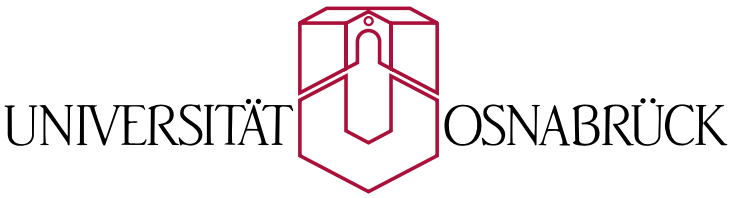


Universität Osnabrück
Institut für Kognitionswissenschaft

Masterarbeit

HRI für Rettungsroboter

Offene Evaluation, Re-Design und Spezifikation der Operatorschnittstelle eines RoboCup Rescue Roboters

Nadja Müller

30. Mai 2006

Erstgutachter: Professor Dr. Joachim Hertzberg
Zweitgutachter: Tobias Thelen, M.A.

Deutsche Zusammenfassung
Das Ziel dieser Masterarbeit ist ein Re-Design der grafischen Schnittstelle des RoboCup Rescue Team Deutschland1 zu entwerfen. Dafür wird das vorhandene Interface anhand von zwei Versuchspersonengruppen evaluiert. Die eine Gruppe besteht aus Experten, die mit diesem Interface seit langem arbeiten. Die andere Gruppe besteht aus mehreren Novizen, die noch nie mit dem System in Berührung gekommen sind und in mehreren Sitzungen eingelernt und danach getestet wurden. Ein aus der „Isometric“ erstellter Fragebogen soll dann Aufschluss über die notwendigen Informationen geben und die Vor- und Nachteile des jetzigen Interfaces preisgeben. Anhand von zusätzlichen Forschungsergebnissen und Design-Guidelines wird ein Re-Design entworfen und beschrieben.

English Abstract
The aim of this master thesis is to develop a re-designed graphical user interface for the RoboCup Rescue Team Germany1. Two groups of subjects evaluate the interface used currently. The first group consists of experts, who have been working on this interface for years. The other group is made up by naive subjects, that have never encountered the interface. Therefore they were trained in several sessions to use the interface and tested afterwards. On the basis of the “Isometric” a questionaire is created to gain knowledge about the advantages and disadvantages of the currently used interface. Considering additional research results and design guidelines, a re-design will be developed and described.

Danksagung

Besonders danke ich meinem Erstgutachter Prof. Dr. Joachim Hertzberg dafür, dass er diese interdisziplinäre Arbeit stark unterstützt und ermöglicht hat. Zudem danke ich für eine sehr freundliche und sorgsame Betreuung meiner Masterarbeit, die vielen interessanten Gespräche - vor und während der Arbeit - und die allzeit offene Türe, für Fragen und Sorgen jedweder Art.

Ebenso danke ich meinen Zweitgutachter Tobias Thelen für die bereitwillige Hilfe, interessante Anregungen und die Zeit, die er sich für mich genommen hat.

Kai Lingemann und Andreas Nüchter danke ich sehr für ihre Geduld, ihre Verbesserungsvorschläge und ihre freundschaftliche Unterstützung.

Ein Dank geht auch an Clemens Gruber für die konstruktive Kritik und richtungsweisende Ideen für die Evaluation. Für weitere fachliche und freundliche Unterstützung danke ich Dr. Jean Scholtz und Sven Laqua.

All den Versuchspersonen und Mitgliedern des Team Deutschland1 gilt besonderer Dank!

Ebenso danke ich meiner Familie, die ich hier oben im Norden sehr vermisse. Für die übermenschliche Unterstützung und Geduld, danke ich meinem Freund Florian aus tiefstem Herzen.

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

Einleitung

1.1 Motivation

Die Schwierigkeit der Telerobotik liegt generell in der präzisen Steuerung eines Roboters, zu dem kein Sichtkontakt besteht. Da in diesem Fall auch nur ein Computerinterface die vom Roboter gewonnenen Daten dem Operator vermitteln kann, muss diese Mensch-Computer-Schnittstelle entsprechend funktional gestaltet sein. Ein solches Interface kann als Filter, aber auch als Verstärker von empfangenen Sensordaten fungieren. Ziel dabei ist es, dem Operator bei der Entscheidungsfindung zu helfen, indem ihm ein „sinnvoller“ Überblick über die Situation vor Ort gegeben wird. Das Potential eines Systems hänge sehr stark von den Fähigkeiten des Operators und der Qualität der Benutzerschnittstelle ab[1], wie Fong et al. in [31] meinen.

USAR Wettkämpfe

Ein sehr spezielles und schwieriges Anwendungsgebiet der Telerobotik ist die RoboCup Rescue League [12]. Die (noch semi-)autonome Erkundung eines Katastrophengebietes durch einen Roboter ist ein wichtiges und notwendiges Betätigungfeld für die Forschung. Diese USAR-Wettkämpfe[2] wurden von der NIST (National Institute of Standards and Technology [6]) erfunden, um ein Testfeld haben, auf das sie sich in ihrer Arbeit beziehen können [27, 48, 49]. Diese Testfelder sollen möglichst realistisch ein (städtisches) Szenario nach einer Katastrophe nachahmen. Die Roboter sollen in diesen Szenarien sicher navigieren bzw. navigiert werden und dabei Opfer finden. Diese vermeintlichen Opfer werden durch Schaufensterpuppen dargestellt. Um Opfer sinnvoll zu lokalisieren, ist eine Karte von Nutzen, die bei der Exploration des Testfeldes entstehen soll bzw. kann. Wie diese Karte entsteht, liegt im Ermessen des jeweiligen Teams. Es gibt drei verschieden schwierige Testfelder. Ein gelbes Testfeld soll ein wenig beschädigtes Gebäude darstellen. Die Opfer darin sind eher einfach aufzufinden. Schwieriger wird es in

[1] Zitierte englische Texte wurden möglichst sinngerecht ins Deutsche übersetzt. In dieser Arbeit wird auf englische Zitate im Allgemeinen verzichtet und eine eigene deutsche Übersetzung tritt an deren Stelle, um den Lesefluss nicht unnötig zu stören.

[2] Urban Search and Rescue

einem orangenen Testfeld, wo der Roboter mehrere Stockwerke des Gebäudes befahren kann und auch mit einigem Schutt zu kämpfen hat. Ein rotes Testfeld ist auch eine Ruine eines mehrstöckigen Gebäudes, doch sind die Zugänge und Übergänge instabil. Zudem sind hier die Opfer nicht sichtbar ausgelegt, sondern müssen zum Teil durch andere Sensoren, wie z.B. durch eine Kohlendioxid-Sensor, gefunden werden.
Die Anzahl der richtig gefundenen und beschriebenen Opfer werden als Pluspunkte angerechnet. Minuspunkte dagegen gibt es bei Kollisionen mit der Gebäudestruktur und mit Opfern. Zudem werden die erlangten Punkte durch die Anzahl der Operatoren geteilt.
Die Zeit, die ein Operator zur Verfügung hat, beträgt zehn Minuten[3]. Durch diese enge Zeitgrenze entsteht beim Operator ein gewisser Stress durch den Zeitdruck, Ehrgeiz und Aufregung. Dieser Stress ist von der NIST gewollt, denn in einer realen Katastrophensituation würden Stress und Ermüdung ebenso auftreten [22, 40]. Gerade deswegen ist es hier besonders wichtig, die Schnittstelle so gut zu gestalten, dass der Operator damit in solchen belastenden Situationen dennoch umgehen und rationale Entscheidungen treffen kann. Eine Flut von Informationen ist ebenso unpassend wie ein Mangel an relevanten Informationen. Ein „gutes“ Interface kann im Wettkampf ausschlaggebend und im Ernstfall lebensrettend sein.
Die Teams müssen nicht nur an den Sensoren und Algorithmen für Navigation und Kartierung arbeiten, sondern auch darauf achten, dass sie ein passendes Interface für all diese Funktionen haben. Es ist der Wunsch des Deutschland1 Teams, das jetztige Interface im Zuge dieser Masterarbeit genauer unter die Lupe zu nehmen und ein Re-Design in Betracht zu ziehen. Schließlich soll das Interface nicht ein vernachlässigstes Stiefkind sein, das wichtige Plätze im Wettkampf kostet.

Zusammenspiel von HCI und HRI

Die Interfaces der Teams sind sehr verschieden [22]. Ebenso variiert die Anzahl der Eingabegeräte. Das Interface für das Team Deutschland1 setzt sich untere anderem aus den verschiedenen Kamerabildern, 2D-Scans und dem 3D-Scan zusammen. Diese Informationen alle für den Operator möglichst leicht zugänglich darzustellen, ist nicht nur eine Herausforderung in der Mensch-Computer Interaction (HCI), sondern auch der Mensch-Roboter Interaktion (HRI). Das bedeutet, dass nicht nur mögliche Funktionen einer Anwendung am Monitor dargestellt werden müssen, sondern auch übermittelt werden muss, in welcher Situation sich der Roboter befindet[4]. Man könnte hier einwenden, dass es nicht wichtig ist, ob der Operator nun den Systemstatus des Computers mitgeteilt bekommen und eine Anwendung bedienen muss oder ob der Operator nun über den Status des Roboters informiert werden muss, - dass es hier also hier um reine HCI handelt. Aber es gibt einen großen Unterschied: Hier geht es nicht nur um Software, sondern hier ist noch ein weit entferntes, teures Stück Hardware im Spiel, dass sicher bewegt werden muss. Eine Folge davon ist, dass wenn eine Aktion mit dem Roboter ausgeführt wurde, diese nicht mehr rückgänig zu machen ist. Wenn der Roboter nun vorwärts navigiert und gedreht wurde, ist dies unwiderruflich passiert. Passiert dabei eine Kollision, kann dieses fatale Auswirkungen

[3] In den früheren Wettkämpfen waren es 20 Minuten. Diese Zeit wurde halbiert, um u.a. mehr Teams die Chance zu einer Teilnahme zu geben.

[4] Dies wird meist mit „human – robot awareness“, einer speziellen Form der „situation awareness“ [48] bezeichnet.

haben. Diesen Fehler kann man nicht durch bessere Fehlerrobustheit des grafischen Interfaces rückgänig machen: Man kann nur versuchen, dem Operator einen solch guten Überblick zu vermitteln, dass die Möglichkeit einer Kollision geringer wird. Das heißt, der Operator sollte in seinen Entscheidungen unterstützt werden. In der HCI dagegen gibt es „Sicherheitsnetze“ – und für den „undo“-Button ist jeder Computernutzer desöfteren dankbar. Die Telerobotik vereint hier also HCI und HRI und kann auch von beiden Feldern wichtige Methoden und Erkenntnisse übernehmen, wie z.B. in [26].

1.2 Der wissenschaftliche Beitrag dieser Arbeit

Im Zuge dieser Arbeit wird ein Re-Design einer Schnittstelle entsehen. Es handelt sich dabei um das Interface des Team Deutschland1. Dabei werden nicht nur Experten nach ihrer Meinung und nach ihren Verbesserungsvorschlägen gefragt, sondern es werden auch Menschen befragt, die das System erst seit kurzem kennen und nutzen. Die dritte Quelle von Informationen für das Re-Design sind Artikel und Forschungsberichte über USAR, Mensch-Roboter-Interaktion und angrenzende Bereiche.
Mit diesem Wissen ausgestattet, wird hier nun getestet, ob es überhaupt möglich ist, die Erkenntnisse auf eine konkrete Schnittstelle anzuwenden. Wie lassen sich abstrakte Richtlinien in dem Re-Design in der Praxis, also an der Schnittstelle des Team Deutschland1, anwenden? Machen die Richtlinien überhaupt Sinn in diesem speziellen Fall? Welche Richtlinien können noch nicht greifen? Widersprechen sich die Erkenntnisse aus der Evaluation mit den Richtlinien? Der wissenschaftliche Beitrag dieser Arbeit wird darin liegen, Informationen bezüglich einer Aufgabe, die auf verschiedene Art gewonnen wurden, einem Anwendungstest zu unterziehen.
Nicht minder wichtig ist der praktische Beitrag dieser Arbeit: Dem RoboCup Rescue Team Deutschland1 eine bessere Schnittstelle für das Kurt3D System zu geben, damit es noch ein bisschen einfacher wird, auch weiterhin mit den weltbesten Teams im Wettkampf mithalten kann.

1.3 Aufbau dieser Arbeit

Diese Arbeit ist in sechs Kapitel aufgeteilt. Das erste Kapitel ist diese Einleitung. Darauf folgt eine Bestandsaufnahme des jetzigen Systems. Dabei wird die Hardware, die grafische Schnittstelle und deren Funktionalitäten genau festgehalten.
Anschließend folgen die Kapitel dem Titel der Arbeit: Das Kapitel 3 (ab Seite 17) beinhaltet die Beschreibung einer Evaluation. Da sie sehr frei gestaltet ist, und nicht immer den eigentlichen Richtlinien einer klassischen Evaluation folgt, wird sie als eine „offene Evaluation“ bezeichnet [36]. Im Kapitel 4 werden weitere Quellen für ein Re-Design angesprochen, da die Ergebnisse der offenen Evaluation nicht alleine als eine gute Basis für das Re-Design dienen sollen. Das daraus resultierende Re-Design wird in Kapitel 5 (ab Seite 57) beschrieben und spezifiziert. Das letzte Kapitel beinhaltet eine Diskussion und einen Ausblick für die weitere Entwicklung des Interfaces.

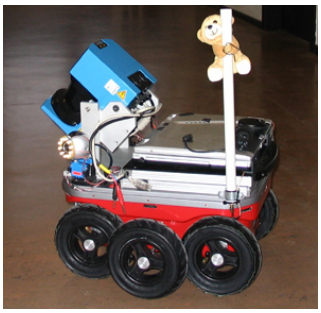
Kapitel 2

Der Kurt3D Roboter und das Interface

Bevor die Evaluation beschrieben und ein Re-Design vorgestellt wird, ist es wichtig, die Funktionalität des Systems, von der ausgegangen wird, genau festzuhalten. Im Mittelpunkt dieser Arbeit stehen die KURT-Roboter, die ihren Einsatz in der empirischen Evaluation finden. Die Ausstattung von diesen Robotern soll im Zuge dieser Arbeit nicht verändert werden, das Gleiche gilt für die Eingabegeräte. Sie werden folglich nur kurz dargestellt. Im Gegensatz dazu soll die GUI umgestaltet werden. Deshalb wird sie hier im Detail beschrieben.
Das „KURT-System“ besteht aus zwei Teilen: dem Roboter und einem anderen Rechner, der als Server und Operatorarbeitsplatz dient. Die Verarbeitung der Daten findet im Operatorcomputer, also off-board mit zwei Ausnahmen (on-board) statt: die Einzelbilder der Kameras (drei Bilder pro Sekunde, JPEG-Format) und die 3D- bzw. 2D-Scanpunkte. Die Kommunikation zwischen dem Roboter und dem Operatorcomputer läuft über WLAN 802.11a.

2.1 Hardware des Kurt3D

- KURT2 Robot Platform (als mobile Plattform)
- Onboard Laptop (zum Sammeln und Senden von Sensordaten und zur Umsetzung der Operatorbefehlen)
- Odometrie (für die Kontrolle der Räder bzw. das Regeln der Geschwindigkeit)
- Neigungswinkel-Sensor (eingebaut, wird aber erst seit Kurzem ausgelesen)
- 3D-Laser Scanner (um 2D-Scans und 3D-Scans zu machen)
- 2 Kameras mit Schwenk-Neigekopf, kein Zoom; (für die Orientierung und Navigation)



(a) Der Kurt3D Roboter.

(b) Der Joystick Saitek Cyborg Evo.

Abbildung 2.1: Teile der Hardware des Systems.

2.2 Operatorausstattung

- Laptop oder Desktop Rechner (für die Steuerung des Roboters)
- Maus (für die Bedienung des Computers)
- Tastatur (unter anderem für die Eingabe von Shortcuts)
- Joystick (für die direkte Steuerung des Roboters und der Kameras): Für die Evaluation wurde der Saitek Joystick Cyborg Evo (Abb. 2.1(b)) verwendet

Tastatur-Shortcuts

Da es sehr viele Shortcuts gibt, sind diese im Anhang A (S. 93) aufgelistet. Sie sind als Kommentar aus der Header-Datei übernommen und geordnet worden.

2.3 Das grafische Benuzterinterface (GUI)

Die Abbildung 2.2 zeigt die GUI, bevor sie Daten darstellt. Die einzelnen Elemente wurden in Kategorien eingeordnet:

- Ein **Display-Funktionselement** ist ein Element in der GUI, das durch seine Funktion einen Einfluss auf das Display nimmt; damit kann die Darstellung verändert werden.
- Ein **Steuerungselement** ist ein Element, dessen Benutzung direkte Auswirkung auf die Bewegung des Roboters hat, z.B. eine bestimmte Aktion auslöst.
- Ein **Informationselement** ist ein Element, das nur informiert, aber keine Aktion oder Veränderung bewirken kann.

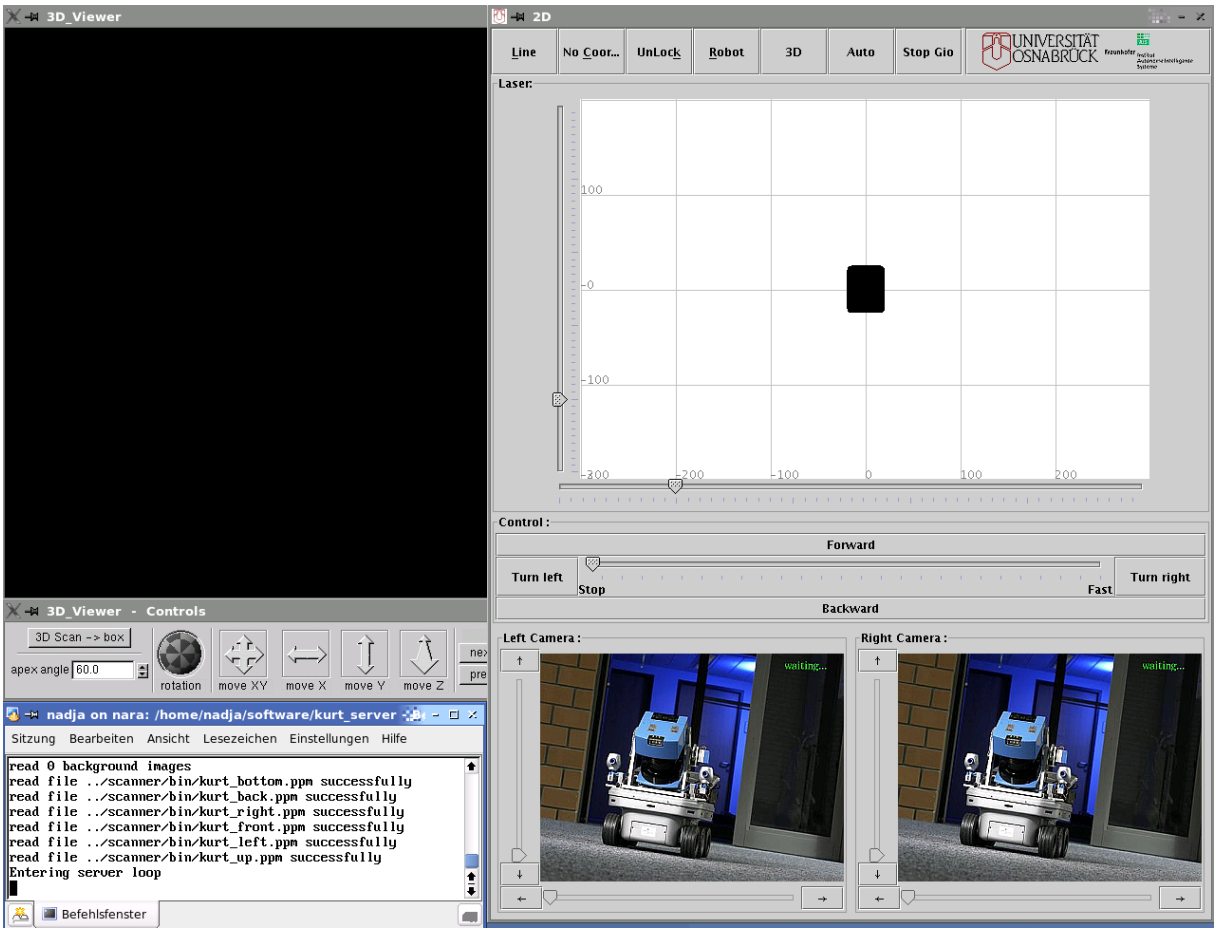


Abbildung 2.2: Die GUI ohne Datendarstellung. In der Mitte, hintern den beiden Fenstern, befindet sich die 3D-Funktionsleiste.

Die Buttonleiste über dem 2D-Viewer

Bemerkungen: Unterstreichungen zeigen die assoziierten Shortcuts an. „default“ gibt an, was der Button beim Start anzeigt (siehe Abbildung 2.3).

- unLine-Line (Verbinden der Punkte im 2D Scan: ausgeschaltet [=unLine] oder eingeschaltet [=Line]; default = Line;) *Display-Funktionselement*
- NoCoor... - Coords (ob die Koordinaten angezeigt werden sollen oder nicht; default = NoCoor) *Display-Funktionselement*
- UnLock - Lock (ob das Gitter fixiert sein soll oder ob die X- bzw. Y- Achse per Mausziehen in den Schiebereglern des 2D-Viewers verzerrt werden kann; default = unlock) *Display-Funktionselement*
- Robot - NoRobot (ob der Roboter im 2D-Viewer angezeigt werden soll oder nicht; default = Robot) *Display-Funktionselement*

Abbildung 2.3: Die Buttonleise über dem 2D-Viewer.

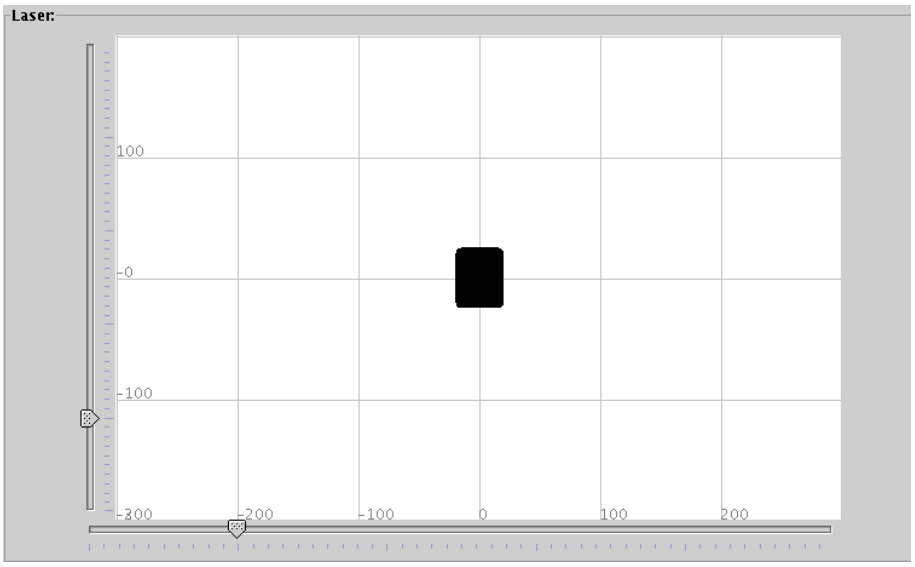


Abbildung 2.4: Der 2D-Viewer mit zentrierter Roboterdarstellung.

- 3D - 3D (ein Klick löst einen 3D-Laserscan aus; wenn ein solcher Scan vom System empfohlen wird, wird der Button rot eingefärbt; dass ein 3D-Scan gerade erstellt wird, ist nicht an dem Button zu erkennen) *Steuerungselement*, *Display-Funktionselement*
- Auto - Auto (ein Klick aktiviert die autonome Navigation; dass der Roboter autonom navigiert, ist nicht an diesem Button zu erkennen) *Steuerungselement*
- stopGio - stopGio (eine zu verfolgende Trajektorie kann per Mauszeiger in den 2D-Viewer eingezeichnet werden; der noch nicht ganz getestete Algorithmus sorgt dafür, dass der Roboter den angegebenen Zielpunkt ohne Umweg verfolgt. Neu erkannten Hindernissen weicht er dabei (noch) nicht aus; auch dieser Button verändert sich nicht durch das Klicken) *Steuerungselement*

Der 2D-Viewer

Die Darstellung des Roboters ist ein schwarzes Rechteck (siehe Abbildung 2.4), dessen Ecken abgerundet sind. Dieses Rechteck ist absichtlich ein wenig größer, ist also in der Relation zu den umgebenden Scanpunkten nicht maßstabsgetreu, um zu gewagte Manöver zu demotivieren. Wenn der Operator sieht, dass genügend Platz vorhanden ist, dann sollte

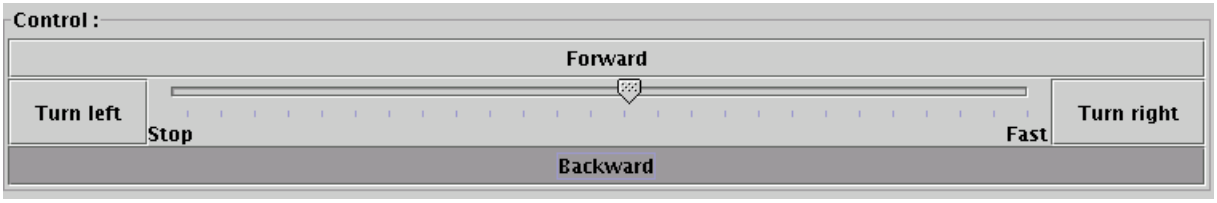


Abbildung 2.5: Die Maussteuerung mit geklicktem Backward-Button.

das wirklich so sein, plus zwei Sicherheitszentimeter. Der Operator wird folglich durch das Display „manipuliert“, ist sich dessen aber bewusst.
Es gibt zwei Darstellungsweisen des Roboters, zwischen denen man durch ein Klick in das Feld des 2D-Viewers gewechselt werden kann: Entweder wird der Roboter in der Mitte oder am unteren Rand des Viewers eingeblendet. Die erste Darstellungsweise wird bei einem 360 Grad Scanner, wie er im System in Hannover eingebaut ist, bevorzugt, da dann auch die Scanpunkte hinter dem Roboter erkennen kann und damit die Navigation für den Operator erleichtert. Die zweite Darstellungsweise ist nur bei einem 180 Grad Scan sinnvoll, da der Bereich hinter dem Roboter nicht erfasst wird. Ein Rückwärtsfahren muss dann durch eine 180 Grad Wendung des Roboters auf der Stelle und Vorwärtsfahren ersetzt werden. Der 2D-Viewer ist also nicht nur ein *Informationselemtent*, sondern auch ein *Display-Funktionselement*, weil man die Darstellungsweise durch ein Klick auf der Viewer verändern kann.
Einige der oben genannten Buttons aus der Knopfleiste (im Bild 1.1; unLine, NoCoors, UnLock, Robot) haben direkten Einfluss auf die Darstellung des 2D-Viewers. Folglich kann es als *Display-Funktionselemente* eingeordnet werden.

Die Maussteuerung

In diesem Teil der GUI (siehe Abbildung 2.5) kann der Roboter mit der Maus gesteuert werden. Diese Steuereinheit besteht aus folgenden Buttons:

- „forward“-Button: ein schmaler, langezogener Button; beim Klicken ohne Geschwindigkeitseinstellung ist eine (mittlere) Geschwindigkeit als default eingestellt, d.h., falls der Operator vergisst, eine Geschwindigkeit einzustellen, bewegt sich der Roboter dennoch vorwärts. Nachdem aber einmal die Geschwindigkeit eingestellt wurde, funktioniert diese default-Funktion nicht mehr. Der Roboter bewegt sich dann also nicht mehr mit der default-Geschwindigkeit, wenn der Schiebregler für die Geschwindigkeit auf „Stop“ geschoben wurde. *Steuerungselement*
- „turn left“-Button: befindet sich links unter dem „forward“-Button. Ein Klick löst eine Rotation auf der Stelle mit einem Winkel von ca. 3 Grad nach links aus; durch permanentes Klicken, also durch ein Nicht-Loslassen der Maustaste, rotiert der Roboter links herum. Dabei kann man dann nicht mehr erkennen, wie oft bzw. wie groß der Winkel der Rotation war. *Steuerungselement*

- „turn right“-Button: das Gegenstück zum „turn left-Button“. Er ist auf der rechten unteren Seite des „forward-Button“ angesiedelt und besitzt die analoge Funktionalität dessen. *Steuerungselement*
- backward-Button die gleiche Größe wie der forward-Button; liegt unter den turn left- und turn right-Button; nur beim Display für einen Roboter mit 360 Grad Scan sichtbar; *Steuerungselement*
- Geschwindigkeitsregler: ein mit der Maus „anfassbarer“ Schieberegler für die Geschwindigkeit. Die Skala ist durch kleine, unbeschriftete Einheitsstriche markiert. Diese lautet am linken Ende „stop“, am rechten Ende „fast“. Er ist also ein *Steuerungselement*, weil er aber auch eine informative Funktion besitzt ist er auch ein *Informationselement*. Dabei ist zu erwähnen, dass, wenn man mit dem Geschwindigkeitsregler am Joystick eine andere Geschwindigkeit einstellt, der Schieberegler in der GUI angepasst wird. (Andersherum funktioniert das leider nicht, da der Geschwindigkeitsregler am Joystick nicht mit einem ansteuerbaren Motor ausgestattet ist.

Die Kamerabilder

Je nach Roboter kann dieser Bereich variieren: Ist nur eine Kamera vorhanden, kann auch nur ein Kamerabild in der Konfigurationsdatei ausgewählt und in der GUI angezeigt werden. Ebenso können auch zwei Kameras, also zwei Kamerabilder, gewählt werden.
Die Kamerabilder (siehe Abbildung 2.8(b)) sind durch „Left Camera“ bzw. „Right Camera“ beschriftet. *Informationselement*
Um jedes Kamerabild herum befindet sich links und unten jeweils ein Schieberegler, dessen Enden auf der linken Seite mit einem Pfeil nach oben am oberen Rand bzw. einem Pfeil nach unten am unteren Rand hat. Analoges gilt für den Skala des Schiebereglers unter jedem Bild (also Pfeil nach links am linken Rand, Pfeil nach rechts am rechten Rand). Man kann auf diese Pfeil-Buttons klicken, um die Kamera nach oben bzw. unten blicken zu lassen, oder eben nach links oder rechts. Diese Bewegungen sind natürlich nur in einem bestimmten, durch die Hardware festgelegten Bereich möglich. Der Öffnungswinkel der Kameras beträgt je 45 Grad, ein Zoom ist nicht möglich. *Steuerungselement*

Der 3D-Viewer

Der 3D-Viewer (siehe Abbildung 2.8(c)) ist ein relativ großes Fenster auf der linken Seite, das die Daten des 3D-Scans als grüne Punktewolke darstellt. Anhand von den Funktionen (siehe unten 3D-Funktionenleiste) kann diese Grunddarstellung variiert werden. *Informationselement*

Die 3D-Funktionenleiste

Dieses Fenster (siehe Abbildung 2.8(a)) zweigt die Optionen für Filter/Algorithmen auf, die für die Darstellung des 3D-Scans aktiviert werden können. Es gibt Menüs, die durch einen

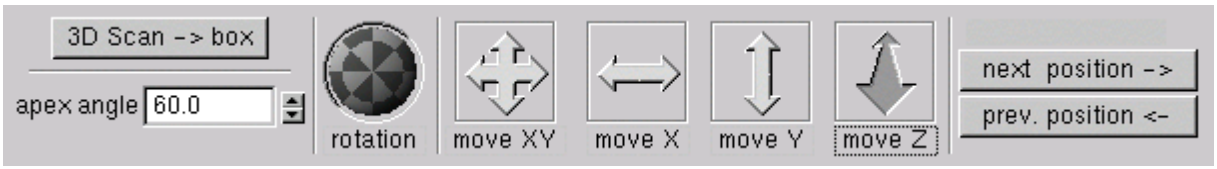


Abbildung 2.6: Die 3D-Steuereinheit mit geklicktem „move *Z*“.

Klick auf den entsprechenden Button mit dem Oberbegriff entweder aufgezeigt werden oder wieder durch einen zweiten Klick hinter den Button mit dem Oberbegriff verschwinden. Eine Auflistung befindet sich im Anhang.

Die 3D-Steuereinheit

In diesem Fenster (siehe Abbildung 2.6) befinden sich wichtige Funktionen, mit denen es möglich ist, die gewonnenen 3D-Scandaten von einer anderen Postion aus zu betrachten oder aber den Blickwinkel einer Position zu verändern.
Links befinden sich zwei Funktionen: ein Button „3D-Scan - box“ und darunter ein Kästchen, um den Öffnungswinkel (apex angle) des Scans im Nachhinein zu verändern (entweder durch Eingabe durch die Tastatur oder durch das Klicken mit der Maus von den Pfeilbuttons).

- „3D Scan - box“: Dieser Button ermöglicht es, dass eine (anders als die Scanpunkte) eingefärbte Box als Markierung für die Opfer in den Scan eingefügt wird. Diese Boxen werden einfach durchnummiert, um sie zu unterscheiden. Weitere Möglichkeiten, das Opfer irgendwie näher zu bestimmen, d.h., dessen Merkmale (z.B.: Bewegung, Kohlendioxid-Gehalt) an die Box zu schreiben oder dem Operator irgendwie in dem Scan ersichtlich zu machen, gibt es nicht. Ist dieser Button geklickt/aktiviert, dann werden die Optionen der 3D-Funktionsleiste deaktiviert, da diese Funktionen dann nicht benutzt werden sollen und können. Die „move“ -Buttons (s.u.) können aber nun die Position der Box innerhalb des Scans verändern. Im Gegensatz zu allen anderne Buttons in diesem Fenster, kann man mit dieser Funktion Scans nachhaltig verändern, aber nicht den Blickwinkel oder die Position verändern. *Display-Funktionselement*
- „Apex angle“: Der Öffnungswinkel, der maximal 180 Grad betragen kann (default ist 60 Grad), kann verändert werden. Wenn er verkleinert wird z.B.: auf 10%, werden die übriggebliebenen 10% des Scans auf die ganze Breite des 3D-Viewer vergrößert/in die Breite verzerrt. Dies kann sinnvoll sein, um besser Details zu betrachten, die dann aber verzerrt dargestellt sind. Die Position wird also anhand dieser Einstellung nicht verändert, der Blickwinkel bleibt aber gleich und der Ausschnitt des Gesehenen verkleinert/vergrößert und damit auch verzerrt. *Display-Funktionselement*
- Die Rotationskugel: Weiter rechts befindet sich eine (schwer zu beschreibende) Rotationskugel, die auf einen zweidimensionalen Kreis proijziert wurde. Durch eine Art kariertes Spinnennetz, das sich verändern kann, wird ein dreidimensionaler Effekt

erzeugt. Die Funktion dieser Rotationskugel liegt in der Möglichkeit, diese Kugel anzuklicken, - und dabei die Maustaste gedrückt zu lassen, und anhand der Maus-Bewegung den Blick im Scan nach z.B. oben zu richten. D.h., dass die Scanpunkte, die weiter oben (in der Y-Richtung) lagen, nun in das Zentrum des 3D Viewers gelangen. Diese Funktion ist von First Person Shootern („Egoshootern“) als „Mouselook“ bekannt, - im Unterschied dazu muss hier aber der Mausbutton gedrückt gehalten werden und der erste (und damit einzige) Klick muss auf der Rotationskugel stattfinden. Mit gedrückter Maustaste kann aber der Zeiger der Maus auf dem ganzen Display (also auch außerhalb des 3D-Viewers) bewegt werden, und der Blickwinkel folgt (bis zu seinen Grenzen). Diese Funktion verändert also den Blickwinkel, nicht aber die Position. *Display-Funktionselement*

– Die „move“ -Buttons: Das gerade beschriebene Prinzip zeichnet auch die anderen Tasten weiter rechts aus: auf den Button klicken und dann die Taste geklickt halten; dann aber ist es möglich, das ganze Display zu benutzen. Mit dem nächsten Button, „move XY“, kann man sich in der X-und der Y-Achse bewegen, aber auch gleichzeitig Bewegungen in beiden Achsenrichtungen, d.h., schräg, machen. Für Bewegungen in der X-Richtung wird der „move X“ -Button benutzt; analog sind die „move Y“- und „move Z“ - Buttons zu verstehen.

