

BERGISCHE UNIVERSITÄT WUPPERTAL

---

AUSARBEITUNG ZUM PROJEKT IM KURS  
MULTIMODALE-MENSCH-MASCHINE-SYSTEME

---

# Inertialsensorbasierte Fernbedienung zur Mensch-Maschine Interaktion

---

B.Sc Felix SCHÜRMANN  
Mat.Nr.: 1110259

6. Februar 2019

# Inhaltsverzeichnis

<b>1 Einleitung</b>	<b>2</b>
<b>2 Grundlagen</b>	<b>3</b>
2.1 ThreeJS . . . . .	3
2.2 Raspberry Pi . . . . .	3
2.3 Adafruit 10-DOF IMU Breakout . . . . .	3
<b>3 Inertiale Messeinheiten</b>	<b>4</b>
3.1 Beschleunigungssensor . . . . .	4
3.2 Gyroskop . . . . .	4
3.3 Magnetometer . . . . .	4
<b>4 Implementierung der inertialen Messeinheit</b>	<b>5</b>
4.1 Signalverarbeitung . . . . .	6
4.1.1 Komplementärfilter . . . . .	7
4.1.2 Pitch & Roll . . . . .	8
4.1.3 Neigungskompensation . . . . .	8
4.2 Kommunikation . . . . .	8
<b>5 Benutzeroberfläche</b>	<b>9</b>
5.1 Einmessen des Bildschirms . . . . .	11
5.2 Funktionalität . . . . .	13
5.2.1 Gestenerkennung . . . . .	13
5.2.2 Translation . . . . .	14
5.2.3 Rotation . . . . .	15
5.2.4 Clear . . . . .	15
<b>6 Herausforderungen</b>	<b>15</b>
<b>7 Ausblick und Machbarkeitsanalyse</b>	<b>16</b>

# 1 Einleitung

Inertiale Sensorik hat breiten Einzug in das alltägliche Leben gefunden. Die meisten aktuellen Smartphones besitzen eine Kombination aus Beschleunigungssensoren, Gyroskopen und Magnetometern. Verwendung finden diese im Smartphone meist zur Lagebestimmung und Unterstützung von Spielen. Die aufkommende Technik im Bereich der virtuellen und augmentierten Realität nutzt diese Sensorik zur Lagebestimmung im Raum und zur Interaktion mit virtuellen Elementen, wobei die Sensorik in Brillen und Geräten zur Handsteuerung verbaut ist. Das Projekt im Zuge des Kurses untersucht hierfür grundlegende Anforderungen und Möglichkeiten, die sich durch die Kopplung von inertialen Messeinheiten zur Steuerung von Objekten in virtueller Realität ergeben. Hierfür wurde eine intertiale Messeinheit mit 10 Freiheitsgraden über einen Raspberry Pi eingebunden. Dieser gibt die Messdaten über ein Websocket an einen Webbrower weiter. Zur grafischen Anschauung dient eine mittels ThreeJS erstellte Szene.

## **2 Grundlagen**

### **2.1 ThreeJS**

ThreeJS ist eine Javascript Bibliothek und API, die verwendet wird um dreidimensionale Computergrafik im Webbrower darzustellen. Hierzu bildet OpenGL die Grundlage. Die Vorteile von ThreeJS finden sich in der gut zu verstehenden Struktur und der großen Community, die viele gute Beispielprojekte liefert.

### **2.2 Raspberry Pi**

Der scheckkartengroße Computer mit ARM Prozessor erfreut sich immer größerer Beliebtheit. Sein günstiger Preis und seine vielfältigen Einsatzmöglichkeiten machen ihn zum idealen Tool für angehende Ingenieure und Bastler. Das hier eingesetzt Modell 2 B besitzt einen CPU Takt von 900 Mhz und einen Arbeitsspeicher von 1 GB.

### **2.3 Adafruit 10-DOF IMU Breakout**

Die inertiale Messeinheit 'IMU Breakout' von Adafruit benutzt das drei Achsen Gyroskop L3GD20H, den Beschleunigungssensor und Kompass LSM303 sowie den Sensor BMP180 zur Messung des barometrischen Luftdrucks.

## 3 Inertiale Messeinheiten

### 3.1 Beschleunigungssensor

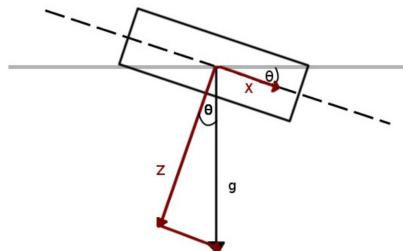


Abbildung 1: Bestimmung der Ausrichtung im Beschleunigungssensor

Wie in Abbildung 1 zu sehen, kann die Orientierung eines Sensors mit Hilfe der Erdbeschleunigung bestimmt werden. Weiterhin kann eine Strecke durch doppelte Integration von Beschleunigungen abgeschätzt werden. Problematisch sind hier die großen Ungenauigkeiten im Signal, die bei doppelter Integration zu exponentiellem Fehler führen.

