Einsatz von Gesten in der mobilen Softwareentwicklung

Wilhelm Büchner Hochschule

Alexander Pann

850621

25. April 2021

Inhaltsverzeichnis

1 Definition

Zuerst muss der Begriff Geste für dieses Anwendungsgebiet definiert werden, da es verschiedene Bedeutung für deses Wort gibt. Die möglichen Definition wurden von Wiktionary [69] entnommen und werden hier zuerst analysiert.
Die erste Bedeutung des Begriffs meint ein Zeichen des Ausdruckes während eines Vortrages, eine nicht relevante Definition. Eine andere Bedeutung sieht eine Geste als Zeichen oder zeichenhafte Mitteilung, die etwas indirekt mitteilen möchte. Diese Definition trifft zu, da einer mobilen Anwendung durch eine Geste etwas mitgeteilt wird, allerdings ist die Definition noch zu ungenau. Die gewöhnliche Bedeutung in der Informatik ist, dass eine Bewegung mit Hilfe von einem Eingabegerätes wie einer Maus oder einem Stift oder mittels Fingern auf einem Touchscreen eine bestimmte Aktion ausgelöst wird. Diese Beschreibung ist akkurat, allerdings fehlt hierbei der Aspekt, dass Gesten auch von Sensoren erkannt werden können. Die allgemeinste Bedeutung ist, dass eine Geste ein bildhaftes oder normales Zeichen ist, dass durch diverse Körperbewegungen oder auch Bewegungen der Finger und deren Konfiguration der Kommunikation dient, allerdings oft nur einzeln und sequenziell ausgeführt werden können. Diese Definition ist sehr treffend, allerdings muss ein Zusammenhang zu Computern hergestellt werden. Die folgende Definition, beschreibt den Begriff Gestenerkennung mit Hilfe der bereits genannten Definitionen für den Begriff Geste: Gestenerkennung ist die automatische Erkennung von Gesten durch den Computer. Gesten können hierbei jegliche menschliche Bewegung sein, die von Sensoren erkannt werden können, wobei meist die sogenannten Berührungsgesten gemeint sind, die durch eine Bewegung der Finger auf einem berührungsempfindlichen Bildschirm entsteht. Bei dieser Form der Mensch-Maschine-Kommunikation werden Informationen von Menschen an die Maschine übertragen beziehungsweise bestimmte Aktionen ausgelöst.

2 Gestenerkennung

Gestenerkennung besteht aus zwei Teilaufgaben: dem Erfassungsprozess, bei dem die Geste in Daten umgewandelt wird und der Interpretation, bei der der Geste eine Bedeutung zugeordnet wird. Dieses Kapitel beschäftigt sich nur mit der zweiten Aufgabe der Interpretation, da der Erfassungsprozess von Sensor zu Sensor sehr unterschiedlich sein kann. Es wird auf die allgemeine Erkennung durch Hardware, durch Heuristik und künstlicher Intelligenz eingegangen und beschäftigt sich näher mit den sogenannten Berührungsgesten, die am häufigsten verwendet werden.

2.1 Erkennung durch Hardware

Um die Erkennung durch die Hardware zu beschreiben, muss zuerst festgestellt werden, welche Sensoren auf den populärsten mobilen Plattformen unterstützt werden. Laut statista.com [63] waren im Oktober 2020 mit 72.92% Android und mit 26.53% iOS die führenden mobilen Betriebssysteme, die zusammen fast 99% des gesamten Marktes abdecken. Daher werden in diesem und folgenden Kapiteln nur diese beiden Plattformen betrachtet und es wird aufgrund der Monopolstellung von Android ein Fokus auf Android gelegt . Die folgende Liste wurde von der Liste der unterstützten Sensoren der Android Plattform der Android Entwicklerdokumentation [25] abgeleitet und stellt die verschiedenen Sensoren vor. Es werden dabei sowohl reine Hardwaresensoren (H) erwähnt, als auch Sensoren, die softwaremäßig (S) verarbeitet wurden:

