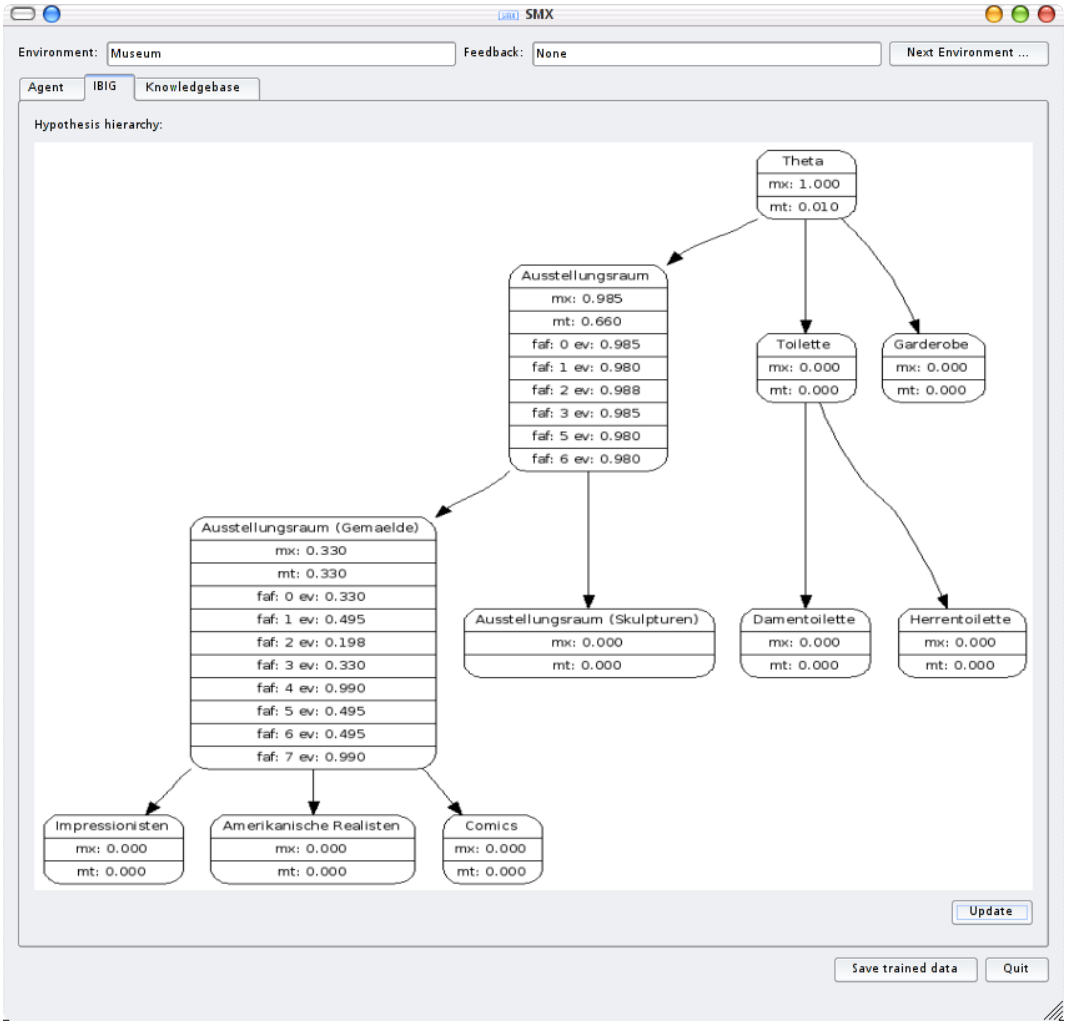

Abb. 39: Evidenzverteilung in der Raumhierarchie des IBIG-Moduls. Zu jeder Raumhypothese wird die Einzelevidenz (*mx*) und die kombinierte Evidenz (*mt*) angezeigt, sowie, welche sensomotorischen Eigenschaften (*faf*) einen Raum unterstützen.

Neben dieser Ablaufsteuerung bietet diese Oberfläche noch weiteren Einblick in das System. Unter dem Registerreiter »IBIG« (9) verbirgt sich eine Ansicht über die Evidenzverteilung in der Raumhierarchie (siehe Abb. 39).

Der dritte Registerreiter »Knowledgebase« gibt einen Einblick, welche sensomotorischen Eigenschaften (*FAF)* dem System bekannt sind und wie sehr sie eine Raumhypothese unterstützen (siehe Abb. 40).

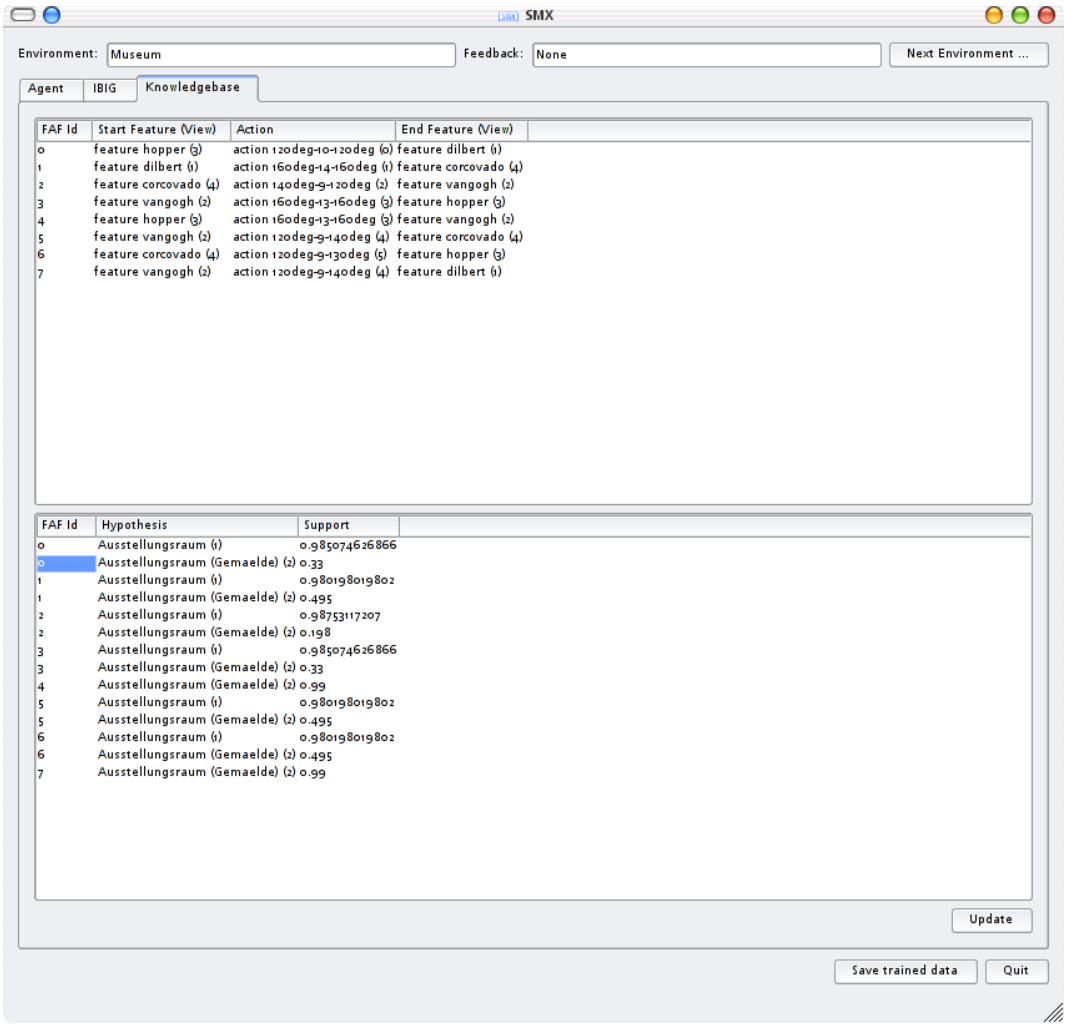

Abb. 40: Die obere Tabelle zeigt die bekannten sensomotorischen Eigenschaften *[view, action, view]* und in der unteren Tabelle ist aufgelistet wie stark ein Tripel eine Raumhypothese unterstützt.

4.7 Implementierung

Nach der funktionalen Beschreibung der Komponenten geht dieses Kapitel genauer auf die konkrete Implementierung ein. Wichtige Methoden werden knapp beschrieben, und es wird ein grober Übrblick über die Klassenstruktur gegeben (siehe Abb. 43 und Abb. 44).

Der *SMX* ist fast ausschließlich in *Python* implementiert[69]. Die Vorteile dieser komplett dynamisch getypten und objektorientierten Programmiersprache sind ihre sehr große Standardbibliothek und das umfangreiche Angebot weiterer Bibliotheken. Dies erlaubt eine Konzentration auf die eigentliche Programmlogik, und die dynamische Typisierung ist bei einem sich stark wandelnden Prototypen auch von Vorteil.

4.7.1 IBIG Inferenzmodul

Das Inferenzmodul lässt sich in drei Komponenten unterteilen:

- eine abstrakte Implementierung einer Hierarchie bzw. eines Baumes
- ein Modul zur Verwaltung der relativen Häufigkeit, in der sensomotorische Eigenschaften in einer Umgebung auftraten
- ein Modul zur Berechnung der aktuellen Evidenzen und des Informationszuwachses

(Raumhypothesen-)Hierarchie

```
agentctrl.util.hierarchy.py
ibig.ibig.py
```

In `agentctrl.util.hierarchy.py` ist eine Hierarchiedatenstruktur in Form eines Baumes implementiert. Die Elemente dieses Baumes `Node` können durch Vererbung angepasst werden, in diesem Fall `IBIGNode`. Diese Klasse repräsentiert eine Umgebungs- bzw. Raumhypothese und speichert den durch die Funktion m_X zugewiesenen Wert `IBIGNode.mx`, d.h. die Kombination der *Simple Support Functions*, sowie die durch m_T bestimmte Gesamtevidenz `IBIGNode.mt`[70]. Zusätzlich können zur Berechnung des Informationszuwachses für etwaige Explorationsschritte potentielle m_X- und m_T- Werte zwischengespeichert werden: `IBIGNode.setPotMX, IBIGNode.getPotMX`[71].

