|  |  |
| --- | --- |
|  | Kinect SA |
|  | Technischer Bericht |

Felder werden durch InStep automatisch ausgefüllt. Bei manuellen Einträgen kann mit Doppelklick der Text selektiert und geändert werden Achtung! Dabei dürfen die Textmarken nicht gelöscht werden, da ansonsten die Verweise in der Fusszeile nicht mehr stimmen.

Änderungsnachweis

| Version | Status | Datum | Beschreibung | Autor |
| --- | --- | --- | --- | --- |
| 0.1 | Erstellung | 14.12.2012 | Merge aus bestehenden Einzeldokumenten | Josua Schmid |
| 0.2 | Revision | 14.12.2012 | Abschnitt Zieldefinition | Josua Schmid |

Inhaltsverzeichnis

[1. Allgemein 4](#_Toc343267846)

[1.1 Aufgabenstellung 4](#_Toc343267847)

[1.2 Erklärung über die eigenständige Arbeit 5](#_Toc343267848)

[1.3 Abstract 5](#_Toc343267849)

[2. Management Summary 6](#_Toc343267850)

[2.1 Umfeld, Vorgehen, Technologie 6](#_Toc343267851)

[2.2 Zusammenfassung der Ergebnisse, Ausblick 6](#_Toc343267852)

[4. Anforderungskriterien 7](#_Toc343267853)

[4.1 Zieledefinition 7](#_Toc343267854)

[4.2 Umfeld 7](#_Toc343267855)

[4.3 Konkrete Anforderungen 7](#_Toc343267856)

[5. Analyse der Gesten 9](#_Toc343267857)

[5.1 Benutzte Gesten anderer Produkte und Projekte 10](#_Toc343267858)

[5.2 Mögliche Gesten 11](#_Toc343267859)

[5.3 Entscheidung 13](#_Toc343267860)

[5.4 Abhängigkeiten der Gesten 14](#_Toc343267861)

# Allgemein

## Aufgabenstellung



## Erklärung über die eigenständige Arbeit

Wir erklären hiermit,

* dass wir die vorliegende Arbeit selber und ohne fremde Hilfe durchgeführt haben, ausser derjenigen, welche explizit in der Aufgabenstellung erwähnt ist oder mit dem Betreuer schriftlich vereinbarrt wurde
* dass wir sämtliche verwendeten Quellen erwähnt und gemäss gängigen wissenschaftlichen Zitierregeln korrekt angegeben haben.
* dass wir keine durch Copyright geschützten Materialien (z.B. Bilder) in dieser Arbeit in unerlaubter Weise genutzt haben.

Rapperswil, den 21.12.2012

Renato Bosshart Josua Schmid

## Abstract

Herkömmliche Industriepanels sind oft ungeeignet für den Realeinsatz. Zum Beispiel benutzen Arbeiter oft Handschuhe oder haben schmutzige Hände - das macht die Bedienung Touch-Panels oder Tastaturen schwierig. Das Ziel dieser Arbeit ist zu zeigen, dass die Microsoft Kinect solche Panels ersetzen kann. Dazu wurde anhand einer Evaluation von geeigneten Gesten und der Kinect-Rahmenbedingungen ein Gestenerkennungs-Framework entwickelt. Sein Zweck ist das stabile Erkennen von Gesten unter den in einer Werkshalle üblichen Störfaktoren.

Die Architektur der Software setzt vorwiegend auf Events. Das garantiert optimale Flexibilität, hatte jedoch zur Folge, dass Speicherlecks analysiert und gelöst werden mussten. Weitere Herausforderungen waren die Zuweisung von Kinect-Skeletten zu eindeutigen Personen, sowie Dauer von Gesten und die Genauigkeit der Skelette. Zur Lösung dieser Probleme mussten jeweils Heuristiken gesucht werden, die sowohl der niedrigen Komplexität der Architektur sowie der leichten Benutzbarkeit der API gerecht werden.