  Wichtig ist zu bemerken, dass man durch die „move“- Buttons die Position (und damit auch als Folge den Blickwinkel) innerhalb des Scans verändert, während man mit der Rotationskugel nur den Blickwinkel, aber *nicht die Position* verändert. *Display-Funktionselemente*

– Die Positions-Buttons: Ganz rechts außen befinden sich zwei Buttons übereinander: oben der „next position ->“ -Button, unten der „prev. position <-“ -Button. Anhand von diesen Buttons kann man die Scans in den 3D-Viewer bringen, die an verschiedenen Positionen gemacht wurden. (Man „geht“ sozusagen die alten Positionen ab.) Ist man z.B. an der Position 5, kann man durch den Button „next position ->“ zu der Position 6 gehen , bzw. der Scan von Position 6 wird dann im 3D-Viewer angezeigt. Würde man danach wieder auf den „prev. position <-“- Button klicken,also zurückgehen, taucht wieder der Scan von Position 5 auf. Die Bedeutung der Pfeile auf den Buttons ist vielleicht eine Orientierungshilfe für den Operator, da er sich die Richtung des Pfeils eher merkt, als die Beschreibung next , bzw. previous. *Display-Funktionselement*

2.4 Grundaktionen und Abläufe

Nicht nur die einzelnen Elemente des Interfaces sind wichtig, sonder auch deren Interaktion. Diese werden hier kurz beschrieben. Die angesprochenen Tastenkürzel können im Anhang A nachgeschlagen werden. Videoaufnahmen der Abläufe befinden sich auf der Webseite `http://www.kos.informatik.uni-osnabrueck.de/hri/`.
Es gibt vier verschiedene Grundaktionen des Operators:

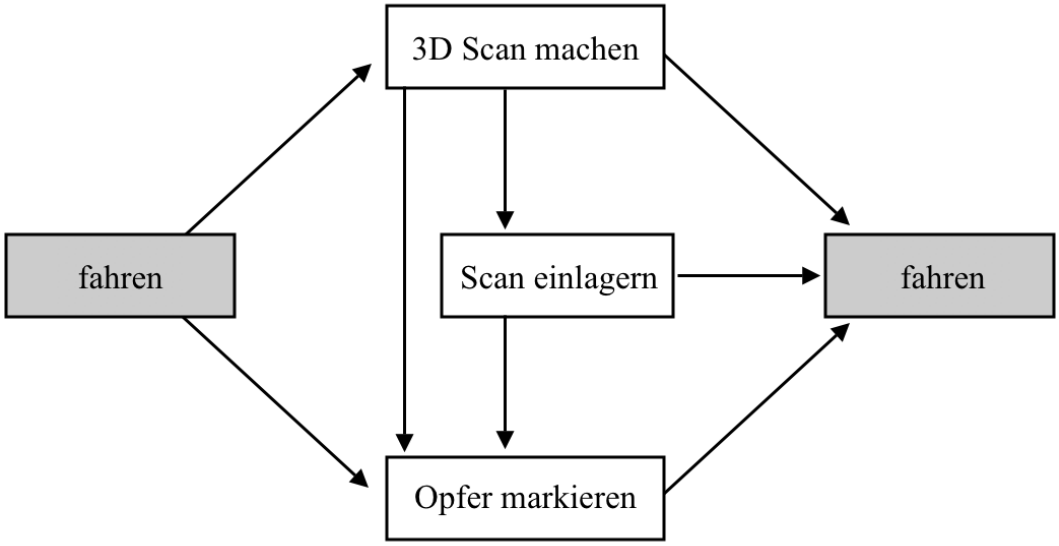


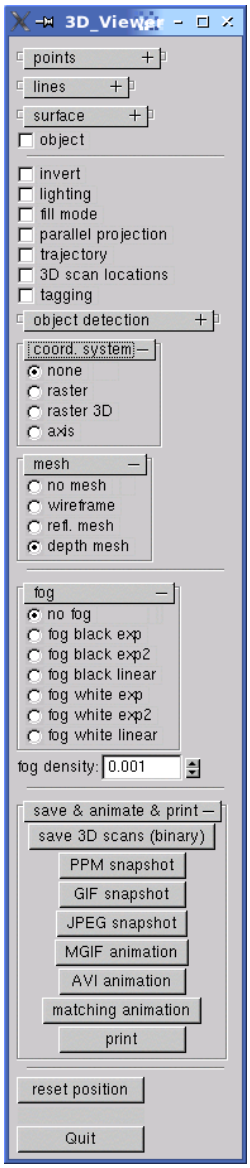
Abbildung 2.7: Mögliche Ablaufschemata.

1. *Fahren*: Dafür stehen dem Operator der Joystick, die Joysticktasten und die Maussteuerung in der GUI zur Verfügung. Von einem Eingabegerät kann ohne Probleme zu einem anderen gewechselt werden. Es tritt eine kleine, kaum wahrnehmbare zeitliche Verzögerung bei der Übertragung zum Roboter auf.
2. *3D-Scan machen*: Ein 3D-Scan kann mittels eines Buttons in der GUI oder durch einen Knopf am Joystick ausgelöst werden. Der Roboter kann nur im Stillstand scannen. Während des Vorgangs bleiben alle Informationen in der GUI gleich, nur im Terminal kann der Operator einige Ausgaben bzgl. des Scans ablesen. Der Scanvorgang ist beendet, wenn die neue, grüne Punktewolke im 3D-Viewer angezeigt wird.
3. *3D-Scan manuell einlagern*: Wenn das System den letzten 3D-Scan nicht gut mit den schon vorhandenen 3D-Scans zur Überlagerung gebracht hat, hat der Operator die Möglichkeit, dies manuell zu korrigieren. Durch ein Tastenkürzel wird der manuelle Modus des 3D-Viewers aktiviert, der letzte Scan wird dann nicht mehr grün, sondern gelb dargestellt. Weitere Shortcuts stehen für Rotationen und Translationen zur Verfügung. Der manuelle Modus kann durch zwei verschiedene Tastenkürzel verlassen werden: Der eine sorgt dafür, dass nun die neue Position des 3D-Scans genau so beibehalten wird; der andere löste einen zweiten Versuch des Systems aus, den Scan nun auf Grundlage der neuen Position besser mit den anderen Scans zu überlagern. Sprich, der 3D-Scan muss nicht auf den Punkt genau manuell eingelagert werden, sondern muss nur in die Nähe der Endposition gebracht werden, da das System den neuen Scan „einklickt“. Mit dem Verlassen des manuellen Modus wird der gelbe Scan dann wieder grün eingefärbt.

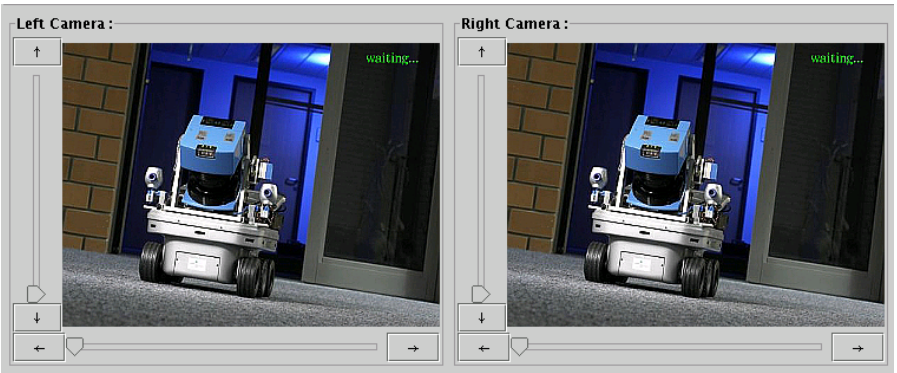
Im Englischen heißt der Vorgang des Systems, den neuen Scan zur Überlagerung zu bringen, „to register“. Das System kann also einen Scan „registrieren“, oder aber der Operator tut dies manuell. Die direkte deutsche Übersetzung des „Registrierens“ wurde hier nicht übernommen, sondern durch den Begriff des (manuellen) Einlagerns ersetzt.

4. *Opfer markieren*: Ein erkanntes Opfer wird durch die „Box“, einem Rechteck bzw. Quader von rötlich eingefärbten Pixeln, kenntlich gemacht. Dazu muss der Operator durch einen Shortcut den „box modus“ aktivieren. Dadurch taucht ein eingefärbtes Pixelfeld im 3D-Viewer auf und kann mit weiteren Tastaturkürzeln an die richtige Stelle bewegt werden. Der Operator verlässt danach den „box modus“ mit dem selben Shortcut, mit dem er ihn zuvor aktiviert hat.

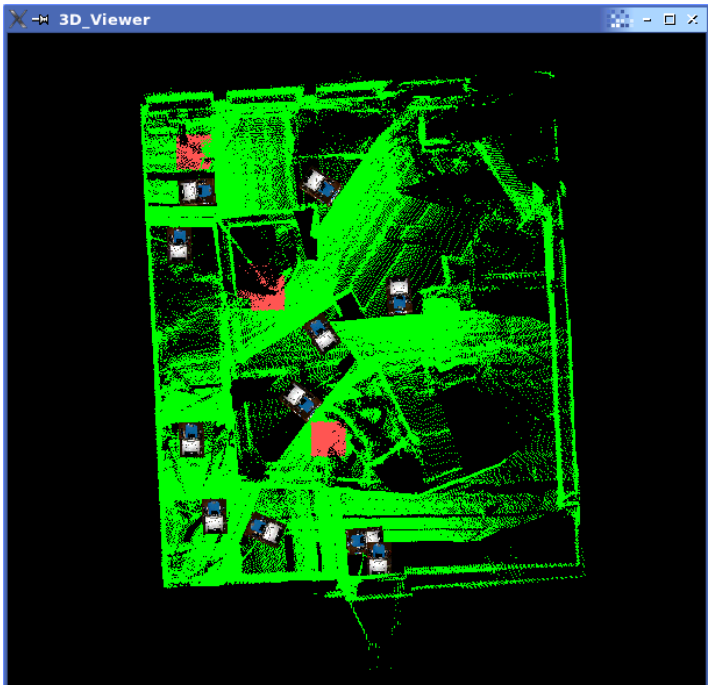
Die möglichen Zusammenhänge der Aktionen sind in der Abbildung 2.7 dargestellt. Wenn ein 3D-Scan vom System gleich richtig eingelagert wird, ist es zwar möglich, aber natürlich nicht notwendig, den Scan manuell einzulagern.



(a) Die teilweise aufgeklappte 3D-Funktionsleiste.



(b) Die beiden Dummie-Kamerabilder.



(c) Der 3D-Viewer mit von oben parallel projizierten Scans, rot markierten Opfern und Scan-Positions-Einblendung.

Abbildung 2.8: Verschiedene Teile der GUI.

Kapitel 3

Die offene Evaluation des vorliegenden Interfaces

3.1 Erläuterung

Diese offene Evaluation des bestehenden Interfaces soll aufzeigen, wie ein Operator eine USAR Mission erfüllt. Dabei gilt es zu erkennen, welche Informationen notwendig sind und welche Informationen die GUI, und damit den Operator, unnötig belasten.

Wie schon kurz in der Einleitung (Kapitel 1.3) angesprochen, handelt es sich hier um eine sogenannte „offene Evaluation“ [36]. Dieser Begriff beinhaltet, dass die Evaluation ein „wesentlicher Bestandteil des Entwicklungsprozesses“ [36] ist. Der Evaluator muss die Umwelt und die Beteiligten, in diesem Falle die Operatoren, in seine Arbeit miteinbeziehen und dem Entwicklungsprozess und Entwicklern als Helfer und Berater zur Seite stehen. Ausgehend von dem vorhandenen System wird eine weitere Entwicklungsschleife in Angriff genommen, um die Verbesserung der Schnittstelle voran zu treiben. Nach dieser Schleife wird vielleicht eine andere folgen, um das hier entstehende Re-Design an die neuen Anforderungen anzupassen.
Ein weiterer Punkt ist, dass es sich um das Re-Design einer sehr speziellen, einmaligen Schnittstelle handelt. Es geht hier nicht um eine weit verbreitete Anwendung, wie beispielsweise ein Textverarbeitungssystem, das von vielen Menschen genutzt wird. Die Zielgruppe besteht aus nur sehr wenigen Personen, deren Ansprüche und Arbeitsweisen hier betrachtet werden sollen. Die Auswahl der Versuchspersonen ist auf diese Zielgruppe abgestimmt und ist sehr klein gewählt. Es geht hier darum, die Schwachstellen und Haken dieses Interfaces zu finden - es geht aber nicht darum, eine „repräsentative“, prozentuale Angabe einer Teilfrage in Händen zu halten. Da es keine Vergleichsmöglichkeiten gibt, wären durch statistische Auswertungen gewonnene Daten zudem nicht aussagekräftig. Das Interesse liegt hier an qualitativen Aussagen, nicht in quantitativen. Betrachtet man diese Einschränkungen, so wird klarer, dass man hier nur auf ein rudimentäres Methodeninventar zurückgreifen kann.

Eine intuitive, heuristisches Vorgehensweise soll dabei helfen, die verschiedenen Facetten dieses Mensch-Roboter-Teams, besser zu verstehen und im Re-Design aufgreifen zu können. In diesem Team gibt es unterschiedliche Ebenen und Abläufe, die man vielleicht nicht erkennt - und damit auch nicht verbessern kann, wenn man eine durch Methodik eingeengten Blick hat. Es ist eine also eine bewusste Entscheidung, auf die „klassischen Maßstäbe“ einer Evaluation zu verzichten. Diese Freiheit ist durch die interdisziplinäre Eigenart dieser Arbeit zu rechtfertigen: Sie ist ein Zugeständnis, um Lücken zwischen den verschiedenen Disziplinen besser überbrücken zu können.

Die Materialien zu dieser Evaluation, also Videos, Bilder und die ausgewerteten Fragebögen, können auf der Webseite `http://www.kos.informatik.uni-osnabrueck.de/hri/` gefunden werden.

3.2 Beschreibung

Dieses spezielle Interface des RoboCup Rescue Team Deutschland1 wird nur von einigen wenigen Personen benutzt. Die Zielgruppe für das Re-Design ist damit sehr klein. Zumal von diesen Personen nur ein oder maximal zwei Personen in einem Wettkampf für die Steuerung ausgewählt werden.

3.2.1 Versuchspersonen

Eine Zielgruppe, zwei Arten von Versuchspersonen

Obwohl die Zielgruppe sehr klein ist, werden für die Evaluation zwei Gruppen von Versuchspersonen hinzugezogen: Novizen und Experten. Die Novizen müssen in das System eingelernt werden, während die Experten damit schon seit Jahren umgehen, bzw. es zu großen Teilen selbst programmiert haben. Für eine Art von vermeintlichen Endnutzern wird vorgeschlagen, etwa fünf bis sieben Versuchspersonen zu nehmen [28,47]. Da die Zielgruppe sehr klein und leicht bestimmtbar ist, kann hier nur von einer Art von Endnutzern die Rede sein. Folglich wären eine Handvoll Versuchspersonen ausreichend. Wie aber auch bei Dumas in [28] erwähnt, ist es durchaus sinnvoll, die Gruppe der Versuchspersonen nicht nur aus einem einheitlichen, sehr stark Zielgruppen bezogenen Bereich zu wählen, sondern die Gruppe inhomogen zu gestalten. Die Experten, die eigentliche Zielgruppe, zu befragen, ist wichtig, da deren Meinungen eine Quelle für viele detaillierte Informationen und Erfahrungen sind. Der inhomogene Teil der Testpersonen, die Novizen, ist nicht minder wichtig: Sie sind noch nicht betriebsblind und können durch neue Ideen und Kritiken das Interface von einer anderen, „frischen“ Perspektive beleuchten. Der große zeitliche Aufwand, um die Novizen einzulernen und zu testen, ist damit eine gute Investition in das Re-Design. Effektiv wurden bei dieser Evaluation neun Versuchspersonen involviert. Die Behauptung, dass fünf Teilnehmer schon bis zu 80 Prozent der Fehler, bzw. 90 Prozent anhand von circa zehn Teilnehmern aufdeckt werden können [28], ist hierbei motivierend.

Experten

Die Gruppe der Experten setzt sich aus zwei Experten vor Ort in Osnabrück [3] und den zwei Operatoren (von Lissabon [10] bzw. Osaka [11]) zusammen. Sie haben seit einigen Jahren mit dem System zu tun und kennen es sehr gut. Problematisch ist, dass sie vielleicht betriebsblind sind, und dass sowohl die Oberfläche als auch die Konfiguration der Eingabegeräte von ihnen entworfen und implementiert wurden. Natürlich gibt es Ideen und Wünsche, für deren Umsetzung aber noch keine Zeit übrig war. Somit stellt die momentane Schnittstelle den bestmöglichen funktionalen Kompromiss dar.
Die Experten werden mit ihren Aussagen wichtige Erfahrungen miteinbringen können, vor allem, da sie die Perspektive von „hinter den Kulissen“ und damit Einblicke in das System haben, die ein eingelernter Novize noch nicht haben kann.

Novizen

Auswahlkriterien Es wurden fünf Studenten ausgewählt, die ein großes Interesse an Computern und Computerspielen haben. Die Einschränkung, dass eine Vertrautheit mit Computern vorhanden sein muss, soll die Grundzüge der sehr speziellen Zielgruppe widerspiegeln. Ein Operator, der wenig Erfahrung mit Computern und deren Handhabung hat, ist extrem unwahrscheinlich. Ebenso ist ein Interesse an Computerspielen von Bedeutung: Erstens, da dort der Umgang mit Maus und Joystick geübt wird; zweitens, damit die Versuchsperson den Umgang und die Steuerung des Roboters mit denen eines Computerspiels vergleichen kann. Im Fokus stehen vor allem First Person Shooter[1], da die Aufgabe, Opfer in einer USAR-Arena zu suchen vergleichbar mit der Suche von Feinden in einem Spiel scheint [39,41]. Dem ist hinzuzufügen, dass in einem solchen Spiel kein Kartografieren notwendig ist. So lange dies noch in den Aufgabenbereich des Operators fällt, ist es dienlich, ein gutes räumliches Vorstellungs- und Orientierungsvermögen mitzubringen. Alle Novizen haben Erfahrung mit 3D-Programmen. Das aber impliziert nicht, dass auch ein gutes Orientierungsvermögen vorhanden ist.
Da die Einarbeitung in mehreren Sitzungen ablief, war es wichtig, dass die Versuchspersonen ein paar Mal für einige Stunden verfügbar war. Die Teilnahme wurde nicht entlohnt oder mit Versuchspersonenstunden belohnt. Neugier, Interesse und die Möglichkeit, mit echten Robotern arbeiten zu dürfen, waren Motivation genug. Sie wurden natürlich darauf hingewiesen, dass sie jederzeit aufhören können und dass diese Evaluation anonym verläuft.

Schrittweises Anlernen Ziel ist es, die Versuchspersonen schrittweise an den Umgang mit dem System heranzuführen. Darunter fällt nicht nur die sichere Steuerung anhand von Maus und Joystick, sondern auch das Kartografieren und das Auffinden/Markieren von Opfern.

1. Der erste Schritt ist es, eine Versuchsperson mit der Situation vertraut zu machen. Zu diesem Zweck wird zuerst erklärt, worum es sich bei der Rettungsrobotik handelt

[1] In Deutschland sind diese umstrittenen Spiele als „Egoshooter“ bekannt.

Stufe	*Was wird erlernt?*	*Wie wird es erlernt?*	*Wo?*
Einführung	Grundwissen, Szenariokenntnis	Erklärungen, Bilder	Computer, Internet
2D-Simulation	Umgang mit Maus-Steuerung, Joystick, 2D-Viewer	freies Explorieren	Computersimulation
KURT2	Situationskenntnis, Verhältnis Realität – Display	Ababreiten von Aufgaben, mit und ohne Sichtkontakt	(Neben-)Zimmer, Flur
Kurt3D	Kartografieren, Opfer finden und markieren	Kartografieren d. Räumlichkeiten, mit und ohne Sichtkontakt; sich selbst als Opfer markieren	(Neben-)Zimmer, Flur
Test	effizientes Kartografieren und Opfermarkieren	(Zeitdruck)	unbekanntes Terrain

Abbildung 3.1: Zusammenfassung des Einlernens

und wie die Forschung momentan z.B. durch NIST Wettkämpfe angetrieben wird. Von den Wettkampfsituationen werden Bilder gezeigt, um der Versuchsperson einen Eindruck von den gestellten Szenarien zu geben.

2. Der nächste Schritt beinhaltet die erste Einarbeitung in die Steuerung des Roboters. Dazu steht eine 2D-Simulation (USARSIM, siehe [35]) zur Verfügung.

3. Im dritten Schritt darf die Versuchsperson zum ersten Mal einen richtigen Roboter steuern. Es handelt sich dabei um einen KURT2, der nicht mit einem 3D-Scanner ausgestattet ist und auch nur eine zentral sitzende Kamera hat. Eine Kartografierung ist damit nicht möglich.
   Indem die Versuchsperson den Roboter mit Sichtkontakt steuert, erlernt sie schnell den Gebrauch der verschiedenen Eingabegeräte. Ob nun die Maus benutzt wird oder der Joystick, steht der Versuchsperson frei. Die Versuchsperson erlernt dabei die „human-robot awareness“ [49], sie bekommt sozusagen ein Verständnis für den Status, die Aktivität und das direkte Umfeld des Roboters. In dieser Phase läuft das so ab, dass die Versuchsperson abwechselnd auf den Roboter und auf den Monitor schaut, um die reale Situation mit der Information am Rechner zu vergleichen. Dies ist ein sehr wichtiger Schritt, denn die Versuchsperson muss ein Gefühl für die Informationendarstellung am Monitor und die Effizienz der Steuerung bekommen. Wird der Roboter auf einen Stuhl zu gesteuert, so erlernt die Versuchsperson einzuschätzen, wie z.B. ein Meter Abstand in Wirklichkeit, dann im Kamerabild und im 2D-Scan aussieht. Nach dem ersten Eingewöhnen wird die Versuchsperson angeleitet, z.B. unter einen Tisch oder ein bisschen schneller zu fahren.
   Ist eine gewisse Vertrautheit sichtbar, wird die Versuchsperson ermutigt, den Robo-

ter in den angegliederten Raum zu steuern, damit kein direkter Sichtkontakt mehr möglich ist. Eine Kollision, ein Durchdrehen der Räder oder Vergleichbares ist durch die offene Tür hörbar. Dieser angegliederte Raum ist schmal und mit Tischen und Stühlen ausgestattet. Folglich ist es eine Herausforderung, dort sicher zu navigieren. Wenn diese gemeistert ist, darf die Versuchsperson den Roboter auf den Flur hinaus fahren. Da dieser Flur sehr lang ist, kann man den Roboter nicht mehr hören und muss sich damit völlig auf die Information am Monitor verlassen. Die Versuchsperson wird dabei darauf hingewiesen, dass viele Büros an diesem Flur liegen. Da der normale Universitätsbetrieb weiterläuft, nehmen die Menschen auch nicht auf den Roboter Rücksicht, d.h., es kann passieren, dass Menschen an dem Roboter vorbeilaufen und dass Türen zum Flur hin aufgehen können. Dies zwingt die Versuchsperson, sehr aufmerksam zu sein und die Informationen am Monitor nicht aus dem Auge zu lassen. Ansonsten darf die Versuchsperson auch per Roboter in die offenen Büros hineinschauen, ihn hineinfahren und sich auf dem Flur nach Belieben schnell und weit bewegen. Die einzige Einschränkung ist die Laufzeit der Roboter-Akkus (ca. 4 Stunden). Die Versuchsperson kann folglich so lange mit dem Roboter fahren wie sie will oder bis die Akkus leer sind. Deswegen werden die Versuchspersonen angehalten, genügend Zeit für dieses Einlernen mitzubringen.

4. Der vorletzte Schritt verläuft analog zu dem Schritt davor. Der einzige Unterschied ist hier, dass der 3D-Roboter zum Einsatz kommt. Die Versuchsperson muss vorher mit dem dreidimensionalen Scannen vertraut gemacht werden. Da die Handhabung des 3D-Fensters am Monitor deutlich mehr Übung bedarf, wird erst wieder mit Sichtkontakt zum Roboter angefangen. Dabei wird die Versuchsperson mit den Möglichkeiten des 3D-Viewers vertraut gemacht und erlernt danach manuelle Einlagerung eines vom System falsch eingepassten Scans in die Karte.
   Die erste Aufgabe ist es dann, diesen Raum, in dem die Versuchsperson sitzt, zu kartografieren; die zweite Aufgabe besteht darin, den in der Realität angegliederten Raum auch in die 3D-Karte einzugliedern. Wenn diese Aufgaben gut gelöst werden, soll die Versuchsperson den Flur kartografieren. Im Weiteren wird der Versuchsperson erklärt, wie Opfer in der 3D-Karte markiert werden. Dies ist, im Gegensatz zu den anderen Aufgaben, sehr einfach und kann sehr schnell erlernt werden.
   Da das Erstellen einer Karte sehr wichtig und auch schwierig ist, wird dabei sehr stark auf die Ergebnisse geachtet. Die Versuchsperson muss einen gewissen Grad an Präzision erreichen, da sie ansonsten in unbekannten Terrain wenig Chancen auf eine gute Leistung hat.

5. Um zu sehen, wie gut die Versuchspersonen in unbekanntem Terrain mit dem Roboter navigieren und kartografieren können, durchlaufen sie alle den gleichen Test: Innerhalb von 30 Minuten soll ein ihnen unbekannter Raum kartografiert werden. In diesem Raum befindet sich ein Parcours (siehe Abbildung 3.2), der ähnlich einer gelben Testarena eines USAR Wettkampfes gestaltet wurde. Ebenso müssen drei Opferteile in der Karte markiert werden.
   Das Display und die Eingabegeräte werden dabei von einer Kamera aufgenommen. In einem Gespräch mit der Versuchsperson nach dem Test wird auch das aufgenomme-

Abbildung 3.2: Bilder des Test-Parcours

(a) Kurt3D im Parcours. (b) Versuchsperson am Arbeitsplatz.

Abbildung 3.3: Roboter und Operator während der Evaluation.

ne Video gezeigt. Anhand des Videos fällt es der Versuchsperson leicht, den Ablauf nochmals nachzuvollziehen und Auskunft darüber zu geben („retropsective thinking aloud“ [28]). Dabei wird auf bestimmte ungewöhnliche Situationen verwiesen und nochmals genauer nachgefragt, was die Intention hinter bestimmten Aktionen war.

Nach dem Test und der Nachbesprechung findet ein strukturiertes Interview statt, siehe Kapitel 3.2.2, das auch mit den Experten durchgeführt wurde.

3.2.2 Strukturiertes Interview

Das strukturiertes Interview dient hier als Gesprächsgrundlage. Die Befragten werden alle mit den gleichen standardisierten und sturktrierten Fragen bzw. Aussagen konfrontiert. Es wird notiert, in wie weit die Befragten der Aussage zustimmen. Diese Werte können danach auch ausgewertet werden, die Ergebnisse sind aber nicht absolut zu sehen sondern

eher als Tendenzen zu betrachten. Am wichtigsten ist es, dass das Gespräch durch die Aussagen in Gang gebracht wird. Da es sich immer wieder um die gleichen Fragen bei jeder Versuchsperson handelt, werden die gleichen Gesprächspunkte bei jedem angesprochen. Dabei wird notiert, was die Versuchspersonen zu einer Aussage ergänzen und wie sie darüber denken. Die Antwortmöglichkeiten sind also einerseits standardisiert, da eine Skala verwendet wird. Andererseits gibt es zusätzlich offene Antwortmöglichkeiten.
Der „Fragebogen“, der eigentlich eine Ansammlung von Aussagen ist und nicht von Fragen, wurde anhand der ISO 9241 Teil 10 [4] erstellt. Dieser Teil der ISO Norm[2] von 1996 bestimmt den Begriff der Usability („Benutzerfreundlichkeit“) näher. Dazu dienen folgende sieben Aspekte:

- Aufgabenangemessenheit
- Selbstbeschreibungsfähigkeit
- Steuerbarkeit
- Erwartungskonformität
- Fehlerrobustheit
- Individualisierbarkeit
- Erlernbarkeit

Ein sinnvolles Werkzeug stellt dabei der IsoMetrics [9] dar. Jeder der oben genannten sieben Aspekte wird anhand von dem Isometrics in Aussageform „verpackt“. Zum Beispiel gibt es 15 Aussagen – sogenannte „items“–, die nur auf die Aufgabenangemessenheit abzielen, zwölf Aussagen drehen sich nur um die Selbstbeschreibungsfähigkeit eines Systems usw. Zwei Aspekte, Individualisierbarkeit und Erlernbarkeit, sind in dieser Evaluation nicht von großer Bedeutung und können deshalb weniger ausführlich behandelt werden.
Den Isometrics gibt es in zwei unterschliedlichen Formen: In der Langform und der Kurzform. Beide haben das Bewertungssystem gemeinsam, welches sich von „stimmt nicht“ über „stimmt wenig“ und „stimmt mittelmäßig“ zu „stimmt ziemlich“ und „stimmt sehr“ und natürlich „keine Angabe“ erstreckt. Der Unterschied liegt darin, dass in der Kurzform die Aussagen nacheinander eingestuft werden sollen, in der Langform hingegen wird nach der Aussage noch weiter nachgefragt, wie wichtig dieser (vorher in der Aussage näher bestimmte) Aspekt für den Gesamteindruck einer Software/einer Schnittstelle ist. Auch hier muss der Befragte seinen Eindruck einschätzen und anhand einer Wichtigkeitsskala („nicht wichtig“, „wenig wichtig“, „mittelmäßig wichtig“, „ziemlich wichtig“ und „sehr wichtig“) mit in die Gewichtung des Aspekts miteinfliessen lassen. Die Aussagen in der Langformformatierung lassen zudem noch Platz für Ergänzungen und Kommentare.
Der Isometrics wurden auf die zu evaluierende Schnittstelle dieses Systems angeglichen. Bei allen Isometric-Items musste dabei darauf geachtet werden, dass sie zwei Ansprüchen genügen: Alle wichtigen Aspekte der Benutzerfreundlichkeit (bzgl. ISO 9241 Teil 10)

[2] In [51] können theoretische Informationen über Richtlinien, Standards und Stilempfehlungen gefunden werden. Dieser Artikel setzt u.a. die ISO 9241 Teil 10 in Zusammenhang mit den anderen Teilen und anderen internationalen Normen.

müssen darin vorkommen und möglichst alle Teile der Schnittstelle müssen angesprochen werden. Im Bereich der langformatigen Aussagen wurde darauf geachtet, dass sie allgemein gehalten werden, um ein Bild von dem Gesamteindruck zu bekommen. Nachdem eine Auswahl der (scheinbar) wichtigsten Aussagen getroffen wurde, wurden diese angepasst, indem einige Formulierungen ausgetauscht wurden. Die kurzformatigen Aussagen dagegen wurden völlig ausgetauscht: Da die Aussagen sich sehr genau auf einzelne Teile der Schnittstelle beziehen, wurde nur die Aussagenstruktur beibehalten, die Inhalte wurden aber völlig umformuliert.
Der Ablauf dieses strukturierten Interviews sieht folgendermaßen aus: Zuerst wird die Versuchsperson mit den speziellen Aussagen über die Schnittstelle konfrontiert. Die Aussage wird dem Befragten laut vorgelesen, der dann einschätzen muss, wie sehr oder wie wenig er der Aussage zustimmt. Diese kurzformatigen Aussagen sind so gruppiert, dass die einzelnen Teile der grafischen Benutzerschnittstelle nacheinander durchgegangen werden. Dabei wird der Befragte darauf hingewiesen, auf welchen Teil des Displays sich nun die Aussagen beziehen. Nachdem die Aspekte des Displays und der Eingabegeräte durchgegangen sind, hat der Befragte einen besseren Überblick über seinen Eindruck von der Schnittstelle. Deshalb folgen darauf die langformatigen Aussagen, die nun einen begründeten kritischen Eindruck der Schnittstelle widerspiegeln können. Von der Versuchsperson werden nun genauere Kommentare und Ideen erwartet. Durch die Reihenfolge der verschiedenen Aussagearten sollen unklare Kommentare wie „Mir gefällt das Display einfach nicht“, für den Befragten begründbarer werden.

3.3 Beobachtungen und Interpretationen

Bei dem stufenweise Anlernen wurden Notizen gemacht, die einen Einblick in die Entwicklung geben und deshalb eine gewisse Wichtigkeit besitzen. Zuerst werden nun die einzelnen Beobachtungen dargestellt, um dann im zweiten Teil die Aspekte eines jeden Teils des Interfaces für sich zusammenzufassen. Die Abbildung 3.4 stellt den Aufbau des folgenden Kapitels grafisch dar, um sie leichter nachvollziehen zu können.

3.3.1 Stufenbezogene Beobachtungen und Besonderheiten

2D-Simulation

Als die Einarbeitung mit der 2D-Simulation begann, war die Simulation noch relativ langsam. Dies führte schnell zu einer gewissen Überdrüssigkeit der Versuchspersonen. Deshalb war die Zeit, die sie dort am Rechner verbrachten, ziemlich gering. Da die Versuchspersonen aber sehr schnell mit der Simulation zurechtkamen, war die Langsamkeit der Simulation nicht so schlimm.
Leider ist auch der 2D-Viewer in der Simulation verhältnismäßig langsam. Das führte dazu, dass für die Navigation die Kamerabilder viel wichtiger waren als der Scan.

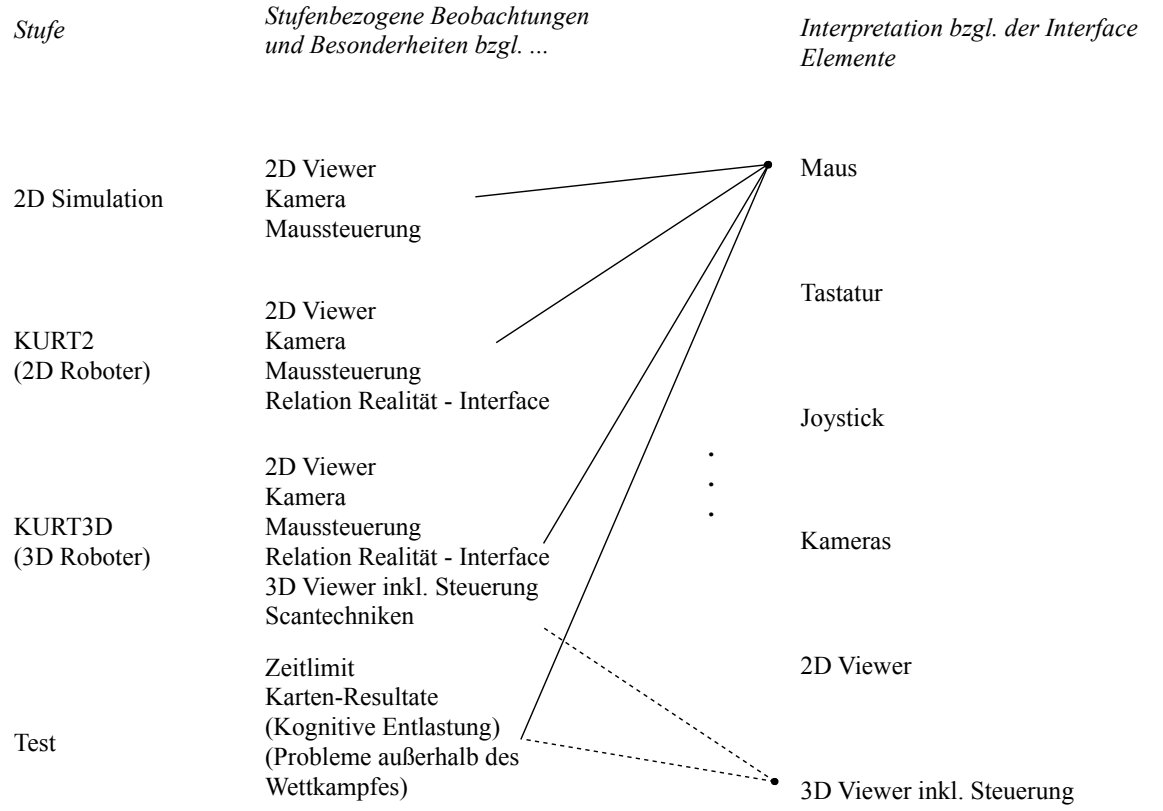


Abbildung 3.4: Darstellung der Kapitelstruktur.

Anfängliche Probleme gab es bei den meisten beim ersten Losfahren. Sie waren verwundert, dass der Roboter nicht losfuhr, obwohl sie die Geschwindigkeit von null auf circa „mittelschnell“ hochgesetzt hatten. Die erwartete Reaktion, dass der Roboter nun auf diese eingetellte Geschwindigkeit beschleunigt, blieb aus, bzw. der Roboter bewegte sich überhaupt nicht. Dass man danach „forward“ klicken bzw. den Joystick nach vorne bewegen muss, benötigte Erklärung.

KURT2 – Der 2D-Roboter

Dieser Roboter besitzt eine zentrierte Kamera, die mit den Slidern am Display oder dem Cooliehead bewegt werden kann. Die meisten Versuchspersonen nutzten diese Funktion anfangs, mussten aber danach feststellen, dass sie nicht mehr wissen, wo die Grundposition ist. Das Fehlen einer Reset-Funktion wurde deshalb von fast allen bemängelt. Da nun

der 2D-Scan deutlich schneller als in der Simulation dargestellt wurde und die Versuchspersonen sich immer stärker auf diese Information einließen, verlor die Kamerainformation an Wichtigkeit. Das wurde dadurch unterstützt, dass der Cooliehead nicht invertiert und die Steuerung der Kamera folglich für einige Versuchspersonen nicht intuitiv war.
Schon in dieser Phase der Einarbeitung merkte man, welche Versuchsperson gewissenhafter mit dem Roboter umgeht. Obwohl alle angewiesen wurde, Kollisionen zu vermeiden, gab es durchaus Fälle, in denen eine Kollision provoziert wurde. Auf Nachfragen wurde dann gesagt, dass es dazu diene, die Grenzen und Relationen von 2D-Scan und Realität auszuprobieren.[3]
Die Versuchspersonen lernten spielerisch mit dem Roboter umzugehen. Es wurde viel mit Geschwindigkeit und Kurvenfahren experimentiert. Folglich war ein Fortschritt und eine wachsene Vertrautheit zu beobachten.

Kurt3D – Der 3D-Roboter

Der Übergang vom KURT2 zum Kurt3D gestaltet sich vor allem durch den Umgang mit dem 3D-Scans als schwierig. Die Versuchspersonen wurden erst eingewiesen und bekamen dann viel Zeit, um sich mit den Ansichten, Bewegungen und dem manuellen Einlagern auseinanderzusetzen. Da alle Erfahrung mit 3D-Programmen mitbrachten, wurden auch entsprechende Erwartungen an den 3D-Viewer und dessen Steuerung gestellt. Der unangenehmste Fehler verwehrte den Versuchspersonen ein für sie intuitives Umgehen mit den 3D-Scans: Klickt man mit der Maus in den 3D-Viewer hinein, stürzt das Programm ab. Die Gewohnheit, mit der Maus die Ansichten zu verändern, wurden somit jedesmal „mit einem Absturz bestraft“, wie es eine Versuchsperson formulierte.
Die Versuchspersonen mussten erlernen, mit welchen Funktionen sie die Scanpunkte sichtbar machen, wie sie die Scanposition („scan location“) anzeigen lassen können und die Ausrichtung des Roboters. Dies passiert alles in dem Menü rechts neben dem Viewer, die sehr viele verschiedene Funktionen und Anzeigemodi zur Verfügung stellt. Meistens ist diese Leiste hinter anderen Fenstern versteckt, da die Fensteranordnung eigentlich für einen 16:9-Monitor optimert ist, der normalerweise im Wettkampf zur Verfügung steht. Diese Funktionen werden einmal angewählt und bleiben dann auch bei weiteren Scans erhalten. Folglich müssen die Fenster nicht andauern verschoben werden. Es fiel den Versuchspersonen dennoch negativ auf.