Statistik

```
agentctrl.statistic.py
```

Die Statistik, mit welcher Häufigkeit eine sensomotorische Eigenschaft in einer Umgebung gefunden wurde, wird in diesem Modul verwaltet[72]. Mit `FAFOccurrences.addOccurrncesForNode(node, faf)` lässt sich eine gefun-

69 http://www.python.org
70 siehe S. 49
71 siehe Kapitel 4.3.4, S. 53
72 siehe Kapitel 4.3.3, S. 50

dene sensomotorische Eigenschaft `faf` für die Umgebung `node` verbuchen. `faf` steht hierbei für »feature-action-feature« und repräsentiert ein sensomotorisches Tripel, wohingegen `node` für einen *Knoten* in der Hypothesenhirarchie steht, also ein Hypothese über eine Umgebung.

Inferenz

`ibig.ibig.py`

Die eigentliche Berechnung der Evidenzen und des Informationszuwachses verschiedener Explorationsschritte ist in der Klasse `IBIG` implementiert[73].

`IBIG.calculatePotentialEvidence(faf)` berechnet die Evidenzverteilung für den Fall, dass in der momentanen Umgebung das sensomotorische Tripel `faf` gefunden wird. Diese Methode wird mit allen möglichen Aktionen, die dem Agenten zu einem gewissen Zeitpunkt zur Verfügung stehen ausgeführt, so dass anschließend mit `IBIG.calculateInformationGain()` und `IBIG.getFAFWithMaxInformationGain()` der Schritt, welcher den höchsten Informationszuwachs verspricht, bestimmt werden kann. Mit `IBIG.doFAF(faf)` wird nach dem Ausführen einer Aktion, wenn das wirkliche sensomotorische Tripel ermittelt wurde[74], die Evidenzverteilung in der Hierarchie entsprechend aktualisiert.

Mit `IBIG.newScene(feedback)` wird ein Wechsel einer Umgebung signalisiert. Der Agent wird z.B. in einem neuen Raum in der VR-Umgebung gestellt, wofür z.B. die Evidenzen in der Hierarchie zurückgesetzt werden. Der optionale Parameter `feedback` ist die Rückmeldung für den Agenten, in welcher Umgebung er sich momentan noch befindet, so dass das System anhand der gemachten Erfahrungen, d.h. die gesammelten sensomotorischen Tripel, lernen kann.

73 siehe Kaptiel 4.3.4, S. 53

74 Vor dem Ausführen einer Aktion und der anschließenden Analyse des dortigen *Views*, fehlt noch der *target View*, d.h. es sind nur die ersten beiden Teile des sensomotorischen Tripels bekannt.

4.7.2 VR-Umgebung

```
agentctrl.blendereventhandler.py
agentctrl.blenderagent2.py
agentctrl.vrcomserver2.py
agentctrl.util.mathutil.py
```

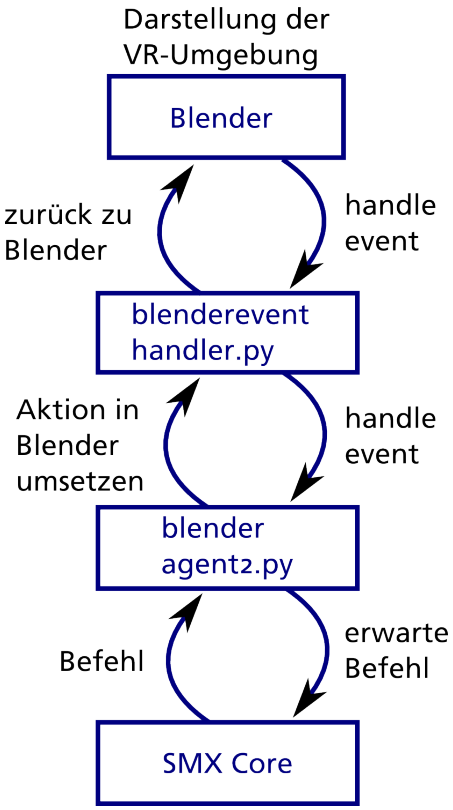

Abb. 41: Steuerung des Agnten (SMX) in Blender

Für die VR-Umgebung wurde, wie bereits erläutert, Blender bzw. dessen Gameengine gewählt. Sie lässt sich nicht komplett integrieren, stattdessen wird aus der laufenden Gameengine periodisch ein Pythonskript (`blenderevent handler.py`) aufgerufen, welches dann Kontrolle über den Agenten in der Umgebung hat. Dieses Skript ist direkt in Blender integriert und ruft lediglich die Methode `BlenderAgent.handleEvent()` auf (siehe Abb. 41). Hier wird entschieden, was der Agent als nächstes macht. Dafür wird über `VRAgent Connector` angefragt, welchen Befehl das *SMX* Kernmodul gegeben hat. Wurde ein Befehl empfangen, wird dieser umgesetzt, indem der Agent in VR-Umgebung bzw. dessen »virtuelle Kamera« über die Pythonschnittstelle von Blender manipuliert wird.

Protokoll

Das auf TCP[75] aufsetzende Protokoll zum *SMX* Kernmodul wird in den Klassen `VRAgentProtocol` und `VRAgentConnector` auf Blender-Seite bzw. `AgentVRProtocol` und `AgentVRControl` auf *SMX*-Seite implementiert. Es unterstützt folgende Befehle[76]:

```
→VRCONNECT
←ACK
```

Dieser Befehl ist die initiale Verbindung der VR-Umgebung mit dem *SMX*.

```
→EXPECTING_COMMAND
```

Nach der Herstellung der Verbindung kann die VR-Umgebung nach Instruktionen »fragen«. Diese Befehle werden im folgenden beschrieben:

```
←GOTO (<x1>, <y1>, <z1>) (<x2>, <y2>, <z2>)
```

75 RFC 793: http://www.ietf.org/rfc/rfc793.txt
76 →: Befehl von Blender(Agent) zum SMX Kernmodul. ←: Entgegengesetzte Richtung

```
→EXPECTING_COMMAND
```

Der *SMX* instruiert den Agenten an den Punkt *(x1, y1, z1)* zu gehen und in Richtung des Punktes *(x2, y2, z2)* zu »schauen«.

```
←JUMPTO (<x1>, <y1>, <z1>) (<x2>, <y2>, <z2>)
→EXPECTING_COMMAND
```

Dieser Befehl unterscheidet sich von **GOTO** nur dadurch, dass der Agent sich nicht zum angegebenen Punkt »bewegt«, sondern direkt dorthin versetzt wird. Das heißt, es ist nicht zu sehen wie er zu diesem Punkt gelangt.

```
←SCREENSHOT
→SCREENSHOT
→<Bildgröße in Bytes>
→<Bilddaten>
→EXPECTING_COMMAND
```

Der Agent wird beauftragt, aus seiner momentanen Position einen visuellen Schnappschuss (Screenshot) zu machen und diesen für die Analyse zurückzuschicken.

Befehle

Um den Agenten bzw. die »virtuelle Kamera« in der VR-Umgebung zu bewegen, bietet Blender zwei Befehle. Mit **setPosition((x,y,z))** kann die Position des Agenten direkt gesetzt werden. Auf eine Aktivierung der Physikengine und der Bewegung des Agenten durch Rotations oder Bewegungsimpulse wurde bisher verzichtet. Eine Bewegung, wie bei dem Befehl **GOTO**, wurde dadurch simuliert, dass die Positionsänderungen durch setPosition((x,y,z)) in kleinen Schritten erfolgten. Die Orientierung des Agenten kann über **setOrientation(matrix)** gesetzt werden. Bei matrix handelt es sich um eine Rotationsmatrix[77]. Eine Orientierung kann durch die Kombination von Rotationen um die

x-Achse $\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{bmatrix}$,

die y-Achse $\begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix}$

und die z-Achse $\begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$ bestimmt werden[78].

Als Vereinfachung bewegt sich der Agent in der VR-Umgebung momentan lediglich in der x-y-Ebene, d.h. der Blick wendet sich nicht nach oben oder unten.

77 siehe Watt 1999

78 Alternativ bietet Blender noch die Möglichkeit die Orientierung des Agenten anstatt über eine Rotationsmatrix über ein *Quaternion* zu setzen.

Orientierungsberechnung

Die Standardausrichtung in Blender ist entlang der »negativen« Y-Achse ($lookat_{standard}$), d.h. jeder Winkel ist relativ zu diesem Vektor angegeben (siehe Abb. 42).