Sensorart	**Typ**	**Beschreibung**
Beschleunigung	H	Messung der Beschleunigungskräfte in m/s^2 in alle Dimensionen (x, y, z).
Raumtemperatur	H	Messung der Raumtemperatur in Grad Celsius.
Gravitation	H/S	Messung der Gravitationskräfte in m/s^2 in alle Dimension (x, y, z)
Gyroskop	H	Messung der Drehung des Gerätes in rad/s in alle Dimension (x, y, z)
Licht	H	Messung des Umgebungslichts (Illumination) in lx (Lux)
Lineare Beschleunigung	H/S	Messung der Beschleunigungskräfte, die auf das Gerät einwirken in m/s^2 in alle Dimensionen (x, y, z).
Magnetfeld	H	Messung des Umgebungsmagnetfeldes in alle Dimensionen (x, y, z) in μT.
Orientierung	S	Messung der Grade der Drehung (Orientierung) des Gerätes.
Druck	H	Messung des Umgebungsluftdruckes in hPa oder mbar.
Annäherung	H	Messung des Abstandes eines Objektes in cm zum Bildschirm eines Gerätes.
Relative Feuchtigkeit	H	Messung der relativen Feuchtigkeit in Prozent.
Rotationsvektor	H/S	Messung der Orientierung eines Gerätes als 3D-Vektor.
Temperatur	H	Messung der Temperatur eines Gerätes in Grad Celsius.

Tabelle 1: Unterstützte Sensoren in der Android-Plattform

Android bietet zum Auslesen der Sensordaten ein eigenes Systemservice an, dass sich *SensorManager* nennt. Damit können die unterstützten Sensoren

aufgelistet werden, aber auch die Sensoren auf Ereignisse abgehorcht werden. Der Code in Listing 1 wurde der offiziellen Dokumentation entnommen und demonstriert das Auslesen eines LichtSensors:

Listing 1: Auslesen eines Lichtsensor in Android (Java)

```
public class SensorActivity extends Activity implements
    SensorEventAdapter {
     private SensorManager sensorManager;
     private Sensor mLight;

     @Override
     public final void onCreate(Bundle savedInstanceState) {
          super.onCreate(savedInstanceState);
          setContentView(R.layout.main);
          sensorManager = (SensorManager)
              getSystemService(Context.SENSOR_SERVICE);
          mLight =
              sensorManager.getDefaultSensor(Sensor.TYPE_LIGHT);
     }

     @Override
     public final void onSensorChanged(SensorEvent event) {
          // The light sensor returns a single value.
          // Many sensors return 3 values, one for each axis.
          float lux = event.values[0];
          // Do something with this sensor value.
     }

     @Override
     protected void onResume() {
          super.onResume();
          sensorManager.registerListener(this, mLight,
              SensorManager.SENSOR_DELAY_NORMAL);
     }

     @Override
     protected void onPause() {
          super.onPause();
          sensorManager.unregisterListener(this);
     }
}
```

Die Liste der Sensoren der iOS-Plattform [3] ist nachfolgend dargestellt.

Einige Sensoren betrachten dabei die Daten auf einer sehr abstrahierten Ebene:

Sensorart	**Beschreibung**
Beschleunigung	Messung der Beschleunigungskräfte.
Rotation	Messung der Rotation des Gerätes.
Licht	Messung des Hintergrundlichtes.
Position der Smart-Watch	Messung der Position einer SmartWatch am Handgelenk.
Schritte	Messung der Schritte des Benutzers.
Besuchte Orte	Daten über oft besuchte Orte des Benutzers.
Tastaturbenutzung	Daten über die Tastaturbenutzung des Benutzers.
Gerätenutzung	Daten über die Nutzung des Gerätes durch den Benutzer.
Nachrichtenappnutzung	Daten über die Nutzung der Nachrichten-Anwendung durch den Benutzer.
Telefonnutzung	Daten über die Menge an Zeit, die der Benutzer bei Telefonanrufen verbringt.

Tabelle 2: Unterstützte Sensoren in der iOS-Plattform

Das Verarbeiten der Sensordaten erfolgt dabei unter Einsatz des Core Motion Frameworks [2]. Es enthält unter anderem die Klasse *CMMotionManager* zum Starten und Managen von Bewegungsservices.

