Hochschule Reutlingen
Reutlingen University

Dokumentation

Bildverarbeitung

Gesture Detection for SCITOS, 3D-Punktwolke,

Daniel Pohlmeyer, Sina Turville

1 Inhalt

1.1 Ausgangspunkt

Zur Gestenerkennung ist bereits die Software Kinetic Space im Einsatz. Hierbei wurden bereits mehrere Gesten eingelernt, die über die Kinect aufgenommen und verarbeitet werden. Je nach Übereinstimmung zwischen der Geste, die die von der Software erkannte Person ausführt und der eingelernten Geste, wird die Erkennung durch einen grünen bzw. roten Balken symbolisiert. Nachdem eine Person erkannt wird, wandelt sich das Farbbild in einen schwarzen Hintergrund, auf dem die Person skelettiert dargestellt wird. Skelettierung bedeutet, dass ein Objekt auf Linien reduziert wird. Die Person wird somit umgangssprachlich in ein Strichmännchen vereinfacht verwandelt. Ziel dieser Gestenerkennung ist es nun, dass der Roboter Leonie über Gesten Fahrbefehle erhält. Momentan wird über UDP-Kommunikation diese Steuerung auch bereits realisiert. Allerdings hat man sich aus Sicherheitsgründen noch auf die Bewegung des Kopfes beschränkt. Die Software für den Roboter ist SCITOS und läuft über das sogenannte MIRA-Center von Leonie.

1.1.1 Weiterführende Arbeit und Ziel dieses Projekts

Aufbauend auf den bereits vorhandenen Gegebenheiten ist es nun Ziel, anstelle der Kopfdrehungen Leonie durch Gesten in Bewegung zu setzen, sowie zu stoppen. Durch bestimmte Gesten, die zuvor eingelernt wurden, soll ein Fahrbefehl in Richtung der Person ausgelöst werden. Durch eine andere, deutlich erkennbare Geste, soll Leonie gestoppt werden können.
In der Theorie wird außerdem die Funktion der Microsoft (MS) Kinect untersucht. Dabei liegt der Fokus auf Triangulation und der Objekterkennung mit Hilfe einer 3D-Punktwolke.Gegebenenfalls soll noch ein Vergleich zur MS Kinect 2 herangezogen werden, welche mit der Technik Time of Flight (TOF) arbeitet.

1.1.2 Vorgehensweise

Nachdem wir uns mit der Software und Leonie vertraut gemacht haben, haben wir den Code live mit Leonie über das MIRA-Center getestet. Bis jetzt reagiert Leonie auf erkannte Gesten mit Kopfdrehen. Dabei wird über das Terminal, sowie über die Software KineticSpace ebenfalls angezeigt, ob eine Geste erkannt wurde und ob eine Kommunikation zwischen Client(PC) und Server (MIRA-Center auf Leonie) stattfindet. Da bis dahin nur bekannt war, dass Leonie mit einer Kopfdrehung antwortet, wenn eine Geste erkannt wurde, musste zunächst bestimmt werden, ob die Drehung auch richtungsabhängig mit Gesten verknüpft ist. Ob bei der Geste "right arm" Leonie auch mit einer Drehung nach rechts reagiert oder ob sie unabhängig der Gesten immer abwechselnd mit einer Rechts- und Linksdrehung antwortet, war hierbei die Fragestellung. Das musste durch schlichtes Ausprobieren getestet werden, da dies vorerst nicht aus dem Code beantwortet werden konnte.

1.2 Hindernisse

Durch die Dokumentation der Vorgänger konnte die bisherige Umsetzung überprüft und in Betrieb genommen werden. Jedoch stimmte die Ausgabe am Terminal nicht mit dem verfügbaren Sourcecode überein. Das erschwerte nachzuvollziehen, wie die Gesten erkannt wurden, ob Gesten konkret erkannt wurden oder nur allgemein, sowie wo und wie die Kopfdrehung implementiert war. Insbesondere was genau hinter dem Befehl *MoveHeadLeftRight* zur Kopfdrehung, der auch manuell im MIRA-Center wie in Abbildung 1 zu sehen ist, aufgerufen werden kann, konnte zunächst durch eigene Recherche nicht geklärt werden.

Hier Bewegungsbefehle über
\robot\Robot.*ServiceName(Übergabeparameter)* aufrufen