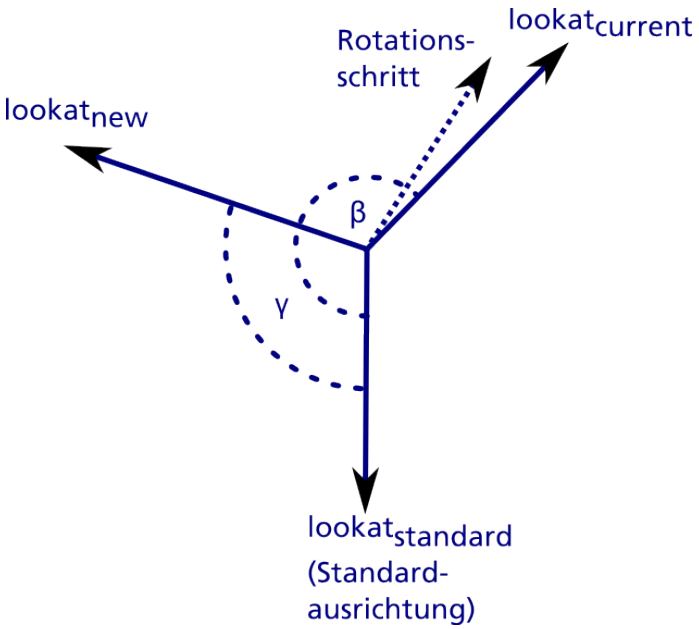

Abb. 42: Bei einer Rotation wird der Agent Schrittweise von $lookat_{current}$ zu $lookat_{new}$ rotiert.

Wenn der Agent nun seine Orientierung von $lookat_{current}$ zu $lookat_{new}$ ändert, tut er dies Schrittweise, um eine kontinuierliche Rotation zu simulieren.

$$lookat_{current} : c = \begin{pmatrix} x_c \\ y_c \end{pmatrix}, lookat_{new} : n = \begin{pmatrix} x_n \\ y_n \end{pmatrix}, lookat_{standard} : s = \begin{pmatrix} x_s \\ y_s \end{pmatrix}$$

$$\beta = \arccos\left(\frac{x_c * x_s + y_c * y_s}{\sqrt{(x_c^2 + y_c^2)} * \sqrt{(x_s^2 + y_s^2)}}\right), \gamma = \arccos\left(\frac{x_n * x_s + y_n * y_s}{\sqrt{(x_n^2 + y_n^2)} * \sqrt{(x_s^2 + y_s^2)}}\right)$$

Zunächst werden die Winkel zu den Beiden Orientierungen β und γ berechnet. Dieser bildet sich aus dem Arkuskosinus des Quotienten des Skalarprodukts durch das Produkt der Längen der beiden Vektoren.

Nun muss noch bestimmt werden, ob es sich um eine Rotation im oder gegen den Uhrzeigersinn handelt, da mit der obigen Formel immer der eingeschlossene Winkel (kleiner 180°) berechnet wird. Ist die Determinante größer als *0,* handelt es sich um eine Rotation gegen den Uhrzigersinn und sonst umgekehrt.

$$det\left(\begin{pmatrix} x_s \\ y_s \end{pmatrix}\begin{pmatrix} x_c \\ y_c \end{pmatrix}\right) = x_s * y_c + y_s * x_c$$

Mit diesem Wissen kann der Orientierungswinkel schrittweise von $lookat_{current}$ zu $lookat_{new}$ angepasst werden. Um aus einem Winkel α eine Orientierungsmatrix zu erhalten, kann die oben erwähnte Rotationsmatrix für die z-Achse gewählt werden:

$$\begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Weiterentwicklung

Bei einer Weiterentwicklung, wie sie im Kapitel 4.8.1 (S. 80) beschrieben ist, würde der Befehl **GOTO** evtl. durch mehrere parametrisierte *Control Laws* ersetzt werden.

4.7.3 Visuelles Analysesystem (Okusys)

```
agentctrl.okusyscomclient.py
```

Die eigentliche visuelle Analyse wird über das externe Programm *Okusys*[79] gemacht. Dieses wird wie die VR-Umgebung über ein auf TCP aufsetzendes Protokoll angesprochen, welches clientseitig in **OkusysComClient** implementiert ist.

```
→INST <Instanzname>
←OK
```

Der Befehl **INST** signalisiert *Okusys*, die angegebene Instanz zu laden (**OkusysComClient.selectInstance()**). Eine Instanz beinhaltet das gelernte Wissen über Bilder, so dass es gesendete Bilder klassifizieren kann.

```
→INSPECT_IMAGE
→<Bilddaten>
→END_OF_INSPECT_IMAGE
←ERGEBNIS
←Maximaler Glaube: <Evidenz für die Hypothese>
←Objekt: <Name der Hypothese> (Nr. <Id der Hypothese>)
←Anzahl der Fixationen: <Anzahl der Augenbewegungen>
```

Mit INSPECT_IMAGE wird beauftragt, das folgende Bild zu analysieren und eine Hypothese darüber anzustellen, um welches Bild es sich handelt (**OkusysComClient.inspectImage()**). Zurückgegeben wird die wahrscheinlichste Hypothese und der Grad mit der sie unterstützt wird.

4.7.4 Knowledgebase

```
ibig.knowledgebase.py
ibig.kbParser.fafkbparser.py
ibig.kbParser.hierarchyParser.py
ibig.kbParser.fafStatisticParsr.py
```

Die Wissensbasis (**Knowledgebase**) aggregiert die wichtigsten Daten des *SMX* und regelt ihre Persistierung. Darunter befinden sich:

- die Hierarchie an Raum- bzw. Umgebungshypothesen: **Knowledgebase.hierarchy**

79 Schill 2001

- die bisher gesammelte sensomotorische Eigenschaften:
  Knowledgebase.features (sensorische Komponente),
  Knowledgebase.actions (motorische Komponente),
  Knowledgebase.fafs (sensomotorische Eigenschaft)
- die Statistik, wie stark eine sensomotorische Eigenschaft eine Umgebung unterstützt: **Knowledgebase.fafStatistic**

Für diese Daten organisiert die Wissensbasis das Laden und Speichern im XML-Format sowie das Hinzufügen und Entfernen von Elementen. Mit loadKnowledgebase() und saveKnowledgebase() werden automatisch alle Daten geladen bzw. gespeichert. Diese Methoden erwarten, dass Knowledgebase mit einem gültigen Pfad (**kbBasePath**) zu einem Verzeichnis mit den XML-Dateien (default: **fafkb.xml**, **hierarchy.xml**, **fafstatistic.xml**) initialisiert worden ist.

Das Format der XML-Dateien ist hier durch eine *Document Type Definition (DTD)*[80] angegeben.

Raumhierarchie:

```
<!DOCTYPE hierarchy [
<!ELEMENT hierarchy (node)>
<!ELEMENT node (node*)>
<!ATTLIST node
   id ID #REQUIRED
   name CDATA #IMPLIED>
]>
```

Beispiel:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<hierarchy>
   <node id="0" name="Theta">
       <node id="1" name="Ausstellungsraum">
           <node id="2" name="Ausstellungsraum (Gemaelde)">
               <node id="3" name="Impressionisten"/>
               <node id="4" name="Amerikanischer Realismus"/>
               <node id="5" name="Comics"/>
           </node>
           <node id="8" name="Ausstellungsraum (Skulpturen)"/>
       </node>
       <node id="10" name="Toilette">
           <node id="6" name="Damentoilette"/>
           <node id="7" name="Herrentoilette"/>
       </node>
       <node id="9" name="Garderobe"/>
   </node>
</hierarchy>
```

Sensomotorische Eigenschaften:

```
<!DOCTYPE featureActionFeatureKB [
<!ELEMENT featureActionFeatureKB
   (features?, actions?, featureActionFeatures?)>
<!ELEMENT features (feature*)>
<!ELEMENT feature EMPTY>
<!ATTLIST feature
```

80 http://www.w3.org/TR/REC-xml/

```
    id ID #REQUIRED
    name CDATA #IMPLIED>
<!ELEMENT actions (action*)>
<!ELEMENT action EMPTY>
<!ATTLIST action
    id ID #REQUIRED
    distance CDATA #REQUIRED
    preActionTurnAngle CDATA #REQUIRED
    postActionTurnAngle CDATA #REQUIRED>
<!ELEMENT featureActionFeatures (featureActionFeatureTriple*)>
<!ELEMENT featureActionFeatureTriple EMPTY>
<!ATTLIST featureActionFeatureTriple
     id ID #REQUIRED
     startFeatureId IDREF #REQUIRED
     actionId IDREF #REQUIRED
     endFeatureId IDREF #REQUIRED>
]>
```

Beispiel:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<featureActionFeatureKB>
    <features>
        <feature id="0" name="Van Gogh"/>
        <feature id="1" name="Edward Hopper - Nighthawks"/>
        <feature id="2" name="Dilbert"/>
    </features>
    <actions>
        <action id="0" distance="12"
            preActionTurnAngle="40" postActionTurnAngle="100"/>
        <action id="1" distance="12"
            preActionTurnAngle="20" postActionTurnAngle="20"/>
    </actions>
    <featureActionFeatures>
        <featureActionFeatureTriple id="0" startFeatureId="0"
            actionId="0" endFeatureId="1" />
        <featureActionFeatureTriple id="1" startFeatureId="1"
            actionId="1" endFeatureId="0" />
        <featureActionFeatureTriple id="2" startFeatureId="1"
            actionId="0" endFeatureId="2" />
    </featureActionFeatures>
</featureActionFeatureKB>
```

Statistik:

```
<!DOCTYPE featureActionFeatureStatistic [

<!ELEMENT featureActionFeatureStatistic
    (featureActionFeature*)>

<!ELEMENT featureActionFeature (ibigNode*)>
<!ATTLIST featureActionFeature
    id ID #REQUIRED>

<!ELEMENT ibigNode EMPTY>
<!ATTLIST ibigNode
    id ID #REQUIRED
    occurrences CDATA #IMPLIED>
]>
```

Beispiel:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

```
<featureActionFeatureStatistic>
    <featureActionFeature id="0">
        <ibigNode id="8" occurrences="3"/>
        <ibigNode id="2" occurrences="8"/>
        <ibigNode id="6" occurrences="4"/>
    </featureActionFeature>
    <featureActionFeature id="1">
        <ibigNode id="1" occurrences="1"/>
    </featureActionFeature>
    <featureActionFeature id="2">
        <ibigNode id="4" occurrences="21"/>
        <ibigNode id="5" occurrences="1"/>
    </featureActionFeature>
</featureActionFeatureStatistic>
```

Um auf die Daten zugreifen zu können, bietet die Wissensbasis die Methoden **`getAction()`**, **`getFeature()`**, **`getFAF()`** und **`getFAFsByStartFeatureAndAction(startFeature, action)`**. Die letzte dieser Methoden gibt eine Liste von sensomotorischen Tripeln zurück, die im angegebenen *Start View* (**`startFeature`**) und der Aktion (**`action`**) übereinstimmen. Anhand dieser Tripel kann das Inferenzmodul den Explorationsschritt mit dem wahrscheinlich größten Informationszuwachs berechnen. Zu diesem Zeitpunkt ist der *Target View* noch nicht bekannt, und daher wird der Informationswachs für alle bekannten *Target Views* berechnet.