Ist ein Scan im 3D-Viewer sichtbar, kann die Perspektive geändert werden. Mit einer fliegenden Kamera kann man sich als Beobachter durch die Punkte hindurch bewegen. Dabei muss die Versuchsperson lernen, mit den Steuerungselementen, die unter dem Viewer angegliedert sind, umzugehen.
Das manuelle Einlagern eines vom System nicht korrekt eingelagerten Scans wurde von einigen Versuchspersonen sehr schnell erlernt, andere aber brauchten viel Zeit, um damit zurechtzukommen. Einige lernten es sehr schnell, indem sie sich die entsprechenden

[3] Vorausgreifend darf hier gesagt werden, dass dieses Ausloten der Grenzen nicht zu einer sichereren Navigation geführt haben.

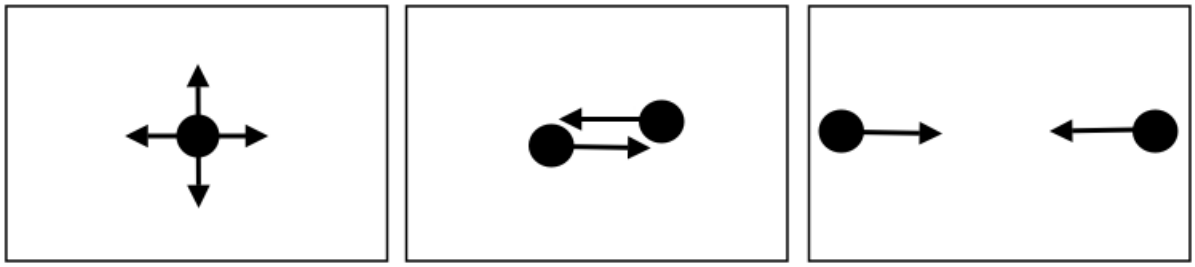


Abbildung 3.5: Die verschiedenen Techniken des Scannens

Tastenkombinationen, von denen eine Liste neben dem Computer bereit lag, durchlasen und dann systematisch ausprobierten. Andere lernten durch einfaches Ausprobieren der Tasten.
Auffällig interessant waren die drei verschiedenen Methoden, mit denen dann die Versuchspersonen den Raum scannten, in dem sie saßen (siehe Abbildung 3.5). Bei der ersten („Windrose“, in der Abbildung 3.5 links) steht der Roboter in der Mitte des Raumes und scannt viermal, wobei er nach jedem Scan um 90 Grad gedreht wird. Die zweite („zentral, überlappend“, in der Mitte der Abbildung 3.5) benötigt nur zwei Scans: Der Roboter fängt mit dem ersten Scan circa zwei Drittel des Raumes ein, wird dann ein Stück nach vorne gefahrend und anschließend um 180 Grad gedreht. Der dann folgende zweite Scan überlappt mit dem ersten und kann gut eingegliedert werden, vorausgesetzt die Überlappung ist groß genug und markant. Die letzte Methode („peripher, überlappend“, rechts in der Abbildung 3.5) benötigt auch nur zwei gegenläufige Scans, die aber mit „dem Rücken an der Wand“ erfolgen. Dadurch entsteht eine sehr große Überlappung.

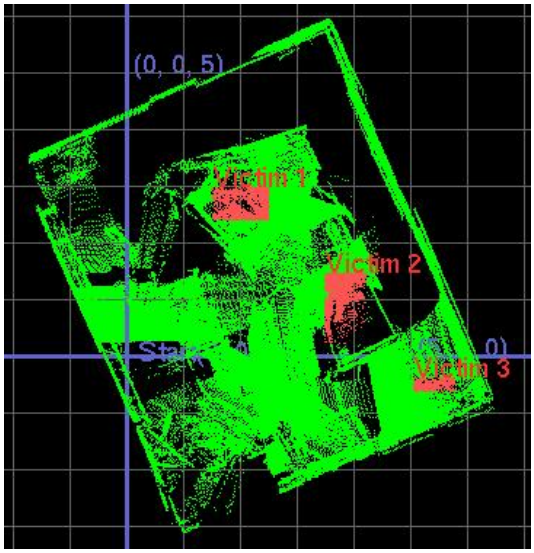
Dabei stellten die Versuchspersonen sehr schnell fest, dass die automatische Einlagerung des Scans meist dann am besten funktioniert, wenn der Roboter möglichst wenig gedreht wurde[4].
Nachdem die Versuchspersonen erlernt hatten, mit dem 3D-Viewer umzugehen, wurde ihnen gezeigt, wie Opfer in der 3D-Karte markiert werden. Dies wurde von allen sehr schnell gemeistert. Dass eine Markierung („Box“) nicht wieder gelöscht werden kann, fiel schnell negativ auf.

Test

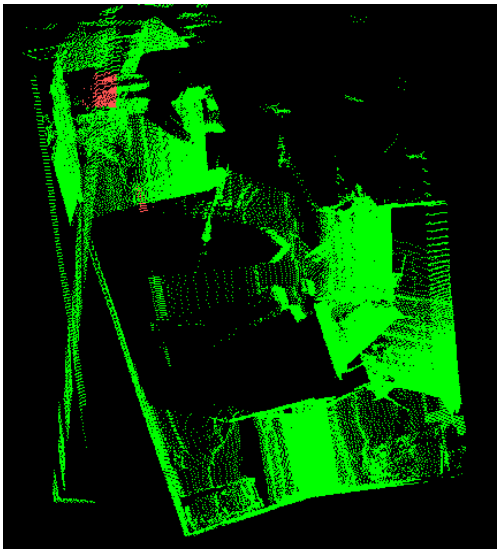
Den Versuchspersonen wurde diesmal ein 30 Minuten Limit gegeben, um einen für sie unbekannten Raum sinnvoll, d.h., die Realität gut widerspiegelnd, zu kartografieren und die Opfer in die Karte einzutragen. Der Schwierigkeitsgrad der Testarena wurde von den

[4]Dies müsste eigentlich dazu führen, dass die Kameras mehr gedreht werden, anstatt den ganzen Roboter zum „Gucken“ zu benutzen. Dies war aber nicht der Fall.

Versuchspersonen sehr verschieden empfunden. Die resultierenden Karten geben diese Meinung über den Parcours wieder: Eine gute Karte wurde von einer Versuchsperson gefertigt, die die Arena als einfach empfand (siehe Abbildung 3.6(a)). Man sieht, dass die Versuchsperson alle drei Opfer rot markiert hat, zudem ist das 3D-Raster eingeblendet: nur wenn diese Funktion aktiviert ist, sieht man die Beschriftung der Opfer mit „Victim X“.



(a) Die erstellte Karte einer Versuchsperson, die sich gut orientieren konnte und den Test als einfach empfand.



(b) Die Karte stammt von einer Versuchsperson, die schnell die Orientierung velor.

Abbildung 3.6: Die unterschiedlichen Resultate.

Die ungenaueste Karte (siehe Abbildung 3.6(b)) stammt von einer Versuchsperson, die schon am Anfang die Orientierung verloren hat und überraschend schlecht navigierte. Obwohl sehr genervt, konnte genau diese Versuchsperson im Nachhinein sehr präzise und für die Evaluation wertvolle Informationen geben. Interessanterweise sind ihr kleine Facetten und Probleme bei der GUI aufgefallen, die ansonsten kein Befragter von alleine erkannt hat.
Bei dem Test zeigte sich sehr schnell, wie gut die Versuchsperson die Situation, in der sich der Roboter befindet, einschätzen kann. Manchmal kam der Roboter zu nah an das Opfer und verschob es. Ebenso eckte der Roboter einige Male an. Zwei Versuchspersonen haben den Roboter sehr präzise gesteuert und sind mit ihm nirgendwo angestoßen.
Da gleich nach dem Test das Video angeschaut wurde, war es kein Problem für die Versuchspersonen, ihre Schritte nachzuvollziehen und zu kommentieren.

Kognitive Entlastung Bei der Nachbesprechung stellte sich heraus, dass es anscheinend sehr wichtig war, als erstes einen 3D-Scan zu machen, um einen Bezugspunkt für die Navigation und folgende Kartierung zu haben. Diejenigen, die erst mit dem Roboter

gefahren sind und erst „als es interessant wurde“ einen Scan machten, hatten Probleme mit der Orientierung und Einlagerung des Scans. Es scheint ein Vorteil zu sein, die kognitive Arbeit des „Verfolgens der Roboterbewegungen im Kopf“ durch einen ersten Scan am Anfang zu minimieren. So muss nicht vor dem inneren Auge der Weg verfolgt, sondern direkt am Monitor nachvollzogen werden.
Eine Funktion, die bei der Anlernphase wenig benutzt wurde, gewann im Test an Bedeutung: Das Anzeigen der Scanposition („scan location“) des Roboters (siehe Abb. 3.7). Diese Darstellung des Roboters im 3D-Viewer erschien wahrscheinlich deswegen als hilfreich, da der Raum unbekannt war. Für ein paar Versuchspersonen war es einfacher, die Bewegungen des Roboters nachzuvollziehen, wenn sie die letzte Scanposition am Monitor sahen. Sicherlich ist es einfacher, sich das letzte Wegstück zu merken, als den ganzen Bewegungsablauf seit Anfang der Fahrt. Der gefahrene Pfad konnte somit vergessen werden und belastete nicht weiter die Arbeit des Operators. Beispielhaft ist die Karte in der Abbildung 3.7, in der die Scanpositionen eingeblendet sind. In der Mitte unten, also direkt an der Türe, wurde gleich zu Beginn der erste Scan gemacht. Der gefahrene Weg kann gut nachvollzogen werden. Diese Versuchsperson hat, trotz Testsituation, ausgesprochen präzise navigiert und kartografiert.

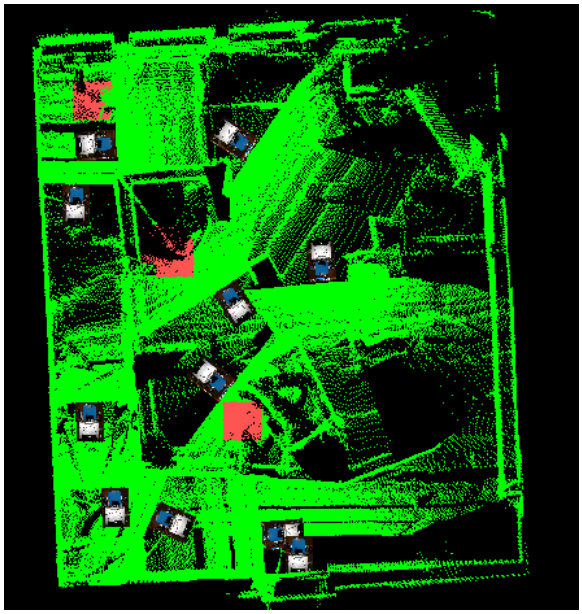


Abbildung 3.7: Eine von einer Versuchsperson erstellte Karte mit Anzeige der Roboter-Positionen.

Probleme außerhalb des Wettkampfes Eine wachsende Problematik, die ein paar Mal in den Nachgesprächen angesprochen wurde, war, dass mit jedem 3D-Scan der Computer langsamer wurde[5]. Dies hatten die Versuchspersonen schon vorher in der Einar-

[5]In der Abbildung 3.7 sind zum Beispiel elf Scans gemacht worden.

beitungsphase bemerkt. Ohne Zeitdruck empfanden sie es aber nicht als schlimm oder belastend. Nun aber wurde dieser Punkt kritischer, umso weniger Zeit sie hatten. Der Operator gerät immer mehr unter Stress und muss dabei mit einem immer langsamer werdenden Computer umgehen.
In einem USAR-Wettkampf dagegen stehen nur 10 Minuten (früher 20 Minuten) zur Verfügung. Folglich können nicht so viele Scans wie in diesem Versuch gemacht werden, wie ein Experte zu dieser Problematik erklärte. Deshalb bemerkt der Operator kaum die wachsende Trägheit des Rechners[6].
Ähnlich verhält es sich mit der manuellen Einlagerung eines Scans: Je mehr Scans gemacht werden, desto mehr redundante Punkte tauchen im 3D-Viewer auf. Eine Versuchsperson meinte, dass „zu viele Punkte“ vorhanden gewesen seien, so dass er den Scan nicht mehr einlagern konnte, weil er fast „nur noch grüne Flächen“ sah. Da er die Scans ansonsten an kleineren Merkmalen ausrichtete, nun aber keine mehr sehen konnte, gab er auf und fing an, Scans zu löschen. Auch dieses Problem stellt sich den Experten im Wettkampf nicht, da, wie oben erwähnt, nicht so viele Scans in der kurzen Zeit gemacht werden können.
Bei dem Markieren der Opfer traten auch manchmal Probleme auf. Die Markierung („Box“) tauchte in dem 3D-Viewer sehr weit draußen auf. Zuerst wurde sie dann von den Versuchspersonen, bei denen das Problem auftrat, nicht gefunden. Erst nach langwierigem Herauszoomen tauchte die Markierung auf und musste anhand der Tasten (im Nummernblock der Tastatur) lange verschoben werden. Das kostete die Teilnehmer nicht nur Zeit, sondern auch Nerven, wie man an den Videos erkennen konnte. Im Nachgespräch stellte sich weiter heraus, dass es vor allem das Vertrauen in die Funktionalität der GUI verminderte. Es wurde vorgeschlagen, dass diese Markierungsbox standardmäßig in der Mitte des letzten Scans auftaucht. Ein Experte meinte dazu, dass ihm das noch nie passiert sei und das ein Operator im Wettkampf leider aus Zeitgründen nur sehr selten überhaupt Opfer markieren kann.
Ein weiterer Kommentar bezüglich der Opfermarkierung war, dass die Farbe „viel zu unauffällig“ gewählt wurde. Die gelbliche Farbe sei in dem Scanbild kaum zu sehen. (Wenn die Markierungsarbeit abgeschlossen ist, wird die Markierung rot.) Zudem wäre es sinnvoll, wenn man die Markierung mit der Maus auf die Stelle im 3D-Scan ziehen könnte.

3.3.2 Interpretation bezüglich der Interface Elemente

In den folgenden Abschnitten soll nun der Fokus nicht mehr auf den einzelnen Stufen des Einlernens (siehe 3.3.1) liegen, sondern auf den verschiedenen Elementen des Interfaces. In der Abbildung 3.4 entspricht dies der rechten Spalte. Hier werden die Beobachtungen, Kommentare und Tendenzen aus den strukturierten Interviews der Novizen und Experten miteinbezogen.

[6] Für eine Anwendung des Systems in einem Ernstfall, wo der Roboter vielleicht eine Stunde operieren müsste, wäre diese Verlangsamung in keinster Weise tragbar, zumal auf den Operator dann noch der Stress des Ernstfalles (anstelle des künstlichen Wettkampfzeitdrucks) wirkt.

Die Eingabegeräte

Für das Re-Design sollen die Eingabegerät eigentlich nicht mit einbezogen werden, denn größere Änderungen an der Hardware sind dabei nicht vorgesehen. Für den Ablauf und den Erfolg einer Mission ist die Auswahl der Eingabegeräte sicherlich sehr relevant. Eine Änderung der Tastenbelegung, eine Erweiterung oder Einschränkung der zugeschriebenen Funktionalität etc., kann großen Einfluss auf den Operator haben.

Die Maus

Betrachtet man die Maus als ein Eingabegerät für die Robotersteuerung, so könnte man sie, nach Meinung der Novizen, wegrationalisieren. Das Navigieren lief bei ihnen nur über den Joystick selbst, bei manchen auch vermehrt über die Joysticktasten, am wenigsten aber wurde die Maus benutzt. Die Gegenaussagen bzgl. des Joysticks und der Joysticktasten bestätigen die Beobachtung: Die Maus wird von den Novizen nicht für die Naviagation benutzt. Zwei Novizen entschieden sich bei der Aussagen, ob mit der Maus eine gezielte Steuerung des Roboters möglich sei, dazu, keine Angabe zu machen. Sie zögen diese Art der Steuerung gar nicht mehr in Betracht und könnten sich nicht mehr daran erinnern, wie es war, mit der Maus zu steuern. Auf die Nachfrage, ob sie denn nicht erst abgewogen haben, wie sie den Roboter am präzisesten steuern könnten, bevor sie die Maus einfach „vergessen“, meinte einer, dass die Joysticktasten doch die gleiche Funktionalität und Präzision hätten.[7]
Den Novizen gegenüber stehen die reinen „Maus-Navigierer“ unter den Experten. Eventuell liegt es daran, dass sie nicht immer einen Joystick zur Hand haben und eine Maus eigentlich immer an dem Rechner angeschlossen ist. Der Aufwand, nur die Maus zu benutzen, ist damit deutlich geringer und das mündet dann vielleicht in eine Gewohnheit. Diese zwei Experten sind der Meinung, dass durch die Maus eine gezielte Steuerung möglich ist. Die anderen Experten nutzen die Joysticktasten am meisten. Den Joystick selber benutzen sie für schnellere Fahrmanöver (z.B. auf einer etwas längeren, freien Fahrt oder um mit mehr Geschwindigkeit über ein kleines Hindernis zu fahren).
Gäbe es für die Maus nur die Aufgabe der Navigation, könnte man sie fast als redundant bezeichnen. Die Maus hat aber noch weitere Aufgaben: Mit ihr kann man die Perspektiven im 3D-Viewer verändern. Diese Funktionaliät ist auch durch Tastenkombinationen, die noch aus „Prä-Maus“-Zeiten stammen, zu erreichen. Eine Funktion, die nicht von der Tastatur übernommen wird, ist das Aktivieren von Fenstern im grafischen Interface. Wenn das rechte Fenster aktiviert ist und der Operator z.B. einen Scan manuell einlagern will, er also die Tastaturkombination „Alt+m“ drückt, so bleibt dieser Shortcut wirklungslos und der manuelle Modus wird nicht aktiviert. Erst wenn mit der Maus das Fenster des 3D-Viewers aktiviert wird, hat der Shortcut die erwünschte Wirkung und der Scan kann eingelagert werden. Einige Versuchspersonen fühlten sich dadurch in ihrem Arbeitsfluss stark gestört, sogar genervt. Dieses Umgreifen ist nicht akzeptabel. Dennoch wurde von

[7] Der Vollständigkeit zuliebe darf hier gesagt werden, dass die andere Versuchsperson einfach (sichtbaren) Spaß hatte, mit dem Joystick zu steuern.

keiner Versuchsperson der „allgemeingültige“ Shortcut „Alt+Tab“ ausprobiert, der das andere Fenster aktivieren kann.
Es sei hier auch erwähnt, dass eine Versuchsperson die Maus für die Bewegung der Kameras benutzte. Die Steuerung am Cooliehead war ihr zu ungenau, vor allem als sie versuchte, den Horizont beider Kameras anzugleichen. Andere Versuchspersonen waren verwundert, dass sie auf die Pfeilknöpfe klicken mussten und nicht einfach den Slider mit der Maus ziehen konnten. Dies ist auch eine Funktionalität, die zwar erwartet wurde und auch durch die Form des Sliders motiviert wird, aber nicht implementiert ist.

Die Zukunft der Maus Die Maus ist in den letzten Jahren zu einem Standardeingabegerät für den Computer geworden. Die Maus ist so wichtig, dass die meisten Nutzer ohne sie mit ihrem Computersystem nicht mehr umgehen können. Die wenigsten Programme sind nur noch mit Tastatur zu bedienen. Wenn nun eine Maus neben einem Rechner steht, will man sie auch benutzen und auch wie im Alltag damit umgehen können. Wie Nutztungsweise bei dieser GUI ist aber so eingeschränkt, dass eine Maus, mit kleinen Ausnahmen, nicht notwendig ist und sogar Fehler im Arbeitsfluss mit provoziert[8].
Diese Arbeit soll keinesfalls den Eindruck erwecken, dass die Maus für dieses Interface nicht sinnvoll ist und abgeschafft werden kann. Im Gegenteil: Die Maus muss mehr Funktionen übernehmen, – Funktionen, die der Nutzer erwartet, gewohnt ist und auch erwünscht. Dabei ist die erste Intuition der Novizen ein wichtiger Hinweis. Sie haben alle gedacht, dass der 3D-Viewer mit der Maus bedient werden kann[9]. Oft nannten die Versuchspersonen Programme, wie z.B. Acrobat Reader oder 3D-Applikationen, bei denen es möglich ist, das gezeigte Objekt mit der Maus „anzufassen“, d.h., es wird angeklickt und kann mit gedrückter Taste dann rotiert oder verschoben werden. Sicherlich kann man weiterhin dagegen argumentieren, dass diese Funktionen alle durch Shortcuts abgedeckt sind und man diese schnell durch häufiges Benutzen erlernt. Dennoch erscheint diese riesige Ansammlung von Shortcuts eher ein Überbleibsel von der „Prä-Maus“-Zeit zu sein, dessen Funktionalität noch nicht sinnvoll dem neuen Eingabegerät übergeben wurde. Für den Perspektivenwechsel wird die Maus sehr wahrscheinlich sinnvoll und intuitiv einsetzbar sein. Ein Nebeneffekt wird sein, dass die beim strukturierten Interview bemängelte Steuerung des 3D-Viewers dadurch verbessert werden könnte.
Der 3D-Viewer hat aber noch mehr Funktionen als nur das Darstellen der Scans. Die größte Schwierigkeit stellt das manuelle Einlagern der Scans dar. Es ist geplant, dass dies in den kommenden Jahren nicht mehr nötig sein wird, da das System die Scans dann perfekt einlagern können wird. Dennoch ist momentan die Problematik noch vorhanden. Es stellt sich damit die Frage, ob das Einlagern mit der Maus einfacher gehen könnte. Eine mögliche, negative Antwort könnte damit begründet werden, dass das Einlagern anhand der Tasten schrittweise abläuft, d.h., der Scan wird immer in einem bestimmten Intervall rotiert oder

[8] Rechtshänder stießen während oder nach der Benutzung der Maus mit der rechten Handkante an einen Button des Joysticks. Dieser Button ist zudem noch unglücklicherweise mit dem Auslösen eines 3D-Scans belegt.

[9] Wie schon oben erwähnt, führt ein Mausklick in diesem Viewer zu einem Systemabsturz. Dieser hat zwei Wirkungsweisen: Einerseits lernt der Nutzer schneller, dass er auf keinen Fall mit der Maus dort hinein klicken darf, andererseits potenziert er die Benutzerunfreundlichkeit.

verschoben. Tippt man zweimal die Taste, dann bewegt er sich um das doppelte Intervall. Mit der Maus gäbe es diese diskreten Schritte nicht, sondern der Scan wäre (eventuell) mit der Maus frei drehbar. Die Erfahrung mit der Tastatursteuerung zeigt, dass der einzulagernde Scan nicht genau in die neue Lage gebracht werden muss, sondern dass eine Annäherung ausreicht. Das System „matcht“ den Scan dann automatisch in die neue, bzgl. der Karte besser passende Lage ein.
Indem man die Ideen der Novizen ernst nimmt und den heutigen Standard für Computerbedienung miteinbezieht, ergibt sich eine Erwartungskonformität. Wenn man dann nicht durch dieses schwer wiegende Argument der Erlernbarkeit die Erwartungskonformität „wegdiskutiert“, ergäbe sich sicherlich eine leichter zu bedienende (also kognitive entlastende) Oberfläche.

Die Tastatur

Dieses Eingabegerät ist nicht ersetzbar. Es stellt sich nicht einmal die Frage, ob es sinnvoll ist, sie mit Shortcuts zu belegen. Es gibt Nutzer, die am liebsten nur die Tastatur für die ganze Mission benutzen würden. Es wurde von einer Versuchsperson anfangs gefragt, ob nicht der Roboter vollständig anhand der Tastatur bedienbar sei[10]. Mehrere rieten dazu, die Maussteuerung in der GUI durch die Cursor-Tasten auf der Tastatur zu ersetzen. Sicherlich sind Tastenkürzel im professionellen Umgang mit Programmen notwendig. Wie schon oben erwähnt, ist es aber auch nicht nur eine Frage der Erlernbarkeit.
Tastenkombinationen sollen zwei Anforderungen genügen: Einerseits intuitiv, z.B. den Anfangsbuchstaben von Befehlen folgend (z.B. das klassische “Control+c“ für „copy“, also Kopieren); andererseits soll ein gewisser sinnvoller Abstand zwischen den Tasten liegen. „Sinnvoll“ kann hier bedeuten, dass der Abstand „sinnvoll klein“ ist („Control+v“ für „paste“, also Einfügen, um das Beispiel weiterzuführen), oder dass der Abstand „sinnvoll groß“ ist (Sicherheitsabstand), damit nicht aus Versehen eine falsche Aktion ausgelöst wird. Zudem sollte der Shortcut immer noch in der Reichweite einer Handspanne liegen und maximal drei Finger einer Hand und niemals mehr als zehn Finger involvieren: Ein Shortcut muss viele Bedingungen erfüllen. Wenn dann die Tastenkombination „Alt+M“ eine andere Funktion verbirgt als “Alt+m“, dann sollte das ein Alarmsignal sein. Die Auflistung im Kapitel A auf Seite 93 und folgende spricht wohl für sich selbst.
Eine Veränderung, die vorgenommen werden kann, ist, eine gute, d.h., eine leicht nachvollziehbare Belegung der Tasten zu entwerfen. Da sich inzwischen sehr viele Tastenkürzel so stark etabliert haben, sind diese kaum mehr abänderbar. Dies soll nicht unterstellen, dass die momentane Belegung schlecht ist. Für einen neuen Belegungsplan der gesamten Tastatur würden nur einige Kürzel verlagert werden. Vergleichbar wäre dies dann mit dem Erlernen des Zehnfingersystems für schnelles Schreiben: Erlernt man es ohne Vorkenntnisse, dann ist es einfach. Hat man aber schon einen eigenen Schreibstil gefunden, fällt es sehr schwer, die alten Gewohnheiten aufzugeben und sich zum neuen System zu zwingen.

[10] Es ist hier hinzuzufügen, dass diese Versuchsperson mit dem 3D-Viewer und dem Einlagern der Scans später mit „genügend vielen“ Shortcuts zu tun hatte.

Der Joystick

Der Joystick ist nicht nur ein Eingabegerät, sondern vereint eigentlich drei verschiedene Eingabegeräte. Im unteren Teil befinden sich die Tasten und die „Flosse“, mit der man die Geschwindigkeit einstellen kann. Ganz oben ist der Cooliehead, der mit der Steuerung aus Auswahl der Kameras belegt wurde (abgesehen von drei Tasten, die andere Funktionen verbergen). Dazwischen ist der Joystick-Stick, also der „Joystick selber“, mit dem der Roboter gesteuert werden kann.

Die unteren Joysticktasten Die Belegung der Tasten ist nicht groß bedacht worden, sondern wurde schnell vor einem Wettkampf gemacht. Sie erhebt weder den Anspruch, intuitiv noch sinnvoll zu sein. Die Anordnung der Tasten suggeriert auch keine offensichtliche Belegung, zumal diese Belegung eigentlich den „Joystick an sich“ in Frage stellt. Die Aussagen des strukturierten Interviews zeigen, dass auch die Versuchspersonen nicht einer Meinung sind, wenn es um die gezielte Steuerung des Roboters geht. Dass die Joysticktasten dabei noch eher diese Funktion erfüllen, ist klar. Dennoch wird dem Joystick an sich nicht unbedingt eine „unkontrollierte“ Steuerung zugeschrieben, sondern die Befragten sind der Meinung, dass er nicht unpräzise aber auch nicht präzise ist.

Der Joystick-Stick In Betracht sollte man bei der Diskussion um den Joystick ziehen, dass der Vorteil der Steuerung mit dem Stick ist, dass die Bewegung des Roboters der des Armes des Operators entspricht. Ein „nach vorne Fahren“ ist auch beim Joystick eine Vorwärtsbewegung. Analoges gilt für die Rückwärtsbewegung. Bei der Rotation nach rechts oder links ist zu bemerken, dass eine Rotation des Joysticks um die vertikale Achse nicht mit der Rotation des Roboters auf der Stelle korrespondiert. Um den Roboter zu drehen, muss der Joystick nach links bzw. rechts „gelegt“ oder gekippt werden. Wäre die Belegung des Joysticks auch bei der Rotation analog zur Rotation zum Roboter, wäre das Kurvenfahren wenig ergonomisch und auf Dauer vielleicht schmerzhaft. (Da das Kippen des Joysticks zu keiner Bewegung des Roboters direkt korrespondieren könnte, erscheint diese Belegung deutlich sinnvoller.) Diese Art von Feedback, die der Körper durch den Joystick bekommt, ist durchaus sinnvoll. Ein Force-Feedback-Joystick würde dies noch verstärken.

Das obere Joystick-Teil Der obere Bereich des Joystick umfasst mehrere Tasten, die um den Cooliehead herum angebracht sind. Die Idee, die Kamerasteuerung unter anderem durch die Bewegung des Coolieheads zu bewerkstelligen, fand bei den meisten Versuchspersonen Anklang. Obwohl sie nicht viel benutzt wurde, gab es auch hier wieder den Unterschied zwischen Nutzern, die eine invertierte Bewegung des Coolieheads bevorzugen, während andere dies eben nicht tun. Bezüglich der Invertierung gab es hier wiederum eine Problematik: Wählt man die eine Kamera aus, so ist die Steuerung invertiert. Die Steuerung der anderen Kamera ist aber nicht invertiert. Aufgrund der „Kameraträgheit“

der Versuchspersonen, fiel das kaum auf, wenn es auffiel, dann stiftet es Verwirrung und Verwunderung.
Eine wichtiger Unterschied zwischen dem 2D-Roboter und dem 3D-Roboter ist, dass auf dem Feuerknopf Kameraroutinen liegen. Dadurch können die vorher einzeln bewegten Kameras wieder in eine der Grundpositionen gebracht werden. Wenn von den Versuchspersonen die Kameras benutzt wurden, dann meist mit diesen Routinen, die beide Kameras nacheinander nach unten, außen und nach vorne bewegen.
Die beiden oberen Tasten wurden kaum benutzt. Mit der linken Taste ist der autonome Modus ein- bzw. ausschaltbar. Die rechte Taste dagegen führt zu einem abrupten Missionsende. Theoretisch stellt sich die Position der Taste, die einen 3D-Scan auslöst, als problematisch dar: Sie liegt zwischen den Kameraauswahltasten und direkt unter dem Cooliehead. Damit liegt sie „mitten in der Kamerasteuerung“. Praktisch aber wurde sie nie aus Versehen betätigt, was nicht zuletzt an den seltenen Kamerabewegungen der Versuchspersonen liegt.

Jede Möglichkeit der Steuerung in diesem Interface hat somit einen Nachteil:

- Die Joysticktasten sind nicht dafür gedacht, die Bewegungen zu kodieren, die eigentlich mit dem Joystick selbst gemacht werden.
- Der Joystick, mit dem eine gezielte, präzise Steuerung schwierig ist.
- (Nicht zu vergessen: Die Maussteuerung. Sie muss in der GUI mit der Maus bedient werden, hat eine sehr schlechte Aufteilung und ist mitten in der GUI zwischen Informationsfenstern eingelagert.)

Für welche Steuerung sich der Operator auch entscheidet, er wird immer, ob bewusst oder unbewusst, mit kleinen Fehlern und Nachteilen zu kämpfen haben. Deswegen wurde schon darüber nachgedacht, den Joystick durch ein Gamepad zu ersetzen. Damit würde die Möglichkeit verloren gehen, mit dem Roboter zügig ein kleines Hindernis zu überfahren. Zudem ist das Einstellen der Geschwindigkeit ungenauer. Folglich wurde das Gamepad als Eingabegerät vorerst wieder verworfen.

Die Kameras

Wie schon oben mehrmals erwähnt, wurden die Kameras sehr wenig bewegt. Die am Monitor dargestellte Information wurde fast nur für das Auffinden von Opfern vor Ort benutzt. Folglich sind die Kameras sehr wichtig, aber bei der Art der Informationsübermittlung in der GUI „stimmt etwas nicht“.

Unzufriedenheit mit den Kameras Im strukturierten Interview schlug sich Zerrissenheit und Kritik nieder.

- Speziell die Novizen empfanden die Kamerabilder als stockend. Dabei ist mit „stockend“ nicht gemeint, dass sicherlich eine gewisse technisch bedingte Verzögerung vorhanden

ist, wie den Versuchspersonen erklärt wurde. Das Stocken bezieht sich dabei auf ein Phänomen, dass bei den Experten, ihrer Aussage nach, normalerweise nicht auftaucht: Die Kamerabilder blieben einfach „stehen“, d.h., ein Bild wird angezeigt und wird nicht von einem neuen abgelöst. Das Bild friert ein und die Versuchspersonen wunderten sich, dass sie z.B. im 2D-Scan eine Person vor dem Roboter vorbei laufen sahen, aber die Kamerabilder diese Information nicht zeigten. Erst als der Roboter wieder ein kleines Stück bewegt wurde, liefen die Bilder wieder flüssig. Dieses Problem tauchte nicht immer auf, anscheinend aber so oft, dass alle Novizen einstimmt der Aussage sehr zustimmten, dass die Kamerabilder zu stockend sind.

– Die Ausrichtung der Kameras, also die Information, wohin die Kameras eigentlich „schauen“, blieb den meisten Versuchspersonen verschlossen. Dass sich diese Information nicht aus den Kamerabildern gewinnen lässt, bestätigten fast alle im strukturierten Interview. Auf die Nachfrage, wie sie denn dann die Ausrichtung der Kameras erkennen können, wurde meist kurz nachgedacht: Die Kameras wurden ja kaum bewegt und das Problem tauchte damit nicht oft auf, da beide Kameras dann einfach nach vorne ausgerichtet waren und dort auch blieben. Die einzige Quelle für diese Information scheint das Vergleichen eines Kamerabildes mit dem 2D-Scan zu sein. Darüber, ob man die Informationen des 2D-Scans sofort mit den Informationen in den Kamerabildern machten kann, gab es verschiedene Meinungen. Einige meinten, dass dies nicht ginge, andere meinten, dass dies ziemlich gut gehe.
– Ob und wie gut Objekte aus dem 2D- und dem 3D-Scan in den Kamerabildern wiedererkannt werden, wurde noch extra abgefragt. Hier spiegelt die Auswertung des Interviews wider, dass die Versuchspersonen zwar besser mit dem 2D-Scan (als mit dem 3D-Scan) in Verbindung mit den Kamerabildern umgehen können, dennoch war das Maß der Zustimmung unterschiedlich. Einige kamen also mit den verschiedenen Informationsfenstern besser zurecht, als andere. Unter diesem Gesichtspunkt ist auch folgender Punkt sehr interessant:
– Dass die Bilder von zwei Kameras vorhanden sind, empfand die meisten Versuchspersonen wenig bzw. gar nicht verwirrend. Nur einer meinte, es verwirre ihn ein bisschen und stimmte der Aussage nur mittelmäßig zu. Es ist also eine Tendenz vorhanden, dass die beiden Kamerabildern nicht verwirrend sind. Angesichts der Beobachtung, dass die Kameras wenig gedreht wurden und im Vergleich zu anderen Informationen wenig benutzt wurden, ist dies nicht überraschend.
– Im Weiteren wurde bemängelt, dass die Steuerung der Kameras im der GUI ungenügend sind: Die Slider sind nicht mit der Maus greif- und verschiebbar.
– Die schon angesprochene Invertierung des Coolieheads bei nur einer Kamera wurde als verwirrend empfunden bzw. beobachtet.

Weitere Ursachen und Anregungen Eventuell hätte man die Novizen mehr zum Kameragebrauch drängen können, wenn man sie direkt mit reinen Aufgaben konfrontiert hätte, die nur mit reiner Kameraarbeit zu lösen gewesen wären. Aber die Art, wie die Mission des Kartografierens und des Opferfindens von den Versuchspersonen erfüllt wird,

sollte nicht erzwungen werden. Schließlich ist es jeder Versuchsperson selber überlassen, mit welcher Information und auf welche Art und Weise eine Mission möglichst erfolgreich abgeschlossen wird. Im Mittelpunkt steht ja die Beobachtung, wie die Versuchsperson die Mission zu erfüllen versucht, was Vorteile und was Nachteile bringen kann und wie erfolgreich sie dabei ist.
Eventuell spielt es aber einer Rolle, dass die bestmögliche, realisierte Kamerasteuerung erst bei dem 3D-Roboter geübt werden konnte. Vorher gab es keiner Kameraroutinen und keine Möglichkeit, ohne Sichtkontakt die Ausrichtung der Kameras wieder in eine Grundstellung zu bringen. Somit haben sich die Novizen von Anfang an wenig auf die Kameras verlassen und sie letztendlich nur noch für das Auffinden von Opfern gebraucht. Bei den Experten dagegen werden die Kameras häufiger gebraucht, um Abstände besser abschätzen zu können und um eine Kollision festzustellen, die durch keinen anderen Sensor feststellbar wäre[11].
Ein weiterer Punkt ist, dass in der GUI nicht immer die rechte Kamera auch die rechte Kamera auf dem Roboter sein muss. Es ist zufällig, welcher USB-Anschluss nun rechts oder links dargestellt wird. Sind die Kameras in der GUI vertauscht, ist dies eine massive Verkomplizierung der Wahrnehmung des Operators. Durch den geringen Gebrauch der Kameras von den Novizen, fiel dies aber bei ihnen kaum negativ auf.
Bezüglich der Kamera gab es einige Anregungen. Sie sollten z.B. über einen Zoom verfügen. Eine Kamera wäre auch ausreichend, wenn sie einen 180 Grad Überblick (oder am besten einen Rundumblick) gewährleisten würde. Viele Versuchspersonen meinten auch, dass die Kamerabilder nicht unbedingt nebeneinander in der GUI dargestellt werden müssen. Die linke Kamera könnte also auch durchaus auf der linken Seite der GUI dargestellt werden, und die rechte auf der rechten Seite. Sie machten aber auch die Einschränkung, dass, wenn die Ausrichtungen der Kameras überlappen, dann die Bilder in der GUI auch nebeneinander dargestellt werden sollten.
Dass die Ergebnisse der Haut- und Bewegungsdetektoralgorithmen in den Kamerabildern dargestellt werden, fand nicht großen Anklang und störte die meisten. Der Operator soll durch die eingelagerten Ergebnisse der Detektionsalgorithmen Hilfestellung bekommen. An sich ist das eine sehr gute Idee, wenn die Algorithmen verlässlich funktionieren. Kann sich der Operator nicht auf die Algorithmen verlassen, belasten die aufblinkenden Farbfelder den Operator und kehren damit den eigentlich hilfreichen Effekt um. Solange diese Algorithmen also nicht sinnvoll und korrekt funktionieren, sollten sie nicht dargestellt werden, um die Kamerabilder nicht mit „irrelevanter Information überfluten“.
Die Kameras sind für die Erfüllung der Mission sehr wichtig. Sie kommen zwar nur relativ selten zum Einsatz, wenn, dann reicht der Informationsgehalt aus, um Opfer zu erkennen. Leider können für das Re-Design nur bedingt Änderungen vorgenommen werden.