2.2 Erkennung durch Heuristik

Mit Heuristik ist gemeint, dass ein Bewertungsverfahren verwendet wird, dass durch Schätzen, Beobachten, Vermuten oder Raten versucht ein Problem zu Lösung (in diesem Fall: die Erkennung einer Geste). Es wird dabei nur eine ungefähre Lösung gefunden, die abhängig von der Bewertung ist. Desto besser das Problem bewertet werden kann, desto besser ist die Schätzung. Dieses Verfahren eignet sich für Gesten, da Gesten nicht eindeutig definiert beziehungsweise erkannt werden können. Dieses Verfahren wurde z.B. verwendet, um Gesten in Echtzeit zu erkennen [47], Handgesten [49][59] und Berührungsgesten zu erkennen [52] oder allgemein die Bewegungen von Menschen zu erkennen [22]. Ein einfacher Einsatz von Heuristik zur Gestenerkennung kann man beispielsweise in der Open-Source-Bibliothek Sensey sehen, mit der Gesten in Android erkannt werden können. Listing 2 beschreibt die Flip-Geste. Bei dieser Geste wird erkannt, ob der Bildschirm des Gerätes nach oben (face up) oder nach

unten ausgerichtet ist (face down). Es werden dabei nur die Daten der Z-Achse des Beschleuningungssensors benötigt:

Listing 2: Flip-Gestenimplementierung in Sensey

```
public class FlipDetector extends SensorDetector {

    public interface FlipListener {

        void onFaceDown();

        void onFaceUp();
    }

    private int eventOccurred;

    private final FlipListener flipListener;

    public FlipDetector(FlipListener flipListener) {
        super(TYPE_ACCELEROMETER);
        this.flipListener = flipListener;
        this.eventOccurred = 0;
    }

    @Override
    protected void onSensorEvent(SensorEvent sensorEvent) {
        float z = sensorEvent.values[2];
        if (z > 9 && z < 10 && eventOccurred != 1) {
            eventOccurred = 1;
            flipListener.onFaceUp();
        } else if (z > -10 && z < -9 && eventOccurred != 2)
           {
            eventOccurred = 2;
            flipListener.onFaceDown();
        }
    }
```

2.3 Erkennung durch künstliche Intelligenz

Aufgrund der Fortschritte auf dem Gebieten des maschinellen Lernens, einem Teilgebiet der künstliche Intelligenz sowie anderen statistischen Techniken wird künstliche Intelligenz in den letzten Jahren auch eingesetzt, um Gesten zu erkennen. Nachfolgend werden einige Techniken und deren Forschungsarbeiten

erwähnt, die künstliche Intelligenz zur Erkennung benutzt haben.

Bei der Gestenerkennung durch Hauptkomponentanalyse [34] [18], bei der versucht wird, Datensätze zu vereinfachen, indem die Menge an statistischen Variablen (wie z.B. mehrdimensionale Sensordaten) auf eine geringe Anzahl aussagekräftiger Linearkombination (Hauptkomponenten) angenähert werden. Ein anderer Ansatz ist die Stützvektormethode [17](englisch: Support Vector Machine [16]), bei der in den Daten durch mathematische Verfahren Muster erkannt werden. Die Daten werden dabei so in Klassen unterteilt, dass möglichst große Freiräume zwischen den Bereichen frei bleiben. Diese Art der Klassifizierung wird auch Breiter-Rand-Klassifikator genannt.

Es können auch verschiedene schwächere Klassifizier zu einem guten Klassifizier kombiniert werden. Diese Technik wird Boosting[60] genannt und kann auch zur Gestenerkennung verwendet werden [23].

Auch der Einsatz von Deep Learning [68] ist möglich [51]. Dabei werden aus dem Gebiet des maschinellen Lernens künstliche neurale Netze verwendet, bei der zwischen der Eingabe- und Ausgabeschicht eine beliebige Anzahl an versteckten Schichten verwendet wird. In [51] und [10] werden Convolutional Neural Networks verwendet, die im Vergleich zu normalen neuronalen Netzen als Eingabe Matrixen verarbeiten können.