### 3.2 Gyroskop

In MEMS Gyroskopen wird die Drehrate ermittelt. Dieses erfolgt durch Ausnutzen der Coriolis Kraft, unter Zuhilfenahme einer auf dem Sensor oszillierenden Komponente. Auch hier wird der Winkel durch Integration der Drehrate bestimmt, was zu einer erhöhten Störanfälligkeit und zu einem sogenannten Drift führt, d.h., dass ein errechneter Winkel nicht konstant bleibt, auch wenn der Sensor sich nicht mehr bewegt.

### 3.3 Magnetometer

Eine weitere Möglichkeit die Ausrichtung zu bestimmen bietet das Magnetometer. Hier wird mit Hilfe des Erdmagnetfeldes die Lage bestimmt. Hierfür muss die gesamte dreidimensionale Ausrichtung des Sensors berücksichtigt werden, was wiederum die Störanfälligkeit dieses Sensors zusätzlich erhöht. Außerdem können lokale Magnetfelder oder Eisen die Sensorik stören.

## 4 Implementierung der inertialen Messeinheit

Zunächst wurde die inertiale Messeinheit über den I2C Bus des Raspberry Pi angebunden. Hierfür wurde der Sensor auf eine ausgemusterte Fernbedienung verbaut um die Anwendung im Feld zu simulieren. Zu sehen ist die Fernbedienung in Abbildung 2. Es bestehen fünf Kabelverbindungen zum Raspberry Pi. Versorgungsspannung, Ground, I2C Daten, Clock und eine Verbindung zu einem GPIO, der das High Signal eines auf der Fernbedienung verbauten Knopf detektiert. Um nun die Daten auszulesen, wurde ein C++ Programm implementiert, das die Daten blockweise über den I2C Bus aus den Registeradressen der IMU ausliest. Wichtig hierbei ist die korrekte Adresse des Registers, die für den jeweiligen Sensor aus dem Datenblatt entnommen wurden.