# Management Summary

In Zusammenarbeit mit M&F Engineering wurde untersucht ob und wie die Microsoft Kinect für die stabile Bedienung von Industriepanels geeignet ist. Als Produkt dieser Evaluation wurde eine Programmbibliothek entwickelt welche stabile Benutzereingaben für jegliche .NET-basierte Applikationen ermöglicht. Die Bibliothek wird durch eine einfache API beschrieben. Jene ist komfortabel zu benutzen, ausführlich dokumentiert und gut zu erweitern. Um Firmenkunden einen Überblick über die Möglichkeiten der Bibliothek und der zugehörigen API zu gewähren, wurde eine GUI-Applikation entwickelt, welche die API implementiert und bildhaft zeigt was die Bibliothek kann.

## Umfeld, Vorgehen, Technologie

Die Gestenerkennungs-Software wird typischerweise in einer Werkshalle eingesetzt. Dabei hat das Verwenden der Microsoft Kinect den Vorteil, dass auch Arbeiter mit schmutzigen Händen oder mit Handschuhen ein Panel bedienen können - bei herkömmlichen Touch-Screens oder Tastaturen wäre dies nur schwierig möglich.

Konkret bedient man in unserem Beispiel-Setting einen grossen, weit entfernten Bildschirm von einem gekennzeichneten Ort in der Halle aus. Auf jenem Bildschirm sind z.B. Details über die Tagesproduktion oder den Maschinenstatus bequem einsehbar. Der Arbeiter geht dafür in den für die Bedienung vorgesehenen Bereich, führt die Anmeldegeste aus und hat danach Zugriff auf die bereitgestellten Informationen.



Abbildung : Beispiel-Setting in der Werkshalle einer Druckerei[[1]](#footnote-1)

## Zusammenfassung der Ergebnisse, Ausblick

Mit dieser Arbeit steht ein stabiles Gestenerkennungs-Framework für Industrieanwendungen zur Verfügung. Es kann mit wenig Aufwand um eigene Gesten erweitert und so direkt den eigenen Anforderungen angepasst werden.

Der mitgelieferte Prototyp einer Slideshow zeigt wie eine Anwendung aussehen könnte. Wenn man seinen Programmcode anschaut sieht man, wie einfach das Benutzen der Framework-API ist.

# Anforderungskriterien

## Zieledefinition

Die Aufgabe der Semesterarbeit ist die Beantwortung folgender Frage:

*„Ist die Kinect für Industrielösungen einsetzbar?“*

Zur Erreichung des Ziels wird ein Prototyp entwickelt, welcher einen ausgewählten Satz an Eingabemethoden demonstriert. Der Prototyp wird hinsichtlich Stabilität analysiert und optimiert. Dazu gehört auch eine technische Grenzwertanalyse, welche die Einsetzbarkeit des Produkts in einem Industrie-Umfeld dokumentiert.

Der zeitliche Ablauf der Semesterarbeit umfasst drei Phasen:

* **Evaluation** bestehender technischer Lösungen und Bedienkonzepte
* **Entwicklung** und Implementierung eines Bedienkonzeptes für einen minimalen Funktionssatz. Das beinhaltet die folgenden Unterschritte pro Bedienkonzept:
  + Implementierung: Grundimplementierung
  + Stabilisierung: Es wird so lange auf Input stabilisiert, bis klar ist, ob das Bedienkonzept industrie-tauglich ist.
* **Dokumentation** der Ergebnisse

## Umfeld

Die in der Semesterarbeit zu erstellende Software wird im industriellen Umfeld eingesetzt – konkret in einer grossen Halle. Es sollen damit Bildschirme bedient werden können, die in bis zu 3m in der Höhe angebracht sind. Die Distanz von Bildschirm zum User beträgt etwa 2 - 10 Meter. Die Raumbeleuchtung befindet sich an der Decke – normalerweise Neon oder Halogen. Es könnte aber auch Tageslicht von oben oder der Seite hinzukommen.

In der Halle bewegen sich normalerweise mehrere Personen, teilweise auch kleinere Fahrzeuge. Die Arbeiter haben unter Umständen Schutzkleidung an oder Handschuhe an.