Die Methoden **`addFeature`**, **`addAction`**, **`addFAF`** fügen der Wissensbasis neue Elemente hinzu, welche bei einer nächsten Speicherung dann auch berücksichtigt werden.

Map

```
agentctrl.map.mapelements.py
agentctrl.map.mapinfoparser.py
agentctrl.map.mapmanager.py
```

Neben den Daten, die in der Wissensbasis verwaltet werden, gibt es noch das Zusatzuwissen bzgl. der Umgebungen. Wie in Kapitel 4.2 beschrieben, bekommt das System eine Liste von *Viewpoints* für jede Umgebung. Diese *Viewpoints* und deren Verknüpfung, d.h. ob der eine *Viewpoint* in unmittelbarer Umgebung von dem anderen ist, werden vom **`MapManager`** verwaltet. Dieser liest das Zusatzwissen mittels des Parsers **`MapInfoHandler`** aus einer XML-Datei ein. Diese hat folgendes Format:

```
<!DOCTYPE hierarchy [
<!ELEMENT map (environment+)>
<!ELEMENT environment (features+, paths+)>
<!ATTLIST environment
    name CDATA #REQUIRED>
<!ELEMENT features (feature+)>
<!ELEMENT feature (name, location, lookat)>
<!ATTLIST feature
    id ID #REQUIRED>
<!ELEMENT name (#PCDATA)>
<!ELEMENT location (#PCDATA)>
<!ELEMENT lookat (#PCDATA)>
```

```
<!ELEMENT paths (path+)>
<!ELEMENT path EMPTY>
<!ATTLIST path
   start IDREF #REQUIRED
   end IDREF #REQUIRED>
]>
```

Beispiel:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<map>
   <environment name="Museum">
      <features>
         <feature id="1">
            <name>Nighthawks</name>
            <location>[-6.279, -0.713, -1.2]</location>
            <lookat>[-11.532, -0.687, -0.037]</lookat>
         </feature>
         <feature id="2">
            <name>Dilbert</name>
            <location>[0.721, 7.287, -1.2]</location>
            <lookat>[0.069, 11.575, 0.182]</lookat>
         </feature>
         <feature id="3">
            <name>Van Gogh</name>
            <location>[6.721, 0.287, -1.2]</location>
            <lookat>[11.553, 0.362, -0.057]</lookat>
         </feature>
         <feature id="4">
          <name>Corcovado</name>
          <location>[0.721, -7.287, -1.2]</location>
          <lookat>[0.138, -11.561, 0.0]</lookat>
         </feature>
      </features>
      <paths>
         <path start="1" end="2"/>
         <path start="1" end="3"/>
         <path start="2" end="1"/>
         <path start="2" end="4"/>
         <path start="3" end="1"/>
         <path start="4" end="2"/>
      </paths>
   </environment>
</map>
```

4.7.5 SMX Kernmodul

```
agctrl2.py
agentctrl.agentcontroller.py
agentctrl.vrcomserver2.py
ibig.mapibigconnector.py
```

Im Kernmodul laufen alle Fäden zusammen, und die Systemsteuerung ist hier implementiert. Die zentrale Klasse **`AgentController`** hat Zugriff auf die VR-Umgebung über **`VRAgentConnector`**[81], auf die Szenenanalyse über **`OkusysComClient`**[82], auf die Wissensbasis **`Knowledgebase`**[83] und das Inferenzmodul **`IBIG`**[84].

81 siehe Kapitel 4.7.2, S. 67
82 siehe Kapitel 4.7.3, S. 70
83 siehe Kapitel 4.7.4, S. 70
84 siehe Kapitel 4.7.1, S. 66

Initialisierung

Die Initialisierung beginnt mit der Verbindug zur VR-Umgebung und zum Szenenanalysesystem. Anschließend wird die Wissensbasis sowie das Kartenzusatzwissen geladen und das Inferenzmodul initialisiert.

Ist dies geschehen, muss die zu explorierende Umgebung mit `AgentController.switchEnvironment(name, feedback)` ausgewählt werden[85]. Bei `feedback` handelt es sich eine optionale Rückmeldung für das System, welchen Raum es gerade exploriert, so dass es seine Fähigkeit Umgebungen zu klassifizieren verbessern kann. Ohne das Feedback versucht der *SMX* seine Umgebung zu bestimmen ohne weiter zu lernen. Die zur Verfügung stehenden Umgebungen und Feedbacks sind über die Methoden `getEnvironmentNames()` und `getFeedbacks()` verfügbar.

Ist die Umgebung ausgewählt, wird aus der Menge der *Viewpoints* einer zufällig ausgewählt, welches dann der Startpunkt des Agenten für die Exploration ist (`_initEnvironment()`, `_getInitialView()`).

Exploration

Mit `AgentController.calcStep()` und `_getNextView()` wird der nächste Explorationsschritt des Agenten berechnet. Die möglichen Schritte werden vom `MapManager` mit `getReachableViews()` ermittelt. Aus dieser Menge werden zunächst die Unbekannten herausgefiltert. Dafür schaut der `MapIBIGConnector` für welche der Schritte bzw. deren *Start View* und Aktion es ein sensomotorisches Tripel in der Wissensbasis gibt. Für diese bereits bekannten[86] sensomotorischen Tripel wird nun der Informationszuwachs berechnet, und der Explorationsschritt, dessen sensomotorische Eigenschaft den höchsten Zuwachs verspricht, wird ausgewählt. Gibt es für einen Schritt keine bekannten sensomotorischen Tripel, wird einer zufällig ausgewählt.

Mit `AgentController.makeStep()` wird der Agent instruiert den ausgewählten Explorationsschritt auszuführen. Anschließend kann der *Target View* mit finishStep analysiert und berechnet werden, so dass das neu ermittelte sensomotorische Tripel komplett ist und die Evidenzverteilung neu berechnet werden kann. Diese Abfolge wird nun so lange wiederholt bis der *SMX* für eine Raumhypothese genügend Evidenz gesammelt hat.

Einblick in den Ablauf

Einen Einblick in den Ablauf des Systems bieten die Methoden `getCurrentHypothesis()`, `getExpectedNextView()`, `getCurrentView()` und `getNextStepDescription()`. Mit getIBIGHierarchyImage() bekommt man eine mit GraphViz und PyDot[87] erzeugte Visualisierung der Raumhierarchie und der aktuellen Evidenz für die einzelnen Hypothesen (siehe).

85 (siehe Abb. 29, S. 50)
86 bekannt bzgl. des *start Views* und der Aktion
87 siehe Anhang

Schließlich gibt es noch die Möglichkeit die erlernten sensomotorischen Eigenschaft und deren Evidenz für bestimmte Umgebungen mit **`saveKB()`** zu speichern.

4.7.6 GUI