Es kann auch das Klassifikations -und Regressesionsverfahren Random Forest [72] verwendet werden, bei der eine Menge von Entscheidungsbäume kombiniert werden. Entscheidungsbäume enthalten formale Regeln zum Klassifizieren von Objekten. Jeder dieser Bäume führt eine Klassifizierung durch. Die Klasse, die am Häufigsten klassifiziert wurde, wird als finale Klassifizierung verwendet. In Kombination mit der linearen Diskriminanzanalyse[20] wurde eine Gestenerkennung auch bereits umgesetzt [58]. Dabei wird eine Trennfunktion (Diskriminanzfunktion) verwendet, die bei der Analyse jeder Beobachtung eine Wertung zuordnet. Aus der Wertung wird die Gruppe bestimmt sowie deren Abgrenzung zu anderen Gruppen. Die bekannteste Trennfunktion ist die Fisher'sche Diskriminanzfunktion.

Auch der Einsatz von k-Nächste-Nachbarn-Klassifikation[53] ist möglich [45]. Dabei werden Trainingsbeispiele abgespeichert und bei der Klassifikation die nähesten k Nachbarn(Auswahl durch eine Distanzfunktion) betrachtet. Die häufigste Klassifikation wird als finale Klassifikation verwendet.

Ebenfalls kann das Problem durch Clusterbildung[55] mit Hilfe z.B. des k-Means-Clustering-Algorithmus[44] gelöst werden. Dabei werden aus einer Menge von Objekten k Gruppen mit ähnlichen Eigenschaften gebildet. Es ist die schnellste Technik zur Gruppierung von Objekten.

2.4 Berührungsgesten

Berühungsgesten sind Gesten, bei der der Benutzer mit einen berührungsfähigen Bildschirm interagiert und dabei Gesten auslöst. Die hier vorgestellten Gesten orientieren sich an der Android-Plattform [27] und den Gesten der iOS-Plattform, die in den User Interaction Guidelines der Entwicklerdokumentation beschrieben sind [5]. Die beiden Plattformen verwenden zum Großteil die selben Gesten. Die Liste der Gesten und deren Anwendungen basiert auf einer Referenztabelle[36]. Die Anwendungsgebiete sind teilweise abhängig von der Aktualität der Software und kann auch nicht vollständig sein. Funktionen in speziellen Modi wie VoiceOver oder TalkBack werden auch nicht aufgelistet.

Tippen wird meist zur Aktivierung von Elementen verwendet.

Doppel-Tippen wird zum Heranzoomen in Apps verwendet.

Dreifach-Tippen wird bei Android zum Wechsel zwischen vergrößerter und nicht vergrößerter Ansicht verwendet.

Zwei-Finger-Tippen wird zum Heranzoomen in Apps verwendet.

Geteiltes Tippen durch zeitverzögertes Tippen mit zwei Fingern wird in iOS für eine VoiceOver-Funktion verwendet.

Geteiltes Doppel-Tippen durch zeitverzögertes Doppel-Tippen mit zwei Fingern wird in Android für eine VoiceOver-Funktion verwendet.

Ziehen(drag) wird für Scrollen und Verschieben verwendet.

Mit zwei Fingern ziehen wird für Scrollen und Verschieben im Zoom-Modus verwendet.

Mit drei Fingern ziehen wird in iOS zum Verschieben des vergrößerten Ausschnitts verwendet.

Mit drei Fingern nach oben wischen wird in iOS zum Verschieben des vergrößerten Ausschnitts verwendet.

Wischen (flick) wird zum horizontalen Scrollen verwendet.

Vertikales Wischen wird zum vertikalen Scrollen und anderen ortsabhängigen Funktionen verwendet.

Vertikales Zwei-Finger-Wischen wird nur in speziellen Modi verwendet.

Länger drücken wird unter iOS zur Cursor-Positionierung und unter Android zur Bearbeitsstartung und Textauswahl verwendet.

Zusammenziehen wird zum Herauszoomen in Apps verwendet.

Spreizen wird zum Hineinzoomen in Apps verwendet.

Dreifach Tippen mit drei Fingern wird nur in speziellen Modi verwendet.

Drehgeste mit zwei Fingern wird zum Rotierung von Objekten oder Bildern in manchen Apps verwendet.

Schütteln wird unter iOS zum Rückgängigmachen verwendet.

3 Gesten-Navigation

Gesten-Navigation wird vermehrt im mobilen Bereich verwendet. Daher werden hier Android und iOS als Fallstudien verwendet.