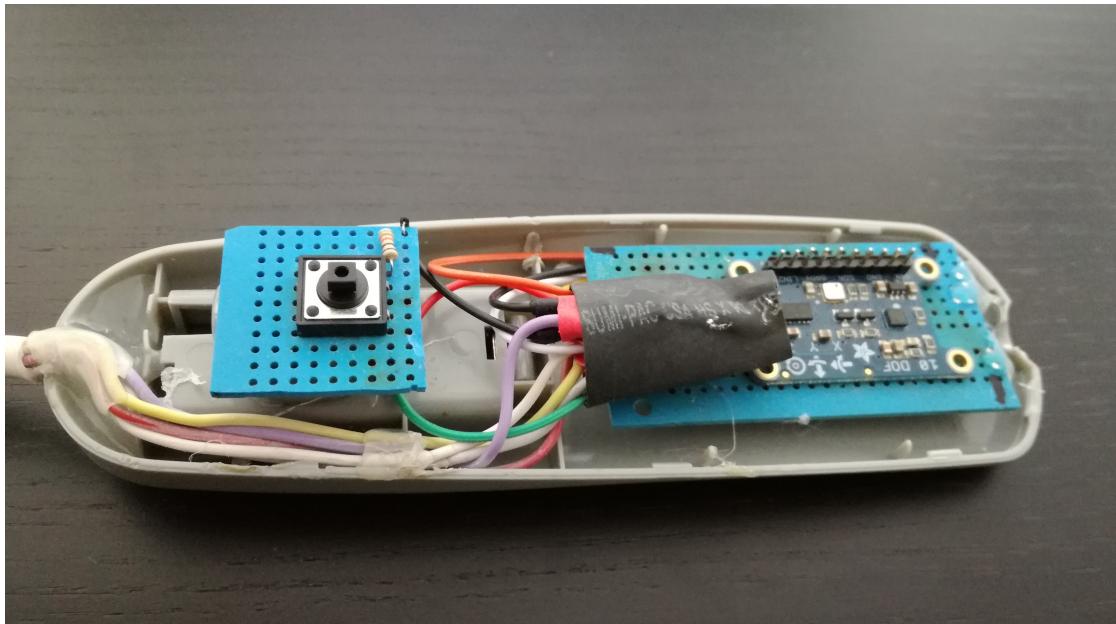


Abbildung 2: Inertiale Messeinheit auf Fernbedienung

## 4.1 Signalverarbeitung

Im gewählten Projekt stellt die Signalverarbeitung sicherlich die höchste Anforderung dar, um eine benutzerobergerechte Anwendung zu ermöglichen. Der erste Schritt besteht darin, die Offsets der Eingangssignale zu ermitteln, um diese vor der weiteren Berechnung zu subtrahieren. Beim Gyroskop wird so versucht, den Drift zu minimieren. Dazu muss in diesem Fall der Sensor flach auf dem Tisch liegen, um eine gewisse Anzahl an Messwerten zu mitteln. Im Anschluß kann das stärker verrauschte Signal des Beschleunigungssensors durch einen Tiefpassfilter geglättet werden. Das Magnetometer muss ortsspezifisch neu kalibriert werden, vor allem um die Störungen durch hartes Eisen zu kompensieren. Diese zeigen sich als Verschiebung der magnetischen Messwerte weg vom Ursprung, siehe Abbildung 3. Zur Kompensation werden Minimal- und Maximalwerte gemittelt und vom Sensorsignal subtrahiert, die vorher durch Schwenken des Sensors im Raum aufgenommen wurden. Nun sind die Grundlagen geschaffen, um die Ausrichtung des Sensors zu bestimmen. Abbildung 4 zeigt die im Projekt implementierte Methode als leichte Abwandlung des in [1] gezeigten Signalflussgraphen. In Kapitel 7 wird im Weiteren auf versiertere Methoden eingegangen, zusammen mit den Anforderungen, die es für eine professionelle technische Umsetzung benötigt.

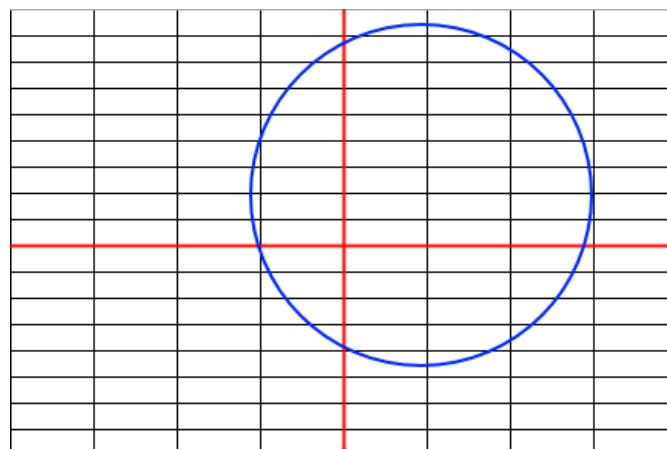


Abbildung 3: Signalflussgraph zur Bestimmung der Ausrichtung [2]

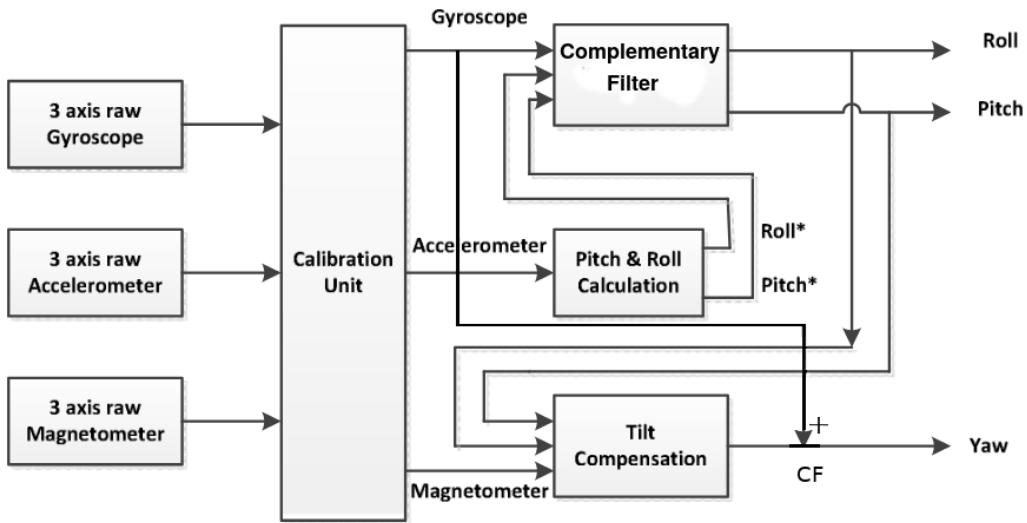


Abbildung 4: Signalflussgraph zur Bestimmung der Ausrichtung [1]