```
gui.agentform2.py
gui.agentgui2.py
gui.waitform.py
gui.nextenv.py
```

Die grafische Nutzerschnittstelle (GUI) basiert auf dem Widgettoolkit *Qt*[88] bzw. dessen Pythonwrapper *PyQt*[89]. Die Dialoge wurden mit Hilfe des grafischen Werkzeugs *Qt Designer* entworfen (siehe Anhang). Die XML-Repräsentation der so erzeugten Dialoge wurden mittels des *Qt user interface compilers (uicc-qt3)*[90] in Pythonklassen umgewandelt. In Klassen, die davon geerbt haben, konnte dann die eigentliche Programmlogik implementiert werden.

88 www.trolltech.com/ (siehe Anhang)
89 www.riverbankcomputing.co.uk/pyqt/ (siehe Anhang)
90 Programm aus dem PyQt-Paket

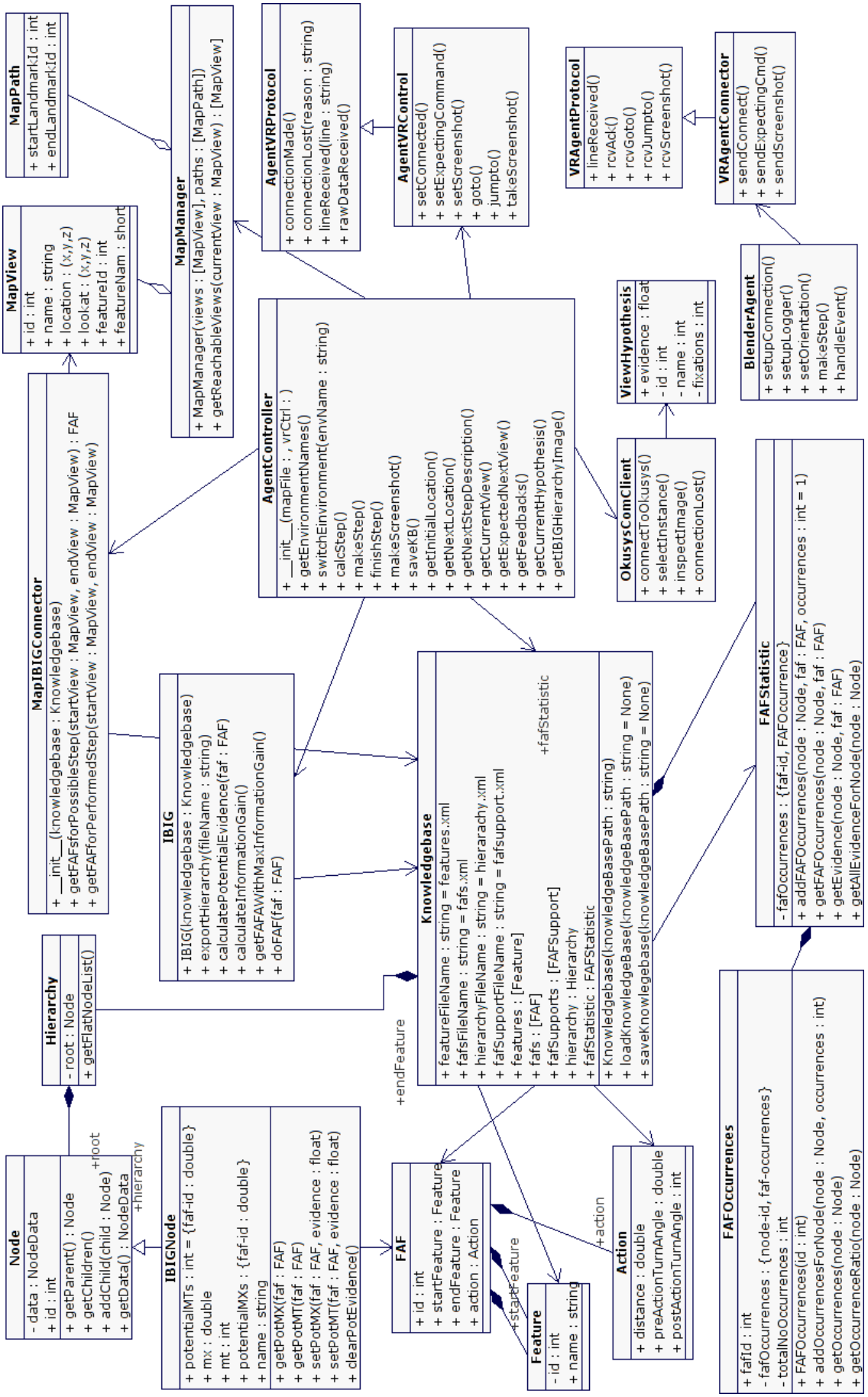

Abb. 43: Klassendiagramm vom Sensorimotor Explorer (Auswahl der wichtigsten Klassen)

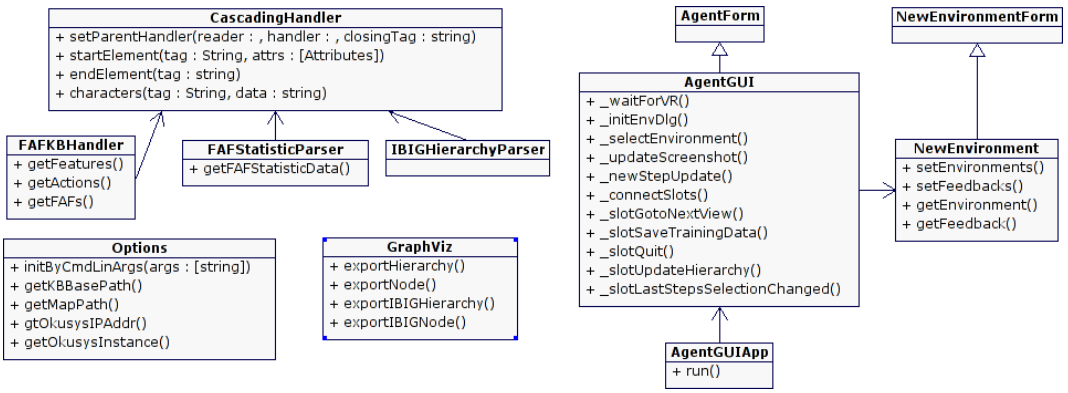

Abb. 44: UML-Diagramm der Klassen für das GUI, den XML-Parser und die Programmoptionen

4.8 Bewertung und weitere Entwicklung

Dieses Kapitel umfasst eine abschließende Bewertung des Systems sowie vielversprechende Schritte der Weiterentwicklung.

4.8.1 Sensomotorische Repräsentation

Vor einer generellen Bewertung der verwendeten sensomotorischen Repräsentation wird sie zunächst mit den beschriebenen Modellen aus dem letzten Kapitel beschrieben.

Ein Vergleich mit SSH, Route Graph und View Graph

In der *SSH* von Kuipers sind auf dem *Causal Level* auch Sequenzen von *Views* und Aktionen definiert, welche jedoch eine leicht abweichende Bedeutung haben: *Views* basieren auf einer anderen Sensorik, nämlich omnidirektionaler Abstandssensoren, und Aktionen sind über sensomotorische *Control Laws* anstatt durch eine Sequenz »Rotation - Geradlinige Bewegung - Rotation« definiert.

Bzgl. des *View Graphs* fällt der Vergleich ähnlich aus. Hier repräsentieren die *Views* Panoramabilder der Umgebung und Aktionen sind über einen Homingalgorithmus mit Referenzbildern, oder Geradlinige Bewegungen in nicht exploriertes Gebiet definiert.

Der *Route Graph* bzw. das dort beschriebene Routensegment ist ein abstrakteres Modell, welches mehr oder weniger auf den *View Graph*, die *SSH* und das sensomotorische Tripel des *SMX* passt.

Wie »sensomotorisch« ist die verwendete Repräsentation?

Ein deutliches Problem der im *SMX* verwendeten Repräsentation ist die Abhängigkeit von vordefinierten *Viewpoints* und die Definition einer Aktion.

Ein sensomotorischen Repräsentationen zugeschriebener Vorteil ist die Fähigkeit aus dem Ziel, zu einem in der Repräsentation vertretenen Ort zu navigieren, zuverlässig eine konkrete Aktion abzuleiten. Dies ist ohne Probleme möglich, da die Aktion in der Repräsentation bereits direkt enthalten ist.

Nun arbeitet der *SMX* momentan in einer diskreten Umgebung mit festen *Viewpoints*, in welcher eine mehr oder weniger geometrische Beschreibung einer Aktion durch zwei Winkel und einer Distanz ausreicht, um zuverlässig zu einem Ort zu gelangen, sofern keine Hindernisse zwischen *Viewpoints* liegen.

Würde der *SMX* nun in einer kontinuierlichen Umgebung mit simulierter Physik navigieren, ergäbe sich durch leicht ungenaue odometrische Informationen von Aktion zu Aktion eine immer größer werdende Fehlabschätzung der eigenen Position. Hier sind die *Control Laws*, welche über eine direkte Relation von sensorischen und motorischen Informationen definiert sind, der Idee von Sensomotorik um einiges näher. Durch diese direkte Kopplung vermindert Kuipers das Problem des sich aufsummierenden Fehlers bei der Abschätzung der eigenen Position beträchtlich.

In gewisser Weise setzt die Repräsentation des *SMX* ein *Sensory* und *Control Level* im Sinne der *SSH* bereits voraus; oder er nutzt zumindest eine abstraktere Definition von (Senso-)Motorik und Aktionen. Ein großer Vorteil des *SMX* ist jedoch die deutlicher ausgearbeitete Strategie Aktionen auszuwählen. Das System kann in einem bestimmten Zustand auf Explorationserfahrungen bzw. erlernte Zusammenhänge zurückgreifen, um eine Aktion vorzuschlagen, welche die maximale Information für die Aufgabe der Lokalisation ergibt[91]. Um diese Strategie auf eine »echte« sensomotorische Grundlage zu stellen bedarf es jedoch einiger Erweiterungen.

Weiterentwicklung

Wenn man den *SMX* nun in einer kontinuierliche Umgebung mit simulierter Physik und ohne vordefinierte *Viewpoints* operieren lässt, ergeben sich einige Herausforderungen.

Wenn man dem Beispiel von Kuipers folgt, werden *Control Laws* benötigt, die jedoch mit einem visuellen Sensor mit beschränktem Sichtfeld arbeiten. Wünschenswert ist dabei, dass sie nicht wie beim *View Graph* auf Referenzbilder der *Target Views* angewiesen sind. Hierbei handelt es sich um eine komplizierte Aufgabe, da die visuelle Sensorik im Gegensatz zu Abstandssensoren, noch keine explizite Rauminformation enthält.

Ein weiteres Problem ist die Ermittlung der möglichen Aktionen an einem bestimmten Ort (*View*). Auch hier stellt einen die visuelle Sensorik vor größere Herausforderungen, als die von Kuipers verwendete.

Mit diesen beiden Fähigkeiten sollte jedoch eine zuverlässige Navigation mit der bestehenden Explorationsstrategie auch in komplexeren kontinuierlichen Umgebungen möglich sein, welches das System einer etwaigen Implementierung in einen Roboter ein wenig näher bringt.

4.8.2 Propagieren von Unsicherheit zwischen Granularitätsstufen

Der *SMX* verwaltet momentan auf zwei Ebenen unsicheres Wissen, ohne den Grad der Unsicherheit von einer Ebene auf die andere zu propagieren. Das visuelle Analysemodul berechnet für einen Schnappschuss eine Hypothese, um was es sich handelt und wie groß die Unterstützung dieser Hypothese ist. Es gibt z.B. zurück, dass es sich bei einem Schnappschuss mit 55%-iger Unterstützung um das Gemälde »Nighthawks« von Edward Hopper handelt.

Diese Information wird auf der Ebene der Exploration nicht mehr genutzt. Dort wird einfach der *View* mit der höchsten Wahrscheinlichkeit als »sicher« angesehen.

Ein Beispiel (siehe Abb. 45): Der Ausgangspunkt ist, dass nach eingehender Exploration auf der Ebene der Raumhypothesen 80% für ein »Büro« spricht und 20% für eine »Küche«. Wenn nun das sensomotorische Tripel [»Tür«, »2 Meter geradeaus und nach rechts drehen«, »Badewanne«] gefunden wird, das die bisher nicht beteiligte Hypothese »Badezimmer« unterstützt, die Unterstützung für den *View* Badewanne jedoch nur bei 35% liegt, stellt sich die