[11] Ein Beispiel wäre eine Kollision mit einem Fenster, das durch keinen Scan wahrgenommen wird. Nur anhand des Suchens mit der Kamera konnte der Operator erkennen, dass ein Rahmen vorhanden war und somit das seltsame Verhalten des Roboters durch eine Kollision mit einem Fenster ausgelöst sein musste.

2D-Viewer

Der 2D-Viewer (siehe Abbildung 2.4 auf Seite 8) spielt für die Navigation die wichtigste Rolle. Die Aussagen, dass es einem schwer fallen würde, sich ohne den 2D-Scan zu orientieren und dass man sich sehr stark auf den 2D-Scan verlässt, fanden große Zustimmung bei den Versuchspersonen. Diese Aussagen sind die am eindeutigsten zugestimmten von allen.
Wie schon oben angesprochen, herrscht eine gewissen Zerrissenheit über die Möglichkeit, die Punkte aus dem 2D-Scan mit den Informationen in den Kamerabildern in Zusammenhang zu bringen. Ebenso unentschieden sind die Versuchspersonen bei der Frage danach, wie schnell sie die 2D-Scan-Information dann mit der Information in den Kamerabildern matchen können. Gerade bei diesen Aussagen ist eine Selbsteinschätzung sehr schwierig und muss relativ gesehen werden. Hier kann man nur die Tendenz herausfiltern, dass keine Häufungen bei dem einen oder anderen Extrem („stimmt nicht“, „stimmt sehr“) auftraten.

Die 2D-Karte Die Idee einer 2D-Karte, also die schon gescannten Daten in dem 2D-Viewer weiter anzuzeigen, tauchte bei jeder Versuchsperson auf und wurde oft wiederholt. Der Gedankengang der Novizen war meist folgender: Wenn es möglich ist, eine dreidimensionale Karte fast automatisch vom System generieren zu lassen, dann müsste es um so einfacher sein, eine 2D-Karte vom System bauen zu lassen. Zudem wäre die Information, die man dabei gewinnen würde, nicht sehr aussagekräftig, da der 2D-Scan nur genau eine Scanebene darstellt. Eine 2D-Karte kann dem Operator keine Information darüber geben, wie die Welt denn aussehen würde, wenn der Scan nur ein paar Zentimerter höher gemacht worden wäre. Es ist also der Informationsverlust einer Projektion, der die 2D-Karte als wenig sinnvoll erscheinen lässt.
Sehr positiv fiel der 2D-Scan durch seine Schnelligkeit, also mit einer guten Darstellung der Realität, auf. Eine kleine zeitliche Verzögerung ist vorhanden, dennoch war diese den Novizen sofort verständlich und fiel damit nicht negativ auf.

Die Button-Leiste über dem 2D-Viewer Interessanterweise ist die Grundeinstellung des 2D-Viewers passend, denn keine der ganzen Funktionen, mit denen man die Ansicht des 2D-Viewers verändern kann, wurde benutzt. Anfangs probierten die Novizen mit den Einstellungen ein bisschen herum, empfanden einiges als weniger praktisch und kritisierten diese teilweise. Im Endeffekt blieben die verschiedenen Möglichkeiten ungenutzt.
Dem Operator werden also Optionen zur Verfügung gestellt, die er offensichtlich fast nicht bzw. gar nicht braucht. Dafür nehmen diese Optionen einen zu großen Teil der verfügbaren, aber eigentlich stark limitierten Fläche der GUI ein. Zudem sind diese Buttons bzgl. der Kategorisierung aus Kapitel 2.3 (Seite 6) sehr inhomogen: *Display-Funktionselemtente* befinden sich direkt neben *Steuerungselementen.* Sie sind weder räumlich getrennt und sind zu dem optisch angeglichen. Unter dem 2D-Viewer ist dann die Maus-Robotersteuerung, zu der die *Steuerungselemente* aus der obernen Button-Leiste besser passen würden.

Slider und Raster am 2D-Viewer Die Möglichkeit, das Raster zu verschieben, wurde kaum von den Novizen genutzt, wie nicht nur dem strukturierten Interview zu entnehmen war. Interessanterweise sind hier im 2D-Viewer die Slider mit der Maus „anfassbar“ und mit gedrückter Maustaste ziehbar. Bei den Slidern neben den Kamerbilder ist zwar ein Sliderpfeil vorhanden, aber er kann nicht benutzt werden. Hier ist also wieder eine Inkonsistenz vorhanden, die Erwartungen seitens des Nutzers schürt, dann aber nicht erfüllt.
Die Grundeinstellungen des 2D-Viewers wurden meist unverändert beibehalten. Es ist fraglich, wie wichtig das Raster für die Navigation ist. Auch bei einen anderen Aussage stimmten die Versuchspersonen größtenteils überein: Die Koordinaten helfen nicht bei der Abschätzung der Abstände der Objekte. Vielmehr sind für einen Operator die relativen Größenverhältnisse wichtig: Ob zwischen dem Objekt und dem Roboter nun ein Meter oder ein halber Meter liegt, ist irrelevant. Relevant aber ist, wie groß der Roboter in der Relation zu dem Objekt zu sein scheint. Wichtig ist das Augenmaß des Operators, um den Abstand einschätzen. Ein Experte meinte, dass eine Passage auf dem Monitor „ziemlich locker und breit“ aussah, aber die Lage vor Ort eigentlich deutlich enger war[12]. Es stellt sich die Frage, wie wichtig die absoluten Daten von den Abständen sind und ob das Raster inklusive dessen Verschiebungsmöglichkeiten nicht meistens ungenutzt bleiben und wegreduziert werden könnten. Andersartige visuelle Hilfen, z.B. Warnungen, wären vielleicht sinnvoller, wenn es eng wird.

3D-Viewer

Wie schon oben beschrieben, ist der Umgang mit dem 3D-Viewer der zeitaufwendigste Prozess des Einlernens. Dabei soll der 3D-Viewer am Ende der Mission die wichtigsten Informationen beinhalten, die für einen Wettkampf oder in einem realen Ernstfall grundlegend sind. Die sichere Sondierung der Lage, die Kartografierung und die Informationen über Opfer stellen den Sinn und Zweck dieser Roboter dar. In diesem 3D-Viewer steckt viel Programmieraufwand und hier wird permanent verbessert und umgeschrieben. Das wichtigste Ziel ist es, dass das System die Kartografierung automatisch und korrekt selbst durchführt. Es kann hier also mit einer stetigen Verbesserung gerechnet werden.
Die Novizen waren alle von der hohen Auflösung und den Möglichkeiten des 3D-Scanners begeistert. Das strukturierte Interview aber zeigt auf, dass der 3D-Viewer (noch) eine untergeordnete Rolle für den Operator spielt und den 2D-Scan bzgl. der Navigation (noch) nicht ersetzen kann.

– Während sich fast alle sehr stark auf den 2D-Scan verlassen, ist bei dem 3D-Scan genau das Gegenteil der Fall: Außer einem Experten verlassen sich die Versuchspersonen nicht sehr stark bzw. ziemlich wenig auf den 3D-Scan. Sicherlich wäre es schwierig, den Roboter nur anhand des 3D-Viewers zu steuern, da die Odometrie bei Rotationen einfach noch zu ungenau ist. Der Roboter fährt dann eben aus der Punktewolke heraus, obwohl er sich eigentlich z.B. hindurch bewegen sollte. Ein Novize schlug vor, dass der Viewer automatisch dem Roboter folgen sollte. Damit meinte er, dass sich

[12] Analoges sagte er auch über die Geschwindigkeit: Was auch dem Monitor schnell aussieht, ist in Wirklichkeit ziemlich langsam.

nicht der Roboter-Avatar durch den Viewer bewegt, sondern dass die Punkte um ihn herum bewegen. Diese Roboter-zentrische Ansicht ist auch im 2D-Viewer, in dem ja nur die Draufsicht (Parallelprojektion von oben) realisiert ist. Im 3D-Viewer dagegen bewegt sich immer der Avatar, nicht die „gescannte Welt“. Damit ist eine Inkonsistenz vorhanden, die in ihrer Entstehung leicht nachvollziehbar ist: Würde man die Punktewolke um den Avatar bewegen, so wäre dies ein großer rechnerischer Aufwand. Es ist also verständlich, dass die Welt aus Punkten statisch ist, und sich nur der Roboter hindurch bewegt, was ein kleinerer rechnerischer Aufwand ist.

– Die Informationen von Kamerabildern und dem 3D-Scan zu verbinden, scheint auch nicht allzu einfach zu sein. Die Versuchspersonen stimmten der Aussage, ob man die 3D-Scanpunkte direkt mit den Objekten im Kamerbild machten kann, weder „nicht“ noch „sehr“ zu. Wichtig ist hier anzumerken, dass die meisten die Draufsicht (Parallelprojektion von oben) im 3D-Viewer verwenden, während nur insgesamt drei von den Versuchspersonen auch auf die Frontalsicht (perspektivische Projektion von vorne) umschalten. Von diesen drei meinen auch alle, dass sie nur wenig oder mittelmässig gut mit dem Verbinden von 3D-Viewer/Kamerabilder umgehen können. Interessanterweise können genau diejenigen mit dieser Relation am besten umgehen, die die Frontalansicht nie benutzen. Der kognitive Aufwand, um Informationen aus der Relation Frontalansicht/Kamerbild zu ziehen, ist ein kleinerer als der der Relation Daufsicht/Kamerabild, da keine mentale Rotation stattfinden muss. Wenn man diese Ergebnisse kritisch betrachtet, gibt es zwei mögliche Folgerungen:

  * Erstens: Diejenigen, die auch oftmals die Frontalansicht nutzen und dennoch ihrer Aussage nach nicht gut mit der Relation 3D-Viewer/Kamerabild umgehen können, haben eine bessere Selbsteinschätzung als diejenigen, die nie mit der Frontalansicht umgehen und dennoch ihrer Aussage nach ziemlich gut mit der Relation 3D-Viewer/Kamerabild umgehen können. Diese bessere Selbsteinschätzung würde dann vielleicht durch mehr Erfahrung begründet sein.

  * Die zweite Folgerung lautet: Der höhere kognitive Aufwand derjenigen, die zwar nie mit der Frontalsicht (perspektivische Projektion von vorne) umgehen, aber besser mit der Relation 3D-Viewer/Kamera umgehen können, weist darauf hin, dass sie diese mentale Rotation „besser beherrschen“ als die anderen. (Wobei sie natürlich keine Vergleichsmöglichkeit haben.)

Im Endeffekt zeigen diese vermeintlichen Folgerungen, mit welcher Vorsicht diese Aussagen betrachtet werden müssen. Da die Einschätzungen aller Aussagen schon sehr relativ zu sehen ist, ist es sehr unwahrscheinlich, auch stichhaltige Relationen zwischen den Aussagen aufzudecken. Nähme man diese Relationen nun hier als „bare Münze“, so könnte man ohne schlechtes Gewissen die inkonsistente Darstellungsweise des 2D- und 3D-Viewers beibehalten. Die Aussagen sind nur Anhaltspunkte, Gesprächsstoff und wertvolle Ideengeber, nicht mehr aber auch nicht weniger. Eine Verbesserung des Designs müsste sich hier gegen

eine Relation von Aussagen stellen, was leicht fällt, da diese Relation sehr vage ist.

Die 3D-Steuereinheit Unter dem 3D-Viewer befinden sich die Buttons, mit denen die Punktewolke im 3D-Viewer bewegt wird (siehe Abbildung 2.8(a) auf Seite 15). Die Meinungen darüber, ob die gewählten Icons dafür eindeutig sind, gehen weit auseinander. Für ein paar sind die sofort verständlich oder schnell erlernbar, für andere sind sie das nicht. Wie schon in anderen Teilen des Interfaces bemerkt man wieder eine Aufspaltung der Nutzer in zwei Lager: Die einen wollen eine invertierte Funktionsweise, die anderen nicht. Es gibt also Versuchspersonen, die sofort mit den Verschiebungen, die in den Buttons implementiert sind, zurecht kommen; denen gegenüber stehen die anderen, die sich immer wieder mit der für sie nicht eingängigen Steuerung schwer taten. Man sah sie förmlich kurz darüber nachdenken, in welche Richtung nun die Bewegung gehen würde und welche Auswirkungen das auf die Ansicht hätte. Ob diese Steuerung nun invertiert sein soll oder nicht ist, ebenso wie bei der Steuerung der Kameras an den Slidern und dem Cooliehead, eine persönliche Vorliebe jedes Einzelnen. Wie eindeutig also ein Icon ist, hängt mit der Erwartung ab, die ein Nutzer hinter einem Icon erwartet. Und dies kann eben verschieden sein.
Ähnlich aufgespalten sind die Meinungen über die Angemessenheit der Steuerung des 3D-Viewers: Man kann aber eine Tendenz zur Nicht-Angemessenheit finden. Zum Beispiel wurde beobachtet, dass die Novizen nach einigem Ausprobieren kaum mehr Gebrauch von der Rotationskugel machten. Zwei Novizen benutzten sie so lange, bis sie sich überhaupt nicht mehr zurecht fanden. Anfangs erschien die Funktion dieser Kugel sehr interessant, am Ende wurde sie gar nicht mehr beachtet. Die Bedienung ist sehr grob und es passieren schnell Fehler, also ungewollte Rotationen. Einer nannte sie „sehr konfus“ und es sei „schwer, sie sinnvoll zu bedienen“. Während der Benutzung taucht dann ein Grafikfehler auf der Kugel selber auf, der diese dann seltsam verzerrt erscheinen lässt. Es ist unmöglich, daraus noch irgendeine Information bzgl. der Orientierung zu machen. Fast alle stellten die 3D-Scans auf den Kopf und verloren schnell den Überblick bzw. die Orientierung. Da der „reset“ Button dort nicht funktionierte, waren die Versuchspersonen schnell dieses „Verzerrens bis zur Unkenntlichkeit“ überdrüssig. So interessant und gut diese Idee der Rotationskugel auch sein mag, an der Umsetzung hat einiges leider nicht perfekt geklappt.

Die 3D-Funktionenleiste Dieses Fenster (siehe Abbildung 2.6 auf Seite 11), in dem viele verschiedene Algorithmen für eine veränderte 3D-Ansicht untergebracht sind, war für die Novizen nicht von großer Wichtigkeit. Auch während eines Wettkampfes werdem diese Funktionen nur in einem extrem reduzierten Maß benutzt. Gegenteiliges gilt für die Experten außerhalb des Wettkampfs: Vor allem beim Aufspüren von Fehlern und beim Testen ist diese Buttonleiste von zentraler Wichtigkeit. An ihr wird oft etwas geändert, weil durch Praktika, Diplomarbeiten und die ständige Arbeit der Mitarbeiter immer wieder einiges Neues hinzugefügt werden kann. Da es inzwischen so viele Funktionen gibt, wurden diese schon in Kategorien aufgeteilt und in ausklappbaren Menüs „versteckbar gemacht“. Es

werden von den Novizen und unter Wettkampfbedingungen also meist, oder auch immer, die gleichen Funktionen benutzt. Dies spiegelt das strukturierte Interview auch wider. Dabei konnte beobachtet werden, dass die Funktionen erst gesucht werden mussten. Sicherlich ist das bei Novizen zu erwarten, da sie sich erst an die Oberfläche gewöhnen müssen. Nach Aussage eines Experten muss auch er erst die Liste kurz durchschauen, um die gewünschte Funktion zu finden. Er hat nicht gelernt, wo genau sich die bevorzugten Funktionen befinden. Hinzu kommt, dass die Funktionen immer wieder verschoben werden. Es wäre angebracht, die wichtigsten und am meisten benutzten Funktionen leichter ersichtlich zu machen, damit weniger Fehler passieren und weniger Zeit auf die Suche verwendet werden muss.

Das Gesamtbild der GUI

Die Meinung der Versuchspersonen über die GUI ist eigentlich im Durchschnitt ziemlich gut. Obwohl die GUI aus vielen verschiedenen Fenstern besteht, erschwert diese Uneinheitlichkeit nicht die Erfüllung der Mission, meinen sie. Die Frage, ob eine einheitliche Gestaltung wichtig ist, fand keine klare Antwort. Hier schlägt sich schon eine Aufspaltung der Meinungen der Versuchspersonen nieder: Die einen, die oftmals mit einer „unerschütterlichen“ Erlernbarkeit argumentieren, und die anderen, welche meinen, dass eine „sinnvoll aufgeräumte GUI“ unter Stress weniger Fehler verursacht.
Dass sich die für die Mission notwendigen Informationen immer auf dem richtigen Platz auf dem Monitor befinden, stimmten die meisten zu. Diese Aussage beinhaltet einige sehr starke Annahmen: Wenn sich die *notwendigen Informationen immer* auf dem *richtigen Platz* befinden, so deutet die Aussage auf eine sehr gute GUI hin. Auch der darauf folgenden Aussage, die einen Teil der vorherigen Aussagen aufnimmt, („Auf dem Bildschirm finde ich alle Informationen, die ich gerade benötige.“) wird zugestimmt. Eine positive Tendenz kann auch aus der (mehr großen als kleinen) Zustimmung bei der Aussage, dass die Proportionen der Fenster ihrer Wichtigkeit/Relevanz entspricht, erkannt werden. Ein großes Lob für diese GUI wird hier klar ersichtlich.
Die Größe der Fenster und der Informationsgehalt derer scheint für die Operatoren angemessen zu sein. Ein Re-Design ist dennoch nicht sinnlos, denn einige Facetten dieses Interfaces sind durchaus verbesserungswürdig und auf den neuesten Stand zu bringen.

Kapitel 4

Die Grundlagen für das Re-Design

Für das Re-Design dieses Interfaces können und müssen mehrere, verschiedene Quellen herangezogen werden. Diese sind:

- *Human-Robot Interaction (HRI)*, bzw. ein sehr spezieller Bereich der HRI: Wichtigste Quelle ist hier die NIST, das National Institute for Standardization and Technology [6] in den Vereinigten Staaten, unter anderem daran interessiert, Telerobotik Interfaces zu standardisieren.
- *Human-Computer Interaction (HCI)*, um die Erkenntnisse der Psychologie bzgl. grafischer Interfaces auszunutzen. Darunter fällt z.B. Aufmerksamkeit, kognitive Auslastung, etc. Hier werden auch Ideen des Informationsdesign in Betracht gezogen.
- *Allgemeine Design-Richtlinien*, bzw. Heuristiken wie sie von z.B. Jakob Nielsen vorgeschlagen werden, um konkrete, sinnvolle Umsetzungen als Ideenquelle auszunutzen.
- *Andere Ansätze* werden ebenso betrachtet. Das umschließt ähnliche Anwendungsbereiche der Fernsteuerung von Robotern oder Avataren.

4.1 HRI-Erkenntnisse

Unter HRI fallen viele Spielarten der Interaktion: Ob es sich um einen persönlichen Serviceroboter handelt, der mit Menschen kommuniziert, wie z.B. PaPeRo [7], oder einen Roboterarm, der einem Arbeiter bei der Montage hilft [13], oder vielleicht einer Art Roboter-Spielgefährten wie die Robbe Paro [5], ist durch den Begriff der „Mensch-Roboter Interaktion“ nicht genauer spezifiziert. In dieser Arbeit handelt es sich um eine rudimentäre Interaktion mit dem Roboter, da er durch elementare Befehle vom Menschen gesteuert wird. Damit fallen einige soziale Aspekte weg, wie z.B. die Anthropomorphisierung, wie sie in [45] beschrieben wird. Im Mittelpunkt stehen also nur sehr ausgewählte Artikel, die sich speziell um grafische Interfaces für Telerobotik, meist im Bezug auf RoboCup Roboter, drehen.
Die NIST treibt die Forschung u.a. durch Austragung der USAR-Wettkämpfe voran. Für

eine Standardisierung der Interfaces in Sinne der NIST muss sehr viel Vorarbeit und intensive Forschung geleistet werden. Folglich sind deren Erkenntnisse für diese Arbeit sehr wertvoll und können direkt umgesetzt werden. Dr. Jean Scholtz, die bei der NIST in der Information Access Division ein Mitglied der Visualization and Usability Group [14] ist, ist in diesem Gebiet sehr aktiv. Sie ist eine der wichtigsten (Co-)Autorinnen[1] in diesem Bereich. Durch die USAR-Wettkämpfe ist für sie ein direkter Zugriff auf die verschiedenen Teams möglich. Da es nicht immer gestattet ist, während des Wettkampfes die Schnittstellen direkt digital aufzunehmen, müssen manchmal die Ergebnisse und Auswertungen von Videos ausreichen. Zudem werden die Operatoren nach ihrer Teilnahme am Wettkampf interviewt. Dieses Interview dauert nicht lange, da die Operatoren die Wettkampf-Arena relativ schnell räumen müssen. (Einen kurzen Überblick kann man sich mit dem Artikel [27] von Drury, Yanco und Scholtz verschaffen.)

4.1.1 Situationskenntnis

Der Begriff der „situation awareness“ oder auch „situational awareness“ in [16], der hier mit „Situationskenntnis“ frei übersetzt wurde, spielt für das Interface-Design eine zentrale Rolle[2]. Gerade hier liegt der Unterschied zwischen der HCI und der HRI: Innerhalb eines durchaus den HCI-Richtlinien folgendem Interface muss die Präsenz des Roboters für den Operator greifbar, „telepräsent“, gemacht werden. Bei der Teamarbeit zwischen Mensch und Roboter tauchen häufig Probleme auf, deren Ursprung für Scholtz et al. in ungenügender Situationskenntnis zu liegen scheint. Deshalb wurde diese nochmals weiter analysiert und wird in fünf verschiedene Typen aufgeteilt:

- Die *Mensch-Roboter-Kenntnis* bezieht sich auf das, was der Mensch über die Situation des Roboters weiß, z.B. über den Ort und die Umstände, in denen sich der Roboter befindet, den Status in dem der Roboter ist, welche Aktivitäten gerade aktuell sind.
- Mit der *Mensch-Mensch-Kenntnis* meint man das Wissen des einen Menschen über andere Menschen, z.B. also die Kollegen, und was diese im Augenblick machen, in welchen „Status“ sie sich befinden.
- Die *Roboter-Mensch-Kenntnis* umfasst das „Wissen“ des Roboters über die Befehle des Menschen, die für die Ausführung einer Aktion verfügbar sind. Zudem beinhaltet sie das Wissen um jegliche vom Menschen entworfene Bedingungen, die es verlangen, einen Befehl nicht zu befolgen oder einen modifizierten Aktionsablauf einzuschlagen.
- Die *Roboter-Roboter-Kenntnis* setzt den Einsatz von mindestens zwei Robotern voraus. Sie beschreibt das „Wissen“ der Roboter von den Befehlen, die ihnen gegeben wurden, und, falls notwendig, Pläne und Taktiken für eine (dynamische) Arbeitsaufteilung und Koordination der Roboter.

[1] Jean Scholtz hat mit Jill L. Drury und Holly A. Yanco einige wichtige Artikel verfasst, z.B.: [24–27, 49], die hier nun im Mittelpunkt stehen. Deren Inhalt wird in eigener Übersetzung und/oder stark paraphrasiert mit eingebunden.

[2] Der Artikel von Drury et al. [24] gibt einen interessanten Einblick in die verschiedenen Definitionen von „awareness“

– Der *Missionsüberblick des Menschen* beschreibt das Wissen über das Gesamtziel der Mission und über die momentanen Schritte, mit denen das Ziel erreicht werden soll.

Diese Aufteilung in die einzelnen Facetten ermöglicht eine feinere Analyse der verschiedenen Arten der Zusammenarbeit von Mensch und Roboter. Je nachdem, wie beide zusammenarbeiten sollen, können einige Facetten vernachlässigt werden, andere dagegen müssen in den Entwicklungsmittelpunkt rücken. Im Falle eines Mensch-Roboter-Teams im RoboCup gilt es, die Anzahl der Operatoren möglichst gering zu halten[3]. Damit ist die *Mensch-Mensch-Kenntnis* hier nicht relevant, da es meist nur einen Operator gibt. Je nachdem, wieviele Roboter eingesetzt werden, gewinnt die *Roboter-Roboter-Kenntnis* an Bedeutung. in dieser Arbeit wird die Steuerung eines einzigen Roboters betrachtet.
Damit stehen die drei übrigen Facetten, *Mensch-Roboter-Kenntnis*, *Missionsüberblick des Menschen* und die *Roboter-Mensch-Kenntnis*, im Fokus. Je besser diese Schnittstellen sind, desto weniger fehleranfällig wird die Zusammenarbeit sein. Folglich sind sie für das Re-Design von großer Wichtigkeit. Konkrete Tipps, wie man die *Mensch-Roboter-Kenntnis* verbessern kann, geben die Autorinnen auch: Man soll dem Operator eine Karte des gefahrenen Weges geben, ihm also mehr räumliche Information zur Verfügung stellen, damit dem Operator die Situation des Roboters klarer wird. Im Weiteren sollte der Operator einiges über den Zustand des Roboters wissen. Darunter fallen u.a. Roboterneigung und -ausrichtung, Sensorenstatus und die Ausrichtung der Kameras in Relation zum Roboter.
Wie man die *Roboter-Mensch-Kenntnis* verbessern kann, hängt sehr stark von den implementierten Algorithmen und Fakten ab. Je mehr „Wissen“ der Roboter über mögliche Befehle etc. hat, desto stärker verändert sich seine Rolle von einem bloßem Werkzeug zu einem Partner[4]. Bis es soweit ist, dass der Roboter einen taktischen Fehler eines Menschen erkennen kann, wird es noch einige Jahre dauern. Auf einer elementareren Ebene aber könnte man an einen Algorithmus denken, der den Roboter vor Kollisionen schützt, obwohl ihn der Operator dorthin (ohne Kollisionsabsicht) gesteuert hat. Der Roboter würde dann den Befehl verweigern und nicht weiterfahren, sondern mit Sicherheitsabstand vor dem vermeintlichen Kollisionsobjekt anhalten. Dieser Modus wird von Baker et al. in [21] „sicherer“ Modus genannt. Sicherlich müsste diese automatische Hilfe auch durch den Operator abgeschaltet werden können[5]. Ein weiterer Modus, der in [21] als „geteilter“ Modus bezeichnet wird, ist teilweise auch schon beim Team Deutschland1 umgesetzt ist. In diesem Modus kann der Operator einen Punkt bestimmen, zu dem der Roboter selbstständig fahren soll.
Auf „GUI-Ebene“ kann man die *Roboter-Mensch-Kenntnis* indirekt verbessern. Da das Roboter-Mensch-Team eigentlich über die GUI „kommuniziert“, spielt diese auch eine elementare Rolle in der Verbesserung der Kommunikation dar. Durch visuelle und akustische

[3]Ein anderer Artikel über die Wichtigkeit der Situationskenntnis in der HRI wird in [17] dargestellt. Dabei handelt es sich um ein Mensch-Roboter-Einsatzkommando für das Aufspüren und Sicherstellen von chemischen, biologische, radioaktiven oder explosivem Gefahrengut („CBRNE device search and rescue“)

[4]Diese Interaktion könnte man dann eine „Peer-to-Peer Human-Robot Interaction“ nennen, wie sie in einem Artikel [29] beschrieben wird, der wiederum von Dr. Jean Scholtz mitverfasst wurde.

[5]Der autonome Modus dieses Systems weist eine ähnliche „Verhaltensweise“ schon auf: Der Roboter dreht sich bei der Detektion eines sich annähernden Objekts in eine andere Richtung. Das ist eine Notwendigkeit für einen autonomen Modus.

Signale verschiedenster Art kann der Operator z.B. gewarnt werden. Im Endeffekt soll der Operator dadurch eine Entscheidungshilfe bekommen. Es könnte sogar soweit gehen, dass dem Operator etwas vorgeschlagen wird. In dem noch vorliegendem Interface ist dies z.B. durch einen rot eingefärbten „3D“-Scan-Button realisiert, der darauf hinweist, dass es nun sinnvoll wäre, wieder einen 3D-Scan zu machen.
Je mehr Entscheidungshilfen der Operator bekommt, desto einfacher könnte das Arbeiten mit dem Roboter werden. Es ist hier wichtig zu betonen, dass der Mensch nur unterstützt werden soll und kann, ihm aber nicht das Mitdenken und Kontrollieren abgenommen werden darf. Es wird immer eine Wahrscheinlichkeit für Fehler in Systemen geben. Sollte einer dieser Fehler während eines Wettkampfes passieren, ist dies bestimmt traurig, aber nicht wirklich tragisch. In einem Katastrophenfall aber könnte das sowohl rechtliche und als auch fatale Folgen für Menschen haben. Die *Mensch-Roboter-Kenntnis* darf nicht durch einen immer autonomeren Roboter vernachlässigt werden [18, 46]. Und da in einem Team jeder auf den Partner aufpasst, braucht es eben auch eine sehr gute *Roboter-Mensch-Kenntnis* im Umkehrschluss, da es eben auch das berühmte „menschliche Versagen“ gibt. Dadurch entsteht eine gegenseitige Kontrolle, aber auch Hilfe. Hat der Operator keine Hilfestellung vom System, kann er deutlich weniger leisten und ist eventuell schnell überfordert. Wie auch Scholtz in [46] anspricht, soll im Mittelpunkt eines Designs eine synergetische und damit intelligentere Team-Interaktion stehen.

„Collaborative Control“

Eine sehr spezielle Idee davon, wie man die Interaktion im Mensch-Roboter-Tean verbessern kann, erklären Fong et al. unter anderem in [30–32]. Dabei muss vorher bemerkt werden, dass das Ziel dieses Ansatzes die Steuerung mehrerer Roboter ist.
Ein Roboter soll nicht nur ein Werkzeug sein, sondern ein Partner. Dies soll durch Dialoge bewerkstelligt werden: Wenn der Roboter z.B. durch widersprüchliche Sensordaten „verunsichert“ wird, richtet er eine Frage an den Operator. Der Roboter soll sich Hilfestellung bei seiner „Wahrnehmung und Kognition“ vom Menschen holen. Damit handelt es sich nicht mehr um eine Mensch-Roboter-Interaktion, sondern um eine Roboter-Mensch-Interaktion, wie Fong et al. betonen. Folglich ist die „Collaborative Control“ das Gegenteil von einer Teleassistenz, die dem Operator Hilfestellung geben soll. Die Kapazität eines Operators kann dadurch besser aufgeteilt werden, da ein Roboter immer dann nach Assistenz fragt, wenn eine problematische Situation eintritt und ansonsten in einem weitgehend autonomen Modus vorgegebene Aufgaben abarbeitet. Konkret schaut dies so aus: Auf die Frage „Wie gefährlich ist dieses Bild?“ würde dann der Operator in diesem Fall mit einer Zahl in einem vorgegebenen Intervall antworten, z.B. „8“. Diese Kommunikation läuft noch über textuelle Ein- und Ausgabe. Vorteil ist dabei, dass der Operator auf Probleme aufmerksam gemacht wird, die auch durchaus „vom Roboter formuliert“ werden: („Ich denke, dass ich feststecke, weil meine Räder durchdrehen. Konnte die Aufgabe X wegen Y nicht erfüllen.“ [30]). Damit muss der Operator nicht erst anhand der Daten eine schnelle Analyse machen, was mit dem System passiert sein könnte, sondern bekommt schon eine Hypothese geliefert. In den Versuchen zeigte sich, dass die Versuchspersonen zwar durch

das Fragen mehr in den Prozess involviert wurden, aber auch schnell von den Fragen genervt waren. Zudem benötigt man für die Antworten schon ein gewisses Expertenwissen. Deshalb ist ein Training im Umgang mit dem Roboter notwendig, – damit verfehlt Fong eines der eigentlichen Ziele, nämlich allen vermeintlichen Nutzern (auch ohne Vorkenntnis) eine Interaktion mit dem Robotern zu ermöglichen.
Ob diese künstliche Unterhaltung sinnvoll ist oder nicht, mag fraglich sein. Was man aber aus dieser Idee filtern kann, ist, dass man bei Fehlermeldungen eventuell noch weitere Informationen einbauen kann. Voraussetzung ist dann natürlich, dass bestimmte Sensorenwerte im System kategorisiert werden können, damit eine Hypothese an den Operator weitergegeben werden kann.

4.1.2 Weitere Design-Richtlinien

Nicht alle von Scholtz et al. beobachteten Probleme werden durch mangelnde Situationskenntnis ausgelöst. Deshalb verfassten sie weitere Tipps für die Verbesserung der Leistung:

- Erstens, die kognitive Last soll vom Operator genommen werden. Je mehr der Operator mental rotieren und „arbeiten“ muss, desto fehleranfälliger ist das ganze System. Durch geeignete Präsentation der relevanten Information, also durch Fusion von Daten[6], können kognitive Prozesse einfach auf die GUI „abgebildet“ und von ihr übernommen werden.
- Zweitens, die Effizienz soll erhöht werden. Das Interface soll es dem Operator möglichst einfach machen, mehrere Roboter gleichzeitig unter Kontrolle zu haben. Das bedeutet, dass z.B. nur ein Fenster am Monitor für mehrere Roboter verwendet werden kann. Im Allgemeinen ist es erstrebenswert, so wenige (redundante) Fenster wie nur möglich in der GUI zu haben.
- Drittens, der Operator soll dabei unterstützt werden, einen angemessenen Modus für den Roboter zu finden. Dies bezieht sich auf Roboter, die mehrere Modi zur Verfügung haben, z.B. (semi-)autonomer Modus, und setzt schon einige *Roboter-Mensch-Kenntnis* voraus.

Die Ideen und Anregungen werden hier als Grundlagen für das Re-Design herangezogen. Da diese relativ allgemein gehalten sind, muss geklärt werden, wie man diese Richtlinien nun für das Re-Design dieses Interfaces anwenden kann. Man muss sich fragen, ob und wie die Richtlinien umgesetzt werden können. Eine weitere grundlegende Frage ist, ob und wie die Ergebnisse der Evaluation zu den Richtlinien passen (siehe Kapitel 3.3.2 auf Seite 30).

4.2 Der Beitrag der HCI

Die in dem Kapitel 4.1 erwähnte Situationskenntnis ist unter anderem von der HCI abgeleitet worden. Abgesehen von der Telepräsenz kann bei der Gestaltung der GUI auf die

[6] Obwohl nur die Rede von der Fusion der Daten ist, sollte auch ein wichtiger Punkt die „Konsistenz“ der Datenpräsentation sein: Konsistent im systemtechnischen als auch im gestalterischen Sinne.

Erkenntnisse der HCI zurückgegriffen werden.
Widerum muss hier betont werden, dass der Begriff der Mensch-Computer-Interaktion ein sehr weitgefächerter ist. Es müssen der Mensch, der Computer und deren Interaktion jeweils analysiert und diskutiert werden. Theorien über Eingabegeräte, Gedächtnisfunktion, mentale Modelle, Geschlechterunterschied, Kostenrechtfertigung, Nutzeranforderung, Richtlinien und Standards sind nur einige von sehr vielen Teilen, welche die HCI ausmachen.

4.2.1 Informations-Design

Für dieses Re-Design macht es Sinn, die HCI unter dem Aspekt des Informations-Designs zu betrachten. Letzteres ist eigentlich eine sehr junge Disziplin, die versucht, aus Rohdaten Informationen zu machen: Der eigentlich sehr alte Grundgedanke dabei ist, dass Daten für die Kommunikation wertlos sind und erst in einen Kontext gesetzt werden müssen, damit Information daraus wird, die besser kommunizierbar ist. Indem Daten organisiert, transformiert und präsentiert werden, sollen sie eine Bedeutung für Menschen bekommen, also zu Informationen werden [50].
Die Überschneidung von HCI und Informations-Design soll hier der Beantwortung der relevanten Fragen dienen: Wie können die Daten des Systems so umgesetzt werden, dass für den Operator am Computer bedeutungsvolle Informationen wahrnehmbar werden?

Die menschliche Wahrnehmung

Der Mensch ist durch seine Sinne einem massiven Informationsfluss ausgesetzt. Da nicht alles in das Bewusstsein gelangen kann und soll, wird eine Vorauswahl getroffen, die man mit „automatischer Informationsverarbeitung“ beschreiben kann [33]. Wird ein Bild *unbewusst* betrachtet, fällt der Blick zuerst auf die Mitte des Bildes, dann meist nach links oben[7]. Der sogenannte Betrachtungspfad führt meist von oben nach unten und von links nach rechts und ist ein Wechselspiel zwischen Fixationen (Element wird für einen relativ langen Zeitraum auf die Fovea abgebildet) und Saccaden (sehr schnelle Bewegungen des Auges) [33].Dem ist hinzuzufügen, dass ansonsten im restlichen Gesichtsfeld unscharf wahrgenommen wird und dann als interessant wahrgenommene Gegenden in der Periphärie angesteuert werden. Zudem gibt es in diesem Gebiet sehr viele, auch widersprüchliche Studien. Diese sollen aber hier nicht weiter diskutiert werden, da der Blick des Operators nicht nebenbei den Monitor streifen wird. Folglich ist es viel interessanter, wie man den Betrachtungspfad steuern kann. Der Mensch kann sich selber Fixationspunkte aussuchen und damit bewusst selektieren. Der Blick kann aber auch durch geschickte Anordnung von Informationen gelenkt werden. Im Prinzip wird daraus ein ständiges Wechselspiel zwischen dem „automatischen, gedankenverlorenem Ankucken eines Bildes“, dem „bewussten Suchen von relevanter Information“ und einer „Verführung des Blickes“ auf die vom

[7] Weil der Blick oft nach links oben fällt, ist auf vielen Webseiten dort das Firmenlogo platziert.