#### 4.1.1 Komplementärfilter

Zur Bestimmung von Pitch und Roll Winkeln wird ein Komplementärfilter verwendet. Die Funktionsweise ist anhand des Auszugs aus dem Code zu erkennen.

```
1 CFangleX=AA*(CFangleX+rate_gyr_x*DT) +(1 - AA) * AccXangle;
```

Hierbei werden schnelle Winkeländerungen durch hohe Gewichtung (Konstante AA) der Gyroskopmesswerte bestimmt. Bei stillstehender Drehrate kommt mit jedem Programmdurchlauf der aus dem Beschleunigungssensor bestimmte absolute Winkel mehr zum Tragen. Diese Funktion bildet den Kompromiss zwischen der absoluten Genauigkeit durch den Beschleunigungssensor und dem verringerten Rauschen und besserer Erkennung von kleinen schnellen Bewegungen durch das Gyroskop. Der Komplementärfilter wurde im Weiteren auch verwendet, um die Ausrichtung auf der Z-Achse zu bestimmen. Hier bildet die Drehrate des Gyroskops die Komponente für schnelle Bewegungen und das Magnetometer korrigiert die absolute Position.

#### 4.1.2 Pitch & Roll

Wie bereits in Abbildung 1 angedeutet, wurden die Winkel durch die geometrischen Beziehungen mit Hilfe der Atan2 Funktion berechnet:

```
1 AccXangle = (float) (atan2(act_acc_y ,act_acc_z )+M_PI)*RAD_TO_DEG;  
2 AccYangle = (float) (atan2(act_acc_z ,act_acc_x )+M_PI)*RAD_TO_DEG;
```

Weiterhin wurde eine Funktion implementiert, die den Gimbal Lock verhindern soll. Dafür wurden die möglichen Winkel auf  $\pm 90$  begrenzt. Beispiel:

```
1 if (CFangleX > 90) CFangleX = (180 - CFangleX);
```

#### 4.1.3 Neigungskompensation

Um die durch Neigung des Sensors verschobenen magnetischen Feldstärken auszugleichen, müssen diese komponentenweise mit der Orientierung neu berechnet werden. Array Index 0 steht für die X-Achse, 1 für Z und 2 für Y. [2]

```
1 magXcomp = magRaw[0]*cos( pitch )+magRaw[2]*sin( pitch );  
2 magYcomp = magRaw[0]*sin( roll )*sin( pitch )+magRaw[1]*cos( roll )-magRaw  
[2]*sin( roll )*cos( pitch );
```

Zudem wurde die Deklination berücksichtigt, die abhängig vom Standort auf der Erde eine Verfälschung des wahren Nordpols zum magnetischen Nordpol erzeugt.

### 4.2 Kommunikation

Um die Daten des C++ Programms weiter nutzen zu können, wurde die Ausgabe im JSON Format gewählt.

Mit Hilfe des Programms "websocketd" wurde ein WebSocket erzeugt, der bei einem Verbindungsaufbau zu diesem automatisch eine Instanz des C++ Programms startet. Anschließend kann mittels HTML/Javascript auf den geöffneten WebSocket zugegriffen werden um die JSON Daten zu verarbeiten. Die übergebenen Daten beeinhalten die Beschleunigungen der Achsen, die durch den CF errechneten Winkel, die Ausrichtung der Z-Achse und das Signal des Knopfes.

## 5 Benutzeroberfläche

Zur Erprobung der Fähigkeiten der Fernbedienung wurde eine Benutzeroberfläche im Webbrowser mit Hilfe von ThreeJS erzeugt. Diese kann über die lokale IP des RaspberryPi aufgerufen werden.