3.1 Android

Unter Android kann man seit Android 10 aus drei verschiedenen Navigationsvarianten auswählen [26]. Bei der klassischen 3-Tasten-Navigation gibt eine Home-Bildschirm-Taste, eine Zurück-Taste sowie eine Taste, die den Benutzer zur Anwendungsübersicht führt. Bei der 2-Tasten -Navigation fällt die zuletzt genannte Taste weg. Bei der Gesten-Navigation wird keine Taste mehr verwendet Um zwischen Bildschirmen, Webseiten und Apps zu wechseln, kann von der linken bzw. rechten Ecke des Bildschirms gewischt werden. Um zum Home-Bildschirm zu gelangen, muss eine Wisch-Weste vom unteren Ende des Bildschirms nach oben ausgeführt werden. Durch eine Wischbewegung vom unteren Rand des Gerätes nach oben sowie halten und loslassen der Geste, werden alle offenen Apps angezeigt. Ein Wechsel der Apps ist durch eine Wischbewegung von links nach rechts am unteren Bildschirmrand möglich. Alle bereits erwähnten Berührungsgesten können ebenfalls zur Navigation verwendet werden. Das Kontrollzentrum kann durch eine Wischbewegung nach unten vom oberen Rand angezeigt werden.

3.2 iOS

Gesten-Navigation ist auch unter der iOS-Plattform seit iOS 13 möglich[4]. Zusätzlich wird hier Face ID zur Authentifizierung verwendet. Die nachfolgende Beschreibung bezieht sich auf das iPhone. Wie bei Android erfolgt das Ein- und Ausschalten durch Tasten am Smartphone. Das Aufwecken des Gerätes aus dem Schlafmodus kann allerdings erreicht werden, indem das Gerät angetippt

oder angehoben wird. Das Entsperren eines iPhones mit Face ID erfolgt durch einen Blick auf das Gerät und anschließender Wisch-Bewegung vom unteren Rand des Bildschirms. Auch die Gesten zum Aufwecken können in Kombination mit Face ID verwendet werden. Der Home-Bildschirm kann erreicht werden, indem vom unteren Rand des Bildschirms nach oben gewischt wird. Durch die selbe Wischbewegung innerhalb eines Apps und darauffolgender Wischbewegung nach rechts am unteren Rand des Bildschirms ist ein Wechsel zu einem anderen App möglich. Wie auch bei Android können Minianwendungen (Widgets) durch eine Wischbewegung nach rechts am Home-Bildschirm angezeigt werden. Durch eine Wischbewegung von der Mitte des Bildschirms nach unten, wird eine Suchfunktion aufgerufen. Das Kontrollzentrum kann durch eine Wischbewegung nach unten vom oberen rechten Rand angezeigt werden. Die Benachrichtigungszentrale kann durch eine Wischbewegung vom oberen Rand nach unten geöffnet werden.

4 Proximic Interactions

4.1 Definition

Ein verwandtes Thema, in dem Gesten eingesetzt werden, ist Proximic Interactions. Es geht hierbei darum, das Konzept der Proxemik auf Technologie anzuwenden, die dann in alltägliche Systeme integriert wird. Die Art und Weise, wie Geräte in unser alltägliches Leben integriert werden, wird als *ubiquitious Computing* bezeichnet. Das Thema proxemische Interaktionen basiert auf der Arbeit des amerikanischen Anthropologen Edward Twitchell Hall Jr [30], der beschrieb, wie Menschen reagieren, wenn Interaktionen in unterschiedlichen Entfernungen stattfinden, und der einige Faktoren wie Geschlecht, Kultur usw. beschrieb, die diese beeinflussen.

Hall arbeitete am Thema Proxemik und schlug ursprünglich acht Dimensionen vor, um es zu kategorisieren:

- Haltung
- Geschlecht
- Relative Körperorientierung
- Berührung
- Retina-Kombinationen
- Wärme

- Riechwahrnehmung
- Lautheit der Stimme

In der Praxis sind die folgenden fünf Dimensionen am wichtigsten: Entfernung, Orientierung, Identität, Bewegung und Standort.