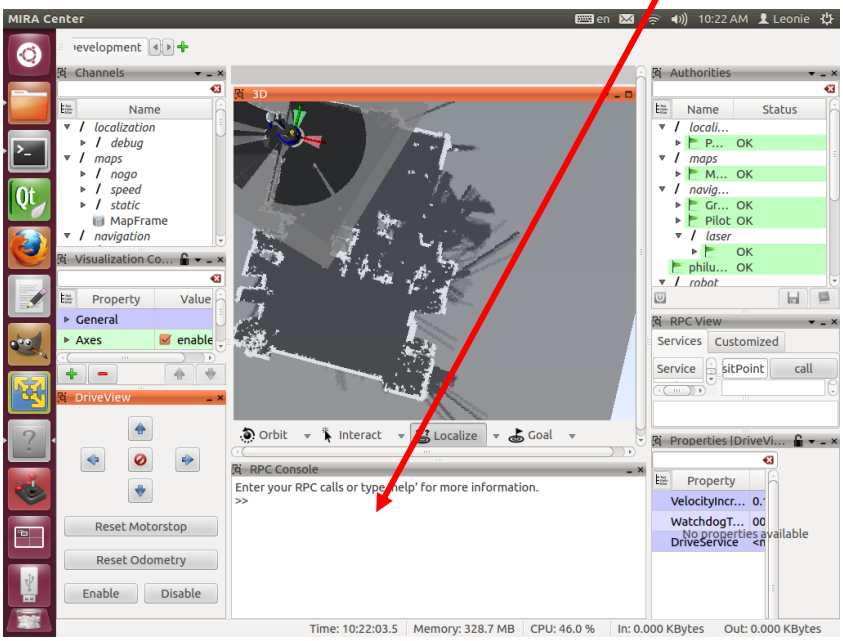

Abbildung 1 MIRA-Center

Allgemein war es teilweise problematisch, Dateien auf Leonie zu lokalisieren. Da mehrere Parteien verschiedene Projekte an ihr durchführten, konnte nicht immer stetig an unserem Projekt gearbeitet werden oder es gab Probleme beim Starten des Programms, sowie der Funktionstüchtigkeit. Wir hatten dadurch mit immer wieder denselben Problemen zu kämpfen.

1.3 Microsoft Kinect[1]

Die MS Kinect verfügt über eine Farbkamera, sowie über zwei 3D-Tiefensensoren, wie in Abbildung 2 dargestellt, auf Infrarotbasis von dem israelischen Hersteller PrimeSense, der mittlerweile von Apple übernommen wurde.

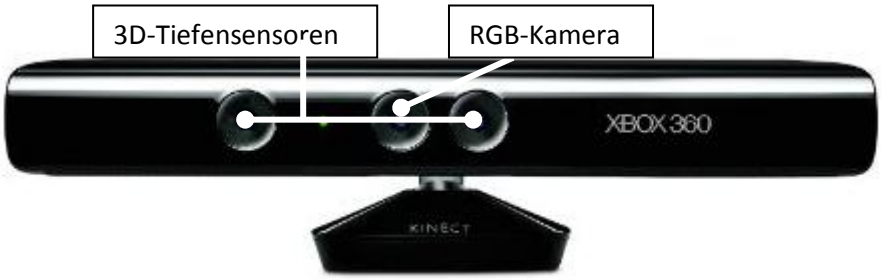

Abbildung 2 xBox Kinect von Microsoft

Der Tiefensensor sendet auf der linken Seite einen flächendeckenden Infrarotstrahl aus, dessen Reflektionen durch Objekte vom zweiten Sensor auf der rechten Seite empfangen werden. Die Infrarotstrahlen bilden eine für das menschliche Auge unsichtbare Punktewolke. Diese kann aber durch eine Digitalkamera, wie in Abbildung 3, visualisiert werden. Der dort enthaltene CMOS-Sensor ist empfindlich gegenüber Infrarot.
Die Farbkamera ist fähig, Farben mit einer Tiefe von 32Bit aufzunehmen und hat eine Bildrate von 30fps. Jeder Punkt besitzt xyz-Koordinaten, welche im ASCII-Format gespeichert werden. Das 3D-Bild kann anschließend durch Triangulation bestimmt bzw. errechnet werden. Aus dieser Punktewolke kann dann über Meshing (=Oberflächenrekonstruktion) Objekte identifiziert werden.

[1] (Hinrichs, Torge, 2014)
(www.xida.de, 2015)

Abbildung 3 Infrarotpunkte von MS Kinect ausgestrahlt
http://www.blogcdn.com/www.engadget.com/media/2010/11/101108-nightvision-01.jpg

Durch die Kamera und die Tiefensensoren werden Bewegungen links und rechts in einem Winkel von je 57 Grad, sowie nach oben und unten je 43 Grad erfasst. Damit ein Objekt erkannt wird, sollte es sich in einem Abstand zwischen 1,2 und 3,5m befinden.

1.3.1 RGB-Kamera

Die Farbkamera arbeitet mit einer Auflösung von 640x480 Pixeln und hat einen Bayer-Farbfilter.

1.3.2 Tiefensensor

Der Tiefeninformationen werden in der Kinect über das Verfahren Light Coding verarbeitet. Hierbei wird eine zufällige Punktewolke erzeugt, welche bei jeder MS Kinect unterschiedlich ist. Aufgrund dessen, dass die Position der Punkte bekannt ist, kann bei anschließend wieder aus der Punktewolke eine Oberfläche erzeugt werden.