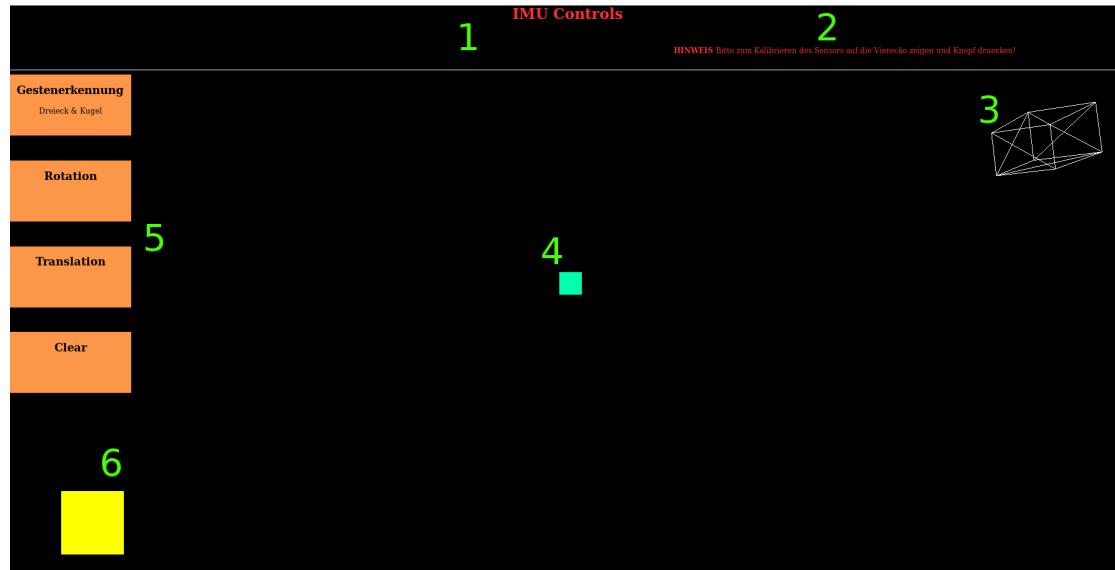


Abbildung 5: Benutzeroberfläche

Die in Abbildung 5 eingezeichnete Nummerierung beschreibt die vorhandenen Objekte wie folgt:

- 1) Überschrift / Beschreibung der Seite
- 2) Kommentarfunktion. Das HTML div gibt Anweisungen zur Benutzung und wird je nach Vorgang aktualisiert.
- 3) Wireframe Würfel zur Anzeige der Lage des Sensors im Raum. Dieser gibt alle Bewegungen in allen Achsen wieder.
- 4) Pointer, der ähnlich wie ein Mauszeiger den auf dem Schirm mit der Fernbedienung anvisierten Punkt anzeigt und verwendet wird um Objekte zu zeichnen oder auszuwählen.
- 5) Menüleiste, die zur Auswahl der Funktionen verwendet wird. Umschalten des Modus erfolgt durch Zeigen mit dem Pointer auf die gewünschte Funktion.
- 6) Eines der Quadrate zum Einmessen des jeweiligen Bildschirms.

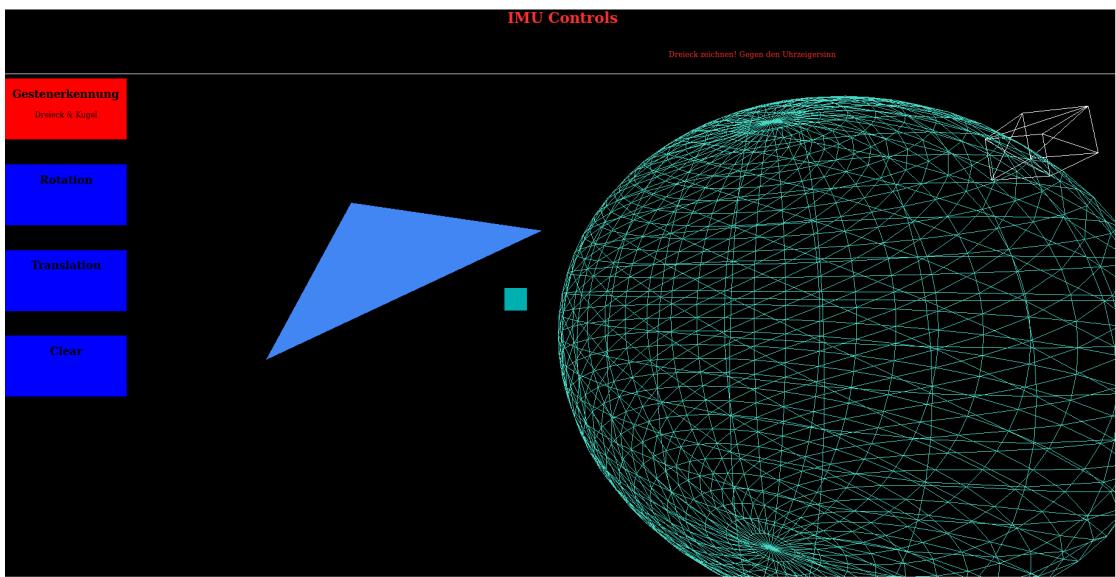


Abbildung 6: Dreieck und Kugel

In Abbildung 6 ist die geänderte Menüführung zu erkennen. Hier ist der Modus zur Gestenerkennung angewählt und somit ist der gewählte Menüpunkt mit rotem Hintergrund gekennzeichnet. Die nicht gewählten Menüpunkte werden in blau dargestellt. Zu sehen sind außerdem Zeichnungen einer Kugel und eines Dreiecks.