91 Die Information ist nur maximal, wenn nach Ausführung einer Aktion der erwartete *View* erreicht wird.

Frage, wie damit umgegangen werden soll. In der bisherigen Implementierung wird die komplette Evidenz einfach der Hypothese »Badezimmer« zugeschlagen, wobei sich doch die Frage stellt, ob die Bildanalyse nicht einfach fehlgeschlagen hat und für den *View* nicht eher die mit 30% unterstützte Hypothese »Schreibtisch« zutrifft.

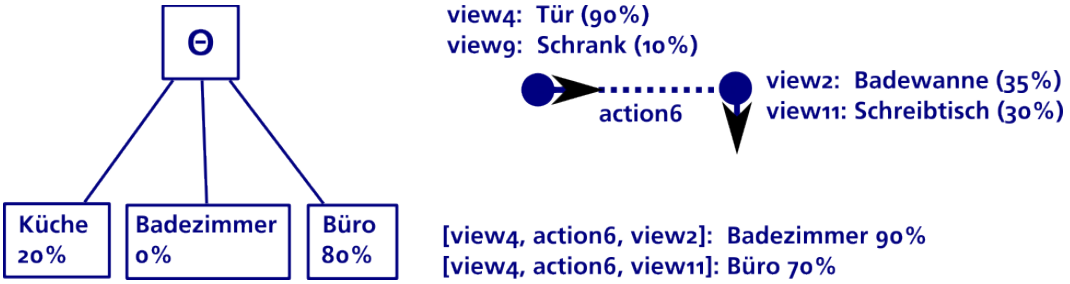

Abb. 45: Die sensomotorische Eigenschaft des Explorationsschrittes (rechts oben) ist besonders bzgl. des *target Views* mehrdeutig. Der ein wenig besser unterstützte *View* »Badewanne« widerspricht der momentanen Evidenzverteilung auf Raumebene.

Eine einfach zu implementierende Möglichkeit ist die direkte Einbeziehung der *View*-Wahrscheinlichkeit in die Unterstützung einer sensomotorischen Eigenschaft für eine Raumhypothese. Im obigen Beispiel unterstützt das Tripel *[View4, Action6, View2]*, das Badezimmer mit 90%. Bezieht man die (Un-)Sicherheit der *Views* (Tür 90%, Badewanne 35%) mit ein erhält man nur noch eine Unterstützung von

$$90\% \cdot \left(\frac{90\% + 35\%}{2} \right) = 56{,}25\%$$

für das Badezimmer. Der Rest wird als Unwissenheit deklariert und an Θ vergeben.

Wenn das Analysemodul für die visuellen Schnappschüsse nicht nur die wahrscheinlichste Hypothese zurückgibt, ergeben sich weitere Möglichkeiten. Wie im Beispiel bei einer gewissen Unklarheit das am besten passende Tripel zu wählen, ist problematisch, da damit forciert wird, dass eine der momentanen gut unterstützten Hypothesen die gesuchte ist. Überzeugender ist eher eine Einbeziehung mehrerer Tripel, wenn einer oder beide *Views* eine ausreichend ähnliche Unterstützung haben. Damit würde im obigen Beispiel nach einem Explorationsschritt noch eine zweite sensomotorische Eigenschaft *[View4, Action6, View11]* in die Berechnung mit eingehen.

5 Schlußwort

In dieser Arbeit wurden Indizien aufgezeigt, die für eine sensomotorische Repräsentation von räumlichen Konfiguration im Gehirn von Menschen und Tieren sprechen. Diese wurden zum einen mit der klassischen Sicht der *Cognitive Map* und neurobiologischen Erkenntnissen und zum anderen mit Repräsentationsformen aktueller Explorationssysteme in Beziehung gesetzt. Hierbei hat sich der Ansatz der *Spatial Semantic Hierarchy* von *Kuipers* als besonders interessant erwiesen, welcher sich in einigen Punkten jedoch noch deutlich an der klassischen Robotik und nicht an biologischen Systemen orientiert.

Der praktische Teil der Arbeit umfasst die Entwicklung des Explorationssystems *Sensorimotor Explorer (SMX)* für eine VR-Umgebung basierend auf den erarbeiteten Ergebnissen und dem Szenenanalysesystem *Okusys*. Der *SMX* besitzt eine visuelle Sensorik mit einer biologisch motivierten Bildanalyse und ist in der Lage, seine Umgebung durch eine lokale Exploration zu klassifizieren, d.h. er erfüllt damit die Aufgabe der Lokalisation. Umgebungen werden anhand sensomotorischer Eigenschaften identifiziert, die während der Exploration gesammelt werden. Die Strategie, nach der die Umgebung exploriert wird, ist durch *IBIG* bestimmt; ein auf der Dempster-Shafer Theorie aufbauendes Inferenzsystem, welches sich am Kriterium des maximalen Informationszuwachses orientiert.

Ein deutlicher Schwachpunkt des bestehenden Systems ist die doch recht abstrakte Implementierung von Sensomotorik. Der *SMX* ist auf eine diskrete Umgebung mit vordefinierten *Viewpoints* angewiesen, so dass wahrscheinlich eine der naheliegendsten Erweiterungen die Entwicklung eines *Control Levels* nach dem Vorbild der *SSH* für die verwendete visuelle Sensorik ist. Damit würde aus der Aktion in den sensomotorischen Tripeln (»rotieren, laufen, rotieren«) ein *Control Law*, also ein eine Art sensomotorische »Bewegungsregel«.

Es stellt sich noch die Frage, inwieweit sich *Control Laws* für die doch recht komplexe visuelle Sensorik in Verbindung mit nicht trivialen Umgebungen konstruieren lassen. Interessant ist hierfür sicherlich, den aktuellen Forschungsstand über die Kodierung und Verarbeitung von (senso-)motorische Abläufen in biologischen Systemen einzubeziehen.

Eine weitere beschriebene Erweiterung ist eine Methode für einen verbesserten Umgang mit verrauschten Sensordaten. Diese ermöglicht das Propagieren von Unsicherheit zwischen den Granularitätsstufen, so dass eine evtl. Ambiguität im Ergebnis der Sensordatenanalyse auf der höheren Ebene der Raumhypothesen berücksichtigt wird.

Mir ist bewusst, dass ich die Themenbereiche Sensomotorik und Raumkognition nur in Ansätzen behandelt habe. Dennoch hoffe ich sowohl die Motivation als auch die Umsetzung einer Integration des motorischen Aspekts in die Repräsentation von Raum glaubhaft dargestellt zu haben. Im Hinblick auf die Entwicklung von Explorationssystemen mit einer zu biologischen Orga-

nismen vergleichbaren Robustheit, kann eine sensomotorische Raumrepräsentation vielleicht einen entscheidenden Beitrag leisten.

6 Danksagungen

Mein Dank gilt allen Personen, die mich bei den Arbeiten zu dieser Diplomarbeit unterstützt haben:

Prof. Dr. Kerstin Schill und Dr. Christoph Zetzsche danke ich für die vielen Ideen und die unkomplizierte Begleitung der Arbeit, und Dr. Thomas Barkowsky danke ich für die freundliche Hinweise zu den Themen *Mental Imagery* und der *Cognitive Map*. Dr. Christoph Flores und Thusitha Parakrama möchte ich für die Unterstützung bei der »Entschlüsselung« und der Erweiterung des Szenenanalysesystems *Okusys* und für die konstruktive Kritik danken.

Und nicht zuletzt danke ich Christopher Galbraith und Daniel Deutsch für die Zerstreuung und Ermutigung während der Diplomarbeit sowie für das Korrekturlesen.

Anhang

A Allgemeines zur Implementierung

Dieses Kapitel beschreibt bei der Implementierung des *Sensorimotor Explorers* verwendete Bibliotheken und Techniken.

A.1 Unittests

Besonders zu Beginn der Entwicklung und für kritische Komponenten wurden sog. *Unittests* entwickelt. Für den Zeitpunkt, an dem das komplette System zum Test von Einzelkomponenten noch nicht zur Verfügung steht, bieten *Unittests* eine bequeme Möglichkeit die Funktionsfähigkeit einzelner Module sicherzustellen. Des weiteren geben sie nach einem *Refactoring* schnell Aufschluss über die Auswirkungen der Modifikationen auf das ganze System. Besonders hilfreich waren die Tests auch bei der Entwicklung der Komponenten, welche etwas komplexere mathematische Funktionen implmentieren; d.h. kontrekt: Geometrische Operationen für die VR-Umgebung und *Dempster-Shafer Theorie* und *IBIG*.