Bildschöpfer als relevant gedachte Information. In diesem Fall ist es noch eingeschränkter: Ein Operator wartet auch bestimmte Informationen, die alle einen bestimmten Platz haben. Die Frage ist bloß, was wann wo nach seiner Aufmerksamkeit verlangt. Es ist bekannt, dass Bewegung Aufmerksamkeit auf sich zieht, und das wird z.B. bei Werbebannern im Internet ausgenutzt. Auffällige Farben erregen auch Aufmerksamkeit. An eine solche Farbe gewöhnt man sich aber[8]. Eine Bewegung bzw. ein ständiges Aufblinken ist dagegen immer wieder ein neuer Stimulus, deshalb ermüden die zuständigen Neuronen nicht so schnell. Folglich ist Bewegung effektiver als Farbe. Dies ist ein wichtiger Gestaltungsfaktor für das Re-Design.

Generell kann man sagen, dass wichtige Informationen im Zentrum (des Bildes/des Monitors) stehen sollten. Und oft fällt im Zusammenhang von Stress der Begriff des Tunnelblicks. Da die Operatoren bei einem Wettkampf unter Stress stehen, liegt die Vermutung nahe, dass das Gesichtsfeld eingeschränkt wird. Auf Nachfragen bei einigen Experten kann man aber durchaus einen Tunnelblick ausschließen, denn „so stark stand dabei noch keiner unter Stress“. An sich ist der Wettkampf eine Situation, an die sich die Operatoren (zu einem gewissen Maße) gewöhnt haben. Zudem ist der Wettkampf keine unerwartete Stresssituation, weil sich ein Operator schon Wochen vor dem Ereignis darauf vorbereitet. Sollte der Operator große private Probleme haben und während dem Wettkampf so stark überfordert sein, – sich selber dieses nicht eingestehen, könnte es zu einem Tunnelblick kommen. Die Wahrscheinlichkeit dafür ist ziemlich gering [15]. Die Möglichkeit eines Tunnelblicks ergäbe sich dann nur noch durch eine Erkrankung (Retinopathia pigmentosa), eine Vergiftung oder durch Alkohol, und kann damit eigentlich ausgeschlossen werden.

Die Fokusmetapher

Der erste Schritt, um aus Daten Informationen zu machen ist, die Daten in einen Kontext zu setzen. Dies wird hier anhand einer Metapher versucht. Die GUI wird also in eine Art leichtverständliches Schema gesetzt, die sich an der menschlichen Wahrnehmung orientiert und auf die Beobachtungen der Evaluation eingeht. Die Benutztung von Metaphern soll also dem Nutzer helfen, leichter und schneller mit den Informationen umzugehen ([20, 23, 50]). Eine der bekanntesten Metaphern im Computerbereich ist die „Desktop-Metapher“: Der Computer-„Desktop“ wird mit einem Schreibtisch, also einem Desktop im Englischen, verglichen.
Um, wie oben angesprochen, das Zentrum als die wichtigste Fläche auszunutzen, ist die Fokusmetapher[9] ein interessanter Ansatz.

Wie die Abbildung 4.1 dargestellt, gibt es verschiedene Ebenen: Den primären, sekundären und periphären Fokus. Bei der Arbeit von Laqua wurde ein Internetportal gestaltet und

[8] Die Eigenschaft der Neuronen, zu ermüden, nennt man auch in der angelsächsischen Literatur „accomodation“. Ein interessantes Beispiel darfür ist der Necker Würfel [43].

[9] Dieser Begriff wurde von Sven Laqua in seiner Bachelorarbeit geprägt [37]. Es stellt sich im Nachhinein als ein glücklicher Zufall heraus, dass der Entwurf des Re-Designs einer Metapher entspricht, die auch schon empirisch ausgewertet wurde. Damit kann Entwurf nicht nur durch die Ergebnisse der Evaluation, sondern auch wissenschaftlich gerechtfertigt werden.

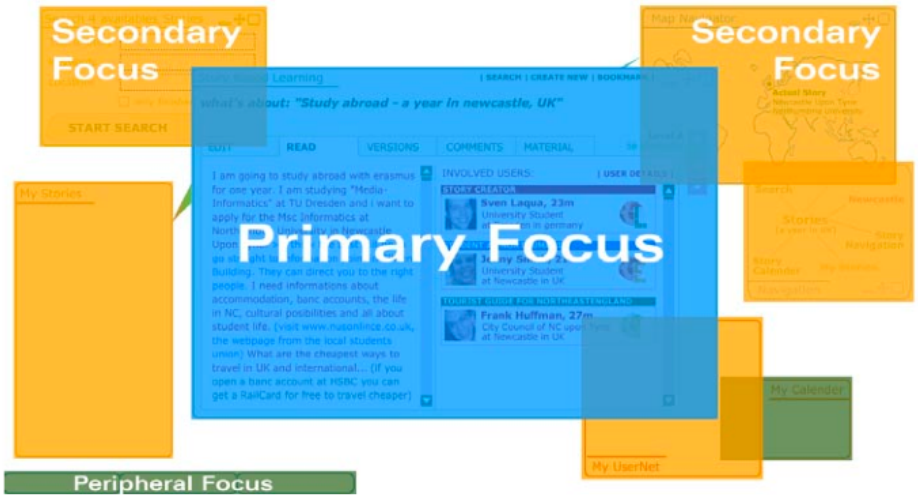


Abbildung 4.1: Schematische Darstellung der Fokusmetapher aus [37].

jede der Boxen stellt eine Art Modul dar, das durch einen Mouse-over-Effekt in den Vordergrund geholt und durch Doppelklicken vergrößert wird [8]. Laqua hat durch Eyetracker-Versuche die Effektivität dieser Metapher gezeigt. Er behauptet, dass die Fokusmetapher die Aufmerksamkeit steigert und den Navigationsaufwand auf dieser Portalwebseite vermindert [38].

4.2.2 Zwei Aufgaben in einer GUI

Um diese Metapher im Re-Design aufzugreifen, muss sie auf dieses System angepasst werden. Der Knackpunkt ist hier, dass es sich nur um einen Bereich handelt, der im zentralen Bereich des Blickes, und damit in der Mitte der GUI, angeordnet sein soll. In diesem System gibt es zwei Aufgaben: Navigieren und Kartografieren (wie auch im Kapitel 4.3.2 angesprochen wird). Dies wurde vor allem im Zuge der Evaluation deutlich. Wenn man eine Versuchsperson beobachtet, gibt es eine zeitliche Abfolge von Aktionen, die sich öfters wiederholen. Da der Operator niemals zugleich navigiert und kartografiert, können die beiden Fenster mit den jeweiligen Informationen immer zu dem Zeitpunkt in den Fokus rücken, zu dem mit ihnen gearbeitet wird. Das bedeutet also, dass während dem Fahren des Roboters die 2D-Informationen (also der 2D-Viewer im alten Design) im Vordergrund ist. Wird ein 3D-Scan ausgelöst, sind die 2D-Informationen nicht mehr wichtig und können in den Hintergrund rücken, um der 3D-Information (3D-Viewer) Platz zu machen. Es gibt also zwei Hauptinformationsfenster in diesem Re-Design und nicht nur eines, wie z.B. bei dem Portal von Laqua. Wichtig ist hierbei, die richtigen „Sprungstellen“ für den Modulwechsel zu finden. Dies wird bei der Umsetzung im Kapitel 5 näher beschrieben.

4.2.3 Allgemeine Design-Richtlinien

Wie schon von Stewart in [51] angesprochen wird, gibt es viele Richtlinien, Tipps und Standards, wie ein User Interface benutzerfreundlich gestaltet werden kann. Stewart beschreibt in dem Artikel die drei wichtigsten Sammlungen von Richtlinien bzw. Heuristiken. Eine davon stammt von Jakob Nielsen. Im Folgenden werden zehn seiner Heuristiken [1] kurz angesprochen, um sie später bei der Umsetzung (Kapitel 5) griffbereit zu haben.

1. *Sichtbarkeit des Systemstatus*: Der Nutzer soll in angemessenen Zeitabständen darüber informiert werden, was momentan im System vorgeht.
2. *Übereinstimmung der Wirklichkeit und des Systems*: Die Sprache und Konzepte, durch die das System mit dem Nutzer kommuniziert, sollten die des Nutzers sein.
3. *Nutzerfreiheit und -kontrolle*: Einem Nutzer soll immer ein Notausgang zur Verfügung gestellt werden, um ungewollte Aktionen rückgängig zu machen. Allgemein sollten „undo“ - und „redo“-Funktionen vorhanden sein.
4. *Konsistenz und Standards*: Begriffe und Aktionen sollten (auch plattformunabhängig) immer gleich sein und möglichst Standards folgen.
5. *Fehlern vorbeugen*: Den Fehlern mit einem durchdachten Design vorzubeugen ist besser als eine guter Fehlermeldung.
6. *Wiedererkennung anstelle von Erinnern*: Objekte, Aktionen und Optionen sollten sichtbar sein, damit der Nutzer sich nicht daran erinnern muss. Gebrauchsanweisungen sollten ebenso sichtbar sein oder einfach zugänglich.
7. *Flexibilität und Effizienz*: Ein System muss einem Novizen dienen können, aber auch Abkürzungen für den Experten bieten.
8. *Ästhetisches und minimalistisches Design*: Irrelevante oder selten benötigte Informationen sollten in Dialogen weggelassen werden, um nicht die relevanten Informationen zu überdecken.
9. *Nutzern sollte Hilfestellung gegeben werden, um die Fehler zu erkennen, zu diagnostizieren und zu beheben*: Diese Informationen sollten eine klare und einfache Sprache haben und ein Lösung zu dem Problem bieten.
10. *Hilfe und Dokumentation*: Auch wenn es besser wäre, dass ein Nutzer ohne das auskommt, kann es dennoch notwendig sein, Hilfe und Dokumentation zur Verfügung zu stellen. Diese sollte einfach zu durchsuchen sein und sich auf die Vorhaben des Nutzers konzentrieren und diese in angemessenen Schritten beschreiben.

Weitere Feinheiten

Bei der Umsetzung dürfen einige Kleinigkeiten nicht vernachlässigt werden. Wichtig sind dabei beispielsweise folgende Punkte:

- Bilder werden (mit Einschränkung[10] besser und schneller als Texte verarbeitet. Die Gründe dafür kann man in der Evolution finden: Vor der Schrift gab es Malereien, z.B. Höhlenmalereien. Sprache wird zudem in einem kognitive belastenden (und evolutionär jüngerem) Teil des Gehirns verarbeitet. Ein Bild dagegen wird in einer älternen Region verarbeitet, die emotiv und damit wenig kognitiven Aufwands bedarf. Eine bildhafte Darstellung ist auch deswegen vorzuziehen, weil ein Bild mehr Informationsgehalt hat als z.B. ein Wort. Eindeutige Icons zu erstellen ist eine hohe Kunst der Informations-Designes [33].
- Lesbarkeit wird durch eine weit gebräuchliche Serifenschrift verbessert. Sie sollte gerade ausgerichtet sein, also Wörter sollten nicht gebogen sein, wie es machmal in Firmenlogos zu sehen ist. Ebenso verschlechtern fette und/oder kursive Schriftarten die Lesbarkeit. (Für längere Texte ist eine einfache und direkte Ausdrucksweise ohne komplizierte Nebensatzkonstruktion entscheidend.)
- Die Farben sollten so gewählt werden, dass ein guter Kontrast zwischen Hintergrund und Vordergrund besteht.
- Durch z.B. Blinken und grellere Farben kann die Aufmerksamkeit auf das Element gezogen werden.
- Auch andere Modalitäten können beansprucht werden: Warnen durch ein akustisches („earcon“) oder haptisches Signal („Force-feedback“).

Ratgeber und Stilempfehlungen gibt es sehr viele, meist vertreten sie ähnliche oder gleiche Ansichten. Interessante Referenzen sind dabei [20, 33, 52]. Zudem gibt es zahlreiche Internet-Referenzen, z.B. `http://www.hcibib.org/`, `http://www.benutzeroberflaeche.de/` oder `http://www.userfocus.co.uk`, um nur einige zu nennen.

4.3 Andere Ansätze

Ein Blick in angrenzende und verwandte Gebiete, in denen Tele-Interfaces vorkommen, erscheint wichtig. Eigentlich müsste es auch in diesen Gebieten eine Vielzahl von Artikeln geben, da eine Schnittstelle für die Akzeptanz und Anwendung entscheidend sein kann. Ebenso beeinflusst eine Schnittstelle die Möglichkeiten des Operators und das Streben danach, eine Aufgabe erfolgreich zu beenden [16] . Leider konnten nicht allzu viele Informationen gefunden werden. Hier soll nun aufgezeigt werden, in welche Richtungen im Zuge dieser Masterarbeit gedacht wurde.

4.3.1 Andere Arbeitsplätze für Operatoren

Ein extrem komplexer Arbeitsplatz ist sicherlich das Cockpit eines Flugzeuges. Interessanterweise wurde oft in Gesprächen angemerkt, dass man doch die Prinzipien eines Cockpits

[10] Man muss hier hinzufügt werden, dass u.a. der Informationsgehalt dabei relevant ist: Ein abstraktes Bild im Museum wird sicherlich nicht „schneller und besser“ verarbeitet als das daneben hängende Schild mit der Aufschrift „Notausgang“.

(a) Einblick in das Cockpit einer Concorde aus [44].

(b) Kontrollstand eines Seaeye-Tauchroboters.

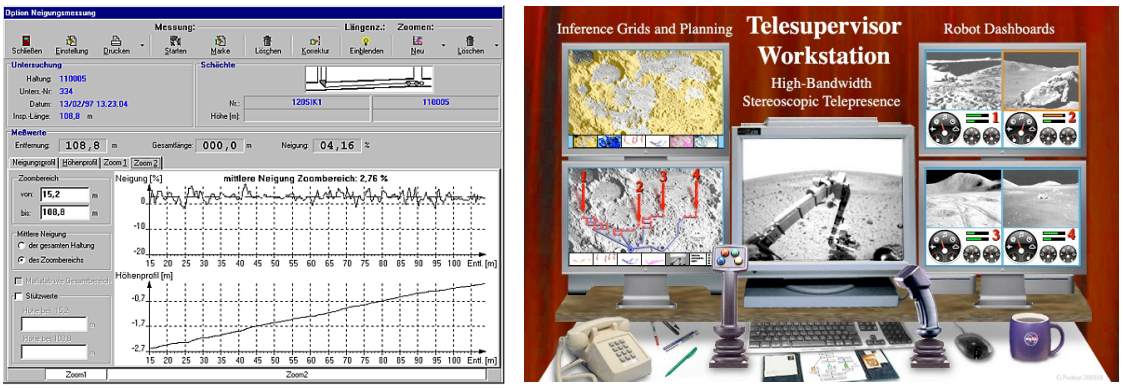
Abbildung 4.2: Andere Operator-Arbeitsplätze.

für das Re-Design verwenden könnte. Pritchett behandelt ausführlich Aspekte der HCI im Artikel [44]. Die Abbildung 4.2(a) zeigt den Arbeitsplatz eines Piloten einer Concorde. Es ist nicht zu übersehen, dass hier eine sehr hohe Komplexität vorhanden ist. Der relevante Unterschied ist, dass ein Pilot in vielen Fällen einfach nach draußen aus dem Fenster schauen muss: Es handelt sich um keine Fernsteuerung, außer wenn Nebel/Wolken den Ausblick verhindern. Der Pilot erfährt zudem Erschütterungen (durch z.B. Luftlöcher) am eigenen Leib, er ist also ein Teil des zu steuernden Objekts. Da viel intensive HCI Forschung in die Gestaltung von Flugzeug-Cockpits fließen musste, kann man für das Re-Design davon in gewisser Hinsicht auch profitieren.

Wenn man sich nun in anderen Telerobotik-Anwendungsbereichen umschaut, fallen einem meist Tiefsee-Tauchroboter und Roboter für Minensprengung ein. Für letzteres wurden keine weiteren Ressourcen gefunden. Auch die Informationen über Tauchroboter sind rar[11]. Die Abbildung 4.2(b) zeigt den Arbeitsplatz eines Operators auf, der einen solchen Roboter steuert. Bekannt ist, dass der Operator einen speziell angefertigten Hand-Controller hat, mit dem er den Roboter steuern kann. Die Roboter sind mit Kameras ausgestattet und teilweise auch mit Manipulatoren. Sicherlich wäre es sehr informativ gewesen, einen Einblick in diese Schnittstelle zu bekommen. Kartografiert wird mit diesen Roboter wahrscheinlich nicht.

Eine sehr fortgeschrittene Form der Teleoperation findet man bei der NASA. Die Roboter, die sich momentan auf dem Mars befinden, „Spirit“ und „Opportunity“, werden von der Erde aus teleoperiert, verfügen aber auch über autonome Navigation. Der zeitliche Faktor der Datenübertragung von der Erde zum Mars - und wieder zurück - ist sehr markant. Die Operator-Schnittstellen auf der Erde sind sehr komplex, da zudem große Mengen präziser

[11] Selbst auf Anfrage bei der Firma Seaeye wurde leider noch keine Auskunft bzgl. des Interfaces erteilt.



(a) Neigungs- und Temperaturmessung der Kanal Analyse Software (Firma IBAK).

(b) Entwurf eines Kontrollstandes für mehrere NASA-Roboter (aus [34]).

Abbildung 4.3: Die Aufgabe des Roboters bestimmt das Interface.

Daten dargestellt werden müssen. Folglich ist ein Monitor sicherlich nicht ausreichend.[12] Weitere Einblicke in eine solche GUI wären sicherlich sehr lehrreich.
In [34] wird eine Überwachungs-Architektur für mehrere Roboter beschrieben, die Mond- und Marsmissionen der NASA eingesetzt werden könnte. Wie der Arbeitsplatz eines Operators aussehen könnte, zeigt Abbildung 4.3(b).
Auch bei der Kanal-TV-Untersuchung kommen teleoperierte Roboter zum Einsatz. Es fällt die Kartografierung weg, da die Topografie der Kanäle bekannt ist. Dennoch müssen beispielsweise Informationen über den Neigunswinkel des Roboters an den Operator gesendet werden. Die Abbildung 4.3(a) zeigt das entsprechende Fenster der Applikation.
Eine GUI ist stark von der Anwendung und dem Umfeld des Roboters abhängig. Diese angesprochenen Ansätze sind Ideengeber für das Re-Design. Leider sind nicht viele detaillierte Informationen über GUIs frei verfügbar. Einen inzwischen etwas älteren Überblick über teleoperierte Fahrzeuge findet man im Artikel [30] von Fong und Thorpe.

4.3.2 Computerspiel als Vorbild

Anfangs wurde desöfteren der Wunsch geäußert, das Re-Design an einem Egoshooter/First Person Shooter zu orientieren. Das würde bedeuten, dass die relevanten Daten nicht in einem eigenen Fenster dargestellt werden, sondern in eine Perspektive (Hintergrund oder Kamerabild) eingebettet werden. Die Analogie von diesen Computerspielen (siehe Abbildung 4.4) und einer Rettungsmission mit einem teleoperierten Roboter scheint gerade Maxwell offensichtlich: Diese Strategie, also das Suchen von Opfern (bzw. Feinden) aus der egozentischen Perspektive, scheint das angemessenste Interface-Paradigma für USAR zu

[12] Bilder einer solchen Schnittstelle kann man auf der Webseite `http://mpfwww.jpl.nasa.gov/MPF/ops/rover.html` finden. Sie stammen noch von einer älteren Mission (1997), als der Rover „Sojourner“ den Mars explorierte.

(a) Die einfachen Anfänge: Doom 1.

(b) Unreal 1: Einlagerung der Karte rechts oben.

Abbildung 4.4: Zwei etwas ältere Screenhots von Egoshootern.

sein [39]. Es hat sich jahrelang in den Spielen bewährt und erscheint als eine vernünftiger Ansatz für einige Systeme.

Das Hauptproblem ist hier, dass es sich bei dem Deutschland1 Team um ein System handelt, das mit 3D-Scans kartografiert. Dieses Kartografieren muss vom Operator ausgelöst, kontrolliert und manchmal auch korrigiert werden. Damit teilt sich das Aufgabenfeld eines Operators eben in zwei große Gebiete auf: Navigation und Kartografierung. Überspitzt formuliert lässt sich hier sagen, dass der Operator für die Navigation die Kartografierungs-Werkzeuge nicht braucht und dass er während des Kartografierens nicht die Informationen benötigt, die bei der Navigation notwendig sind (siehe Kapitel 4.2.2). Solange diese beiden Aufgabenfelder noch nicht technisch verbunden werden können, muss eine hybride GUI zum Einsatz kommen.

Zusammenfassend kann man sagen, dass ein Egoshooter „zu einfach gestrickt“ ist, da keine Kartografierung vorgesehen ist. Das Paradigma kann also nur sehr bedingt umgesetzt werden. Interessant dabei ist die Art, wie in diesen Spielen relevante Informationen geschickt in die GUI einlagert werden (siehe Abbildung 4.4), oder wie der Spieler vor etwas gewarnt wird.

Kapitel 5

Umsetzung und Spezifikation

5.1 Richtlinien und Evaluationsergebnisse

Wie schon im Kapitel 4 beschrieben, wird die Umsetzung von mehreren Seiten beeinflusst. Nun müssen die aus der Evaluation gewonnen Erkenntnisse mit den Empfehlungen der Richtlinien verglichen werden. Erst dann kann der Designprozess in Angriff genommen werden.
Die Design-Richtlinien, wie sie z.B. in Scholtz, Drury und Yanco [25] auftauchen, werden hier auch nochmals kurz aufgeführt:

- *1. Richtlinie*: Verbesserung der Situationskenntnis.
- *2. Richtlinie*: Verringerung der kognitiven Last des Operators.
- *3. Richtlinie*: Effizientere Gestaltung der GUI.
- *4. Richtlinie*: Mehr Entscheidungshilfen für den Operator.

Im Kapitel 3.3.2 wurden die Ergebnisse der Evaluation detailliert beschrieben. Hier werden sie nochmals sehr kurz zusammengefasst:

- Die Darstellung von Funktionen sollte präzise und treffend sein, damit die Erwartung des Operators erfüllt wird (siehe *2. Richtlinie* oben).
- Die verwendeten Begriffe sollten verständlicher sein (siehe *2. Richtlinie*).
- Soweit möglich, sollte eine größere Fehlerrobustheit vorhanden sein, u.a. eine „undo"-Funktion (siehe *3. Richtlinie*).
- Sicherheitsabstände sollten in der GUI besser eingehalten werden (siehe *2. und 3. Richtlinie*).
- Wichtige Befehle sollten besser und schneller auffindbar sein (siehe *2. und 3. Richtlinie*).
- Eine einheitlichere Gestaltung ist nicht notwendig, aber wünschenswert (siehe *2. und 3. Richtlinie*).

- Dem Operator sollten mehr Entscheidungshilfen gegeben werden (siehe *4. Richtlinie*).
- Die Größe der Fenster entspricht ungefähr ihrer Wichtigkeit bzw. Relevanz. (Da diese eine positive Aussage ist, kann darauf geachtet werden, dass dies im Sinne der *1. Richtlinie* so beibehalten wird.)
- Der Informationsgehalt der Fenster ist ausreichend. (Auch dies entspricht der *1. Richtlinie* und kann beibehalten werden.)

Dieser grobe Vergleich zeigt auf, dass keines der Evaluationsergebnisse den Richtlinien zu widersprechen scheint, sondern dass jedes sinnvoll einer Richtlinie zugeordnet werden kann. Wichtig ist nun, weiter im Sinne der Richtlinien zu denken und sich nicht nur auf die Ergebnisse der Evaluation zu beschränken. Eine Versuchsperson kann zwar sagen, was an der GUI stört, sie kann aber nicht unbedingt formulieren, wie die GUI besser wäre. Deshalb muss man hier bei dem Re-Design unbedingt über den „Evaluations-Tellerrand“ schauen.
Für das Re-Design werden die einzelnen Elemente des Interfaces neu entworfen. Betrachtet man aber die Elemente als isoliert von einander, könnte der Arbeitsfluss des neuen Interfaces gestört werden. Die Elemente sind nicht von der Mission zu trennen, ebenso wie die Mission nicht ohne die Elementen betrachtet werden kann. Obwohl nun hier zuerst die Elemente beschrieben werden, stehen sie immer im Kontext eines der Szenarios, die weiter unten beschrieben werden.

5.2 Von Elementen zu Modulen

Um nochmal die Kategorisierung der einzelnen Elemente von 2.3 (Seite 6) aufzugreifen, hier eine kurze top-down Beschreibung. Die GUI besteht aus:

- *Informationselementen*
- *Display-Funktionselementen*
- *Steuerelementen*

Ein *Informationselement* kann also von einem *Display-Funktionselement* verändert werden. Diese beiden Elementarten gehören zusammen, während die *Steuerelemente* einen Effekt auf die Navigation mit dem Roboter haben. Zudem sind die *Steuerelemente* redundant auf den Joystick bzw. die Joysticktasten gelegt.
Dass in der folgenden Beschreibung der Begriff „Modul“ verwendet wird, soll nicht nur anzeigen, dass es sich nicht um die Elemente des älteren Designs handelt. Ein Modul besteht aus mehreren Komponenten. Beispielsweise entspricht das 3D-Modul nicht nur dem 3D-Viewer wie im Kapitel 2 beschrieben, sondern beinhaltet auch noch die dazugehörigen *Display-Funktionselemente* in veränderter Form.

5.2.1 Die drei Module

Zum jetzigen Zeitpunkt macht es noch Sinn, die klare Trennung von 2D- und 3D-Informationen beizubehalten. In ihr spiegelt sich auch die Abfolge der zwei Aufgaben, das Navigieren und

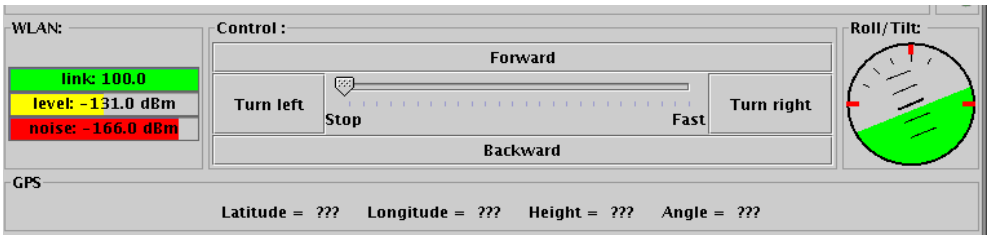


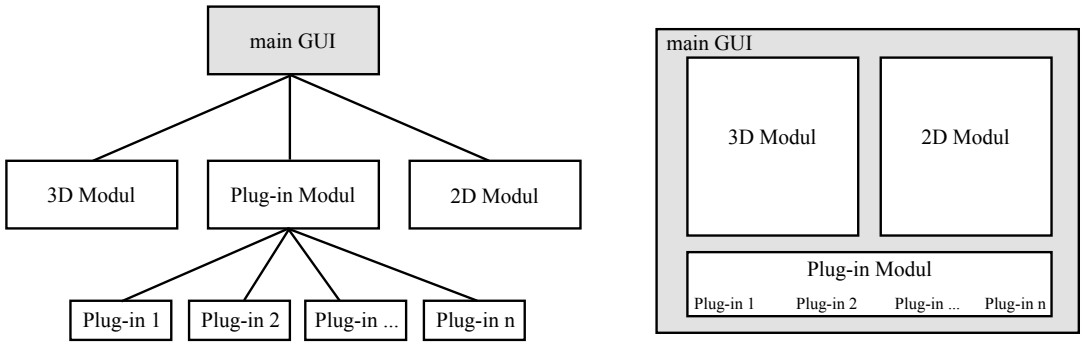
Abbildung 5.1: Momentane Einbindung der neuen Sensoren.

das Kartografieren, wieder. Die Elemente, die dann noch „übrig bleiben“, werden in einem dritten, neuartigen Modul zusammengefasst. Es folgt nun erst eine kurze Beschreibung der drei Module, damit man einen ersten Eindruck von der Grundidee hinter diesem Re-Design gewinnen kann. Zudem werden sie hier in Relation zu den im Kapitel 5.1 angesprochenen Richtlinien gesetzt. Eine genaue Beschreibung der Module folgt in Kapitel 5.5. Dort wird auch auf den Zusammenhang zwischen Design-Entscheidung und Evaluationsergebnissen eingegangen.

Das 3D-Modul wird unterschiedliche Perspektiven darstellen und verschiedene Funktionen für die Kartografierung und die Opfermarkierung bereitstellen. Damit vereint sie folgende drei Elemente der alten GUI: Den 3D-Viewer, die 3D-Funktionsleiste und die 3D-Steuereinheit, die im Kapitel 2.3 beschrieben wurden. Die Vereinigung dieser Elemente soll dazu führen, dass dieser Teil der GUI effizienter zu bedienen ist. Da die Maus viele Funktionen übernehmen soll, muss der Operator nicht ständig mit verschiedenen Shortcuts Rotationen und Translationen auslösen. Der Effekt ist eine Verringerung der kognitiven Last durch eine intuitivere Bedienung.

Das 2D-Modul ist relativ einfach strukturiert und leistet einen erheblichen Beitrag an der Situationskenntnis. Hier ist es möglich, viele Informationen zu fusionieren. Dennoch müssen die Informationen für die Navigation sehr klar und intuitiv bleiben. Der in der alten GUI sichtbare Purismus ist hier sehr sinnvoll und kann durch einige, wenige Ergänzungen durchaus zur einer gewichtigeren Entscheidungshilfe für den Operator werden.

Im Plug-in Modul finden die übrig gebliebenen Informationen ihren Platz. Eines davon ist das eigentlich immer präsente Konsolen-Fenster. Dieses wurde noch nirgendwo erwähnt, weil es die GUI startet und damit nicht nur eine Informationsfunktion, sondern auch eine Metafunktion hat. Aber auch diese durchaus für den Operator wichtigen Informationen werden einen Platz in der GUI bekommen müssen. Gleiches gilt für die Kamerabilder, inklusive deren Steuerung, die auch integriert werden müssen. Da zudem im Verlauf dieser



(a) Schematische Darstellung.

(b) Abbildung in der GUI

Abbildung 5.2: Idee und Umsetzung der Modul – Plug-in Hierarchie.

Arbeit die Hardware und die GUI erweitert wurde, können folgende Informationen auch in das Re-Design integriert werden:

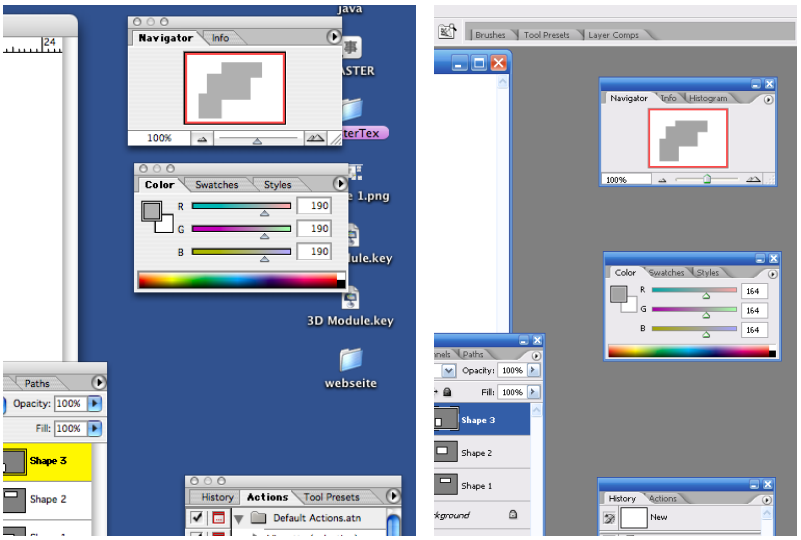
- Mikrofon-Eingabe
- Künstlicher Horizont
- WLAN-Qualität
- Kohlendioxid-Messung
- GPS-Anzeige

Die momentane Einbindung dieser neuen Daten zeigt Abbildung 5.1.

Die Namensgebung des Plug-in Moduls begründet sich durch die dahinter stehende Hierarchie „Modul – Plug-in“ (siehe Abbildung 5.2). Theoretisch können auch die Module als Plug-ins betrachtet werden. In dieser Arbeit werden sie dennoch „Modul“ genannt, um die Hierarchie deutlicher zu machen. Zudem sieht es das Konzept dieser Arbeit vor, dass ein Modul nur bei sehr einschneidenden Änderungen, beispielsweise an der Hardware, verändert werden sollte.

Der GUI-Hintergrund

Die drei Module erhalten einen einheitlichen Hintergrund, der in Abbildung 5.2(b) grau dargestellt ist. Die Module sollen nicht auf dem normalen Betriebssystem-Hintergrund liegen, sondern in eine einheitliche Fläche integriert werden. Sicherlich spielt dabei die persönliche Vorliebe eine gewisse Rolle. Dennoch ist diese Entscheidung begründbar, indem man die menschliche Wahrnehmung berücksichtigt (siehe Kapitel 4.2.1): Die auf einem Monitor dargestellten Informationen konkurrieren um Aufmerksamkeit [19]. Wenn nun bei einer Applikation der Hintergrund inklusive dekorativem Hintergrundbild (Wallpaper/Desktop picture), Icons und Ordner teilweise durchblitzen, entsteht ein größerer



(a) Durchscheinendes Hintergrundbild, Icons und Ordner.

(b) Einheitlicher Hintergrund.

Abbildung 5.3: Screenshots der gleichen Applikation unter Mac OS X und Windows XP.

Konkurrenzkampf (siehe Abbildung 5.3). Durch einen einheitlichen GUI-Hintergrund soll dies vermieden werden. Die Applikation wirkt dadurch „schwerer“ und einnehmender, da sie sozusagen im Vollbildmodus läuft. Da sie die einzig wichtige für den Wettkampf ist, fällt es nicht schwer, die gewollte Leichtigkeit und Möglichkeit der Background Selection[1] zu vernachlässigen.
Durch den homogenen Hintergrund werden die Module optisch stärker als eine Einheit wahrgenommen. Zudem wird kein Platz für ungewollte Ablenkung gelassen und fördert das Zusammenspiel der Module, das im Kapitel 5.3 beschrieben wird.

Die Menüleiste

Um globale Funktionen in die GUI zu integrieren, bietet es sich an, diese in der Menüleiste unterzubringen. Diese wird sich, wie es auch in den meisten Applikationen allgemein üblich ist, ganz oben über die ganze Breite der GUI erstrecken. Dort können z.B. folgende Funktionen ihren Platz finden: das Speichern bestimmter Teile der GUI in verschiedenen Dateiformaten, das Drucken von wichtigen Informationen aus der GUI (3D-Karte, Opferinformationen), etc.
Sehr wichtig sind aber auch die Befehle „undo“ und „redo“, mit denen man entweder eine Aktion rückgängig machen kann, bzw. doch nochmals ausführen kann, falls es fälschlicherweise rückgängig gemacht wurde. Diese Notausgänge, wie sie auch schon in den Design-

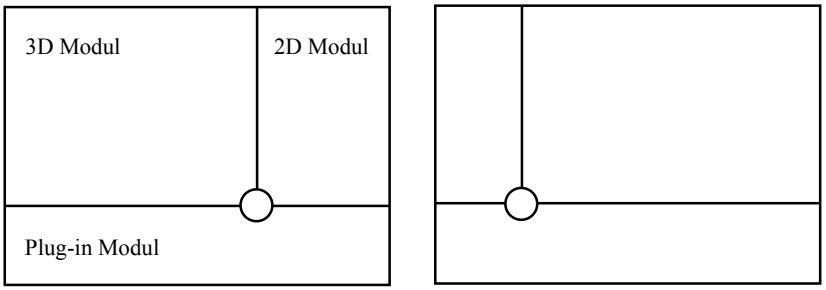
[1] Background Selection ist die Auswahl eines Objekts in einem inaktiven Fenster, das z.B. durchaus von einer anderen Applikation stammen kann [20].

Heuristiken von 4.2.3 auf Seite 51 als dritter Punkt angesprochen wurden, geben dem Operator mehr Kontrolle und Freiheit. Fehler können also durch einen minimalen Aufwand wieder rückgängig gemacht werden. Folglich wird die GUI fehlerrobuster, wie es auch in der ISO 9241 Teil 10 beschrieben wird. (Diese ISO wurde schon im Kapitel 3.2.2 auf Seite 22 angesprochen.)
Dennoch muss diese neugewonnene Freiheit sofort wieder eingeschränkt werden: Zu beachten ist nämlich, dass diese „undo/redo"- Befehle nicht global sind, denn wenn der Roboter nach vorne navigiert wurde, kann dies nicht mehr rückgängig gemacht werden. Es kann zwar das versehentliche Löschen eines 3D-Scans rückgängig gemacht werden, aber die Befehle können nicht der Robotersteuerung dafür dienen, den Roboter wieder auf dem selben Weg rückwärts aus der Sackgasse fahren zu lassen, indem der Operator zwanzig mal nacheinander die „undo"-Funktion aufruft. Folglich sind diese Befehle nicht immer zugängig zu machen und sollten ausgegraut werden.
Eine große Errungenschaft wird dieses „undo"-Sicherheitsnetz bei der manuellen Einlagerung eines Scans haben: Um eine falsche Bewegung wieder zu beheben, muss der Operator nicht die entsprechende Gegenbewegung machen. Da in der alten GUI die Einlagerung anhand von Shortcuts gesteuert wurde, war dies besonders schwierig, da der Operator erst den Shortcut für die entsprechende Gegenbewegung finden musste. Folglich wird nun der Operator weniger kognitiv belastet und auch weniger gestresst sein, da er das Sicherheitsnetz unter sich weiß.