1.3.3 MS Kinect 2 - Time of Flight Technologie (ToF)

Der Nachfolger der Kinect setzt mit einem ToF-Sensor neue Maßstäbe in der Tiefenerkennung. Anders als bei dem Verfahren mit der IR-Punktwolke, ist für die ToF-Technologie nur ein Sensor erforderlich. Die 3D-Szenen werden in Echtzeit erfasst. Außerdem ist die Aufnahme der Szene durch die hochenergetischen Lichtimpulse weniger licht- und oberflächenabhängig. Das 3D-Bild kommt nun durch die Zeitmessung zustande, wie lange ein Lichtimpuls braucht, um vom Sender zum Objekt und wieder zurückzukommen. Da die Lichtgeschwindigkeit mit $c = 3{,}8 * 10^8 \frac{m}{s}$ konstant und bekannt ist, kann dann zusammen mit der gemessenen Zeit die Distanz des Objekts errechnet werden. Wird dieses Verfahren über den Raum angewendet, entsteht ein Tiefenbild.

1.4 3D-Punktewolke

Die 3D-Punktewolke beschreibt die Gesamtmenge aller gemessenen Einzelpunkte.

2 Codiertes Lichtschnittverfahren zur Objekterkennung

Wie bereits bei der MS Kinect erwähnt, ist es möglich Objekte anhand von strukturiertem Licht zu erkennen. Hierzu gibt es verschieden Verfahren. Eines davon ist das codierte Lichtschnittverfahren. Dieses bietet vor allem Vorteile, wenn es darum geht, Kanten, Erhöhungen oder Tiefen eines Objekts zu erkennen. Betrachtet man sich einmal die Abbildung 4, stellt man fest, dass die Lichtstreifen entlang der Kanten des Objekts in einem anderen Winkel weiterverlaufen. Bei Tiefen sind die Streifen sogar teilweise unterbrochen oder verzerrt. Dadurch entsteht das Problem, dass die einzelnen Streifen nicht mehr eindeutig zuzuordnen sind. So könnten an Kanten, etc… die Steifen falsch zugeordnet werden und das Objekt wird nicht mehr richtig erkannt.

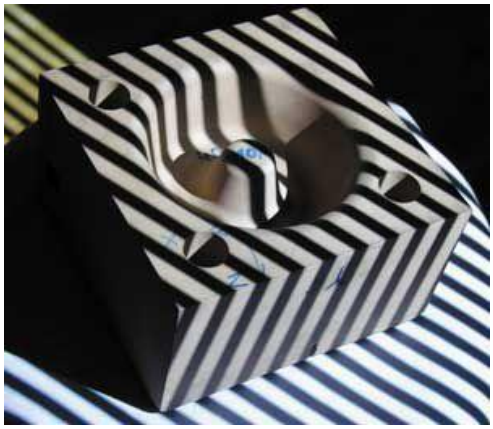

Abbildung 4 Objekt wird mit Licht angestrahlt, das durch Muster in Streifen geteilt ist

Mit dem codierten Lichtschnittverfahren wirkt man diesem Problem entgegen, indem man wie in Abbildung 5 vorgeht. Für das Verfahren sind ein Projektor (1) und eine Matrixkamera (2) notwendig. Nun bestrahlt man das Objekt zuerst mit einem Muster bestehend aus zwei Spalten. Die Spalten verdoppeln sich dann pro Muster mit jedem Durchlauf. Ist nun eine dunkle Stelle bei der ersten Belichtung am Objekt, wird das Ergebnis mit dem ersten Muster mit einer 0 codiert. In der Abbildung 5 ist das auch mit dem zweiten, somit verdoppelten Spaltenmuster der Fall. Beim dritten Durchlauf fällt dann Licht an diesem Punkt auf das Objekt. Dieser Durchlauf wird somit mit einer 1 codiert. So geht man nun bis zum letzten Muster vor. Je nachdem, ob dann entlang diesem Streifen am Objekt eine dunkle oder helle (bzw. beschattete oder beleuchtete) Stelle vorliegt, wird dieser Bildpunkt entsprechend mit 1 und 0 codiert. Anhand dieses eindeutigen Codes ist es möglich, auch die "geknickten" Lichtstreifen wieder richtig zuzuordnen. Die Matrixkamera selbst liefert ein 2D-Bild, das aus Zeilen und Spalten besteht. So ist jedem Bildpunkt eine x- und eine y-Koordinate zugeordnet. Über die eindeutig codierten Lichtpunkte, können dann entsprechenden Winkel, in denen die Kamera zu den Lichtpunkten steht, ermittelt werden. So kann dann die Entfernung bzw. Objekttiefe berechnet werden.