### Störfaktoren

Unter Umständen ist die Kinect kleinen Erschütterungen oder Vibrationen ausgesetzt. Durch Maschinen kann es zu Lärm oder Staubemissionen kommen. Ebenso können Wärmequellen in der Halle entstehen und sich allenfalls bewegen. Störende Beleuchtung kann künstlicher oder natürlicher Natur sein. Fabrikbeleuchtung oder -Heizung kann Licht im Infrarot- oder UV-Bereich freisetzen – das gilt auch für direktes Sonnenlicht. Da in Zukunft immer mehr LED-Beleuchtungen eingesetzt werden, müssen auch sie in dieser Arbeit berücksichtigt werden. Abgesehen von Personen und Fahrzeugen kann es auch andere Objekte haben, die sich bewegen können (z.B.: ein Kran).

## Konkrete Anforderungen

### Umgebung

* Ca. 2-10m Distanz zur bedienenden Person
* Personen sollen sich nicht überwacht fühlen (BigBrother-Problem)
* Mehrere Personen können zuschauen, es bedient jedoch immer nur jemand gleichzeitig. Mehrere Personen sollen von der Software zwar generell unterstützt aber nicht konkret implementiert werden.
* Signalisierung der Einsatzbereitschaft des Systems: Sind Inputs möglich? Hat es zu viel Licht? Person zu weit entfernt, etc.
* Anmeldung der zu bedienenden Person (z.B.: vorbeilaufen soll keine Events triggern)
* Automatische und manuelle Abmeldung (z.B.: was passiert beim Schuhebinden)
* Realisierbarkeit mit einer Kinect
* Aktive und passive Nutzer
* Möglichst unabhängig von oben erwähnten Störfaktoren

### Workflow

* Ausführbare Aktionen/Gesten:
  + Blättern
  + Scrollen
  + Zoomen
* Einfache Gestik: Personen wollen nicht den „Hampelmann machen“ oder sich exponieren.
  + Eindeutig
  + Einfach lernbar
  + Intuitiv
* Gesten sollten nach maximal zwei Versuchen erkannt werden, irrtümliche Inputs sollten nicht vorkommen.
* Einfache Kalibrierung
* Workflow soll intuitiv und nicht zu träge sein, jedoch auch nicht zu empfindlich: guter Tradeoff gesucht

# Analyse der Gesten

Damit die Begrifflichkeiten in der Semesterarbeit immer klar sind, wird im Folgenden definiert welche Gesten mit welchen Aktionen verknüpft werden.

|  |  |
| --- | --- |
| Geste | Beschreibung |
| Push | Eine Bewegung, bei der die Hand schnell nach vorne bewegt wird. |
| Pull | Die Hand wird von einer Position vorne schnell zum Körper gezogen. |
| Pinch Zoom | Durch die Bewegung von beiden Händen zueinander wird herausgezoomt, wenn sich die Hände auseinanderbewegen wird hineingezoomt. |
| Push and Pull Zoom | Durch das Heranziehen eines Objektes wird es grösser, durch das wegstossen kleiner. |
| Wischen | eine schnelle Handbewegung nach links oder rechts, die Bewegung kommt aus dem Ellbogen, nicht nur aus dem Handgelenk. |
| Joystick | Es wird die relative Handposition zu einem definierten Punkt des Körpers verwendet. Darum herum gibt es eine Deadzone, in der nichts passiert. Ausserhalb von dieser wird in diese Richtung gescrollt. Die Geschwindigkeit ist abhängig von der Distanz. |
| Armwinkel | Der Winkel der zwischen Hand, Schulter und Hüftgelenk aufgespannt wird. |
| Winken | Hin und her bewegen der Hand, die sich oberhalb des Kopfes befinden muss. Die Bewegung kommt aus dem Ellbogen. |
| Ausgestreckt | Schulter, Ellbogen und Hand bilden eine Linie. Der Arm zeigt vom Körper weg. |
| Körper lehnen | Eine Position, wobei der Oberkörper nach vorne, hinten, links oder rechts geneigt ist. Die Körperachse ist dabei gekrümmt in dieser Richtung. |
| Handgeste | Eine Geste, die nur von der Hand ausgeführt wird. Diese Gesten können wir mit unserer Ausgangslage nicht erfassen. Daher werden wir nicht weiter darauf eingehen. |
| Laser Pointer | Durch Verlängerung einer Controllerachse oder Körperteils erhält man am Durstosspunkt durch die Bildebene einen Punkt wo ein Cursor dargestellt wird. |
| Absolutes Scrolling | Ein Punkt auf der Bildebene wird fixiert. Das Bild wird genau dem nachfolgenden Bewegungsmuster folgen und der Cursor(falls vorhanden) bleibt auf dem gleichen Bildpunkt bestehen. |
| Scrolling mit Momentum | Das Bild wird beim Loslassen mit der gleichen Geschwindigkeit weiterbewegt. Mit einer Dämpfung wird es während einem Zeitinterwall abgebremst. |
| Emulation | Controller wird als Maus und Tastatur emuliert |