## 5.1 Einmessen des Bildschirms

Damit die inertiale Messeinheit den Pointer korrekt darstellen kann, muss bekannt sein, wie die Lage des Schirms oder der Projektion des Bildes im Raum ist. Daher wird der Schirm zunächst eingemessen, indem der Nutzer hintereinander mit der Fernbedienung auf die leuchtenden Quadrate in den Ecken des Schirms zeigt und den Knopf drückt. Hierbei werden die gemessenen Winkel in X,Y und Z eingespeichert. Anschließend werden die nötigen Werte berechnet.

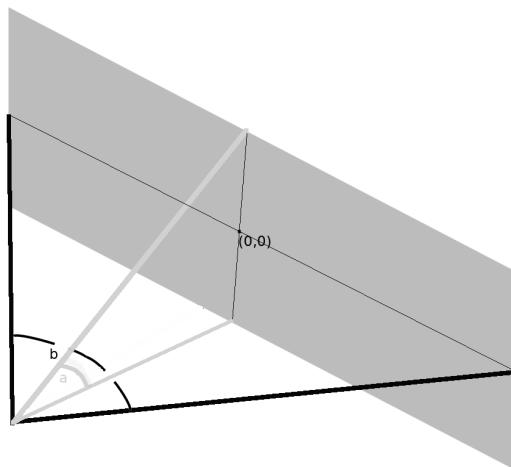


Abbildung 7: Einmessen des Bildschirms

Der gezeigte Code ermittelt die Lage und Größe des Schirms. Hierbei werden die Ränder durch den jeweiligen Mittelwert der beiden Messungen der Achsen bestimmt. Anschließend wird der gesamte Winkelbereich des Schirmes ermittelt. Hierbei muss aufgrund der 360 Grad Darstellung der Z-Achse die Anzeigerichtung des Schirms berücksichtigt werden um ggf. rechte und linke Seite zu vertauschen. In Abbildung 7 steht a für den Winkel PitchDif, b für YawDif.

```
1 function calcWindow() {  
2     leftYawAngle = (pl1AngleZ+pl3AngleZ) / 2;  
3     rightYawAngle= (pl2AngleZ+pl4AngleZ) / 2;  
4     upperPitchAngle = (pl3CFangleY+pl2CFangleY) / 2;
```

```

5 bottomPitchAngle = (pl1CFangleY+pl4CFangleY) /2;
6 YawDif=Math.abs(rightYawAngle-leftYawAngle);
7 if(YawDif>=180){
8   if(leftYawAngle>=rightYawAngle){
9     leftYawAngle=leftYawAngle-360;
10    YawDif=Math.abs(rightYawAngle-leftYawAngle);
11  }
12 }else{
13   rightYawAngle=rightYawAngle-360;
14   YawDif=Math.abs(rightYawAngle-leftYawAngle);
15 }
16 }
17 PitchDif=Math.abs(upperPitchAngle-bottomPitchAngle);

```

Wie in Abbildung 7 zu sehen, steht in ThreeJS der Mittelpunkt der Szene für die Bildschirmkoordinaten (0,0). Um die Position des Pointers zu berechnen, werden lineare Funktionen gebildet, die mit Hilfe der bestimmten Mittelpunkte der realen Winkel und den Koordinatenpunkten im Anzeigesystem die Verschiebung des Pointerpunktes pro realem Winkel beschreiben. Dies wird vom unteren Code beschrieben.

```

1 function calcPointPos(){
2 var m = 2*pixelBreite/YawDif;
3 var mP = 2*pixelBreiteP/PitchDif;
4 var midP= (upperPitchAngle+bottomPitchAngle)/2;
5 var mid= (leftYawAngle+rightYawAngle)/2;
6
7 if (leftYawAngle>=rightYawAngle){
8 m=-m;
9 }
10
11 var xx = msg.AngleZ-mid;
12 var yy = msg.CFangleY-midP;
13 YPos= yy*mP;
14 XPos= xx*m; }

```

Berücksichtigt werden muss auch hier die Möglichkeit der Vertauschung von rechtem und linkem Bildschirmrand. XPos und YPos bilden nun die Koordinaten des Pointers auf dem Schirm und bilden die Grundlage für weitere Berechnungen.

## 5.2 Funktionalität

Beim Starten des Programms kann zunächst ein Dreieck gezeichnet werden. Hierfür werden 3 Vertices hintereinander durch Knopfdruck und Zeigen mit dem Pointer aufgenommen. Im Anschluß stehen die Funktionen des Programms zur Verfügung.

### 5.2.1 Gestenerkennung

Es wurde eine Gestenerkennung implementiert, die mit dem Pointer gezeichnete Objekte erkennen kann. Hierfür kann in den Gestenerkennungsmodus umgeschaltet werden um dann eine Geste aufzunehmen. Im Uhrzeigersinn können entweder ein Dreieck oder ein Kreis mit dem Pointer gezeichnet werden, um zwischen den Modi zum Zeichnen eines Dreiecks oder einer Kugel auszuwählen. Die in Abbildung 8 zu sehenden Gesten können ausgegeben werden.