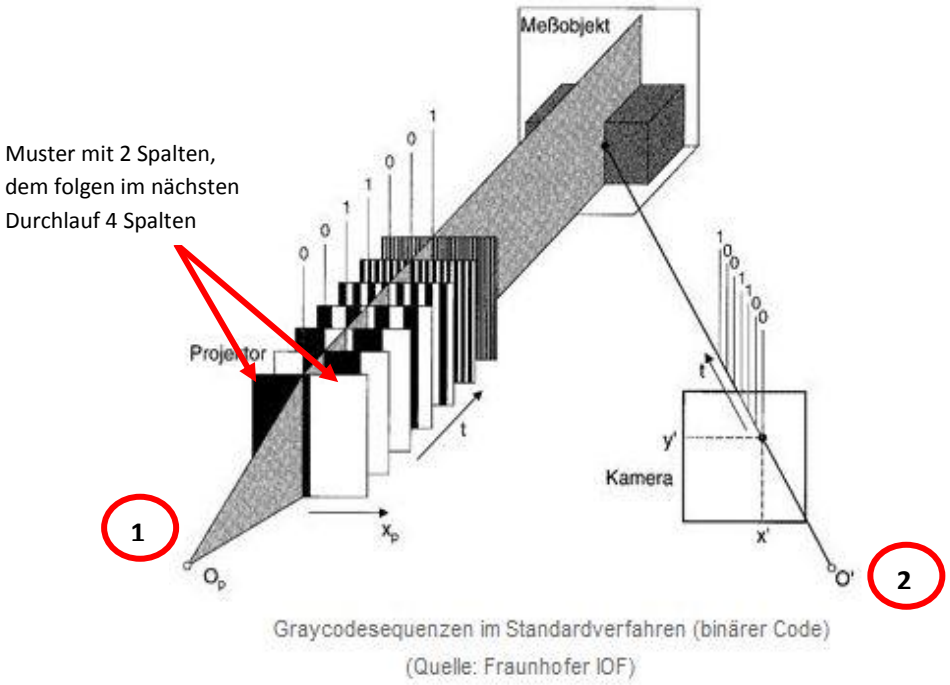

Abbildung 5 Vorgehen codiertes Lichtschnittverfahren

3 Gestenerkennung via Kinect[2]

Um Person zu erkennen gibt es verschiedene Möglichkeiten. Neben der Detektion auf Grundlage der Hautfarbe, ist das auch über Gestenerkennung bzw. Merkmalserkennung einer Person möglich. Dieser Ansatz soll im Folgenden beschrieben werden. Wie bereits im Abschnitt 1.3 bereits erwähnt, werden mit der Kinect 20 Körperpunkte erkannt über die RGB-Farb-Kamera und den Tiefensensoren. Dabei wird ein Skelettbild der Person erstellt, siehe Abbildung 6.

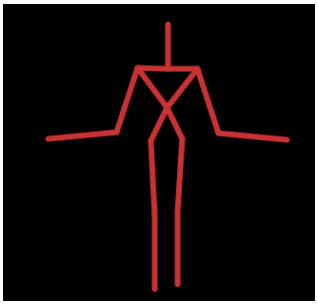

Abbildung 6 Skelettbild bzw. Joints&Bones-Darstellung

Es stellt den Menschen vereinfacht mit Knochen und Gelenken dar. Diese Skelettinformationen können dann zusammen mit den Tiefendaten ausgewertet werden. Hierbei werden auch noch die Winkel während Bewegungen mit einbezogen. So kann beispielsweise die Handposition ermittelt werden. Die Handfläche selbst wird als Punktemenge bzw. -wolke visualisiert. Über die Nachbar-Pixelsuche und radiale Histogramme kann dann die Handform ermittelt werden. Die Kinect stellt hierbei die Hand über zwei Skelettpunkte dar.
Eine Geste ist dabei als Folge von Merkmalsvektoren bzw. Merkmalsvektorsequenz zu verstehen. Ein Merkmalsvektor entsteht aus den Winkeln, die bei einer Körperhaltung einer Person entstehen. Um aus diesen Vektoren nun eine Geste erkennen zu lassen, muss zuvor ein definierter Start- und Endzeitpunkt definiert werden. Das erfolgt in der Software Kinetic Space ganz automatisch. Will man hier eine Geste einlernen, geht man in das dafür vorgesehene Interface, startet das Recording und stellt sich vor die Kinect mit der entsprechenden Geste. Nach 3 Sekunden ist die Geste über diese Software eingelernt. Dahinter stecken diese Merkmalsvektorsequenzen (MVS). Über die eingelernten Vektoren können nun Klassen gebildet werden, die dann für temporäre MVS zum Vergleich herangezogen werden. Die Klassenzuteilung erfolgt dann über den k-Nearest-Neighbour-Algorithmus. Man stellt sich dafür einen Raum mit verschiedenen, eingelernten MVS vor. Irgendwo dazwischen liegt nun die temporäre MVS. Um diese nun der entsprechenden Klasse, also Geste, zuzuordnen, bezieht man die umliegenden MVS mit ein. Die einfache Klassifikation würde sich dann nach der häufigsten Anzahl eines MVS in dem Betrachtungsraum richten. Besser ist jedoch die gewichtete Variante, bei der die Summe der Kehrwerte der Distanz des temporären MVS zu den umliegenden MVS gebildet wird. So "gewinnt" die am nächsten liegende Klasse. Die Distanz selbst wird bestimmt über die zeitliche Betrachtung einer MVS von Zeitpunkt x bis Zeitpunkt y, wobei gelten muss x<y. Hierfür gibt es mehrere Betrachtungsweisen, wie den euklidischen Abstand oder die Pearson-Distanz.
Über Mean shifts werden die Gelenke der Person erkannt.

[2] (Schmutzler, 2013)

4 Roboter Leonie per Gesten steuern

4.1 Systemstart

Der Ordner mit sämtlichen Dateien und Informationen kann durch folgenden Befehl im Terminal geöffnet werden.