5.3 Das Zusammenspiel der Module

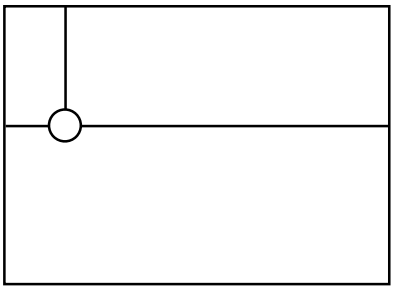
Um den Modulen noch mehr Zusammenhalt zu geben, werden sie durch ein Konzept verbunden, das auch Dynamik erlaubt. Die Module werden alle auf dem Bildschirm angeordnet, wie Fenster. Jedes Modul ist also von der Größe der anderen zwei Module abhängig. An der Ecke, an der sich alle drei Fenster berühren, wird ein neues Element eingesetzt: Eine Art Kreis, oder auch eine Kugel, die man durch eine drag&drop-Bewegung frei verschieben kann. Die drei Fensterecken bleiben dabei aber immer zusammen. Durch das Verschieben der Kugel werden die Fenstergrößen der Module variabel. Man könnte sagen, die Kugel zieht die Fenster – folglich trägt sie den Namen „Dragger". Die Abbildung 5.4 soll die beschriebene Dynamik verdeutlichen.

Die in Kapitel 4.2.1 angesprochene Fokusmetapher bildet eine weitere Grundlage für das Re-Design. Sie wird hier angepasst, indem sie durch eine spezielle Variablilität erweitert wird. Aus den folgenden Beschreibungen wird hervorgehen, dass die Fokusmetapher im Sinne von Laqua eigentlich nur ein Modus von mehreren ist.
Wie schon im Kapitel 4.2.2 angesprochen, müssen während der Navigation die dafür notwendigen Informationen im Fokus sein: In der alten GUI waren das der unverzichtbare 2D-Viewer und die Kamerabilder, während des Kartografieren der 3D-Viewer. Wie im Kapitel 3.3.2 beschrieben, zeigte die Evaluation, dass die Kameras nur wenig genutzt wurden. Wenn, dann brauchte man sie während der Navigation primär um Opfer zu finden. Die Kamerabilder und die Kamerasteuerung werden unter anderem deshalb nicht mit in das 2D-Modul hineingenommen. Ein Vorteil davon wird im Kapitel 5.4 beschrieben.



(a) Das 3D-Modul ist im Fokus.

(b) Das 2D-Modul ist im Fokus.



(c) Hier ist das Plug-in Modul fokussiert.

Abbildung 5.4: Schematische Darstellung der Dynamik anhand des „Draggers“.

Der Fokusmetapher folgend, wird nun während der Navigation das 2D-Modul im Vordergrund stehen. Das heißt hier, dass das rechte Fenster groß gezogen und das 3D-Modul folglich stark verkleinert wird (siehe Abb. 5.4(b)). Umgekehrt nimmt das 3D-Modul sehr viel Platz ein, wenn die 3D-Scans im Mittelpunkt stehen sollen (siehe Abb. 5.4(a)). Als Konsequenz kann man natürlich auch das Plug-in Modul, wie in Abb. 5.4(c) gezeigt, größer darstellen.
Ein Modul soll nicht „sinnlos klein skaliert“ werden, sondern ab einer gewissen, unbedienbaren Größe nur noch symbolhaft dargestellt werden. Beispielsweise könne man auch ein Icon in die Mitte des verkleinerten Moduls setzen. Ein Problem gibt es dann bei der Verkleinerung des 2D-Moduls, wenn der Operator mit der Maussteuerung navigieren will: Da das Modul klein ist, wird er wahrscheinlich nicht sinnvoll auf die Steuerung zugreifen können, bzw. wenn die Steuerung klickbar ist, dann könnte sich, abhängig vom Modus, das 2D-Modul vergrößern. Dadurch wird die Maussteuerung auch vergrößert und auf dem Monitor bewegt. Hier ist es also notwendig, dass der Operator erst einmal in das 2D-Modul klickt, damit es groß wird, und dann erst anfängt, zu navigieren.

5.3.1 Die verschiedenen Modi

Dem Operator soll die Entscheidungsfreiheit gegeben werden, in wie weit er die Dynamik der Fokusmetapher ausnutzen will. Es hängt also vom ihm ab, in wie weit er sich von der GUI unterstützen lassen will. Die Auswahl der Modi erfolgt durch das Kontext-Menü, das durch einen rechten Mausklick auf den Dragger sichtbar wird, wie es bei Computern usus ist. Falls dies nicht implementiert wird, kann dies auch z.B. in einer Konfigurationsdatei zur Verfügung gestellt werden.

Der voll-automatische Modus sorgt für eine den Aufgaben des Operators angemessene Dynamik der GUI. Der Dragger wird automatisch von links nach rechts geschoben (3D-Modul groß: Kartografieren) und von rechts nach links (2D-Modul groß: Navigieren). Wichtig ist hierbei, wann dieser Tausch stattfindet: Theoretisch ist es sinnvoll, dass der Dragger das 3D-Modul vergrößert, *wenn ein 3D-Scan ausgelöst wird*. Sobald der Roboter weiter gesteuert wird, schiebt der Dragger das 3D-Modul nach links und gewährt dem 2D-Modul seinen Platz.

Der halb-automatische Modus gibt dem Operator mehr Freiheit: Sollte er doch ein Modul größer dargestellt haben wollen, als es vom voll-automatischen Modus veranschlagt wurde, dann kann er mit dem Dragger die Proportionen beliebig verändern. Erst durch den Trigger des 3D-Scan-Auslösens wird der Dragger wieder automatisch versetzt. Ein wichtiger Aspekt wäre in diesem Modus, dass das System die vom Operator neu eingestellten Proportionen speichert und *nicht* wieder auf die voll-automatischen Proportionen zurückgreift. Die Option, eine Mindest- bzw. Maximalgröße eines Moduls festzulegen, wäre sicherlich sinnvoll.

Der statische Modus vefügt über keine Dynamik. Der Operator legt eine feste Proportion der Module für die ganze Mission fest. Im Optimalfall könnte dies im Kontext-Menü des Draggers eine Option sein.

5.3.2 Extreme Dragger-Aktionen

Ebenso erlaubt dieses Re-Design, dass nicht unbedingt alle drei Module angezeigt werden müssen, sondern dass der Operator den Dragger so verschieben kann, dass nur ein oder zwei Module sichtbar sind. Dies könnte notwendig sein, um das 3D-Modul im Vollbildmodus zu sehen. Im Wettkampf wird dies nicht oft passieren. Diese Optionen sind in der Abbildung 5.5 beispielhaft dargestellt.

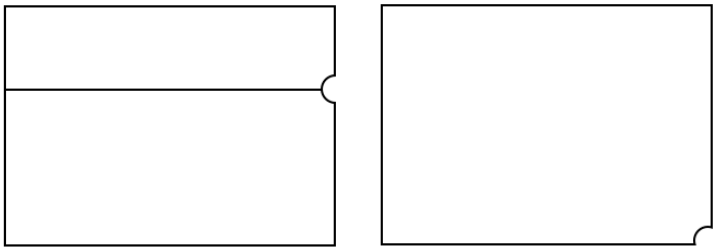
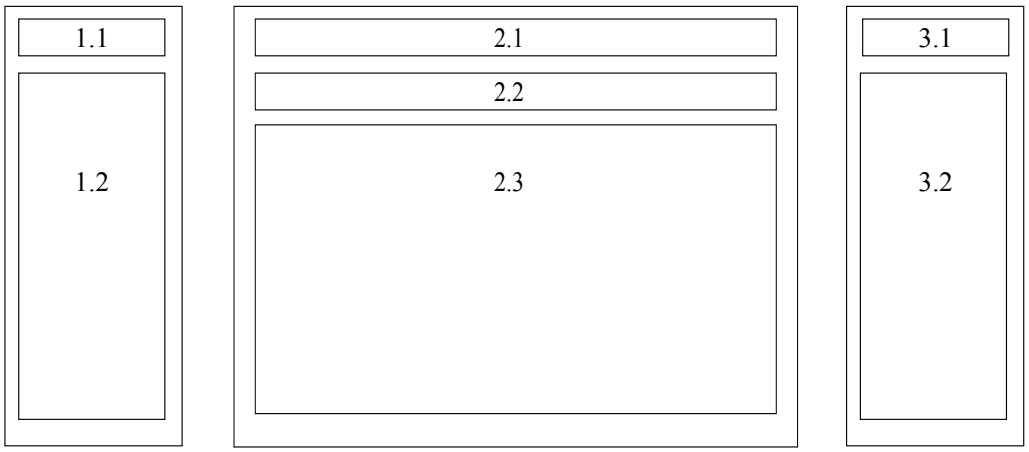


Abbildung 5.5: Darstellung der Möglichkeiten, nur ein oder zwei Module anzeigen zu lassen.

5.4 Einfache Erweiterung durch Modularität

Ein Grundgedanke dieses Re-Designs ist es, die Module modular zu gestalten und in Zukunft auch so zu halten. Das bedeutet, dass eine Art von Information nicht in zwei Module aufgeteilt werden soll. Dies widerspricht in einem gewissen Maße der Fusion von Daten, wie sie auch im Kapitel 4.1.2 angesprochen wurde. Diese Entscheidung wurde bewusst getroffen, um die GUI nachhaltiger zu gestalten. Wenn sich an der Hardware etwas verändert, kann man durch die Modularität dieses besser in der GUI umsetzen, da es nur einen Bereich, aber nicht die ganze GUI betrifft.
Dieses Prinzip wird vorallem in dem Plug-in Modul sehr deutlich: Dieses Modul beherbergt die kleineren Informations-Displays, die als Plug-ins bezeichnet werden. Alle Plug-ins müssen ein bestimmtes Interface implementieren, um eine vorgegebene Schnittstelle für das Plug-in Modul zu bieten. Der Vorteil liegt hier in der Zu- und Abschaltbarkeit eines Plug-ins und in der sehr einfachen Integrierung von Neuerungen. Zum Beispiel sind die Kamerabilder dort gut als Plug-in aufgehoben: Obwohl dieses Re-Design für den Kurt3D Roboter mit zwei Kameras gedacht ist, kann man dieses Kamera-Plug-in durch ein anderes Plug-in (für z.B. nur eine Kamera) ersetzen. Würde man in dem 2D-Modul auf Kameradaten zurückgreifen, müsste man auch das 2D-Modul anpassen.
Die Modularität gewährt also Austauschbarkeit. Der Preis ist, wie gerade schon angesprochen, dass Daten nicht beliebig fusioniert werden sollten. Im Zuge des Designprozesses fiel auf, dass eine „radikale“ Datenfusion nicht immer zwanghaft die GUI, und damit die Situationskenntnis des Operators, verbessert. Wird zuviel Information z.B. in einem Bild kodiert, wird es für den Operator schnell unübersichtlich: Fusion kann auch zur kognitiven Last werden (siehe Diskussion im Kap. 6.1.3). Deshalb gilt es, ein sinnvolles Maß an Datenfusion an den richtigen und möglichen Stellen zu finden. Noch können aus technischen Gründen bestimmte zusammengehörige Daten nicht vereint werden. Geeignete Stellen sind somit noch rar. Um später ein richtiges Maß zu finden, wäre eine weitere Evaluation nach der Implementierung wichtig, um den Entwicklungs-Zyklus weiter zu betreiben und zu verbessern.



(a) Die Scan-Liste. (b) Die 3D-Darstellungsfläche. (c) Die Markierungsliste.

Abbildung 5.6: Eine schematische Darstellung der Modul-Untereinheiten.

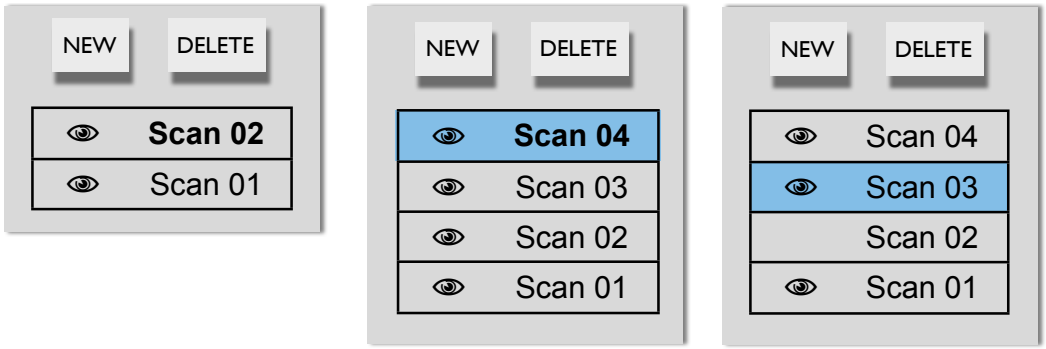
5.5 Beschreibung der einzelnen Module

5.5.1 Das 3D-Modul

Das 3D-Module besteht aus drei Teilen, die hier als Modul-Untereinheiten bezeichnet werden. Links befindet sich eine Liste mit den Scans (Abb. 5.6(a)), in der Mitte die 3D-Darstellungsfläche (Abb. 5.6(b)), rechts eine Liste der Markierungen (Abb. 5.6(c)). Es folgt nun eine genauere Beschreibung der einzelnen Modul-Untereinheiten.

Die Scanliste

Im oberen Teil dieser Modul-Untereinheit (Abb. 5.6(a), 1.1) befinden sich zwei Buttons, „New“ und „Delete“. Mit ersterem kann man einen 3D-Scan auslösen, mit letzterem kann ein 3D-Scan wieder gelöscht werden. Wie im Anhang A beschrieben, war es in der alten GUI möglich, mit der Taste F11 den letzten 3D-Scan zu löschen oder ihn durch entsprechende Tastenkürzel manuell einzulagern. Einen Zugriff auf die älteren Scans ist nur durch das Löschen der jüngeren Scans möglich. Dieser Zugriff soll nun dem Operator durch die sogenannte „Scanliste“ gegeben werden, die in der Abb. 5.6(a) als 1.2 gekennzeichnet ist. Hier werden die 3D-Scans nacheinander aufgelistet. Diese Abfolge soll in der Abbildung 5.7 verdeutlicht werden: Wird nach dem Scan 01 ein zweiter Scan gemacht, so wird dieser über dem Vorgänger eingereiht. Da er neu und unberührt ist, wird er zudem mittels fetter Schriftart kenntlich gemacht. In der mittleren Darstellung, 5.7(b), wurde vom Operator der noch neue, vierte Scan angeklickt und damit ausgewählt. Auch in der 3D-Darstellungsfläche wird dieser Scan dann farblich von den anderen Scans abgehoben. Im



(a) Liste nach dem zweiten 3D-Scan.

(b) Der neue, vierte 3D-Scan ist ausgewählt.

(c) Der zweite Scan ist nicht sichtbar, der dritte ausgewählt.

Abbildung 5.7: Vergrößerte Darstellung der Scanliste aus Abb. 5.6(a).

markierten Zustand kann dieser Scan mit dem „Delete“-Button gelöscht werden. Für ein ungewolltes Löschen steht das „undo“- Sicherheitsnetz der Menüleiste zur Verfügung.

Wenn ein neuer Scan einmal markiert wurde, wird er danach auch in regulärer Schriftart dargestellt, wie es auch in Abbildung 5.7(c) zu sehen ist. In der selben Abbildung kann man auch eine weitere Neuerung erkennen: Es ist das Sichtbarmachen bzw. Unsichtbarmachen eines Scans. Hier ist beispielsweise der zweite Scan unsichtbar gemacht worden. Er wird also nicht mehr in der 3D-Darstellungsfläche angezeigt. Wenn der Operator wieder den Scan sichtbar machen will, muss er auf die Stelle klicken, wo das Auge eigentlich wäre. Dieses Prinzip des Ein-/Ausblendens soll die Beobachtung aus dem Kapitel 3.3.1 (Seite 29) aufgreifen: Je mehr Scans gemacht werden, desto mehr redundante Punkte gibt es. Folglich werden einige Stellen fast flächig grün. Das manuelle Einlagern kann dadurch erschwert werden. Indem man nun Scans beliebig un-/sichtbar machen kann, bekommt der Operator mehr Freiheit zugesprochen. Im Wettkampf kann ein Scan einfach ausgeblendet werden, wenn der Operator sich nicht ganz sicher ist, wie der Scan eingelagert werden sollte. Damit geht dieser Scan nicht verloren. Wenn man Ende des Wettkampfes die 3D-Karte ausgedruckt wird, kann der Operator auswählen, welche Scans angezeigt bzw. mitgedruckt werden sollen.
Dieses Prinzip wird beispielsweise im Adobe Photoshop verwendet: Dort werden sogenannte Layers verwendet, die nicht nur ein-/ausblendbar sind, sondern auch noch gesichert und anderweitig manipulierbar sind.

Erweiterbarkeit Diese GUI soll nachhaltig gestaltet werden. Folglich sind durchaus Plätze für zukünftige Erweiterungen freigehalten worden. Diese werden hier kurz, auf die Nummerierung der Abbildung 5.6(a) bezogen, benannt.

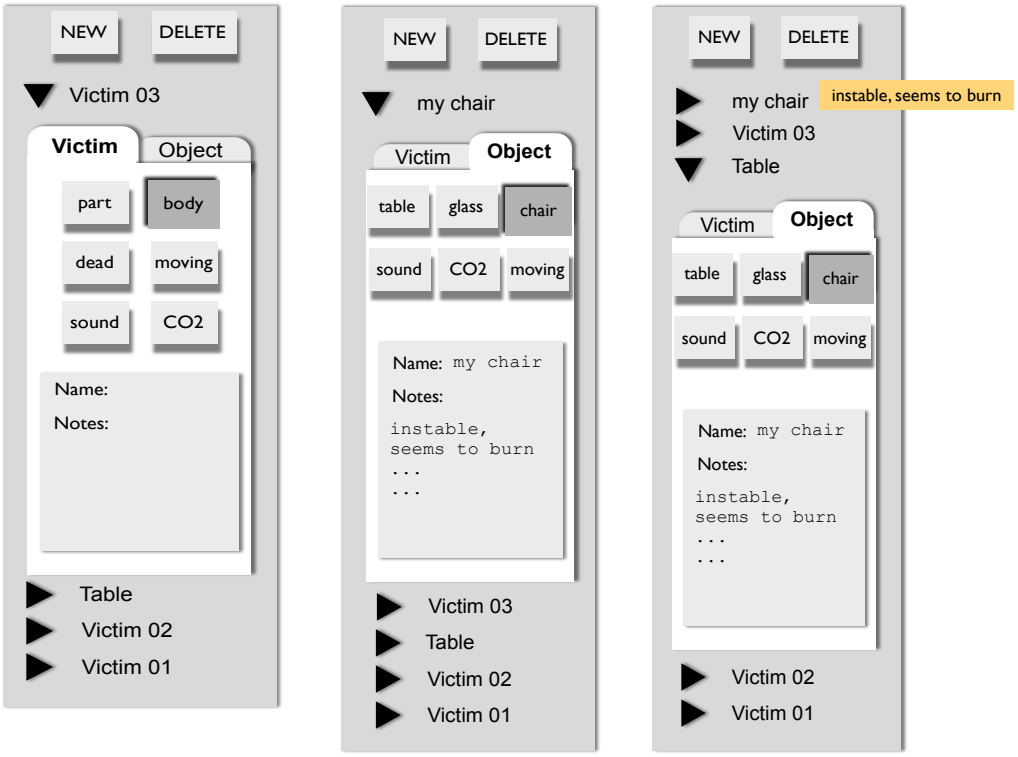
– 1.1: Wenn in Zukunft noch grundlegende Funktionen hier gewünscht werden, können diese hier ihren Platz finden.

– 1.2: Während des Wettkampfes werden nicht sehr viele Scans gemacht. Die Scanliste wird deshalb nicht allzu lang werden. Im unteren Teil der Liste können Icons für bestimmte Funktionen implementiert werden. (Wichtig ist es aber, die Modularität beizubehalten. Folglich müssen die Funktionen auch 3D-Modul-Funktionen sein.)
Sollten ähnliche Funktionalitäten wie die Sichtbarkeit implementiert werden, bietet es sich an, die Liste durch weitere Icons (neben dem „Auge“) anzubringen.

Die Markierungsliste

Diese Markierungsliste (Abb. 5.6(c), ist optisch das symmetrische Pendant zu der Scanliste. Die Modul-Untereinheit 3.1 enthält auch ein „New“- und „Delete“-Button. Mit dem „New“-Button wird hier eine Markierungsbox in die Mitte des letzten Scans gelegt. Der „Delete“-Button dagegen löscht eine Box aus dem Scan heraus. Wieder ist der Zugriff auf eine Box durch eine Liste gewährleistet, die nach dem gleichen Prinzip wie in der Abbildung 5.7 gezeigt wird. Diesmal aber geht es nicht um die 3D-Scans, sondern um Markierungen von Opfer und markanten Objekten/Stellen im 3D-Scan.
In der alten GUI konnte man mit einem Tastendruck („b“, siehe Anhang A) in den sogenannten Box-Modus gelangen und gleichzeitig eine Box einblenden. Es gab keinen Platz für diesen Modus in der GUI, sondern war nur durch einen die Taste „b“ einzuschalten. Sicherlich wäre der Operator im Wettkampf nicht darüber erfreut, wenn er doch den Shortcut vergisst und dann ein Opfer nicht markieren kann. Es ist daher sinnvoll, dieses Opfer-Markieren auch in der GUI sichtbar zu machen und nicht hinter einem Tastenkürzel zu verstecken.

Opfer und Objekte Es dreht sich aber in dieser Markierungsliste nicht alles um Opfer, die doch ziemlich selten zu finden sind. Ein Experte meinte, dass es für ihn wichtig wäre, sich um ein Objekt drehen zu können. Um ein Objekt als Zentrum für eine Rotation im Raum zu bestimmen, muss es erst markiert werden. Da diese Markierung die gleiche Markierung wie die eines Opfers mit einer Box ist, kann man das Prinzip auch darauf anwenden. Sicherlich wäre es nun redundant, eine Modul-Untereinheit für die Markierung von Opfern und eine für die Markierung von Objekten zu schaffen. Folglich gibt es hier nur Boxen, die erst im Nachhinein vom Operator als eine „Box für ein Opfer“ oder als eine „Box für ein Objekt“ bestimmt werden. Merkt der Operator später, dass das vermeintliche Opfer eigentlich gar keines ist, kann er es auch im Nachhinein noch als ein Objekt einstufen oder aber auch mit dem „Delete“-Button löschen. (Auch hier gibt es wieder die Sicherheit der „undo-/redo“-Funktion der Menüleiste.)
Die Unterscheidung zwischen Opfer und Objekt findet in der Markierungsliste statt: Dort gibt es zwei verschiedene Reiter – einen für Opfer, einen für Objekte. Wählt man den Opfer-Reiter aus, so erscheinen Icons, mit denen Wettkampf-relevante Informationen bzgl. des Opferzustandes vom Operator angeklickt werden können. Beispielsweise, ob es sich nur um ein Körperteil handelt, oder ob noch von dem Opfer Lebenszeichen (Kohlendioxid, Bewegung, Geräusche) ausgehen. Durch die Option, ein Opfer in der GUI gleich näher zu



(a) Ein neues Opfer wurde markiert und hier angelegt.

(b) Ein Objekt wurde markiert, hier angelegt und benannt.

(c) Tooltip: Beim Drüberfahren mit der Maus können Informationen angezeigt werden.

Abbildung 5.8: Aufbau und Darstellungen der Markierungsliste.

bestimmen, soll dem Operator Arbeit außerhalb des Rechners abgenommen werden: Er soll also nicht zu Bleistift und Papier greifen, um Notizen zu machen, denn diese Informationen sollen ihren Platz in der GUI finden.
Der Reiter für die Markierung eines Objekts unterscheidet sich hier in den Informationen, die eingegeben werden. Während die Opfer einfach automatisch durchnummeriert werden, wäre es bei Objekten wichtiger, ihnen einen Namen/Titel zu geben, beispielsweise „Tisch“, „Loch“ oder „Glaswand“. In diesem Reiter liegt es vorallem an dem Operator, welche Icons/Textfelder er gerne haben möchte.
Wenn der Operator auf „New“ klickt, wird eine Box in den letzten 3D-Scan gesetzt. Eine Box wird, wie schon in der alten GUI, eine lokal begrenzte Einfärbung der Punkte sein. Um aber die Kritiken und Probleme der Novizen mit der Box aufzugreifen (siehe Kapitel 3.3.1), soll hier nochmals hervorgehoben werden, dass die Box in der Mitte des letzten 3D-Scans auftauchen soll. Es ist nicht akzeptabel, dass eine Box „irgendwo“ versteckt ist und sie erst weit verschoben werden muss.
In der Abbildung 5.8 werden diese Abläufe bzgl. der Markierungsliste nochmals dargestellt.

Erweiterungen Diese Untereinheit bzw. deren Icons/Texteingabefelder sind stark von den individuellen Befürfnissen eines Operators abhängig. Da die einzelnen Opfer/Objekte aufgeklappt werden können, ist keine freie Fläche wie bei der Scanliste absehbar. Im Bereich 3.1 könnten nach Bedarf wiederum benötigte Icons eingesetzt werden.

Die 3D-Darstellungsfläche

Diese Modul-Untereinheit ist sicherlich das Herzstück des Moduls (Abb. 5.6(b)). Wie schon weiter vorne im Kapitel 3.3.2 beschrieben, soll die Maus mehr Funktionen übernehmen. Dieser Entwurf des Interfaces baut folglich darauf auf: Die Maus wird das Hauptwerkzeug für die Bedienung des 3D-Moduls. Mit ihr sollen nicht nur die Scans im Modul bewegt, sondern auch Markierungen vorgenommen werden. Dadurch verändert sich der Arbeitsfluss vorallem bezüglich der Eingabegeräte stark, und die Bedienung des 3D-Moduls wird dadurch vereinfacht.

Die verschiedenen Projektionen Auch hier wird das Reiter-Prinzip wieder aufgegriffen: Der Operator kann sich mehrere Reiter anlegen, die jeweils verschiedene Perspektiven darstellen. Wenn die GUI zum ersten Mal gestartet wird, sind nur zwei Standard-Reiter vorhanden: Die (am meisten benutzte) parallele Projektion (auf die X/Z-Ebene), und ein Präferenzen-Reiter. Ein Kontext-Menü, das durch einen Rechtsmausklick auf den ersten Reiter ausgelöst wird, beinhaltet verschiedenste Projektionen. Indem man eine Projektion anklickt, wird ein Haken vor den entsprechenden Schriftzug im Kontext-Menü gesetzt und ein neuer Reiter mit der gewählten Projektion taucht auf.
Jeder Operator wird verschiedene Vorlieben bzgl. der verschiedenen Projektionen haben. In der alten GUI hab es nur zwei Ansichten, zwischen denen man mit der „F1“-Taste hin- und herschalten konnte. Nun muss sich der Operator nicht mehr auf nur zwei Projektionen reduzieren, sondern kann sich seine Favoriten aussuchen. Dabei wird hier vorausgesetzt, dass der Operator sich mit den Möglichkeiten genau auseinandersetzt und die jeweiligen Vor- und Nachteile in Bezug auf bestimmte Aktionen abwägt. Damit wird der Vorteil der Mausbenutzung hier offensichtlich: Der Operator hat Zugriff auf mehrere verschiedene Projektionen, ohne sich die entsprechenden Shortcuts merken zu müssen: Wieder eine Verringerung der kognitiven Last.

Schneller Zugriff auf Filter Der Präferenzen-Reiter ist eigentlich eine Ansammlung der zuschaltbaren Filter, die einst in der 3D-Funktionsleiste vorhanden waren (siehe Abb. 2.8(a), Seite 15, bzw. Beschreibung im Anhang B). Einige Funktionen, z.B. die des „save & animate & print“ Menüs, sollten in die Menüleiste verlagert werden. Um aber Funktionen, die oft benutzt werden, schneller zu aktivieren, sind über den Reitern Icons/Buttons vorgesehen. Damit ist ein schneller Zugriff gewährleistet und das Suchen in der Funktionsleiste, wie es beispielsweise im Kapitel 3.3.2 auf Seite 41 beschrieben wurde, entfällt.

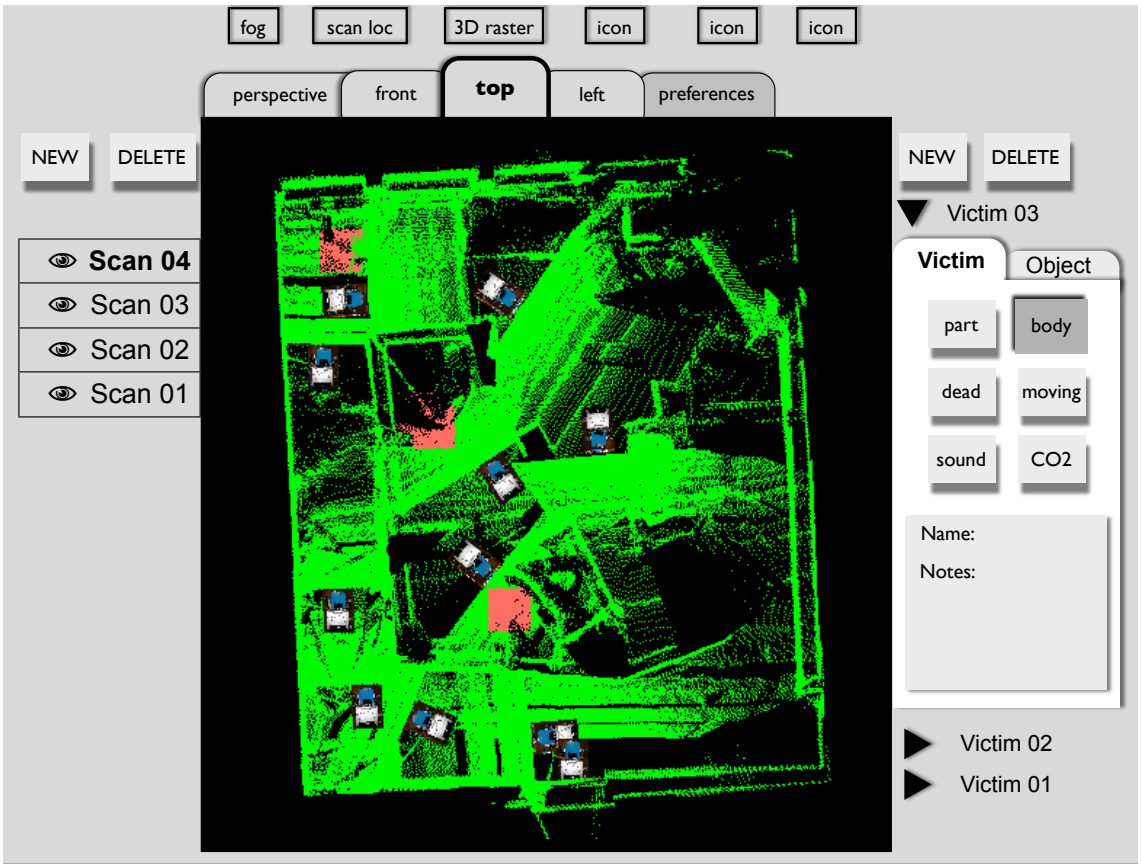
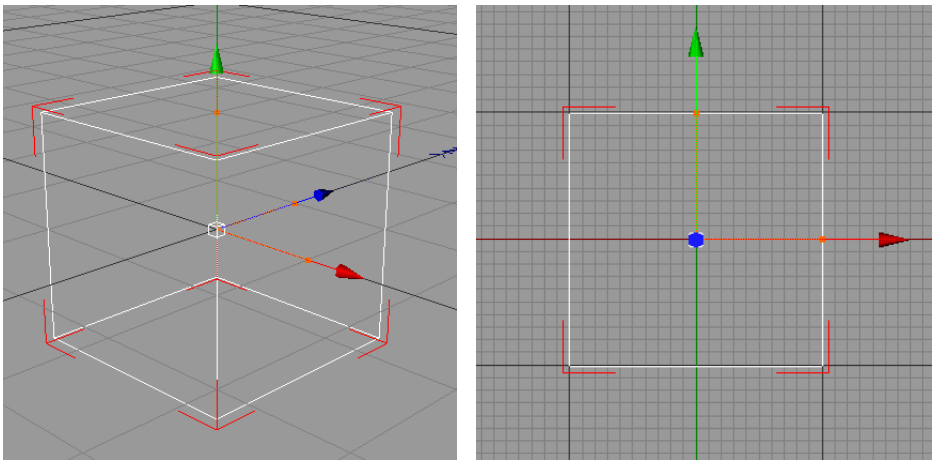


Abbildung 5.9: Entwurf des ganzen 3D-Moduls

Das Koordinatenkreuz Um besser mit den Scans umgehen zu können, wird auf die Einblendung eines Koordinatenkreuzes zurückgegriffen. In 3D-Programmen ist dies üblich: Abbildung 5.10 zeigt beispielsweise die Darstellung des Koordinatenkreuzes im Programm „Cinema 4D“. Wählt man also einen Scan oder eine Markierungsbox aus, so erscheint in dessen Mitte ein Koordinatenkreuz. Um die entsprechende Achse für die Translation/Rotation auszusuchen, klickt man mit der Maus auf eine der Achsen des Koordinatenkreuzes. Genau eine Achse wird ausgesucht, indem man mit dem Mauszeiger auf den vorderen Teil des Achsenpfeils kommt. Dieser wird dann vollständig eingefärbt, damit man weiß, dass nun eine Achse aktiviert ist (siehe Abbildung 5.11(a)). Wenn man eine Ebene aktivieren will, nähert man sich dem Ursprung der beiden Achsenpfeile (siehe Abbildung 5.11(b)). Dann werden beide Pfeile als aktiviert gekennzeichnet. Durch ein erneutes Auswählen einer anderen Achse bzw. anderer Achsen wird eine alte Aktivierung inaktiv. Man kann aber auch die Ansicht auf alle Scans verändern, indem man weder einen Scan noch eine Markierungsbox aussucht und mit dem Mauszeiger über die 3D-Darstellungsfläche fährt. Der Mauszeiger wird dann zu einer „Hand“. Mit gedrückter Maustaste kann dann die Ansicht nach links oder rechts, oben oder unten verschoben werden. Um die Ansicht



(a) Darstellung in der perspektivischen Projektion.

(b) Darstellung in der parallelen Projektion.

Abbildung 5.10: Das Koordinatenkreuz im Programm „Cinema 4D“.

näher heranzuholen, ist das Scrollrad vorgesehen.

Bei der Translation und Rotation der Scans ist es relevant, in welcher Projektion man dies machen will.

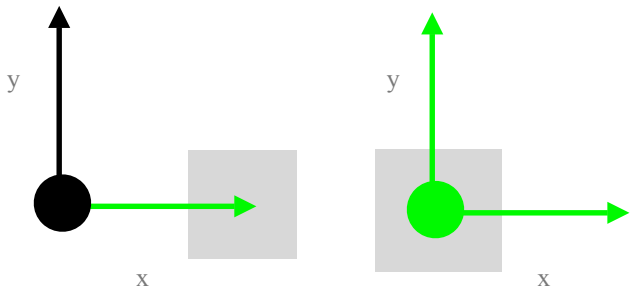
- Hat man eine parallele Projektion vor sich, so kann man nur in der Ebene oder entlang einer der beiden Achsen translatieren. Ebenso ist es hier nicht möglich, um die ebeneinschließenden Achsen zu rotieren.
- In der perspektivischen Projektion sind alle drei Achsen zugänglich. Es sind Translationen entlang einer Achse erlaubt, nicht aber in einer Ebene. Rotationen sind auch nur um jeweils eine Achse möglich.

Es wurde absichtlich davon abgesehen, eine Rotation um mehrere Achsen zuzulassen, da es schnell zu Verwirrung führen kann. Das Paradebeispiel dafür war in der alten GUI die sogenannte Rotationskugel (siehe Kapitel 2.3 auf Seite 11), die nicht gerade Anklang bei den Versuchspersonen fand, wie in Kapitel 3.3.2 (Seite 41) beschrieben wurde.

3D-Scans bewegen

Da es einen Unterschied gibt, ob man einen Scan bzw. alle Scans bewegen will, und dem Setzen einer Markierungsbox, ist es wichtig, auf eine Inkonsistenz in der Maustastenbelegung hinzuweisen.

Die Maustasten beim Scanbewegen Das Basismodell einer Maus, von dem hier ausgegangen wird, ist eine Zwei-Button-Maus mit einem Scrollrad. Generell ist das *Scrollrad*



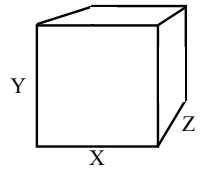
(a) Aktivierung einer Achse an der Pfeilspitze.

(b) Aktivierung einer Ebene am Ursprung.

Abbildung 5.11: Aktivierungzonen am Koordinatenkreuz in einer Parallelprojektion.

für die „Zoomfunktion“ zu gebrauchen. Die ganze 3D-Darstellungsfläche wird „vergrößert“, unabhängig davon, ob etwas aktivert oder in den Listen ausgewählt ist.
Die *linke Maustaste* ist für die Translationen zuständig. Das heißt, dass man mit gedrückter Maustaste entlang einer Achse oder in der von den beiden Achsen eingeschlossenen Ebene den Scan bzw. die Scans verschieben kann. Der *rechten Maustaste* obliegt die Rotation. Mit der gedrückten, rechten Maustaste kann also um die vorher aktivierte Achse rotiert werden, wenn man sich in der perspektivischen Projektion befindet. Bei der Ansicht einer parallelen Projektion kann man um die nicht sichtbare Achse rotieren, indem man die Ebene der anderen beiden Achsen aktiviert und dann mit der gedrückten rechten Maustaste rotiert.
Wird in der Scanliste nur ein einzelner Scan ausgesucht, dann taucht das Koordinatenkreuz in der Mitte dieses Scans auf. Dieser aktivierte, einzelne Scan wird dann unabängig von all den anderen Scans translatiert bzw. rotiert. Dies entspricht dem *manuellen Einlagern in der alten GUI*. Wird der Scan in der Liste nach dem neuen Ausrichten wieder „entwählt“, dann darf das System von der neuen Position des Scans ausgehend nochmals versuchen, den Scan besser einzupassen. Wird der Scan in der Liste nicht „entwählt“, so bleibt der neu positionierte Scan dort liegen, wo er manuell hingeschoben wurde und darf nicht vom System selbstständig neu eingelagert werden.
Die Größe der Darstellung des Koordinatenkreuzes ist dabei aber immer gleich groß zu halten, damit die Achsen auch immer mit dem Mauszeiger sinnvoll greifbar bleiben und nicht zu klein werden.