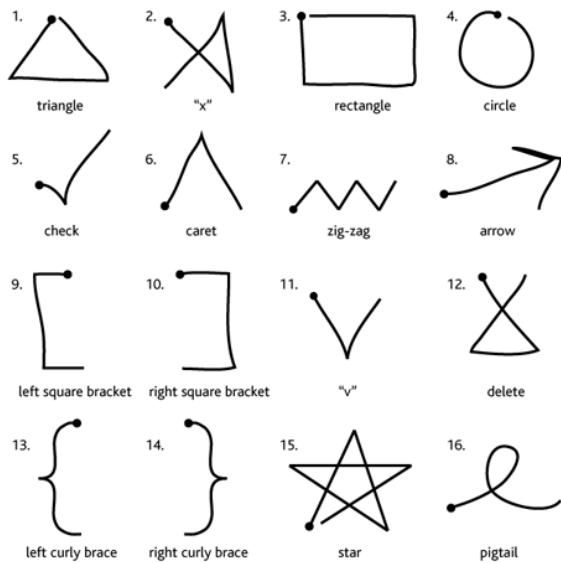


Abbildung 8: Erkennbare Gesten [3]

Zur Gestenerkennung wurde das Script ”\$1 Unistroke Recognizer”[3] verwendet. Zur Nutzung wird bei Knopfdruck ein Array aus Point Objekten erstellt, welches an das Script übergeben wird. Dieses wurde als Leichtgewicht entwickelt und ermöglicht die Erkennung mit weniger als 100 Zeilen Code. Der Algorithmus hierfür besteht im Wesentlichen aus vier Schritten. Zunächst werden die aufgenommenen Punkte neu gesampled. Hierbei werden die aufgenommenen Punkte

durch  $N$  äquidistante Punkte ersetzt. Im zweiten Schritt wird die aufgenommene Geste rotiert. Dies geschieht zunächst anhand der Anordnung von Punktpfaden bzw. den Winkeln zwischen diesen. Dann wird die aufgenommene Geste um ihren Mittelpunkt auf einen Nullwinkel gedreht, siehe Abbildung 9.

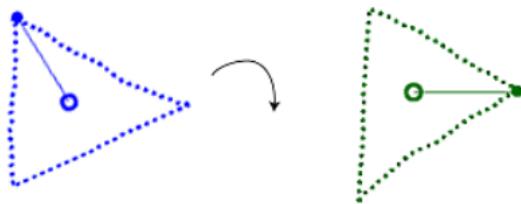


Abbildung 9: Rotieren der Geste zum indikativen Winkel [3]

Als dritter Schritt wird die Geste skaliert und translatiert, um auf eine Referenzfläche zu passen. Im vierten und letzten Schritt werden nun die Gesten mit den vorgegebenen Templates verglichen. Hierbei werden die Abstände zwischen den Punkten berechnet und nach dem nächsten Nachbarn gesucht. Weiter wird eine zusätzliche Rotation durchgeführt um das globale Minimum zwischen den Winkeln der Pfadpunkte aus Template und Geste zu finden. Die Begrenzung dieses Algorithmus findet sich im Erkennen von Gesten im zweidimensionalen Raum und kann daher nicht bei spezifischen Orientierungen oder Streckungen der Gesten funktionieren. Weiterhin können Ovale nicht von Kreisen, Quadrate von Rechtecken unterschieden werden usw. Die aber durchaus gute Erkennungsrate und sehr schnelle Berechnung bietet eine probate Alternative zu aufwendigen Methoden mittels bspw. maschinellem Lernen.

### 5.2.2 Translation

Wenn mit dem Pointer die Fläche zur Translation angefahren wird, schaltet das Programm in den entsprechenden Modus und es kann ein Objekt mit Knopfdruck erfasst und anschließend durch Neigung der Fernbedienung in die entsprechende Richtung verschoben werden. Um das Objekt zu erkennen wurde das Raytracing Verfahren implementiert.

### 5.2.3 Rotation

Die Auswahl des Objektes erfolgt wie bei der Translation. Nun kann durch Neigen der Fernbedienung nach links oder rechts das Objekt gedreht werden. Die Rotationsrate ist hierbei abhängig von der Neigung.

### 5.2.4 Clear

Mit der Auswahl der Clear Fläche werden alle gezeichneten Objekte in der Szene gelöscht.