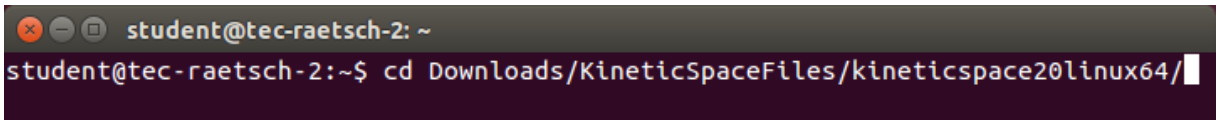

Abbildung 7 Ordner aufrufen

Um die Gestensteuerung umzusetzen, muss zuerst das MIRA-Center auf Leonie (Server) gestartet werden, sowie das C++-File philunitname geöffnet werden. Dazu gibt man folgenden Befehl im Terminal ein.

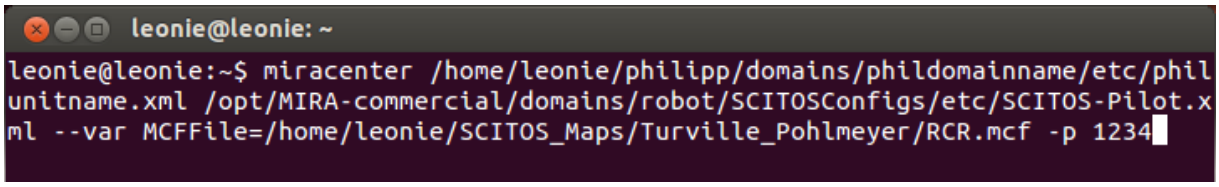

Abbildung 8 MIRA-Center über Terminal starten

Um die Software Kinetic Space zu öffnen, muss der Befehl wie in Abbildung 9 in das Terminal eingegeben werden. Zuvor muss die MS Kinect über USB an den PC angeschlossen werden, sowie an die Stromversorgung. Mit Enter bestätigt man den Befehl und es öffnet sich Kinetic Space. Falls die Meldung kommt, dass die MS Kinect noch nicht angeschlossen sei, die USB-Verbindung der Kinect nochmals trennen und die Software beenden. Dann muss die Kinect wieder angeschlossen werden, kurz warten und dann erst die Software öffnen. Es dauert immer ein paar Momente bis die Kinect wirklich verbunden ist.

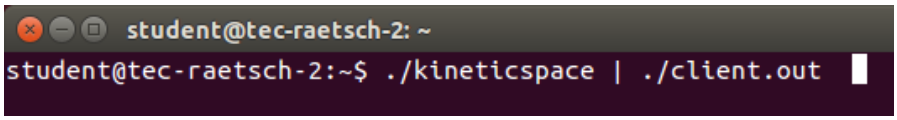

Abbildung 9 Befehl zum Starten von Kinetic Space

Das Interface sieht dann folgendermaßen aus.

Dieser Aufruf startet sowohl Kineticspace wie auch client.out. Kineticspace gibt die erkannten Gesten im Terminal aus. Durch eine sogenannte Pipe werden die Daten an die Software client.out weitergegeben, welche diese Daten dann 1:1 an Leonie sendet.
Es ist zu beachten, dass sich der Rechner mit der Kinect sowie Leonie beide im selben Netztwerk (Robocup WLAN) befinden müssen. Die Leonie hat die IP-Adresse 192.168.1.201. Wird diese IP-Adresse verändert, muss auch das Programm client.out entsprechend abgeändert werden.

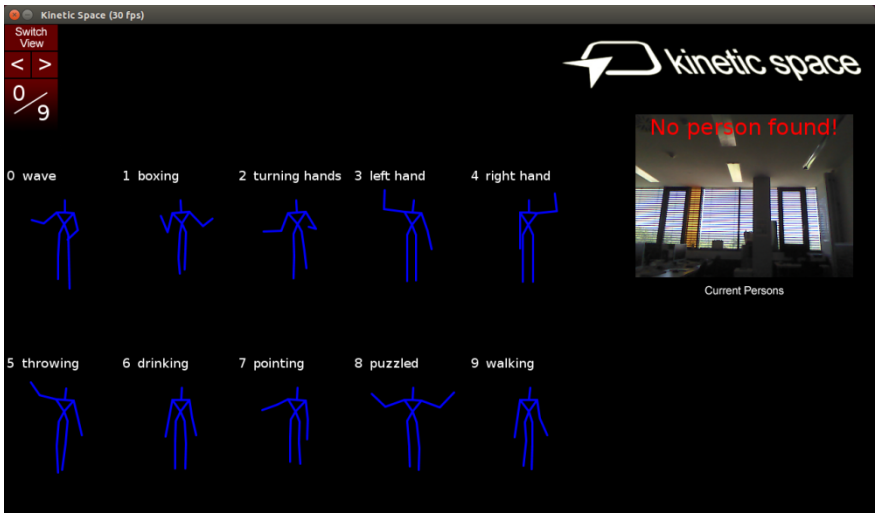

Abbildung 10 Kinetic Space by Matthias Wölfel

Bis jetzt wurde noch keine Person erkannt. Dazu stellt man sich vor die MS Kinect, bewegt die Arme und bewegt sich auf die Kinect zu. Ist eine Person erkannt, ändert sich das Kamera-Farbbild in ein Skelettbild, dass auch Joints&Bones-Darstellung genannt wird, siehe Abbildung 11. Gegenlicht könnte zu Problemen bei der Erkennung führen, deswegen ist es besser, wenn man sich vor eine Wand stellt.