Die Maustasten beim Setzen einer Markierungsbox Wurde in der Markierungsleiste eine neue Box initialisiert, dann muss diese erst in den Scan eingepasst werden, damit die richtigen Punkte der Wolke markiert werden. Die Translation einer Box funktioniert wie bei dem Scanbewegen anhand der linken, gedrückten Maustaste. Die Inkonsistenz, die hier nun bewusst Einzug erhält, ist, dass die rechte Maustaste nicht für die Rotation der Box zuständig ist, da diese nicht rotiert werden muss. Sie ist stattdessen für die Skalierung der Boxengröße zu gebrauchen, wenn man eine Parallelprojektion vor sich hat. Ist man



	Skalierungscursor am Koordinatenkreuz		Handcursor in der Darstellungsfläche	
	linke Maustaste	rechte Maustaste	linke Maustaste	rechte Maustaste
Scan ausgewählt (Koordinatenkreuz angezeigt)	Translation des Scans entlang X, Y oder Z	Rotation des Scans um X, Y oder Z	Translation des Ansicht entlang X, Y oder XY	Rotation der Ansicht um Mittelpunkt (Z)
Box ausgewählt (Koordinatenkreuz für Box angezeigt)	Translation der Box entlang X, Y oder Z	Skalierung der Box in X, Y oder Z	Translation der Ansicht entlang X, Y oder XY	Rotation der Ansicht um Mittelpunkt (Z)
nichts ausgewählt	/	/	Translation der Ansicht entlang X, Y der XY	Rotation der Ansicht um Mittelpunkt der Scanpunktewolke

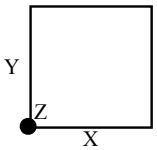
Abbildung 5.12: Tabelle für eine perspektivische Projektion.

dagegen in der perspektivischen Projektion, ist der Mittelpunkt der Box nun das Zentrum für die Rotation. Man kann sich also (mit Einschränkung) „um ein Objekt herumbewegen“. Wie schon in Kapitel 5.5.1 angesprochen, war dies ein Wunsch eines Experten, der damit realisiert wird.
Die beiden Abbildungen 5.12 und 5.13 auf den Seiten 74 bzw. 75, fassen nochmals die Funktionsweisen der Maustasten unter den verschiedenen Bedingungen tabellarisch zusammen.

5.5.2 Das 2D-Modul

Der 2D-Viewer der alten GUI war für die Orientierung am wichtigsten, wie schon im Kapitel 3.3.2 (Seite 38) beschrieben wurde. Der Umgang damit war leicht zu erlernen. Die Darstellungsweise wird beibehalten, um die schon vorhandene Klarheit im Umgang auszunutzen. Im Kapitel 6 wird diese Entscheidung nochmals diskutiert.

Der Roboter-Avatar Die Ansicht im 2D-Modul wurde im allgemeinen so beibehalten, wie es von Vornherein gegeben war. Die Buttonleiste über dem alten 2D-Viewer (siehe Kap. 2.3, S. 7) wird deshalb wegfallen. Die *Display-Funktionselemente*, die darin ihren Platz fanden, werden nun in in einem Kontext-Menü zugänglich sein. Übrig bleiben die *Steuerelemente*, also folgende Buttons: *„3D“, „Auto“* und *„StopGio“*. Die beiden ersten werden weiter offen in dem 2D-Modul angeordnet werden, damit sie schnell zugänglich sind (siehe Abb. 5.14). Der *„StopGio“*-Button wird auf eine etwas andere Art in das Modul integriert: Er wird als eine Raute auf dem Roboter-Avatar dargestellt, die durch drag&drop im 2D-Bereich zum Zielpunkt gesetzt werden kann. Damit ist der Algorithmus automatisch



	Skalierungscursor am Koordinatenkreuz		Handcursor in der Darstellungsfläche	
	linke Maustaste	rechte Maustaste	linke Maustaste	rechte Maustaste
Scan ausgewählt (Koordinatenkreuz angezeigt)	Translation des Scans entlang X oder Y	Rotation des Scans um Z	Translation des Ansicht entlang X, Y oder XY	/
Box ausgewählt (Koordinatenkreuz für Box angezeigt)	Translation der Box entlang X oder Y	Skalierung der Box in X oder Y	Translation der Ansicht entlang X, Y oder XY	/
nichts ausgewählt	/	/	Translation der Ansicht entlang X, Y der XY	/

Abbildung 5.13: Tabelle für eine parallele Projektion.

aktiviert. Er ist gestoppt, wenn der Roboter sich diese Raute wieder „einverleibt“ hat, also die Raute wieder in der Mitte des Avatars ist. Oder aber der Operator klickt auf das Stop-Symbol, das während des Ablaufs des Algorithmus in der Mitte des Avatars dann sichtbar ist. Durch das Stoppen wird die Raute wieder automatisch auf dem Avatar angezeigt, um den Abbruch der selbstständigen Navigation anzuzeigen. Diese grafische Darstellung des Algorithmus soll deren Gebrauch motivieren, weil sie immer schnell greifbar ist. Die GUI sollte dadurch im Sinne der NIST-Richtlinien (S. 57) effizienter werden.

Der Avatar wird nicht nur die „GioRaute“, sondern wird auch die Maussteuerung in sich tragen. Dies ist wahrscheinlich der intuitivste Weg, um die Steuerung mit dem gesteuerten Objekt in Zusammenhang zu bringen. Einen ähnlichen Ansatz kann man auch bei dem INEEL Interface in [21] sehen. Dass bei deren GUI dann aber der „Press to quit robot programs“- Button in die Mitte des Roboter-Avatars gesetzt wurde, scheint im Sinne der ISO 9241 Teil 10 bzgl. der Fehlerrobustheit, Stichwort Sicherheitsabstände, nicht sinnvoll zu sein. Eine solch nachhaltige Funktion sollte eigentlich in der Menüleiste untergebracht werden und nicht im Zentrum der Robotersteuerung liegen.

Farbliche Kodierung Um dem Operator Entscheidungshilfen zu geben, kann eine farbliche Kodierung der Scanpunkte und der Maussteuerung eingeschaltet werden. Scanpunkte, die sehr nah am Roboter sind, werden rot eingefärbt. Ebenso werden die Steuerungs-Buttons der korrespondierenden Richtung rot eingefärbt, um dem Operator darauf hinzuweisen, dass dort ein Objekt ist und er vielleicht besser in eine andere Richtung navigiert. Welche andere Richtung das sein kann, wird auch in den Steuerungs-Buttons kodiert:

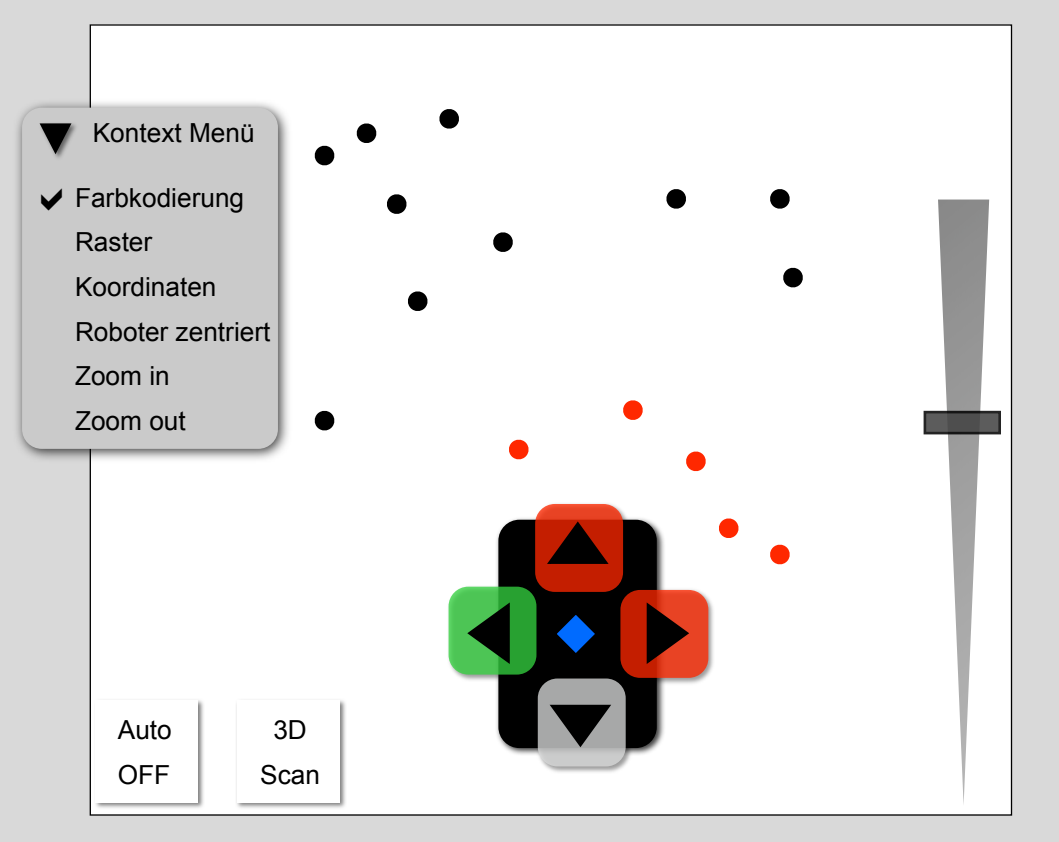


Abbildung 5.14: Entwurf des 2D-Moduls

Grüne Farbe verweist darauf, dass dort keine nahen 2D-Punkte erfasst wurden, graue Farbe bedeutet, dass keine Daten vorliegen (z.B. für die Steuerung nach hinten, wenn der 180 Grad Scanner angeschlossen ist). Diese farbliche Kodierung kann im Kontext-Menü zu- bzw. abgeschaltet werden.

Die Geschwindigkeitsreglung Es gibt hier zwei Möglichkeiten, wie man die Geschwindigkeit mit der Maus einstellen kann. Erstere ist, dass man das Scrollrad dafür verwendet. Zweite ist, dass man einen Geschwindigkeitsregler oder -schieber in dem Modul unterbringt. Beides hat Nach- und Vorteile. Benutzt man das Scrollrad, so ergibt sich eine Inkonsistenz mit der Benutzung des Mausrades im 3D-Modul, wo es als Zoom dient. Eine Zoomfunktion ist in dem 2D-Modul wohl zu vertreten, dennoch erscheint es als praktischer, diese in das Kontext-Menü zu verlegen, da sie selten genutzt wird. Ein Nachteil ist, dass das Scrollrad aus Versehen gedreht werden könnte, und damit ist nicht genügend Sicherheitsabstand gewährleistet.
Der Vorteil bei einer Darstellung eines Geschwindigkeitsreglers ist, dass dem Operator im-

mer offensichtlich ist, welche Geschwindigkeit eingestellt ist. Zwar wird die Geschwindigkeit nicht oft verändert, dennoch kann es eine wichtige Information sein. Der Geschwindigkeitsregler stört de „Purismus“ des 2D-Moduls, hat aber keine funktional-relevanten Nachteile. In diesem Re-Design wird ein Geschwindigkeitsregler angezeigt werden, da diese Information dem Operator zugänglich sein sollte. Ob man nun aber das Scrollrad (modul-konsistent) mit einer Zoomfunktion belegt, oder mit einer Geschwindigkeitsreglung, sei dem Operator überlassen, der für sich selbst die Nach- und Vorteile abwägen muss.

Das Kontext-Menü Durch einen Rechtsmausklick in das 2D-Modul ausgelöst, taucht das Kontext-Menü auf. Dort befinden sich die Funktionen, die selten benutzt werden. Hier ist wieder darauf zu achten, dass es klar ersichtlich ist, ob eine Option aktiviert ist. Eine Aktivierung wird meist mit einem Häkchen vor der Option kenntlich gemacht. Diese Prinzip sollte hier auch übernommen werden, zumal es schon im 3D-Modul so geplant ist.

Erweiterungen Für neue, wichtige Algorithmen können Buttons in das Modul eingebettet werden. Wichtig ist, dass es nicht zu viele werden und dass ein Sicherheitsabstand eingehalten wird, da hier *keine „undo“-Option* gegeben ist! Wenn ein Algorithmus aktiviert wurde, kann er nur abgebrochen, aber nicht rückgängig gemacht werden. Es empfiehlt sich deshalb dann einen Notfall-Stopp-Button leicht zugänglich in dem Modul unterzubringen. *Navigations-Modi*: In der alten GUI gibt es eine sehr gute Entscheidungshilfe: Der 3D-Scan-Button wird rot eingefärbt, wenn ein 3D-Scan gemacht werden soll. Dies funktioniert inzwischen sehr zuverlässig. Zweifellos wird dies mit in das Re-Design übernommen werden. Weiter Entscheidungshilfen bzgl. der Auswahl der Navigations-Modi können hier noch nicht gegeben werden, da es noch keine anderen Modi, abgesehen vom dem Gio-Algorithmus, gibt. Wenn es in Zukunft weitere Modi geben wird, könnten sie, ähnlich wie der eingefärbte 3D-Scan-Button, in das Modul integriert werden.

5.5.3 Das Plug-in Modul

Plug-ins sind aus vielen verschiedenen, gängigen Applikationen bekannt. Sie erlauben es dem Nutzer z.B. bestimmte Informationen zu einem bestimmten Zeitpunkt zur Verfügung zu haben, neue Funktionen zu ermöglichen, Applikationen miteinander zu verbinden, etc. Wenn ein Nutzer seine Applikation mit einer neuen Funktion ausstatten will, ist es praktisch, wenn in der Applikation Plug-ins vorgesehen sind. Durch die festgelegte Schnittstelle muss der Nutzer sich nur um einen relativ kleinen Teil der Programmierung kümmern.
Um den stetigen Verbesserungen und Erweiterungen des Systems Platz zu geben, bietet sich ein Modul an, für das Plug-ins geschrieben werden können.

Die Filterfunktion der GUI

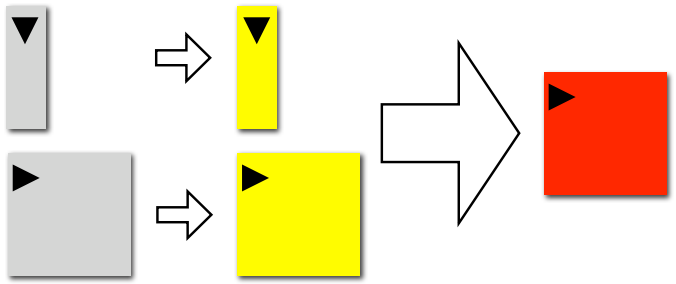
Welche Informationen/Funktionen relevant sind, ist nun nach der Evaluation deutlich klarer. Aber nicht alles kann in eine der beiden Schubladen „wichtig“ / „unwichtig“ gesteckt

werden: Der zeitliche Faktor ist entscheidend. Allgemein gesagt heißt das, dass eine Information zu einem Zeitpunkt völlig irrelevant ist, weil sie sich der assoziierte Wert im Normalbereich befindet. Kommt der Wert aber in einen kritischen Bereich, wird diese Information immer wichtiger für den Operator. Um die GUI nicht von Anfang an zu überladen, werden bestimmte Elemente nur in kritischen Situationen „auf sich aufmerksam machen“, z.B. durch Blinken oder durch ein akustisches Signal.
Etwas Blinkendes auf dem Bildschirm zieht Aufmerksamkeit auf sich, kann aber auch sehr schnell nerven und den Operator belasten. Deswegen soll dem Operator die Möglichkeit eingeräumt werden, diesen blinkenden Alarm duch einen „Hotkey“ auszuschalten. Es liegt im Ermessen des Operators, diesen Hotkey zu benutzen: Es ist ein zur Kenntnis nehmen der Warnung und eine bewusste Entscheidung, diese vorerst einmal zu ignorieren. Diesem Luxus muss aber ein „Timer“ entgegen gesetzt werden: Ist der Wert nach einiger, genau zu bestimmender Zeit noch immer im kritischen Bereich, so löst der Timer trotz dem vorherigen Benutzen des Hotkeys einen weiteren Alarm (visuell oder akustisch) aus. Auch diese beiden Funktionen brauchen einen Platz in der GUI, damit der Operator den Hotkey klicken kann, der auch durch einen Shortcut auslösbar ist. Wenn dadurch der Timer aktiviert wurde, soll der Operator davon optisch Kenntnis nehmen können (z.B. eine Sanduhr als Statusanzeige).

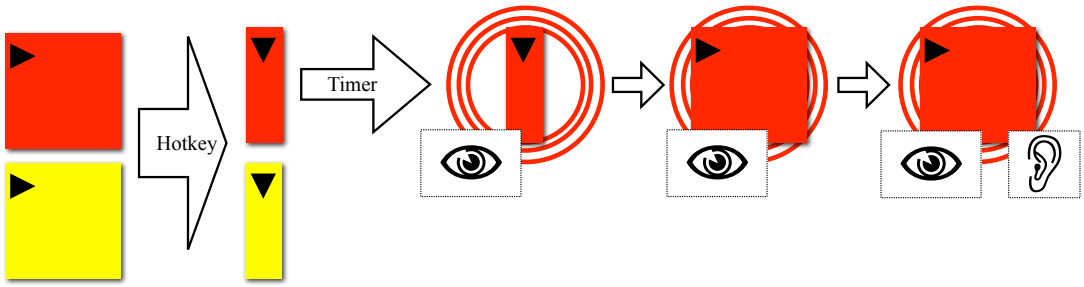
Die Dynamik der Plug-ins Die Dynamik des Plug-in-Prinzips soll sich nicht nur in der Austauschbarkeit niederschlagen. Sie haben auch verschiedene Größen und Zustände:

- *Größe*: Ein Plug-in hat zwei verschiedene Größen: minimiert und maximiert. Im minimierten Zustand zeigt ein kleiner Pfeil an, dass das Plug-in minimiert ist. Darunter befindet sich ein Icon, welches charakteristisch für das Plug-in ist. Ebenso kann unter dem Icon zusätzlich die wichtigste Information dargestellt werden. Unten links sollte immer ein Icon für die Entfernung des Plug-ins aus dem Modul sein.
- *Verschiedene Zustände*: Um die Filterfunktion umzusetzen, gibt es mehrere Zustände, in dem sich ein Plug-in befinden kann. Diese sind wiederum farblich kodiert: Gelb im warnenden Zustand, rot im Notfall. Zudem muss das Plug-in auch im roten Zustand blinkend angezeigt werden und akustisch auf sich mit einem Earcon aufmerksam machen können.

Die Größe und der Zustand sind voneinander abhängig. Die Abbildung 5.18 soll diese darstellen. Wenn ein Schwellenwert überschritten wird, wird das Plug-in gelb einfärbt, bleibt aber in seinem minimierten oder maximierten Zustand (Abb. 5.15(a)). Wenn aber ein Notfall eintritt, wird ein minimiertes Plug-in auf jeden Fall maximiert. Falls mehrere Plug-ins kritische Bereiche anzeigen sollten, würde im schlimmsten Fall ein Scrollbalken der neuen Breite der Plug-ins Herr werden. In diesem Fall hat der Operator sicherlich mehr zu tun, als sich um den unschönen Scrollbalken zu kümmern.
Der Hotkey minimiert sowohl die gelben Warnungen und auch die rot gefärbten Notfälle, wie die Abb. 5.15(b) illustriert. Der Timer, der individuell in den Präferenzen eingestellt werden kann, bewirkt das Auslösen von weiteren Alarmen, damit diese nicht in Vergessenheit geraten. Stufenweise wird der Operator dann permanent auf diesen Notfall aufmerk-



(a) Übergang: Normalzustand, Warnung, Notfall.



(b) Funktion des Hotkey und des Timers.

Abbildung 5.15: Schematische Darstellung der Abhängigkeit von Größe und Zustand.

sam gemacht: Erst durch Blinken, dann durch Maximierung und Blinken, dann zusätzlich noch mit akustischen Signalen.
Im weiteren gibt es noch ein Kontext-Menü, das mit einem Rechtsmausklick auf das Plug-in-Modul aufgerufen wird. Dort befindet sich eine Auflistung der zur Verfügung stehenden Plug-ins, die angewählt werden können. Klickt man also bei einem Plug-in (aus Versehen) auf das Schließen-Icon, so kann man es schnell wieder in das Modul eingliedern. Im weiteren ist an der rechten, untere Ecke eines jeden maximierten Plug-ins eine Skalierungsecke vorgesehen: Greift man mit der Maus diesen Eckbereich, so kann man das Plug-in größer und kleiner ziehen.

Das Plug-in Interface Durch Vorgabe einer Programmierschnittstelle, einem sogenannten Interface, werden bestimmte Funktionen der Plug-ins zur Implementierung vorgeschrieben. So sollten beispielsweise die Funktionen zur Alarmierung eingehalten werden, falls diese für das Plug-in notwendig sind. Wichtig sind auch Funktionen zur grafischen Darstellung (z.B. Farbe, Position), zum Mini- und Maximieren, Schließen, sowie zur Festlegung von Icons.

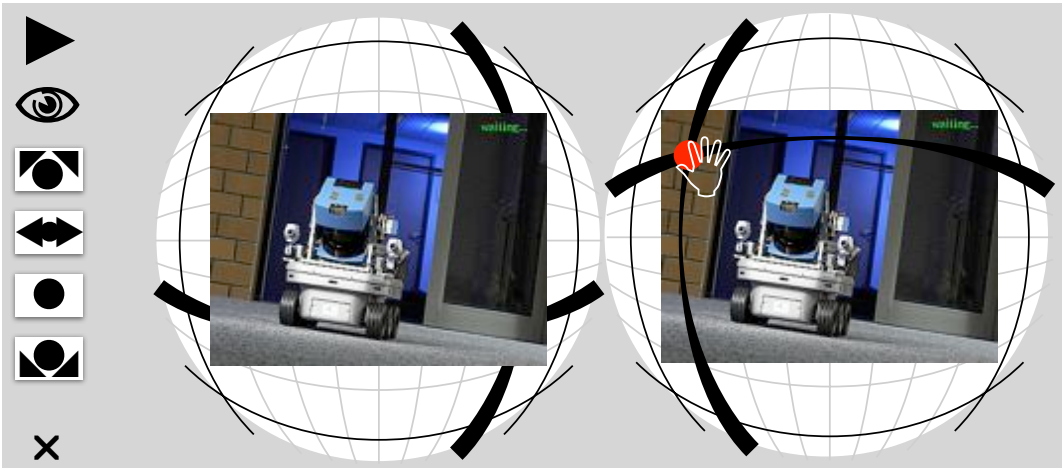


Abbildung 5.16: Ein Entwurf für Kameraroutinen, Kamerabilder und Kamerasteuerung.

Das Kamera-Plug-in

Der Entwurf dieses Plug-ins kombiniert viele Facetten der Kameras. Es ist der Anspruch, sowohl die Ausrichtung, die Kamerabilder, die Kamerasteuerung und die Kameraroutinen möglichst kompakt darzustellen. Oftmals wird die Ausrichtung der Kameras separat von den Bildern in die GUI integriert, da die Ausrichtung immer den Bezug zum Roboter zeigen sollte, um die Situationskenntnis, wie im Kapitel 4.1.1 (S. 44), zu verbessern. Separiert man diese Ausrichtung wieder von den Kamerabildern, so muss der Operator immer noch mitdenken, wie nun die Relation zwischen den beiden ist. Er bekommt also eine Information in zwei Elementen der GUI mitgeteilt. Ebenso gestaltet sich dann eine gute Integrierung der Kamerasteuerung sehr kompliziert.
Der Entwurf in der Abbildung 5.16 zeigt die Verschmelzung von allen Funktionen und Informationen. Es wurde dabei viel Wert auf die Intuitivität gelegt. Wie gut dieser Entwurf funktioniert, kann nur eine Implementierung/Test zeigen. Es folgt eine Beschreibung der einzelnen Facetten:

- Die Darstellung von Längen- und Breitengraden soll dem Operator den Eindruck vermitteln, das Zentrum der Kamera zu sein, die entlang dieser Linien gedreht werden kann. Sie sind *grau* dargestellt.
- Die horizontale Drehgrenzen sind durch zwei *dünne schwarze* Längengrade links und rechts dargestellt. Analog sind die vertikalen Drehgrenzen als Breitengrade oben und unten vorhanden. An den Stellen, wo sie sich kreuzen, beispielsweise links oben, werden die Extrempunkte der Kameradrehung dargestellt. Ab dort kann der Operator die Kamera nur noch nach unten oder rechts bewegen. Bewegt er sie, um das Beispiel weiterzuführen, nach rechts, so „liegt der Kamerakopf im Nacken“ und bewegt sich von links nach rechts.
- Die *dicken schwarzen* Linien zeigen die momentane Ausrichtung an. Da der Mensch,

gemäß der Gestalt-Prinzipien, die Linien auch hinter dem Kamerabild fortführt, kann er grob vermuten, wo sie sich kreuzen. Damit sollte die Ausrichtung in jedem Moment klar sein: Egal ob die Linien sich hinter oder neben dem Kamerabild kreuzen. Es entsteht sozusagen ein (gebogenes) Fadenkreuz, das noch weiter ausgenutzt werden soll:

- Die Kamerasteuerung wurde ebenso in diese imaginäre Halbkugel eingelagert: Bewegt der Operator die Maus in eine Kamera-Kugel hinein, so treten die dicken schwarzen Fadenkreuz-Linien in den Vordergrund des Bildes. Dort, wo sie sich kreuzen, taucht wieder ein Dragger auf, den man durch eine drag&drop-Bewegung in den durch die vier dünnen schwarzen Linien begrenzten Bereich verschieben kann. Die dicken schwarzen Linien folgen dem Dragger dabei. Nach der Bewegung, oder wenn der Operator die Maus aus dem Bereich bewegt, treten die Linien wieder in den Hintergrund.
- Am linken Rand des Plug-ins sind die Kameraroutinen untergebracht. In diesem Entwurf sind vier vorgesehen, die aber beliebig vom Operator festgelegt werden können. Der oberste Button symbolisiert die Routine, dass beide Kameras nach außen, oben gedreht werden. Der Button darunter dreht die Kameras auf dem Äquator nach außen. Der vorletzte zentriert die Kameras, während der letzte dann nach außen, unten blicken lässt. Ist eine Routine aktiv, so kann dies durch Einfärbung (blau, grün) auch dort angezeigt werden. Damit wird die Ausrichtung dann für den Operator noch schneller ersichtlich. Die Buttons sind auch im minimierten Zustand angezeigt, damit man sie anklicken kann. Damit dies sinnvoll ist, muss das Plug-in maximiert werden.
- Wird eine Kamera mit dem Cooliehead am Joystick angewählt und gesteuert, sollte das Fadenkreuz auch in den Vordergrund treten. Damit erkennt der Operator, welche Kamera gerade ausgewählt ist und gesteuert werden kann. Diese Information war in der alten GUI in der Konsole ablesbar und soll nun, leicht erkennbar, am Ort des Geschehens dargestellt werden. Der Blick des Operators muss nicht mehr von Konsole zu Kamerabild wandern: es kann effizienter gearbeitet werden, ohne Informationssuche.

Weitere Plug-ins

Die nun folgenden Plug-ins sind deutlich einfacher strukturiert als das Kamera-Plug-in. Sie sind alle relativ ähnlich, was angesichts des Plug-in-Interface auch erwünscht ist. Diese Entwürfe sind für den Programmierer nur Ideengeber und grobe Anhaltspunkte. Was dabei überlegt aber werden soll, sind folgende Punkte:

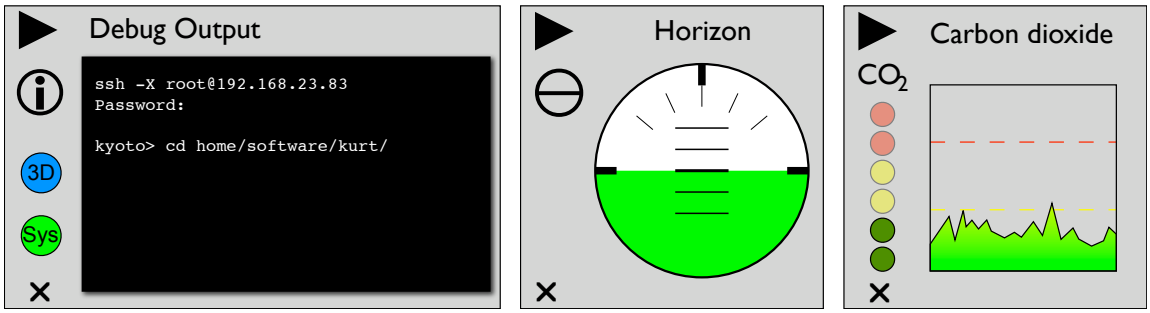
- Welche Funktion/Information ist die wichtigste? Diese sollte möglichst auch im minimierten Zustand zugänglich gemacht werden. Im maximierten Zustand sollte dann aber diese *nicht ein zweites Mal* angezeigt werden und ihren Platz auf der linken Seite behalten. Zudem sollte sie konsistent zu den übrigen Funktionen/Informationen im maximierten Zustand gestaltet sein. Falls es keine wichtigste Funktion/Information gibt, kann auch im minimierten Zustand nichts angezeigt werden.

- Auch wenn für die sinnvolle Anzeige ein minimierter Zustand ausreichend ist, wird das Plug-in eine maximierten Zustand haben. Dies wird vom Interface vorgegeben und sollte nicht überschrieben werden. Der maximierte Zustand ist für Warnungen und Notfälle von zentraler Wichtigkeit (siehe Abb. 5.18). Die durch die Maximierung entstehende Fläche kann man für Hinweise auf die Gründe oder gezielte Warnungshinweise nutzen, falls diese systemtechnisch möglich ist.
- Durch die Farbänderung im Warnungs-/Notfallzustand ist die Farbwahl von Schrift, Buttons und Icons besonders Acht zu geben: Eine warnende, gelbe Schrift kann auf gelben, warnendem Hintergrund nicht gelesen werden. Die Farben sind allgemein aus dem neutralen Bereich zu wählen, um die GUI homogener erscheinen zu lassen und um den wichtigen, markanten Farben mehr Geltung zu verleihen.

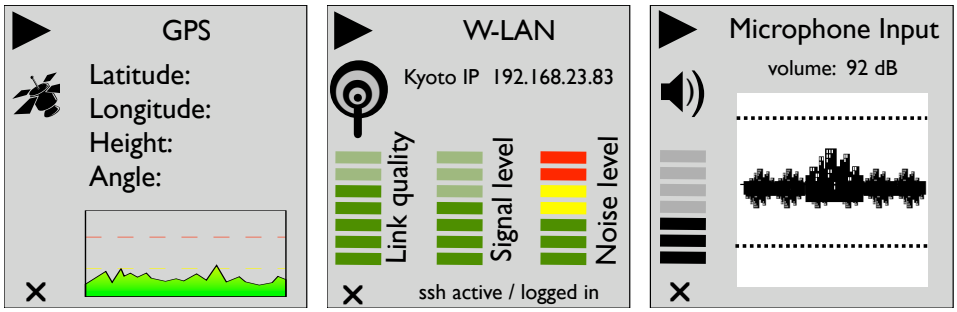
Die Konsole Die Konsole war in der alten GUI wichtig, um zu sehen, ob das System läuft, ob ein 3D-Scan gerade gemacht ist und wann er fertig ist. Ebenso konnte man ablesen, welche Kamera nun mit dem Cooliehead angesteuert wird, was nun vom Kamera-Plug-in übernommen wird. In der Konsole werden andauernd neue Systemdaten angezeigt. Im Normalbetrieb kann der Operator diese ignorieren, dennoch ist es ständige Bewegung im Blickfeld des Operators. Wie schon in Kapitel 4.2.1 (S. 48) angesprochen, zieht Bewegung Aufmerksamkeit auf sich. In Zukunft soll diese Konsole im maximierten Zustand nur noch als „Debug Output“ dienen, nicht aber den Operator im Wettkampf ablenken.
Anhand des Konsolen-Plug-ins (Abb. 5.17(a)) soll dem Operator mehr Information über den Zustand des System gegeben werden: Wenn ein 3D-Scan gemacht wird, was ja durchaus ein paar Sekunden dauert, zeigt der „3D“-Button dies an, indem er „aufleuchtet“. Ob das System läuft, kann der „Sys“-Button anzeigen, den man sich vielleicht wie die LED-Anzeige an einem Laufwerk vorstellen kann. Alles, was aus den Konsolen-Daten an Informationen extrahiert werden kann, sollte möglichst in eine andere, besser zugängliche Darstellungsweise umgesetzt werden. In wie weit das möglich und gewollt ist, liegt in den Händen des Operators. Er kann vielleicht mit dem rohen Systemoutput mehr als mit einem blinkenden Button anfangen.
Sinnvoll wären in diesem Plug-in auch kleine Skripte, die beispielsweise den Server neu starten o.ä. Jedes Skript könnte durch einen Klick auf einen Button aufgerufen werden.

Der künstliche Horizont Noch ist Anzeige des künstlichen Horizonts (Abb. 5.17(b)) sehr träge. Würde der Roboter zur Zeit in eine gefährliche Lage kommen, könne man dies schneller an den anderen Sensoren erkennen. Der Roboter würde also schneller umfallen, als dass sich das Plug-in maximieren würde. Vielleicht wird dieses Problem in Zukunft gelöst – oder dieses Plug-in wird nicht oft einen Platz im Modul zugewiesen bekommen.

Die Kohlendioxid-Anzeige Für dieses Plug-in (Abb. 5.17(c)) müssen erst sinnvolle Schwellenwerte gefunden werden, die den Warnungs-/Notfallzustand auslösen könnten.



(a) Die Konsole. (b) Der künstliche Horizont. (c) Die Kohlendioxid-Anzeige.



(d) Die GPS-Anzeige. (e) Die W-LAN-Anzeige. (f) Die Mikrofon-Eingabe.

Abbildung 5.17: Die Entwürfe verschiedenster Plug-ins.

Die GPS-Anzeige Für die Outdoor-Mission des Kurt3D ist dies sicherlich ein sehr wichtiges Plug-in (Abb. 5.17(d)), das speziell an die Ansprüche des Operators angeglichen werden muss. Eventuell ist eine grafische Darstellung besser zugänglich als die Daten.

Die W-LAN-Anzeige Die momentane W-LAN-Anzeige (Abb. 5.1, S. 59) bedient den Operator mit genauen Werten und einer Anzeige mit Statusbalken. Falls diese Daten für den Operator wichtige Informationen darstellen, sollten sich in diesem Plug-in (Abb. 5.17(e)) auch angegeben werden.

Die Mikrofon-Eingabe Erst, wenn sinnnvolle Algorithmen Störgeräusche, z.B. Publikumslärm, identifizieren können, kann auch ein Schwellenwert festgelegt werden, wann der Operator von diesem Plug-in (Abb. 5.17(f))gewarnt werden soll.

5.6 Die Eingabegeräte

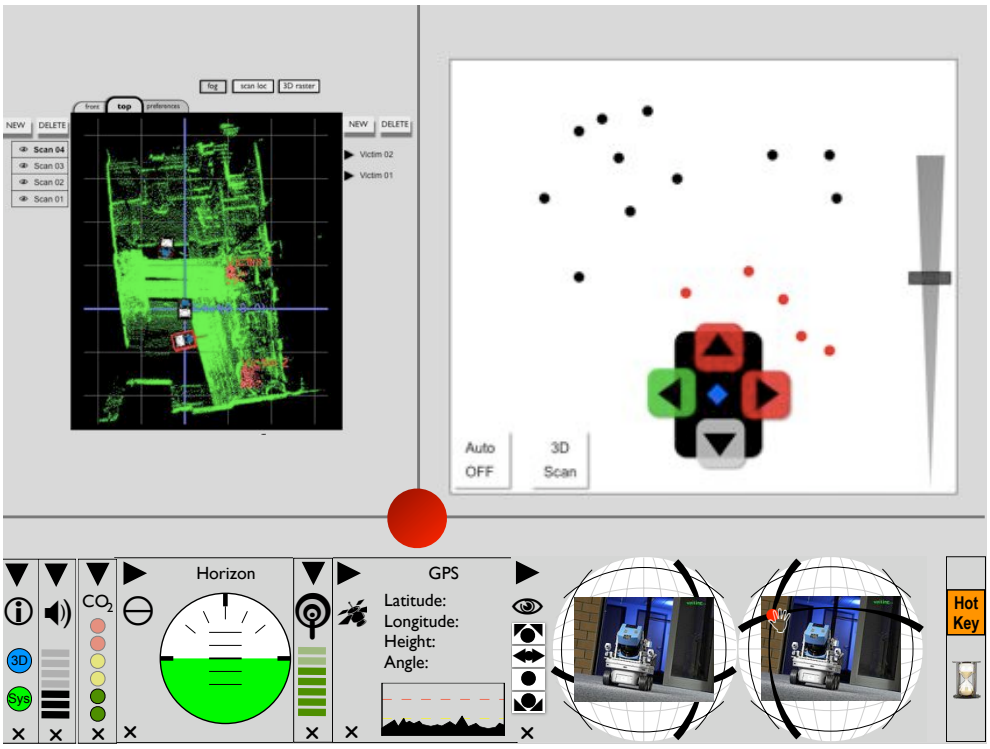
Nachdem nun die grafische Schnittstelle entworfen wurde, sollte noch kurz etwas bezüglich der Eingabegeräte ergänzt werden. Die Tastaturbelegung muss überdacht werden: Mit

der Benutzung der Maus im 3D-Modul werden viele Shortcuts wegfallen können. Ob sich ein Operator nun auf die Maus umstellen wird, oder nicht, kann hier nicht vorhergesagt werden. Jedenfalls ist es durchaus sinnvoll, für den „Notfall“ eine sinnvolle Belegung der Tasten zu haben. Was auf jeden Fall ergänzt werden sollte, ist die Navigation mit den Cursor-Tasten. Sie bieten sich durch ihre Anordnung auf der Tastatur schon als intuitive Steuerung an.
Der Joystick hat sich als sehr beliebt herausgestellt. Die Frage, ob er durch ein Gamepad ersetzt werden sollte, kann man an dieser Stelle nicht beantworten. Gerade aber die Kopplung zwischen Bewegungsrichtung des Joystick und der resultierenden Bewegung des Roboter, scheint den Joystick sinnvoll zu machen. Ein Gamepad mit einem kleinen Joystick („Jog“) würde diesen Vorteil auch haben. Der Nachteil des Joystick ist sicherlich die unpräzise Steuerung, die nicht unbedingt für enge Wettkampf-Passagen zu empfehlen ist.