## Benutzte Gesten anderer Produkte und Projekte

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Device | Eigenheiten und Details | Auswahl | Zoom | Scrolling | Sonstiges |
| Xbox mit Kinect | Individuell für jedes Game, die meisten Games haben Tutorials. Kein allgemeines Konzept | Meist Push, teilweise Handerkennung | 2-Hand Pinch | Wischen, teilweise Joystick | Zusätzliche Aktionen mit z.B. bestimmtem Armwinkel Anmeldung mit Winken |
| Kinect am PC | Noch keine kommerzielle Software, meist spezifische Prototypen oder Emulatoren | individuell |  |  |  |
| Interactive Wall | 3 Beamer, 1 Kinect  Media Center | Push | - | Wischen | Anmeldung mit ausgestreckten Armen, 2 Cursor pro User |
| FAAST | Maus und Tastatur Emulator | Arm strecken | - | Körper lehnen oder Handposition |  |
| The Leap | PC Steuerung, auch mit Emulation, hohe Genauigkeit | Individuelle Handgeste | 2 Finger Zoom oder Push & Pull | Wischen | Noch nicht erhältlich |
| Sixense | Sehr genau, kurze Distanz, zwei Controller, nur Emulation | Button | Button | Button | Laserpointer |
| Glove Pie | Wii Controller am PC | Button | - | - | Nicht zuverlässig |
| Webcam | Objekt oder Handtracking am PC | Handgeste | - | Wischen | Ungenau, störungsempfindlich |
| 3D-Maus | Keine Gesten | Button | Push+Pull | Seitlich bewegen |  |
| Minority-Report | Handgesteuerte Navigation im 3D Raum. Aktionen mit Gesten | Handgeste | 2 Hand Pinch oder Handgeste | Handposition | Kleine Gesten |
| Wii | Ein Controller pro Spieler | Button | - | - | Laserpointer |
| PS Move | Ein Controller pro Spieler | Button | - | - | Genau und schnell |
| Eye Toy | Kollisionsdetektion mit Webcam | - | - | - |  |
| Light-Gun | Point and Click | Button | - | - | Keine Positionsdaten |
| Touchscreen | Weit verbreitet | Klick | 2-Finger Pinch | Absolutes Scrolling | Nur 2D |

## Mögliche Gesten

### Anmeldung

Diese Geste ist sehr wichtig, um dem System mitzuteilen, dass man es jetzt bedienen möchte. Es ist auch schwierig dafür eine gute Geste zu finden, die dem User klar ist. Es ist für einen User ungewohnt, dass er einem System signalisieren muss, dass er interagieren möchte. Alle anderen Inputsysteme reagieren sofort auf einen Input.

Arme nach aussen Halten

Diese Geste ist ziemlich eindeutig, was Fehlaktivierungen minimieren würde. Der Nachteil ist, dass diese Geste relativ viel Platz braucht, was andere Personen behindern könnte. Zudem braucht es eine Information, da diese Geste nicht intuitiv ist.