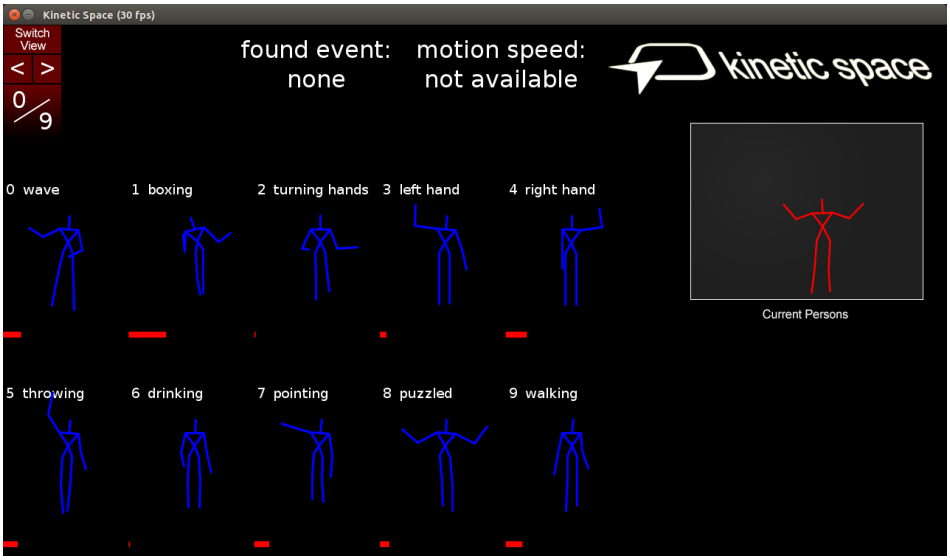

Abbildung 11 Kinectic Space - Person erkannt

Führt man nun eine entsprechende Geste wie in den Vorgabe durch und wird diese erkannt, wird das durch einen grünen Balken unterhalb der entsprechenden Geste signalisiert. Darüber hinaus werden oben am Bildschirm die Nummer, sowie die Beschreibung unter "found event" angezeigt.
Um neue Gesten einzulernen oder bestehende zu konfigurieren, klickt man links oben auf "Switch View". Es erscheint der folgende Screen.

Körperpartien gewichten

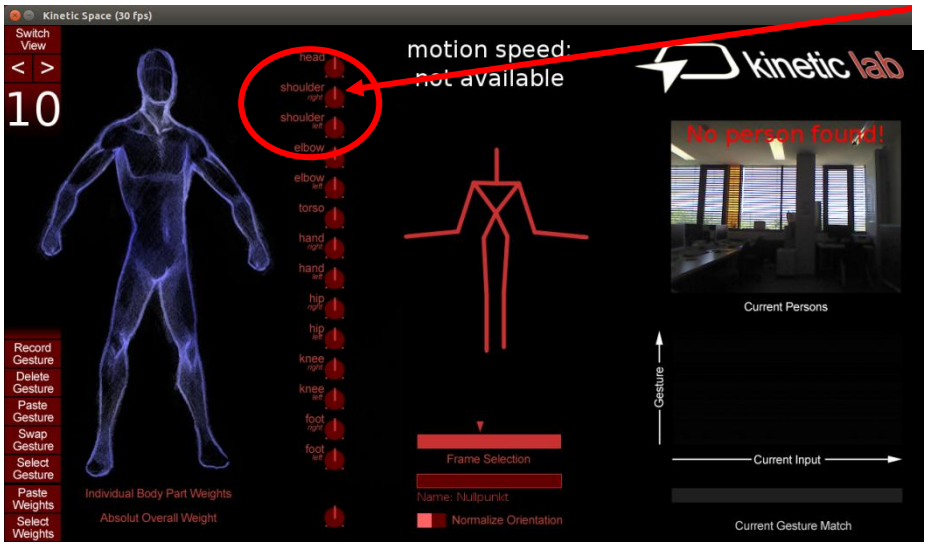

Abbildung 12 Trainmode

Links am Rand wählt man "Record Gesture", sobald eine erkannte Person vor der Kinect steht. Dann sollte man möglichst ruhig mit der entsprechenden Geste stehen bleiben. Dynamische Gesten sollten so gleichmäßig wie möglich ausgeführt werden, um die Nachahmung später zu erleichtern. Dann läuft ein Countdown und das Recording wird gestartet. Die Beendigung wird angezeigt und man kann nun unter dem Skelettbild einen Namen eingeben. Außerdem ist es möglich die Körperpartien unterschiedlich zu gewichten, d.h. für die Erkennung liegt der Fokus nur auf bestimmten Körperregionen. So erhöht man ggf. die Erkennungswahrscheinlichkeit.