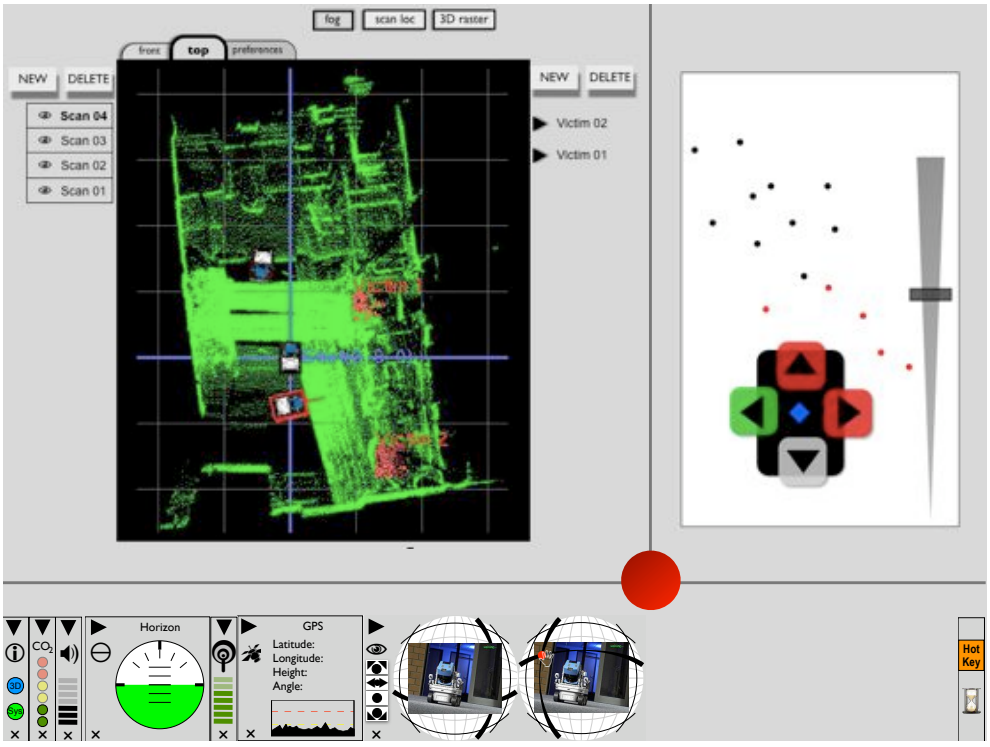
5.7 Mock-ups

Die Abbildungen 5.18(a) und 5.18(b) zeigen den gesamten Entwurf der GUI, wie er beschrieben wurde. Die obligatorische Menü-Leiste müsste oben noch ergänzt werden. Die technische Umsetzung wurde bei diesem Entwurf nicht berücksichtigt. So kann eventuell die Umsetzung des Draggers nicht 1:1 erfolgen, sondern nur vom Grundkonzept der Unterteilung her. Ein Dragger sollte dennoch nahe am Konzept umgesetzt werden, da er die Bedienung erheblich vereinfachen kann.

Die Abbildung 5.19 zeigt eine Einbettung in eine Umgebung, die mit dem Programm JFormDesigner [2]. Es wurde ein Hintergrund erzeugt, in den danach die Modul eingebettet wurden. Es handelt sich nicht um einen vollständig funktionierenden Prototyp, sondern nur um eine grafische Aufbereitung, um einen realistischeren Eindruck zu erwecken. Das Plug-in-Modul und das 2D-Modul wurden dabei farblich an den Standard-Hintergrund angeglichen.



(a) Entwurf mit Fokus auf dem 2D-Modul.



(b) Entwurf mit Fokus auf dem 3D-Modul.

Abbildung 5.18: Zusammenstellung der einzelnen Module.

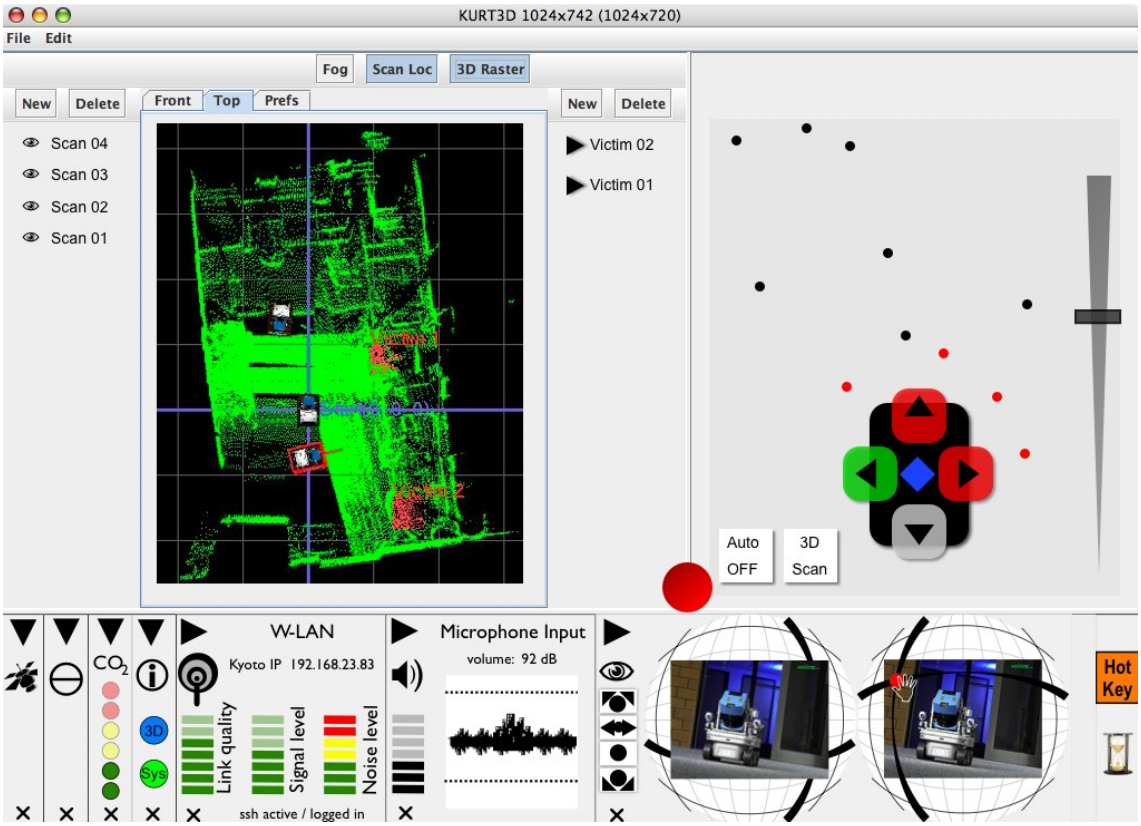


Abbildung 5.19: Eingebetteter, realisitischerer Entwurf der grafischen Schnittstelle.

Kapitel 6

Diskussion

6.1 Diskussion des Ergebnisses

In dieser Masterarbeit wurde ein Re-Design für die grafische Schnittstelle des RoboCup Rescue Teams Deutschland1 entwickelt. Die Ergebnisse der offenen Evaluation und Richtlinien aus angrenzenden Gebieten wurden zur Hilfe genommen. Es folgt nun eine kritische Betrachtung der einzelnen Schritte und Module.

6.1.1 Die Dynamik

Die Dynamik, die dieser GUI zugrunde gelegt wurde, kann anhand von Beobachtungen und Aussagen der Versuchspersonen begründet werden: Der Operator muss sich zwei verschiedenen Aufgaben stellen, dem Navigieren und dem Kartografieren. Solange das noch nicht in einem einzigen Anzeigefeld passieren kann, müssen diese beiden Bereiche sinnvoll getrennt werden. „Sinnvoll“ bezieht sich hier auf den zeitlichen Ablauf der Mission, denn der Operator macht nicht beides auf einmal, sondern immer nacheinander. Folglich wurde die Fokusmetapher (Kap. 4.2.1), die Rücksicht auf die menschliche Wahrnung nimmt, an diese GUI angepasst. Während des Kartografierens können die anderen Sensordaten durchaus in den Hintergrund, also den sekundären Fokus, rücken. In dieser GUI werden sie verkleinert dargestellt und mit entsprechenden Mechanismen ausgestattet, um die Aufmerksamkeit des Operators zur jeder Zeit auf sich ziehen zu können.
Die verschiedenen Modi, mit denen die GUI bewegt werden kann, sind also durch die Evaluationsergebnisse und die Fokusmetapher zu rechtfertigen. Wo aber konnte in dem Re-Design den Richtlinien nicht gefolgt werden? Die Richtlinien sind oft widersprüchlich, wie ein Beispiel im Kapitel 6.1.4 zeigt.
Durch die Dynamik haben die Informationen und Funktionen keinen „Stammplatz“ mehr. Der Operator hat zwar eine Ahnung, wo welche Information angezeigt wird, weiß aber nie sicher, wo genau sie sich befindet. Sind die Plug-ins kleiner skaliert, so rücken die

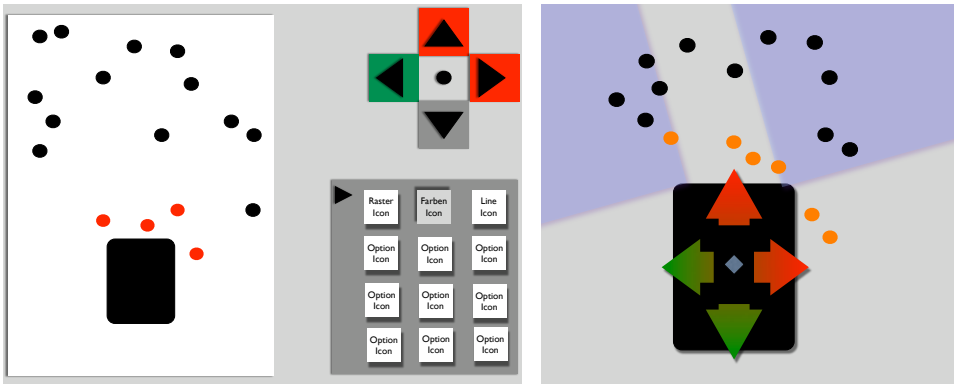
Informationen und Funktionen zusammen. Das ist leider ein schwer zu vermeidender Nebeneffekt, der in Kauf genommen wurde. Der Vorteil der Dynamik und des Fokussieren erschien wichtiger. Der entstehende Nachteil kann durch die verschiedenen Darstellungsweisen (Warnung/Notfall, siehe Abb. 5.15(b)) ein wenig ausgeglichen werden. Macht kein Plug-in auf sich aufmerksam, so ist kein Wert in einem kritischen Bereich. Sicherlich ist dabei die zugrunde liegende Programmierung jedes einzelnen Plug-ins essentiell: Dort kann der Programmierer einem Plug-in seinen „Sinn“ implementieren. Zusammenfassend kann über die Dyamik gesagt werden, dass sie notwendig ist und die dadurch entstehenden Nachteile möglichst gut und unauffällig absorbiert wurden.

6.1.2 Die Module

Der Aufbau der GUI anhand von Modulen ergibt sich im gewissen Maße durch die Dynamik. Dennoch ist ein wichtiger Punkt die Modularität: Ein Modul kann auch für sich alleine stehen – bzw. auch alleine in der GUI angezeigt werden (Kap. 5.3.2). Es soll nicht von einem anderen Modul abhänig sein, weder in der Anzeige als auch in der Informationsverarbeitung. Sicherlich bringt auch dies Nachteile mit sich: Beispielsweise könnte man die Ausrichtung der Kameras auch im 2D-Modul darstellen. Ebenso kann man dann noch die Steuerung der Kameras einlagern, wie es auch in Abb. 6.1(b) probiert wurde. Dennoch wird empfohlen, dass, solange sich die GUI noch in der Entwicklung befindet, die Module auch modular gehalten werden. Innerhalb eines Moduls könnten unter Vorbehalt Abhängigkeiten auftauchen. Es wäre zum Beispiel auch ein Zusammenspiel der Plug-ins denkbar: Falls ein kritischer Wert in einem Plug-in registriert würde, könnte auch ein anderes, situativ abhängiges Plug-in mit aktiviert werden. Man denke vielleicht an den Zusammenhang von Kohlendioxid und Haut-/Bewegungserkennung etc.
Eine wichtige Neuerung scheint die Menüleiste zu sein, die in Applikationen eigentlich obligatorisch ist. In ihr kann der Operator die wichtigen „undo-/redo“-Funktionen, für die GUI allgemein wirksame und modulspezifische Funktionen finden. Zudem kann er auch hier Skripte unterbringen und die Applikation beenden. Die Menüleiste ist für das Gesamtbild und den Zusammenhalt der GUI essentiell. Sie bildet also eine Metaebene zwischen den Modulen und dem System.

6.1.3 Der schmale Grat der Fusion

Die vor allem von der NIST empfohlene Datenfusion ist durchaus kritisch, wie schon in Kap. 5.4 kurz angesprochen wurde: Fusion kann auch zur kognitiven Last werden. Hier, bezüglich dieser GUI, muss betont werden, dass nicht nur Daten, sondern auch Funktionen verschmolzen werden sollten. Folglich wurde die Richtlinie der NIST noch durch die Funktionalität erweitert. Sicherlich ist es auch eine Frage des funktionalen Designs. Aber sowohl das Vorstellungsvermögen des Designers, als auch die menschliche Wahrnehmung des Nutzers sind limitiert. Darstellung hat bekanntlich seine Grenzen. Die Abbildungen 6.1 sollen den Prozess am 2D-Modul illustrieren. Den ersten Entwurf des Moduls zeigt



(a) Die Maussteuerung liegt weit außerhalb. Ein extra angezeigtes Menü gewährt Zugriff auf Befehle, die eigentlich fast nie benutzt werden. Zudem fehlt noch der Geschwindigkeitsregler.

(b) Die Ausrichtung der Kameras wird hier durch einen dunklen Schatten nur in der Ebene angezeigt: Es fehlt die räumliche Ausrichtung, die Steuerung der Kameras und der Geschwindigkeitsregler.

Abbildung 6.1: Frühe Entwürfe des 2D-Moduls.

Abb. 6.1(a). Die Maussteuerung ist noch ausgelagert: Die Fusion von dem bewegten Objekt und dessen Steuerung drängt sich auf. Ebenso ist das Menü an der rechten Seite zwar schon als minimierbar angedacht, dennoch ist es fraglich, ob es Sinn macht, Funktionen so offensichtlich anzubieten, die nur sehr selten in dem Versuch benutzt wurden. Der nächste Schritt, der in Abb. 6.1(b) gezeigt wird, war ein Entwurf, der schon wieder über das Ziel hinausschießt. Die Ausrichtung der Kameras in der Ebene ist relativ einfach darzustellen. Nachteil ist, dass die Scanpunkte dabei nicht mehr so klar erkennbar sind. Schafft man es dann noch, die räumliche Ausrichtung der Kameras durch einen Farbverlauf in Dreiecken darzustellen, so kann dies sicher praktisch sein. Dennoch ist nun der Nachteil, dass man gar keine Scanpunkte mehr in dem Ausrichtungsbereich erkennen kann. Der Operator bekommt zwei schlecht zu separierende Informationen. Denkt man sich nun noch eine praktische, räumliche Steuerung dazu, so liegt es eigentlich wieder nahe, das zu steuernde Objekt mit seiner Steuerung (wie oben) zusammen darzustellen. Sollte man dann eine Lösung zwischen Funktionalität, funktionaler Eleganz und Minimalismus gefunden haben, dann erkennt man, dass die Kamerabilder selbst noch gar keinen Platz haben.
Die Endfassung des Entwurfs (S. 76) geht also einen Mittelweg zwischen kognitiver Entlastung und Fusion.

6.1.4 Die Entwürfe der Module

In jedem Modul stecken Überlegungen, wie man die angedachte Struktur und Dynamik möglichst funktional und intuitiv umsetzen kann. Dabei unterscheidet sich jedes Modul sehr stark von den anderen.

Das 3D-Modul

Durch die Evaluation wurde ersichtlich, dass es die größte Schwierigkeit für den Operator ist, mit den 3D-Scans umzugehen. Sicherlich ist dies kein sonderlich überraschendes Ergebnis. Dennoch wurde klarer, wieso es sich schwieriger darstellt, als es sein müsste: Kritische Punkte sind hier die verschiedenen Ansichten der Scans und die Tastaturbedienung. Ersteres hätte vielleicht gelöst werden können, indem man dem Operator zwei Ansichten auf einmal anbietet. Die Darstellungsfläche hätte also auch zweigeteilt werden können, damit man in beiden Fenstern gleichzeitig Rotationen und Translationen beobachten kann. Dennoch würden bei diesem System die Bewegungen nur noch halb so schnell berechnet und dargestellt werden können, da zwei Ansichten auf einmal gerendert werden müssten. Da gerade dies ein wunder Punkt in der 3D-Darstellung ist, wurde diese Aufteilung verworfen. Parallele und perspektivische Projektionen sollen dem Operator helfen, ein besseres Verständnis für die räumlichen Verhältnisse zu entwickeln. Beispielsweise sind Grössenverhältnisse besser in Parallelprojektionen ersichtlich.

Ebenso ist das Markieren von Objekten und Opfern deutlich einfacher, wenn man einen schnellen Zugriff (anhand von Tabs/Reitern) auf gewohnte Ansichten hat. Der zweite Punkt, die Tastaturbedienung, wurde duch die Mausbedienung eleminiert: Dadurch, dass die Box mit der Maus anfassbar/skalierbar wird, fallen auch die Tastaturkürzel dafür weg. Die Darstellung der Rotations-/Translationsmöglichkeiten mittels eines Koordinatenkreuzes, wie es in 3D-Programmen üblich ist, sollte die Arbeit zudem noch weiter erleichtern. Je mehr Funktionen für den Operator sichtbar gemacht werden, desto „optisch beladener“ wird das Modul. Hier musste die Minimalismus-Heuristik von Nielsen (S. 51) vernachlässigt werden, da dieser ansonsten weitere Heuristiken zum Opfer gefallen wären (beispielsweise „Wiedererkennung anstelle von Erinnern“).

Das 2D-Modul

Die Überlegungen beim Entwurfsprozess für dieses Modul wurden schon in Kapitel 6.1.3 dargestellt. Durch die optische Leichtigkeit dieses Moduls wirkt das 3D-Modul noch „schwerer“. Aber auch hier steht wieder die Funktionalität, und (noch) nicht die Ästhetik im Vordergrund. Donald Norman vertritt die Meinung, dass attraktives Design Dinge besser funktionieren lässt [42]. Wenn die Gestaltung der Funktion deutlich besser gerecht wird, und die Funktionen auch eine bessere Gestaltung zulassen, könnte auf eines ansprechendes/emotionales Design mehr geachtet werden.

Das Plug-in-Modul

Der Vorteil dieses Moduls liegt in der Austauschbarkeit der Plug-ins. Dennoch gibt es einen Nachteil: Es sind nicht beliebig viele Plug-ins auf einmal darstellbar. Dies ist aber kein Problem, das GUI-spezifisch ist. Folglich muss der Operator seine Plug-ins gezielt auswählen und immer wieder aussortieren. Betrachtet man das Konzept dieses Moduls

kritisch, so bemerkt man eine starke Abhängigkeit von der Art, wie der Operator es organisiert. Ist es mit allen möglichen, eigentlich nicht notwenigen Plug-ins überladen, so ist zwar das Konzept dahinter ausgereizt worden, dennoch erzielt es nicht die angedachte Wirkung: essentielle Informationen im Blickfeld zu haben.

6.1.5 Kompromisse

Dieses Re-Design musste, wie gerade ausgeführt, Kompromisse eingehen: Man kann nicht allen Richtlinien folgen, wenn beispielsweise die dafür notwendigen Algorithmen noch nicht existieren. Ideen, sowohl für Funktionen sowie für Gestaltungsmöglichkeiten, sind reichlich vorhanden.
Für diese Arbeit galt es, mit der Zeit zu gehen und ein passendes Re-Design für die Gegenwart und die nahe Zukunft zu entwerfen. Durch den modularen Aufbau wird der GUI diese Nachhaltigkeit verliehen: Man kann Plug-ins neu schreiben, erweitern und austauschen. Das Gleiche gilt für die Module. Die GUI wird nicht zerstört, wenn man etwas herausnimmt. Zudem ist sie durch den Dragger sehr dynamisch. Dem Operator soll Last abgenommen werden, aber er soll sich nicht bevormundet vorkommen. Deshalb wurde entschieden, verschiedene Modi in der Dynamik der Module zu entwerfen. Das, wie auch viele andere kleinere Features, liegen in der Hand des Operators.

Die offene Evaluation

Gerade während der offenen Evaluation wurde deutlich, dass es sehr verschiedene Vorstellungen von einem guten System und einer guten GUI gibt. Man hätte alles auf das „kleinste gemeinsame Vielfache“ reduzieren können, oder aber, wie hier geschehen, der Vielfalt der Experten durch Wahlmöglichkeiten gerecht werden. Vielleicht zahlt sich hier der offene Blick einer offenen Evaluation aus. Im Nachhinein kann man nicht die Frage beantworten, ob eine präzise, empirische Evaluation ein „besseres“ Ergebnis geliefert hätte. Sicherlich wäre es aber ein anderes Ergebnis gewesen. Kritisch betrachtet hat dieses intuitive Vorgehen den Nachteil, dass eine nun folgende Evaluation nicht direkt mit dieser offenen Evaluation verglichen werden kann. Eine methodisch korrekte Evaluation hätte also vielleicht eine gute Basis für weitere Auswertungen geliefert, aber eventuell auch den Blick eingeengt.

6.2 Ausblick: Was die Zukunft bringen kann

Nachdem nun die einzelnen Facetten und Entscheidungen kritisch betrachtet wurden, ist es an der Zeit, weiterzudenken. Der naheliegenste Schritt ist die Programmierung dieses Re-Designs.
Das Erahnen der neuen Algorithmen und Ideen, die in nächster Zeit wahrscheinlich implementiert werden, wird einige Türen im Design öffnen können. Eine der wichtigsten Neuerungen kann sein, dass die 2D-Scandaten in den die 3D-Information mit eingelagert

werden. Dies würde zu einer notwendigen Datenfusion führen. Für die GUI kann das bedeuten, dass es anstelle des 3D- und 2D-Moduls nur noch eines geben wird. In diesem neuen Modul wäre dann Navigation und Kartierung möglich. Damit fiele auch das mentale Rotieren durch den Operators weg, da er dann nicht mehr die 2D-Datenpunkte aus der Parallelprojektion (von oben) in die perspektivische Projektion der 3D-Darstellungsfläche umwandeln müsste. Wenn es also technisch gelänge, den Roboter-Avatar, wie in einem Egoshooter, durch die Räume zu fahren, wäre das sicherlich ein großer Fortschritt.
Das manuelle Einlagern des Scans ist immer noch eine zeitraubende Arbeit für den Operator. Das Ziel ist es, dass das System einen neuen 3D-Scan von alleine einlagern (registrieren) kann und eine perfekte 3D-Karte baut. Mit großer Wahrscheinlichkeit wird die Zukunft diese Erleichterung für den Operator bereit halten.
Momentan wird Kurt3D für den Outdoor-Gebrauch ausgestattet. Im Zuge des entsprechenden Studienprojekts werden sicherlich einige Neuerungen Einzug in das System erhalten. Die Einbindung dieser Erweiterungen wird hoffentlich durch die Modularität, speziell durch das Plug-in-Modul, deutlich erleichtert werden.

Anhang A

Shortcuts

Die Tastatur-Shortcuts

- Veränderung der Ansicht im 3D Viewer:

  'up' / 'down' rotation around the x-axis
  'left' / 'right' rotation around the y-axis
  '7' / '1' rotation around the z-axis
  '9' / '3' translation along the z- axis
  '8' / '2' translation along the y- axis
  '4' / '6' translation along the x- axis
  F1 switch to orthogonal top view an back saves everything
  F2 change view about 90 deg
  F11 delete last 3D scan
  'N' traverse 3D scan positions
  'n' show next scan (in combination with all points mode)
  '<' and '>': enlarge / decrease opening angle

- Filter für den 3D Viewer

  'p' switches point mode between none/reduced/all (single scan)/ all (all scans)/ octree/octree with remision values
  'L' show lightning
  'l' switches line mode between none/reduced/all
  's' switches surface mode between none/reduced/all/as polygons/as quadtree
  'o' switches object mode between none/reduced/all
  'F' switches fog mode between none/fog-type1/fog-type 2/...
  'i' show inverted 3D scan

- Shortcuts für das Einlagern („matching“) eines Scans

  'm' show scan matching

CTRL+ALT+'m' move last scan manually without rematch
ALT+'m' move last scan and rematch afterwards
'M' animate scan matching and save as mgif
F4 unmatch last 3D scan

- Shortcuts für das Markieren von Opfern (anhand einer „box“)

'b' enters box mode/toggles volume box
ALT+'8' moves box up (y-direction)
ALT+'2' moves box down (y-direction)
ALT+'4' moves box left (x-direction)
ALT+'6' moves box right (x-direction)
ALT+'9' moves box into depth (z-direction)
ALT+'3' moves box from depth (z-direction)
ALT+'+' enlarges box (x-direction)
ALT+'-' scaling down box up (x-direction)
ALT+'/' enlarges box (y-direction)
ALT+'*' scaling down box up (y-direction)
ALT+'0' enlarges box (z-direction)
ALT+',' scaling down box up (z-direction)
ALT+'d' saves selected data and removes it

- Shortcuts für Animationen und Bilder

'a' plays animation
'A' saves animation as mpeg
'G' saves animation as mgif
'S' saves current view as PPM
'J' saves current view as JPEG
'K' animate Pipe matching

- Sonstige Shortcuts

F3 saves all scans binary
F8 execute finalize Scans
'r' reset position
'R' reset position and VCR
'O' switches projection mode between orthogonal/projective
'y' toggles show of trajectory
'Y' toggles show of the robot's positions at 3D scan
'x' animates robots trajectory
't' toggles textures
'T' toggles tagging mode
„space“ tag single position

'd' enters object detection mode
'I' localize detected objects using icp
'z' toggles ground plane
'f' toggles fill mode
'P' show Pipe
'q' toggles wireframe
'W' toggles wireframe and widers opening angle, depth values are drawn
'w' toggles wireframe and widers opening angle, remission values are drawn

Anhang B

3D Funktionsleiste

Die 3D-Funktionsleiste - Die einzelnen Funktionen

Hier folgt nun eine sehr kurze Beschreibung der verschiedenen auswählbaren Filter/Algorithmen in ihrer normalen Reihenfolge:

- Menü „points“:

  none zeigt keine Punkte an
  red. points zeigt eine reduzierte Punktemenge an
  all points zeigt alle Punkte an
  all points (n) da mehrere 2D Scans übereinanderliegen können durch die Taste „n“ verschiedene Scans ausgesucht werden
  octree Octree Darstellung
  octree (rem) Octree Dastellung mit Remisionswerte
  edge points Darstellung der Kantenpunkte
  floor plan Dummie-Button für Funktionen (Debugging/Testen)

- Menü „lines“:

  none zeigt keine Linien an
  red. lines zeigt nur bestimmte (“reduzierte“) Linien an
  all lines zeigt alle Linien an, die in der scanslice gefunden wurden

- Menü „surface“:

  none zeigt keine Oberflächen an
  red. lines zeigt nur bestimmte Linien an
  all lines zeigt nur bestimmte Linien an
  object Versuch der Darstellung von Raumvolumen
  invert farbliche Invertierung der Darstellung
  lightning Setzen einer Lichtquelle (default rechts unten)
  fill mode zeigt den Umriss der Flächen

parallel projection bestimmte 3D auf 2D Projektion, Beeinflussung der Tiefendarstellung

trajectory zeigt die Orientierung des Roboters beim 3D Scan durch eine Linie an

3D scan locations blendet Roboteravatar und dessen Scanposen ein

tagging aktiviert mit Tasten; „Screenshots“ für Animation später

object detection zeigt Rechtecke um erkannte Objekte

– Menü „coord. system“:

none zeigt kein Koordinatensystem an

raster zeigt Raster; Taste F1 für Draufsicht; sinnvoll als Maßstab für eine grobe Übersicht

raster 3D zeigt dreidimensionales Raster an und die Opferlabels

axis zeigt nur die X- und Y- Hauptachsen

– Menü „mesh“:

wireframe verbindet Nachbarpunkte zu Drahtgitter

refl. mesh Drahtgitter mit Deflektionswerten

depth mesh Drahtgitter mit Tiefendaten

– Menü „fog“: („fogs“ sind nicht untereinander kombinierbar!)

no fog kein Nebel

fog black exp legt einen exponentiell wachsenden, schwarzen Nebel über die Darstellung

fog black exp2 legt einen quadratisch-exponentiell wachsenden, schwarzen Nebel über die Darstellung

fog black linear legt einen linear wachsenden, schwarzen Nebel über die Darstellung

fog white exp legt einen exponentiell wachsenden, weißen Nebel über die Darstellung

fog white exp2 legt einen quadratisch-exponentiell wachsenden, weißen Nebel über die Darstellung

fog white linear legt wiederum einen linear wachsenden, weißen Nebel über die Darstellung

fog density die Werte für die Nebeldichte kann durch Eingabe einer Zahl oder das Klicken der Hoch-/Runter-Pfeile verändert werden

– Menü „save & animate & print“: Dieses Menü beinhaltet bestimmte Funktionen, um Filme als im avi-Format abzuspeichern, etc.

reset position sollte den Scan wieder in die Ausgangsposition bringen

Quit schließt die Applikation

Bemerkung; Einige dieser Funktionen sind im Zuge von Praktikas und Workshops entstanden. Sie sind alle *Display-Funktionselemente*, manche aber funktionieren nicht.

Literaturverzeichnis

[1] Heuristics for User Interface Design, http://www.useit.com/papers/heuristic/heuristic_list.html.

[2] JFormDesigner - Java/Swing GUI Designer, http://www.jformdesigner.com/.

[3] Knowledge-based Systems, http://www.inf.uos.de/kbs/.

[4] KommDesign.de - Texte, Ergonomie/Usability, http://www.kommdesign.de/texte/din.htm.

[5] Mental Commitment Robot [PARO], paro.jp/english/index.html.

[6] National Institute of Standards and Technology, http://www.nist.gov.

[7] NEC Personal Robot Research Center, http://www.incx.nec.co.jp/robot/.

[8] pinc - platform for intercultural communication, http://www.go2pinc.net.

[9] Project IsoMetrics - Homepage, http://www.isometrics.uni-osnabrueck.de.

[10] RoboCup 2004 Official Site, http://www.robocup2004.pt/.

[11] RoboCup 2005 Osaka, Japan, http://www.robocup2005.org/.

[12] RoboCup-Rescue Official Web Page, http://www.rescuesystem.org/robocuprescue/.

[13] Robotersysteme - Mensch-Roboter-Kooperation und Assistenz, http://www.ipa.fraunhofer.de/Arbeitsgebiete/robotersysteme/assistenz/.

[14] VUG Home Page, http://www.itl.nist.gov/iad/vvrg/.

[15] Persönliche Kommunikation mit Dr. S. Müller, Neurologie, Südklinikum Nürnberg, März 2006.

[16] J. A. Adams. Critical Considerations for Human-Robot Interface Development. In *AAAI Fall Symposium on Human-Robot Interaction*, 2002.

[17] J. A. Adams. Human-Robot Interaction Design: Understanding User Needs and Requirements. In *Proceedings of the 2005 Human Factors and Ergonomics Society 49th Annual Meeting*, 2005.

[18] J. A. Adams and R. P. Paul. Human Management of a Hierarchical Control System for Multiple Mobile Agents. In *Proceedings of 1994 IEEE International Conference on Systems, Man and Cybernetics*, 1994.

[19] J. Adolf and T. Holden. *Human Research Facility (HRF) Human-Computer Interface (HCI) Design Guide.* National Aeronautics and Space Administration, 1997.

[20] Inc. Apple Computer. Apple Human Interface Guidelines, 2005.

[21] M. Baker, R. Casey, B. Keyes, and H. A. Yanco. Improved Interfaces for Human-Robot Interaction in Urban Search and Rescue. In *Proceedings of the IEEE Conference on Systems, Man and Cybernetics*, 2004.

[22] G. Cockton, D. Lavery, and A. Woolrych. Inspection-based Evaluations. In J. A. Jacko and A. Sears, editors, *The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies and Emerging Applications.* Lawrence Erlbaum Associates, Inc., 2003.

[23] K. Cox and D. Walker. *User Interface Design.* Prentice Hall, 2nd edition, 1993.

[24] J. Drury, J. Scholtz, and H. Yanco. Awareness in Human-Robot Interactions. In *Proceedings of the IEEE Conference on Systems, Man and Cybernetics*, 2003.

[25] J. L. Drury, D. Hestand, H. A. Yanco, and J. Scholtz. Design Guidelines for Improved Human-Robot Interaction. In *Proceedings of CHI2004*, 2004.

[26] J. L. Drury, J. Scholtz, and H. A. Yanco. Applying CSCW and HCI Techniques to Human-Robot Interaction. In *Proceedings of the CHI 2004 Workshop on Shaping Human-Robot Interaction*, 2004.

[27] J. L. Drury, H. A. Yanco, and J. Scholtz. Using Competitions to Study Human-Robot Interaction in Urban Search and Rescue. *ACM CHI Interactions*, pages p. 39–41, 03/04 2005.

[28] J. Dumas. User-based Evaluations. In J. A. Jacko and A. Sears, editors, *The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies and Emerging Applications.* Lawrence Erlbaum Associates, Inc., 2003.

[29] T. W. Fong, I. Nourbakhsh, R. Ambrose, F. Heckel, R. Simmons, A. Schultz, and J. Scholtz. The Peer-to-Peer Human-Robot Interaction Project. In *AIAA Space 2005*, 2005.

[30] T. W. Fong, C. Thorpe, and C. Baur. Advanced Interfaces for Vehicle Teleoperation: Collaborative Control, Sensor Fusion Displays, and Remote Driving Tools. *Autonomous Robot*, 11(1):77–85, 2001.

[31] T. W. Fong, C. Thorpe, and C. Baur. Robot as Partner: Vehicle Teleoperation with Collaborative Control. In *Multi-Robot Systems: From Swarms to Intelligent Automata.* Kluwer, 2002.

[32] T.W. Fong, C. Thorpe, and C. Baur. Robot, Asker of Questions. *Robotics and Autonomous Systems*, 42, 2003.

[33] Kira Hagge. *Informations-Design.* Number 40 in Konsum und Verhalten. Physica-Verlag Heidelberg, 1994.

[34] E. Halberstam, L. Navarro, R. Conescu, S. Mau, G. Podnar, A. D. Guisewite, H. B. Brown, A. Elfes, J. Dolan, and M. Bergerman. A Robot Supervision Architecture

for Safe and Efficient Space Exploration and Operation. In *Tenth Biennial International Conference on Engineering, Construction, and Operations in Challenging Environments: Earth Space 2006 Conference*, 2006.

[35] J. Hertzberg, K. Lingemann, and A. Nüchter. USARSIM – Game-Engines in der Robotik-Lehre. In A. B. Cremers et al., editor, *Informatik 2005 – Informatik LIVE*, volume 1, pages 158–162. Gesellschaft für Informatik, 2005.

[36] H. Kromrey. Evaluation – ein vielschichtiges Konzept. *Sozialwissenschaften und Berufspraxis*, 2:105–132, 2001.

[37] S. Laqua. User-Interface-Design einer CSCL-Plattform zur interkulturellen Kommunikation. Master's thesis, Technische Universität Dresden, 2003.

[38] S. Laqua and P. Brna. The Focus-Metaphor Approach: A Novel Concept for the Design of Adaptive and User-Centric Interfaces. In *Interact*, 2005.

[39] B. A. Maxwell, N. Ward, and F. Heckel. A Human-Robot Interface for Urban Search and Rescue. In *Proc. of AAAI Mobile Robot Competition and Exhibition Workshop*, 2003.

[40] B. A. Maxwell, N. Ward, and F. Heckel. A Configurable Interface and Architecture for Robot Rescue. In *AAAI Mobile Robotics Workshop*, 2004.

[41] B. A. Maxwell, N. Ward, and F. Heckel. Game-Based Design of Human-Robot Interfaces for Urban Search and Rescue. In *Proceedings of CHI 2004*, 2004.

[42] D. A. Norman. *Emotional Design: Why we love (or hate) everyday things.* Basic Books, 2004.

[43] R. C. O'Reilly and Y. Munakata. *Computational Explorations in Cognitive Neuroscience. Understanding the Mind by Simulatin the Brain.* A Bradford Book, 2000.

[44] A. R. Pritchett. Human-Computer Interaction in Aerospace. In J. A. Jacko and A. Sears, editors, *The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies and Emerging Applications*. Lawrence Erlbaum Associates, Inc., 2003.

[45] B. Reeves and C. Nass. *The Media Equation: how people treat computers, television, and new media like real people and places.* Cambride University Press, 1996.

[46] J. Scholtz. Evaluation Methods for Human-System Performance of Intelligent Systems. In *PerMIS (Workshop)*, 2002.

[47] J. Scholtz. Usability Evaluation, 2004.

[48] J. Scholtz, B. Antonishek, and J. Young. Evaluation of Human-Robot Interaction in the NIST Reference Search and Rescue Test Arenas. In *Proceeding of PerMIS*, 2004.

[49] J. Scholtz, J. Young, H. A. Yanco, and J. L. Drury. Evaluation of Human-Robot Interaction Awareness in Search and Rescue. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2004.

[50] N. Shedroff. Information Interaction Design: A Unified Field Theory of Design. In R. Jacobson, editor, *Information Design*, pages 267–293. MIT Press, 1999.

[51] T. Stewart and D. Travis. Guidelines, Standards, and Style Guides. In J. A. Jacko and A. Sears, editors, *The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies and Emerging Applications*. Lawrence Erlbaum Associates, Inc., 2003.

[52] S. Watzmann. Visual Design Principles for Usable Interfaces. In *The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies and Emerging ApplicationsHuman-Computer Interaction Handbook:*. Lawrence Erlbaum Associates, Inc., 2003.

Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

Osnabrück, 30. Mai 2006