Slide to Unlock

Diese Geste wird 99% der User bekannt sein, es braucht keine Anleitung dafür. Dafür ist die Erkennung schwieriger, da die Position der Hand nicht klar definiert ist und es wird häufig zu irrtümlichen Aktivierungen kommen. Dafür braucht diese Geste wenig Platz. Apple hat darauf ein Patent, man müsste abklären, wie allgemein das formuliert ist.

Winken

Diese Geste ist bei den meisten Systemen umgesetzt und ziemlich selbsterklärend. Ein kleiner Text: „Winken zum bedienen“ wird es allen Usern klar machen. Die Erkennung dieser Geste ist etwas einfacher als Slide to Unlock, aber auch hier können selten Fehlaktivierungen vorkommen. In den ersten Wochen des Einsatzes könnte es bei den Mitarbeitern zu Missverständnissen kommen, wenn sie eine Person winken sehen, da dies eine übliche Geste zwischen Personen ist um Aufmerksamkeit zu erlangen. Das exponiert den User und lenkt andere ab in einer grossen Halle. Nach einer gewissen Zeit wird dieses Problem aber verschwinden.

Militärischer Gruss

Die militärischen Grüsse sind allgemein bekannt und intuitiv. Es stellt sich die Frage, ob die Benutzer bereit sind, vor einem elektronischen System zu salutieren. Das System erscheint u.U. unsympathisch und wird nicht akzeptiert.

Verbeugen

Intuitive Geste, die jedoch u.U. unbequem sein kann (Arbeitskleidung). Zusätzlich stellt sich die Frage, ob eine Verbeugung nicht als Unterwerfung verstanden wird und somit keine Akzeptanz bei den Benutzern findet.

### Zoom

Zoomen ist etwas weit verbreitetes, hier gibt es eigentlich nur zwei intuitive Gesten.

Pinch

Pinch Zoom ist auf Touchscreens sehr stark verbreitet und dürfte daher jedem User klar sein. Durch den Umstand, dass die Hände in sehr viele Richtungen zueinander und auseinander bewegt werden können, ist die Erkennung hierbei schwieriger und könnte mit anderen Gesten interferieren. Es wird auch zu Fehlaktivierungen kommen, was jedoch nicht kein Problem darstellen wird, da es sich nur um kleine Zoomänderungen handeln wird. Unklar ist ebenfalls, wann die Zoomfunktion aktiviert werden soll und wann der User seine Hände bewegt, damit er nachher eine Zoomfunktion auslösen kann. Zudem werden für diese Geste beide Hände benötigt, in unserem Umfeld wäre eine einhändige Bedienbarkeit sicher ein Vorteil.

Push/Pull

Je nach Darstellung ist auch diese Geste sehr intuitiv und für jeden User logisch. Diese Geste ist sehr einfach zu erkennen und eindeutig. Dadurch wird es zu wenigen Fehlaktivierungen kommen. Jedoch ist auch hier unklar, wann der User zu einer solchen Bewegung ansetzen will und wann er sie wirklich ausführen will. Diese Geste kann jedoch nur mit einer Hand bedient werden.

Wenn man diese Geste verwendet kann man logischerweise Stossen und ziehen nicht mehr für eine Auswahl verwenden.

Kreis Zeichnen

Hierbei kann der User mit einer Hand einen Kreis in die Luft zeichnen und je nach Richtung wird hinein der herausgezoomt. Das ist zwar nicht besonders intuitiv, dafür aber schnell erklärt und es kann ziemlich stabil erkannt werden. Es wird wenig Interferenz mit anderen Gesten haben. Eine sicherte Erkennung ist aber erst ab einem bestimmten Kreisdurchmesser möglich.

### Scrollen/Blättern

Bei den meisten evaluierten Systemen wird Scrollen dem Blättern gleichgesetzt. Blättern wird dabei als Einrastfunktion für Scrollen umgesetzt.

Wischen

Diese Geste wird praktisch bei allen anderen Projekten verwendet. Da man die Geschwindigkeit der Bewegung analysieren kann, werden einerseits Fehlaktivierungen minimiert, andererseits kann man die Distanz und die Geschwindigkeit auf die Aktion übertragen.