4.2 Wie Leonie das Laufen lernte

Nachdem der C++-Code der Datei philunitname (Pfad auf Leonie zu dem C-File: /home/leonie/philipp/domains/phildomainname/src/philunitname.c) in seinen Funktionen geklärt wurde, ist zunächst die Fragestellung aufgekommen, welche Befehle (Services) im MIRA-Center für das Fahren und Stoppen implementiert sind. Durch "Trial and Error" wurde dies über den Konsolenaufruf im MIRA-Center /robot/Robot/.*Service(Übergabeparameter)* herausgefunden. Nach intensiver Recherche und der Hilfe von Herr Felix Ostertag war klar, dass es keinen direkten Fahrbefehl im MIRA-Center gibt für das einfache Geradeaus-Fahren. Daher wurde die auf der *http://www.mira-project.org/MIRA-doc/domains/navigation/Pilot/*vorgeschlagene Lösung getestet. Die Funktion setTask() verfügt über die Subtasks, die Leonie dazu veranlasst, die vorgegebenen Koordinaten anzufahren und sich anschließend um 180° zu drehen. Ausgelöst wird der Fahrbefehl über die zuvor eingelernte Geste 12, was jedoch auch in der Codesequenz leicht geändert werden kann. Die Koordinaten beziehen sich auf die im MIRA-Center eingelernte und ausgewählte Karte. Die Beschreibung zum Einlernen einer Karte ist in Abschnitt 4.3 zu finden. Um den geänderten Code jedoch zuerst kompilieren zu können, muss im Terminal der Befehl *make* eingegeben werden und mit Enter bestätigt werden, siehe

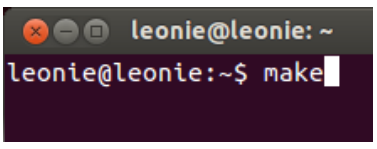

Abbildung 13 Modifizierten Code kompilieren

4.3 Karte einlernen

Da Leonie momentan noch vorgegebene Koordinaten benötigt, um sich fortzubewegen, ist es notwendig eine Karte oder auch mehrere von der gewünschten Umgebung anzulegen. Hierfür öffnet man das MIRA-Center wie in Abbildung 8. Anschließend führt man die folgenden Schritte aus.

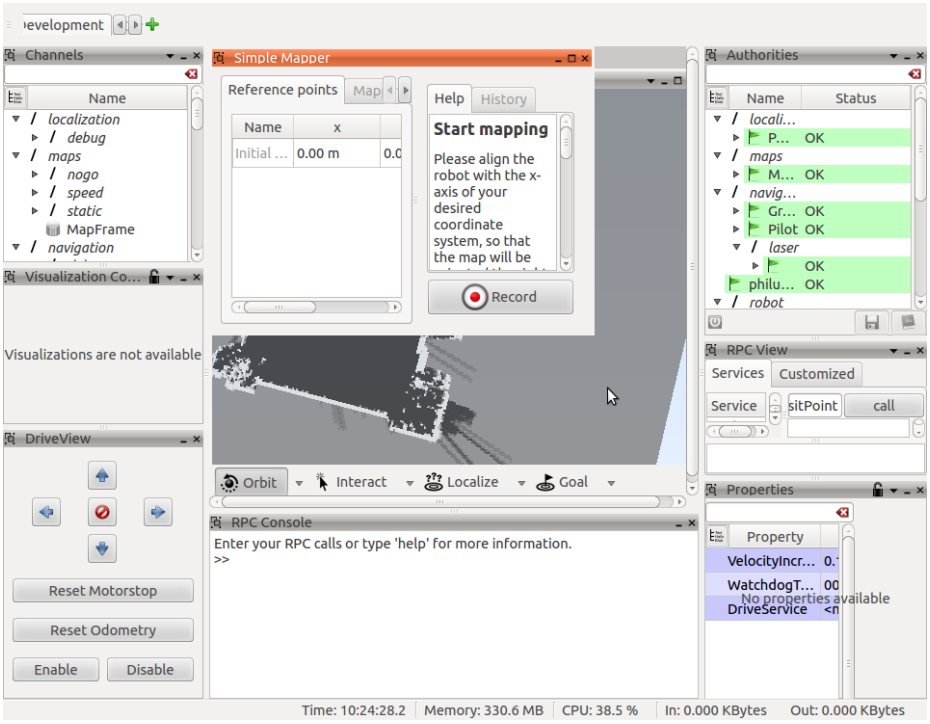

Abbildung 14 Mapmenü öffnen

Nachdem das Miracenter gestartet wurde, kann man durch CTRL+D das „Choose a view“ Fenster öffnen (siehe Abbildung 14). Hier kann der Simple Mapper geöffnet werden. Klickt man auf Record, zeichnet Leonie ihre Umgebung auf. Schiebt man den Roboter nun durch den Raum, entsteht so eine Karte. Hierbei ist darauf zu achten, dass man nach Möglichkeit gerade Bahnen fährt und Kurven nicht in Bögen sondern in 90° Kurven fährt.