## 6 Herausforderungen

Die im Projekt aufgetretenen Herausforderungen fanden sich vor allem in den ungenauen Daten der Sensorsignale. Am problematischsten zeigte sich das Magnetometer, das in manchen Situationen einigermaßen brauchbare Signale zur ungefähren Bestimmung der Ausrichtung lieferte, in anderen wieder nicht. Vermutlich sind hier die magnetischen Störfelder im Raum doch zu groß. Aufgrund des Alters des Sensors ist aber auch ein mechanischer Defekt des Sensors nicht auszuschließen. Auch musste die Fernbedienung wieder umgebaut werden, da die Signalkabel über metallische Klemmen verbunden waren, die sich als ferromagnetisch herausstellten. Es müsste zur weiteren Einschätzung die Situation mit einem anderen Magnetometer geprüft werden. Hiermit könnte möglicherweise der sich als maßgeblich herausgestellte Drift des Gyroskops besser kompensiert werden. Aufgrund des Drifts verschiebt sich der Pointer in der Benutzeroberfläche zu sehr, sodass eine Rekalibrierung vorgenommen werden muss, die hier provisorisch durch Drücken der Leertaste zum Zurücksetzen des Pointers auf die Nullposition implementiert wurde.

Bei der Kombination der Daten zwischen Magnetometer und Gyroskop musste aufgrund der bereits beschriebenen Problematik größere Gewichtung auf das Gyroskop gelegt werden, da auch die Erkennung der Neigungsänderung nicht optimal die dadurch entstehenden Fehler des Magnetometers ausgleicht, was zu einem Abdriften zur Seite bei einer Bewegung des Pointers nach oben und unten führte. Die hier verwendeten Methoden der Signalverarbeitung sind nicht ausreichend um die Lage des Sensors im Raum über größere Zeit korrekt zu bestimmen. Die errechnete

ten Winkel durch den Komplementärfilter unterliegen weiterhin den Auswirkungen des mechanischen Drückens des Knopfes, wodurch die Beschleunigung kurzzeitig eine Änderung der Pointer Position auslöst, die zu Fehlern beim Setzen der Vertices für bspw. das Zeichnen der Dreiecke verursacht.

Weiterhin gibt es Fehler durch zu große Neigung des Sensors. Effekte des Gimbal Locks machen sich sehr stark bemerkbar. Es gibt noch weitere Anomalien bei der Benutzung, bspw. beim Einmessen des Schirms über den Nulldurchgang. Diese könnte möglicherweise programmtechnisch behoben werden, aber der Übergang zwischen  $360^\circ$  und  $0^\circ$  zeigte sich mit Diskontinuitäten und Sprüngen , bspw. von  $355^\circ$  auf  $5^\circ$  .

Die Berechnungen der Szeneumgebung müssten weiterhin verbessert werden, um eine Funktion auf allen Auflösungen und Browzern zu gewährleisten. Zudem findet das verwendete Raytracing Verfahren nicht immer die Objekte im Raum. Hier scheint es Unterschiede zwischen Kugeln und Dreiecken zu geben.

## 7 Ausblick und Machbarkeitsanalyse

Trotz der doch immensen Herausforderungen an die Fernbedienung scheint die Umsetzung möglich. Im Bereich der virtuellen Realität und Spiele, z.B. der Wii, werden ähnliche Applikationen eingesetzt. Die hier vorgeschlagene Idee könnte sich als unterstützendes Tool für Präsentationen ergeben. Der Bedarf zum Vorstellen von dreidimensionalen Konstruktionen oder FEM Berechnungsergebnissen ist sicherlich vorhanden. Hier könnte ein Einsatz mit der Virtual Reality Modeling Language gegeben sein, um so die Präsentationsqualität von bspw. 3D-PDFs zu verbessern, in dem Objekte einfacher untersucht werden können. Um die Herausforderungen zu bewältigen müsste hier ein Ansatz mit Kalman Filter oder Gradient Descent Algorithmus zur Signalverarbeitung angewendet werden um die Schätzung der Winkel zu verbessern und den Drift des Gyroskopes auszugleichen. Zur Umgehung des Gimbal Locks sollte eine Darstellung in Quaternionen gewählt werden. Auch ist zur Rekalibrierung ein Infrarot Sender vorstellbar, der im Bereich der Mitte über oder unter dem Schirm montiert ist und somit einen Referenzpunkt bietet.

## Quellen

- [1] Fatameh Abyarjoo - Implementing a Sensor Fusion Algorithm for 3D Orientation Detection with Inertial/Magnetic Sensors  
<http://franciscoraulortega.com/pubs/Algo3DFusionsMems.pdf>
- [2] Magnetometer Tilt Compensation <http://ozzmaker.com/compass2/>
- [3] Gestures without Libraries, Toolkits or Training: A \$1 Recognizer for User Interface Prototypes <http://faculty.washington.edu/wobbrock/pubs/uist-07.01.pdf>