Joystick

Diese Bewegung ist nicht ganz so intuitiv wie andere, dafür hat sie den Vorteil, dass man nicht „nachgreifen“ muss. Zudem sind bei dieser Bewegung der Platzbedarf und der Bewegungsradius grösser. Je nach Gestaltung des GUIs ist auch diese Geste intuitiv und sollte allen Nutzern klar sein.

Oberkörper bewegen

Diese Geste ist nicht intuitiv und braucht sicher eine Anleitung. Zudem ist sie für unser Umfeld mit spontanen Nutzern nicht wirklich geeignet. Unter Umständen ist sie auch unbequem. Aus technischer Sicht ist diese Geste wahrscheinlich schwierig auszuwerten falls der Benutzer weit weg steht.

### Cursor bewegen

Joystick

Hierbei werden die Cursorfunktion und das Scrolling kombiniert. Das ist stabil erkennbar und hat keine Interferenz mit anderen Gesten. Es wird den Usern intuitiv klar sein.

Zeigen

Dabei wird mit dem Unterarm auf einen Punkt gezeigt. Das ist sicher ungenauer in der Erkennung und benötigt wahrscheinlich eine Kalibrierung für jede neue Session. Zudem wird die Nutzung der Geste „Push“ verunmöglicht. Für den User ist diese Geste sehr intuitiv.

2D Mapping der Handposition

Die Handposition wird vor dem Körper mit angewinkeltem Arm ausgewertet. Diese Geste ist für User gewöhnungsbedürftig und nicht sehr stabil erkennbar. Dafür interferiert sie nicht mit anderen Gesten und es wäre möglich, pro Hand einen Cursor darzustellen. Klicken könnte dann durch Zeigen umgesetzt werden.

### Auswählen

Diese Geste ist sehr wichtig, jedoch nicht ganz einfach umzusetzen, da es bei anderen Technologien dafür immer einen Button oder eine intuitive Lösung gibt. Das ist bei unserer Lösung nicht der Fall. Durch die grosse Distanz wird es zudem unmöglich Handgesten zu erkennen. Deshalb ist es wichtig beim finalen Programm darauf zu achten, dass möglichst wenige Selektionen gemacht werden müssen.

Stossen

Diese Geste ist bei anderen Projekten am häufigsten umgesetzt und dürfte dem User intuitiv einigermassen klar sein. Dafür ist es bei dieser Bewegung sehr wahrscheinlich, dass währenddessen der Cursor bewegt wird, was zu einer Fehlaktion führen wird.

Spezielle Gesten mit der anderen Hand

Hier sind verschiedene Gesten denkbar z.B. Stossen, Winken oder ausstrecken. Dadurch werden Fehleingaben minimiert, jedoch ist das einiges weniger intuitiv und benötigt zudem beide Hände.

Grab

Dabei wird ein Objekt gepackt und zu sich gezogen. Das ist etwas einfacher zu erkennen, Cursorverschiebungen werden weniger häufig vorkommen. Je nach Gestaltung des GUIs ist diese Geste intuitiv klar.

Nicken

Diese Geste ist intuitiv relativ klar und sollte gut zu erkennen sein. Fehlaktivierungen sind denkbar, wenn sich der User gerade mit jemand anderem unterhält. Das kann jedoch anhand der Blickrichtung korrigiert werden.

Timer

Wenn der User seinen Cursor nicht bewegt, beginnt ein sichtbarer Timer abzulaufen. Wenn dieser abgelaufen ist, gilt die Selektion, wenn er bewegt wird. Problematisch ist hierbei eine saubere Kalibrierung. Ebenso muss erkannt werden, wann der User etwas zeigen möchte und wann er wirklich etwas selektionieren möchte.

Thumb Up

Eine sehr intuitive Geste, jedoch nicht machbar mit Kinect auf diese Distanz. Zudem wäre sie nicht für spezielle Arbeitskleidung (Handschuhe) einsetzbar.

### Spezialaktionen (spezielle Aktionen)

Solche sind für unser Projekt nicht nötig, ausser man würde sie für die Anmeldung einsetzten. Dadurch haben wir uns nicht auf solche konzentriert bei der Recherche.