Für die Implementierung der Unittests wurde auf das Framework **`unittest`** aus der Standardbibliothek von Python zurückgegriffen.

A.2 Python

Für die Implementierung des *Sensorimotor Explorers* wurde die Programmiersprache *Python*[92] verwendet. Die Vorteile dieser objektorientierten Skriptsprache sind ihre sehr große Standardbibliothek und die große Anzahl an verfügbaren, freien Bibliotheken, sowie ihre dynamische Typisierung.

Ersteres ermöglicht eine Konzentration auf die Implementierung der zentralen Programmlogik. Es stellt z.B. kein großer Aufwand dar die Hypothesenhierarchie mit den aktuellen Evidenzen zu visualisieren und in der *GUI* anzuzeigen, da es mit *PyDot* und *Graphviz* eine einfache Schnittstelle und Bibliothek zur Visualisierung von beliebigen Graphen gibt.

Die dynamische Typisierung ist bei der Entwicklung eine Prototypen von Vorteil, da feste Schnittstellen sich meist erst im Laufe der Entwicklung ergeben. Wenn die Entwicklung des *SMX* einen stabileren Stand erreicht, lohnt es sich evtl. Teile des Programms in eine strenger typisierte Sprache wie z.B. *Java* zu portieren. Diese Sprache hätte unter anderem den Vorteil, dass mit *Jython*, ein *Python*-Interpreter in *Java*, ein weicher Übergang möglich ist.

A.3 PIL

Für die Vorverarbeitung der visuellen Schnappschüsse aus der VR-Umgebung für das visuelle Analyseprogramm wurde die *Python Imaging Library* verwendet[93]. Sie erlaubt eine einfache Skalierung und Konvertierung der Bilder in das gewünschte Format.

92 http://www.python.org
93 http://www.pythonware.com/products/pil/

A.4 XML, SAX-Parser

Zur Persistierung von Daten, wie z.B. die Statistik und die Hypothesenhierarchie, wurden ausschließlich Dateien im XML-Format[94] verwendet. Zum Einlesen dieser Dateien wurde der *SAX-Parser*[95] aus der Standardbibliothek von Python verwendet. *SAX* steht für *Simple API for XML* und ist eine standardisierte Schnittstelle für XML-Parser. *SAX-Parser* arbeiten über *Callback*-Funktionen und zeichnen sich im Gegensatz zu *DOM-(Tree-)Parsern*[96] meist durch einen niedrigen Bedarf an Speicher- und Rechenkapazität aus.

A.5 Qt, QtDesigner, PyQt

Die grafischen Nutzerschnittstellen (GUIs) basieren auf dem *Widgetset Qt*[97] bzw. dessen *Python*-Wrapper *PyQt*[98]. Dieses *Widgetset* ist u.a. Grundlage des weit verbreitete Desktopumgebung *KDE*[99].

Die Dialoge wurden mit Hilfe des visuellen Entwicklungswerkzeugs *Qt Designer*[100] (siehe Abb. 46), erstellt und dann mit Hilfe des *user interface compilers* von *PyQt*, in eine für *Python* verwendbare Form gebracht.

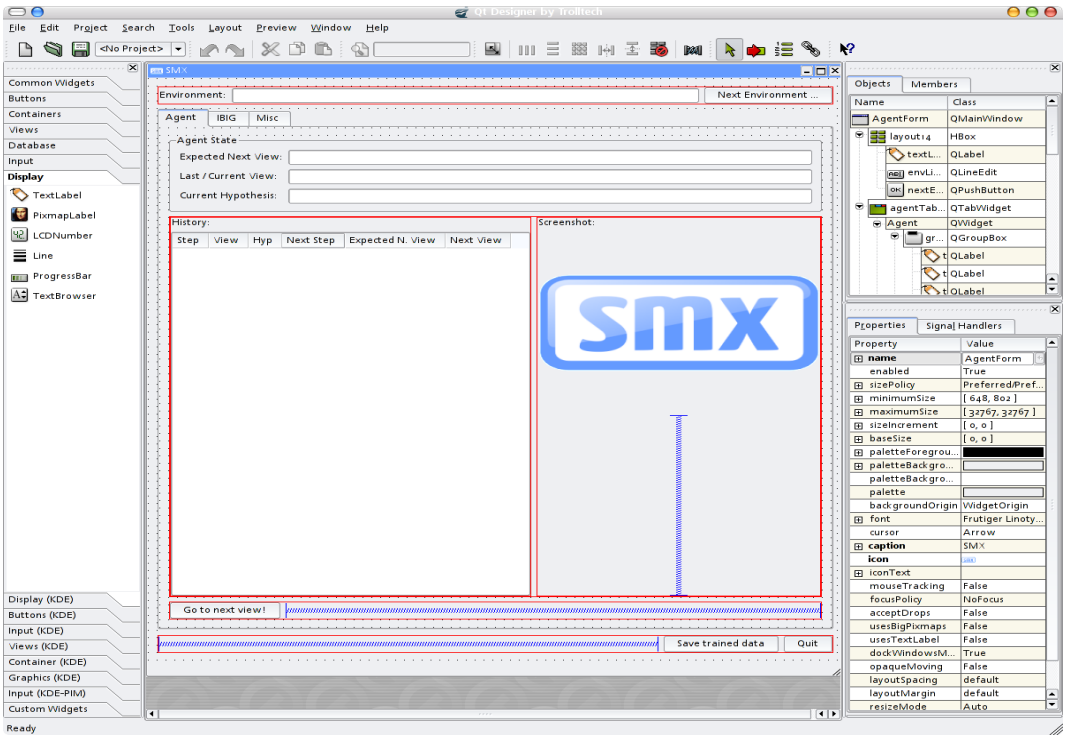

Abb. 46: Screenshot vom *Qt Designer*

94 http://www.w3.org/TR/2004/REC-xml-20040204/
95 http://www.saxproject.org/
96 http://www.w3.org/DOM/
97 http://www.trolltech.com/
98 http://www.riverbankcomputing.co.uk/pyqt/
99 http://www.kde.org/
100 http://www.trolltech.com/products/qt/designer.html

A.6 Graphviz, PyDot

Für die Visualisierung der Hypothesenhierarchien und der Evidenzverteilung wurde *Graphviz*[101] verwendet; hierbei handelt es sich um ein Softwarepaket zur Visualisierung von beliebigen Graphen. Aus diesem Paket wurde *dot*, bzw. der Pythonwrapper PyDot[102], verwendet, da es sich besonders zur Darstellung von Hierarchien eignet (siehe Abb. 47).

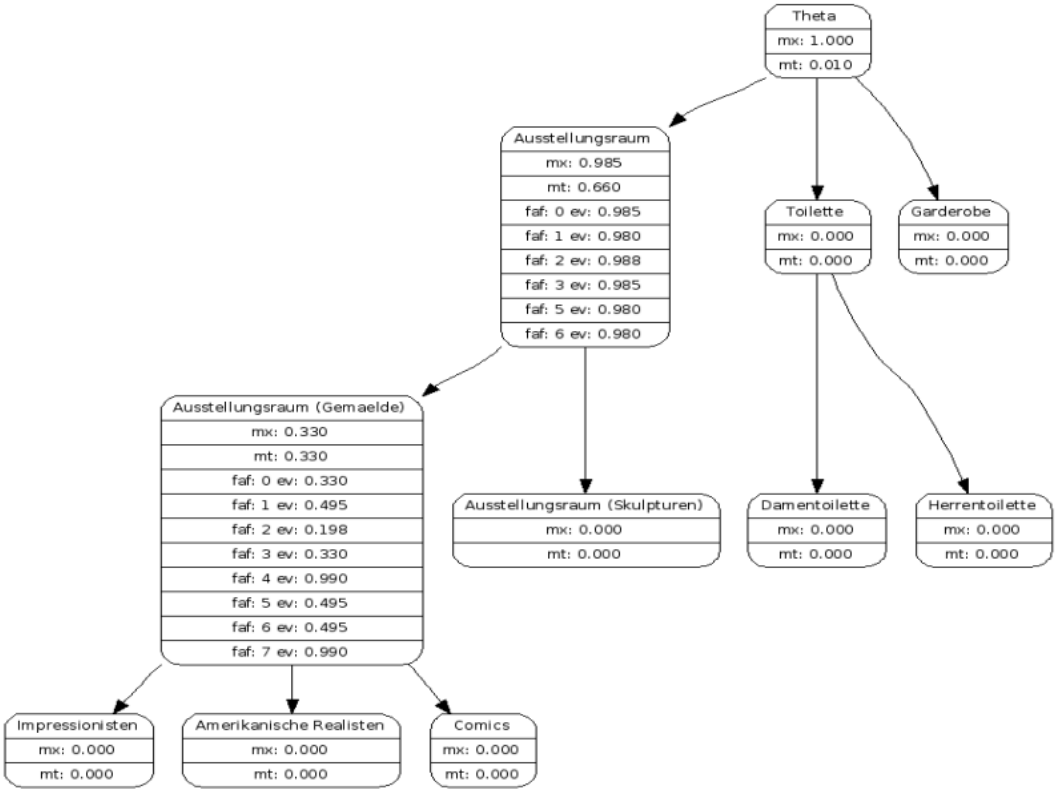

Abb. 47: Mit Graphviz und PyDot erzeugte Darstellung der Raumhypothesenhierarchie mit unterstützenden sensomotorischen Eigenschaften (*faf*) und aktueller Evidenzverteilung (*mx, mt*).

A.7 Twisted

Für einen Großteil der Rechnernetzkommunikation wurde auf das auf Python basierende Framework *Twisted*[103] zurückgegriffen. Dieses ereignisgesteuerte Framework forciert eine Trennung zwischen der logischen Protokoll- und der Transportschicht.

101 http://www.graphviz.org/
102 http://dkbza.org/pydot.html
103 http://twistedmatrix.com/trac

B CD-Inhalt

Die beigefügte CD enthält den Quellcode des *SMX* , sowie zwei VR-Umgebungen für die *Blenderengine*.

Der Quellcode befindet sich im Ordner **src** ist in hierarchisch angeordnete Module unterteilt:

- **agctrl2**
- **agentctrl**
  - **agentcontroller**
  - **map**
    - **mapelements**
    - **mapinfoparser**
    - **mapmanager**
  - **okusyscomclient**
  - **spatial**
    - **spatialEntities**
  - **util**
    - **aexceptions**
    - **cascadingHandler**
    - **exceptionmsg**
    - **graphviz**
    - **hierarchy**
    - **mathutil**
    - **options**
    - **simpleclient**
    - **synchronize**
  - **vrcomserver2**
- **gui**
  - **agentgui2**
  - **agentform2**
  - **nextenvform**
  - **waitform**
- **ibig**
  - **ibig**
  - **kbParser**
    - **actionParser**
    - **fafkbParser**
    - **fafParser**
    - **fafStatisticParser**
    - **featureParser**
    - **hierarchyParser**
  - **knowledgebase**
  - **mapibigconnector**
  - **statistic**

- **test**

- **okusysmock**
- **testAgentctrl**
- **testFAFKBParser**
- **testFAFStatistic**
- **testHierarchy**
- **testIBIG**
- **testKnowledgebase**
- **testmapibigconnector**
- **testMapinfoparser**
- **testmapmanager**
- **testMathutil**
- **testoptions**
- **vrcomserver2test**

Die VR-Umgebungen für Blender befinden sich im Ordner **vr**.

Ausführen des Systems

Zum Starten des Systems muss das Modul **agctrl2.py** ausgeführt werden und anschließend kann die *Blenderengine* mit einer VR-Umgebung gestartet werden. Der Aufruf **agctrl2.py --help** zeigt mögliche Kommandozeilenoptionen.

Systemvoraussetzungen

Das System wurde unter dem Betriebssystem *Debian Gnu/Linux*[104] entwickelt und benötigt folgende Pakete[105]:

- blender >= 2.41
- python2.3
- python2.3-qt3
- python2.3-qtext
- python2.3-kde3
- python2.3-twisted
- python2.3-unit
- python-imaging

104 http://www.debian.org
105 Es wurden Pakete aus dem Entwicklungszweig *testing* und *unstable* verwendet.

Abbildungsverzeichnis

Literaturverzeichnis

Anderson, John R., Kognitive Psychologie (1988), Heidelberg: Spektrum

Aurenhammer, Franz, Voronoi Diagrams - A Survey of a Fundamental Geometric Data Structure. (1991) ACM Computing Surveys 23 , 345-405