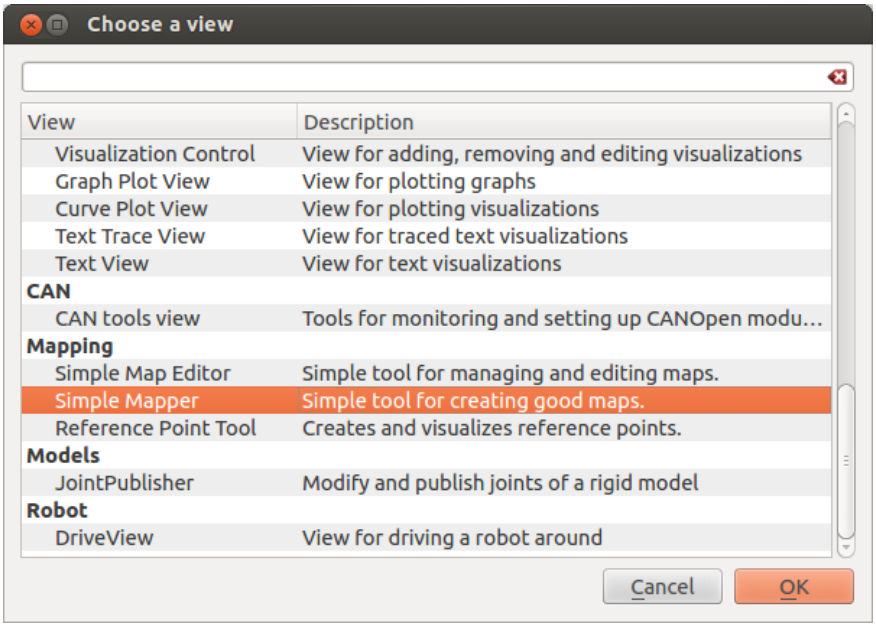

Abbildung 15 Mapview auswählen

Öffnet man im „Choose a View“ Fenster den Simple Map Editor, können zuvor gespeicherte Karten bearbeitet werden. Beispielsweise lassen sich sogenannte NoGo-Areas hinzufügen, welche es Leonie verbieten an die entsprechende Stelle zu fahren. (Siehe Abbildung 16)

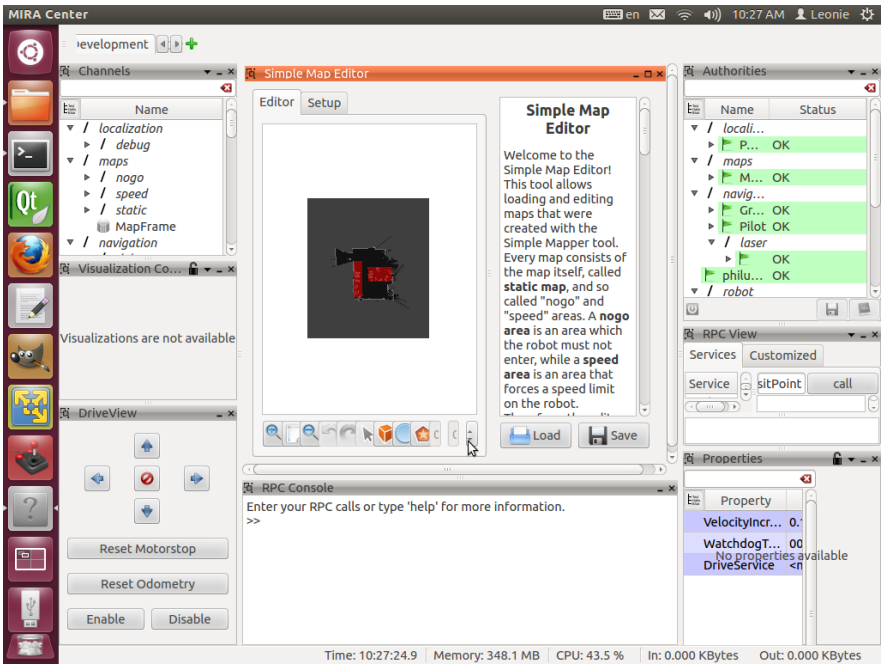

Abbildung 16 Simple Map Editor

4.4 Die Nächsten Schritte

Nachdem nun das Anfahren bestimmter Koordinaten über eine Geste möglich ist, sehen wir nun das Potential, die Gestensteuerung noch auf das Stoppen während der Fahrt, sowie das Zurückfahren zu den Ursprungskoordinaten und das Weiterfahren zu den Zielkoordinaten nach einem Stopp auszubauen.

Zukünftig kann versucht werden, Leonie solange geradeaus fahren zu lassen, wie eine bestimmte Geste über die MS Kinect erkannt wird. Als Notstop kann eine weitere Geste definiert werden. Ansonsten soll Leonie stoppen, sobald keine Geste mehr erkannt wird. Sinnvollerweise kann eine Richtungsänderung über die Gesten 3 & 4 (arm left, arm right) umgesetzt werden. Für diese flexible Fahrsteuerung muss dann auch die MS Kinect auf Leonie angesprochen werden können.