Bestimmter Winkel zwischen Körper und Armen

Diese Geste ist nicht besonders intuitiv und braucht eine Anleitung. Dafür ist sie gut zu erkennen.

### Abmeldung

Die Abmeldung geschieht automatisch nach einem Timeout wenn der Benutzer das Sichtfeld der Kinect verlässt. Eine manuelle Abmeldung könnte durch das nochmalige Ausführen der Anmelde-Geste umgesetzt werden.

## Entscheidung

Wir haben uns entschieden, zwei verschiedene Bedienmodi einzuführen. Der erste ist die Standardbedienung mit den zwei Gesten *Zoom* und *Blättern* und eventuell mehreren Auswahlgesten. Der zweite Modus ist die Joystick-Bedienung. Dieser Modus kann verwendet werden um komplexere Bedienungen wie Zeigen auszuführen. Er bildet die Funktionen *Zoom*, *Scrollen*, *Zeigen*, *Auswählen* ab. Die Implementierung dieses Modus hat für uns niedrige Priorität.

Beide Modi werden ergänzt um die *Anmelde*-Geste.

### Anmeldung – *Winken*

Hier haben wir uns für Winken entschieden, da diese Geste sehr intuitiv ist und wenige Fehlaktivierungen auslösen wird. Zudem benötigt ein Hinweis für diese Geste nur sehr wenig Platz auf GUI.

Slide To Unlock wäre ebenfalls sehr intuitiv gewesen, hätte aber viel Platz auf dem GUI benötigt und die Erkennung ist unsicherer.

Die anderen sind unterwürfige Gesten, welche bei den Benutzern nicht gut ankommen werden. Wer will sich schon einem PC unterordnen?

### Zoom – Pinch-Zoom

Aufgrund der Intuitivität haben wir uns für Pinch-Zoom entschieden, obwohl beide Hände für die Geste benötigt werden. Es ist schwierig, zu erkennen, wann die Geste getriggert werden soll und wann der Nutzer die Geste beendet hat. Beispielsweise möchte ein User schlicht seine Hände zurückziehen.

In Kombination mit dem Joystick-Mode werden wir Push-/Pull Zoom verwenden, aufgrund der eventuellen Interferenzen.

Der Zoom-Ring war zwar früher bei analogen Devices vorhanden, ist jedoch im Moment nicht als Geste üblich und könnte deshalb nicht richtig verstanden werden.

### Scrollen/Blättern – *Swipe/Joystick*

Für einfache Applikationen, die keinen Cursor benötigen und wo jeweils nur wenig geblättert werden muss (z.B. PowerPoint) werden wir *Wischen* implementieren. Für komplexere Anwendungen werden wir, falls die Zeit reicht, den Joystick verwenden, da dieser sehr gut erkannt werden kann und wir damit zugleich auch einen Cursor erhalten würden.

Den Oberkörper zu bewegen ist zu anstrengend und nicht mit allen Arbeitskleidern gut möglich.

### Cursor bewegen – *Joystick*

Wir haben uns hier dafür entschieden, dass wir nur einen Cursor verwenden werden, wenn die Software im Joystick-Modus bedient wird. In vielen Programmen ist es nämlich nicht nötig, einen Cursor zu haben. Dabei können wir die Position innerhalb einer Deadzone einfach für Cursorbewegungen verwenden.

Das Zeigen zu implementieren wäre sicherlich interessant gewesen. Wir vermuten, dass die User wahrscheinlich mit den Fingern zeigen würden. Das Erkennen von Fingern bei unserer Bedienentfernung scheitert an den technischen Möglichkeiten der Kinect. Die Implementierung der Zeigengeste mittels Arm wäre zwar technisch u.U. machbar, aber zu komplex für den Umfang unserer Arbeit.

Ein 2D Mapping der Handposition entspricht eigentlich ziemlich stark dem Modus, den wir für den Joystick verwenden, daher ist jene Geste nicht mehr zu beachten.