Banquet, J.P.; Gaussier, Ph.; Quoy, M.; Revel, A.; Burnod, Y., A Hierarchy of Associations in Hippocampo-Cortical Systems. (2005) Neural Computation 17 , 1339-1384

Bilkey, David K., In the Place Space. (2004) Science 305 , 1245 - 1246

Decety, Jean; Grèzes, Julie, Neural mechanisms subserving the perception of human actions. (1999) Trends in Cognitive Sciences 3 , 172-178

Duchon, Andrew P.; Warren, WilliamH.; Pack Kaebling, Lesli, Ecological Robotics. (1998) Adaptive Behaviour 6 , 473-508

Elfes, Alberto, Sonar-based real-world mapping and navigation.. (1987) IEEE Journal of Robotics and Automation 3 , 249-265

Frans, M.; Schölkopf, B.; Mallot, H.; Bülthoff, H., Learning View Graphs for Robot Navigation. (1998) Autonomous Robots 5 , 111-125

Fyhn, M.; Molden, S.; Witter, P.; Moser, E.; Moser M.-B., Spatial Representation in the Entorhinal Cortex. (2004) Science 305 , 1258 - 1264

Hafting, Torkel; Fyhn, Marianne; Molden, Sturla; Moser, May-Britt; Moser, Microstructure of a spatial map in the entorhinal cortex. (2005) nature 435 ,

Hommel, Bernhard; Müsseler, Jochen; Aschersleben, Gisa; Prinz, Wolfgang, The Theory of Event Coding: A framework for perception and action planning. (2001) Behavioral And Brain Sciences 24 , 849-j937

Jensen, Finn, Bayesian networks and decision graphs (2001), New York: Springer

Knauff, M.; Kassubeck, Jan.; Mulack, Thomas; Greenlee, Mark W., Cortical activation evoked by visual mental imagery as measured by fMRI. (2000) Neuroreport 11 , 3957-3962

Knierim, James J.; Kudrimoti, Hemant S.; McNaughton, Bruce L., Place Cells, Head Direction Cells, and the Learning of Landmark Stability. (1995) The Journal of Neuroscience 15 ,

Kosslyn, Stephen M., Image and Mind (1980), Cambridge, Massachusetts: Harvard University Press

Kosslyn, Stephen M., Image and Brain (1994), Cambridge, Massachusetts: MIT Press

Krieg-Brückner, B.; Frese, U.; Lüttich, K.; Mandel, C.; Mossakowski, T., Specification of an Ontology for Route Graphs. (2004) Lecture Notes in Computer Science 3343 , 390 ff

Kuipers, Benjamin, The 'Map in the Head' Metaphor. (1982) Environment and Behaviour 14 , 202-220

Kuipers, Benjamin, The Spatial Semantic Hierachy. (2000) Artificial Intelligence 119 , 191 - 233

Laeng, Bruno; Teodorescu, Dinu-Stefan, Eye scanpaths during visual imagery reenact those of perception[...]. () Cognitive Science 26 , 207-231

Leutgeb, S.; Leutgeb J.; Treves, A.; Moser, M.-B.; Moser, E., Distinct Ensemble Codes in Hippocampal Areas CA3 and CA1. (2004) Science Express 305 , 1295 - 1298

Mallot, H.; Bülthoff, H.; Georg, Philipp; Schölkopf, B.; Yasuhara, K., View-based cognitive map learning by an autonomous robot.. (1995) Proceedings ICANN 1995 2 , 381-386

Mast, Fred W.; Kosslyn, Stephen M., Eye movements during visual mental imagery. (2002) Trends in Cognitive Sciences 6 , 271-272

Möller, R., Perception Through Anticipation. A Behaviour-Based Approach to Vis. Perc.. (1999) S. 169-176; in Riegler, A.; Peschl, M.; von Stein, A. (Hrsg.): Understanding Representation in the Cog. Sciences, New York: Kluwer Academic

Moore, Tirin, Shape Representations and Visual Guidance of Saccadic Eye Movements. (1999) Science 285 , 1914-7

Moravec, H.P., Sensor fusion in certainty grids for mobile robots.. (1988) AI Magazine , 61-74

O'Keefe, J.; Conway, D. H., Hippocampal place units in the freely moving rat. (1978) Experimental Brain Research 31 , 573-590

Prinz, Wolfgang, A common coding approach to perception and action (1990), : Springer

Prinz, Wolfgang, Perception and Action Planning. (1997) European Journal of Cognitive Psychology 9 , 129-154

Remolina, Emilio; Kuipers, Benjamin, Towards a general theory of topological maps. (2004) Artificial Intelligence 152 , 47-104

Röfer, Thomas, Panoramic Image Processing and Route Navigation. (1998)

Roosendaal, Ton;Selleri, Stefano, The Official Blender 2.3 Guide (2004), : No Starch Press

Sanders, A.F., Stage analysis of reaction processes (1980), : North-Holland Publishing Company

Schill, Kerstin, Decision support systems with adaptive reasoning strategies. (1997) Lecture Notes In Computer Science 1337 , 417-428

Schill, Kerstin; Umkehrer, Elisabeth; Beinlich, Stephan; Krieger, Gerhard, Scene analysis with saccadic eye movements: Top-down and bottom-up modeling. (2001) Journal of Electronic Imaging 10 , 152-160

Schoelkopf, Bernhard; Mallot, Hanspeter, View-based cognitive mapping and path planning. (311-348) Adaptive Behavior 3 , 1995

Schubö, Anna, Interferenz von Wahrnehmung und Handlungssteuerung (1998),

Aachen: ShakerVerlag

Shafer, Glenn, A mathematical theory of evidence (1976), Princeton: Princeton University Press

Shortliffe, Edward H.; Gordon, Jean, A method for managing evidential reasoning in a hi. (1985) Artificial Intelligence 3 , 323-357

Siegel, A.W.; White S.H., The development of spatial representations of large-scale environments. (1975) Advances in Child Development and Behavior 10 , 9-55

Taube, Jeffrey S.; Muller, Robert U.; Ranck Jr., James B., Head-Direction Cells Recorded from the Postsubiculum [...] I.. (1990) The Journal of Neuroscience 10 , 420-435

Taube, Jeffrey S.; Muller, Robert U.; Ranck Jr., James B., Head-Direction Cells Recorded from the Postsubiculum [...] II.. (1990) The Journal of Neuroscience 10 , 436-447

Thrun, Sebastian, Learning Metric-Topological Maps for Indoor Mobile Robot Navigation. (1998) Artificial Intelligence 99 , 21-71

Tolman, Edward C., Cognitive Maps in Rats and Men. (1948) Psychological Review 55 , 189-208

Tversky, Barbara, Cognitive Maps, Cognitives Collages, and Spatial Mental Models. (1993) Spatial Information Theory 716 , 14-24

Watt, Alan H., 3D Computer Graphics (1999), : Adison Wesley

Werner, Steffen; Krieg-Brückner, Bernd;Herrmann, Theo, Modelling Navigational Knowledge by Route Graphs. (2000) , 295-316

Werner, Steffen; Krieg-Brückner, Bernd;Herrmann, Theo, Modelling Navigational Knowledge by Route Graphs (2000), : Springer-Verlag

Wexler, Mark; Kosslyn, Stephen M.; Berthoz Alain, Motor processes in mental rotation. (1998) Cognition 68 , 77-94

Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig ohne fremde Hilfe angefertigt habe. Alle Stellen, die ich wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Schriften übernommen habe, habe ich als solche kenntlich gemacht.

Bremen, 13. April 2006

Johannes Wolter