### Auswählen – spezielle Gesten/Nicken

Auch hier sind wir davon abhängig, ob wir uns in einem Bedienmodus befinden, der einen Cursor zur Verfügung hat, oder nicht. Im Joystick-Modus haben wir keine Handgesten mehr zur Verfügung und eine Geste mit der anderen Hand empfinden wir nicht als besonders intuitiv. Deshalb haben wir uns für Nicken entschieden, da es eine gut erkennbare Geste ist.

In einem Bedienmodus ohne Cursor, wo es nur wenige Objekte mit Interaktionsmöglichkeiten geben wird, können wir wenige Gesten auf diese Objekte legen, die wir schon implementiert haben, z.B. Stossen oder Nicken. Für eine finale Software müsste man hier allenfalls noch ein paar wenige zusätzliche Gesten definieren, was den Benutzer aber nicht überfordern darf. Das heisst, dass nicht mehr als 4 Bedienelemente zur gleichen Zeit aktiv sein sollten, andernfalls wird der Benutzer die Übersicht verlieren. Mittels animierter Icons kann man die Nutzer gut auf die zu verwendende Geste hinweisen.

Diese Geste ist sehr wichtig, jedoch nicht ganz einfach umzusetzen, da es bei anderen Technologien dafür immer einen Button oder eine intuitive Lösung gibt. Das ist bei unserer Lösung nicht der Fall. Durch die grosse Distanz wird es zudem unmöglich Handgesten zu erkennen. Deshalb ist es wichtig beim finalen Programm darauf zu achten, dass möglichst wenige Selektionen gemacht werden müssen.

Einen Timer werden wir nicht verwenden, da dies ein stabiles Inputsignal voraussetzt und den uneingeschränkten Fokus des Nutzers.

Thumb-Up wäre sehr intuitiv ist aber auf diese Distanz sicher nicht mehr erkennbar und fällt daher raus.

## Abhängigkeiten der Gesten

Im folgenden Abschnitt wird nur auf die Gesten eingegangen, die wir für unser Projekt verwenden, da es sonst unnötig kompliziert werden würde und andere Gesten für unser Projekt nicht von Belang sind.

Wir ordnen die Gesten in Datengruppen ein. Die Datengruppen beschreiben, welche Daten(Skelett-Punkte) in die Berechnung einbezogen werden müssen.

|  |  |
| --- | --- |
| Hand rechts | * Winken * Joystick * Wischen * Pinch-Zoom * Push * Pull |
| Hand links | * Pinch-Zoom |
| Kopf | * Nicken |
| Relative Position  (Abhängig von anderem Körperpunkt) | * Winken * Joystick * Push * Pull |
| Bewegungsrichtung | * Wischen * Pinch-Zoom |
| Bewegungsgeschwindigkeit | * Winken * Wischen * Nicken |
| Unterscheidung Hin- und Rückbewegung | * Wischen (zwei verschiedene Gesten) * Pinch-Zoom (zwei verschiedene Gesten) * Nicken * Winken |
| Unterscheidung Gestenstart- und Ende | * Wischen * Pinch-Zoom |

### Bemerkungen

* Für Pinch-Zoom müssen viele Faktoren analysiert werden. Es ist die komplizierteste, jedoch kann man für die anderen sicher am stärksten profitieren.
* Winken ist eher unabhängig zu anderen Gesten. Es besteht eine geringfügige Ähnlichkeit zu Wischen.
* Joystick verwendet nur die relative Position, dafür ist man auf ein stabiles Inputsignal angewiesen. Diese Geste sollt man erst machen wenn man den Input schon stabilisiert hat.
* Wischen ist ähnlich zu Pinch, aber etwas einfacher, da man nur eine Hand auswerten muss.
* Nicken hat eine ähnliche Charakteristik wie Winken, jedoch müssen andere Skelett-Punkte analysiert werden.
* Push/Pull gehören zum Joystick, es ist lediglich eine andere Achse, die analysiert werden muss.

1. Quelle: http://www.ib-hammrich.de/referenzen.php5 [↑](#footnote-ref-1)