Ubiquitious Computing ist eng mit proxemischen Interaktionen verbunden und bedeutet, dass wir in unserem Alltag von Technologie umgeben sind und diese so eng in unser Leben integriert ist, dass wir sie tatsächlich nicht unterscheiden können. Dies wird auch als "verkörperte Interaktion"[19] bezeichnet. Während die meisten modernen Geräte dies zu bieten scheinen, sind nur wenige tatsächlich in der Lage, irgendwelche proxemischen Dimensionen zu erkennen, die zum Alltag des Menschen gehören. Die Implementierung von proxemischer Interaktion in Ubicomp-Systemen würde in vielen Situationen als wertvoller Input dienen, um die Interaktion von Menschen mit diesen Geräten zu vermitteln [29]. Viele Systeme beinhalten räumliche Beziehungen zwischen Menschen und Geräten, wenn auch sehr einfache. Gängige Beispiele sind moderne Mobiltelefone, die ihre Bildschirme abschalten, wenn der Benutzer nicht auf das Gerät schaut. Solche Systeme interpretieren die Eingaben in der Regel auf einer groben Ebene, während sie andere relevante Dimensionen wie die Identität des Betrachters völlig außer Acht lassen.

4.2 Fallstudien

4.2.1 Interaktives Whiteboard

[37] haben ein interaktives Whiteboard mit Hilfe von C, dem Microsoft Table PC SDK und dem SMART Board SDK entwickelt. Es kann mehrere Personen erkennen und handelt anhand der proximalen Zonen:

Persönliche Zone in der sich Personen an der Tafel befinden, auf sie zeigen und Textmanipulationen vornehmen können.

Soziale Zone in der die Tafel aufgrund der Entfernung nicht berührt werden kann, so dass sie nur zum Betrachten verwendet wird.

Öffentliche Zone die von er Entfernung weiter weg ist als die soziale Zone.

4.2.2 Die proxemische Media-Player-Anwendung

Die Anwendung von [9] reagiert auf sich nähernde Menschen und deren Orientierung (implizit). Die Interaktionen der Anwendung können für unterschiedliche Entfernungen konfiguriert werden: Ein Mobiltelefon kann als Zeiger auf

den Fernseher fungieren (als mobiler Token). Videos werden durch Gesten (Schnippen der Hand) ausgewählt. Der Mediaplayer pausiert in verschiedenen Situationen: wenn jemand ans Telefon geht, wegschaut oder wenn der Fernseher blockiert ist. Die Anwendung erkennt auch den festen Funktionsraum. So kann sie den Titel des gerade laufenden Mediums anzeigen, wenn eine neue Person den Raum betritt. Auch andere Geräte können als mobile Token fungieren und den Player steuern.

5 Anwendungsgebiete

Abgesehen von den Plattformen Android und iOS gibt es auch andere Anwendungsgebiete von Gesten, von denen in diesem Kapitel einige vorgestellt werden sollen.

[39] haben ein System entwickelt um Gesten an einem mobilen Roboter zu erkennen und interpretieren. Das System ist ein dreidimensionales Echtzeit-Gestenerkennungssystem, dass sich an Bord eines mobilen Roboters befindet und zwischen sechs verschiedenen menschlichen Gesten unterscheiden kann, die durch Armbewegungen gekennzeichnet sich. Die Gesten wurden mit Hilfe Schwarzweißbilder einer Kamera erkannt. [62] haben einen Algorithmus entwickelt, der auf maschinellen Lernen basiert, um Gesten zu erkennen, die in der Luft ausgeführt werden. Dazu wird eine RGB-Kamera benötigt, die in gängigen Smartphones eingebaut ist. Der Algorithmus läuft in Echtzeit und auch in schlechten Lichtbedingungen. [71] entwickelten einen anderen Ansatz, der sich Sideswipe nennt und stattdessen das GSM-Signal zur Erkennung der Gesten verwendet. [57] haben eine Technik entwickelt, bei der Benutzer selbst Bewegungen aufzeichnen können, die dann als Gesten erkannt werden. Dabei können Berührungsgesten erstellt werden als auch Gesten die von gängigen Sensoren wie z.B. Beschleuningungssensoren, Gyroskopen und Orientierungssensoren abhängig sein können. [40] erweiterten das Konzept der Benutzer erstellten Gesten um die Möglichkeit, damit mobile Telefone, öffentliche Anzeigen und Tabletops zu verbinden.

[21] haben eine mobile Authentifikationsmethode entwickelt, die sich FAST nennt und bei der Berührungsgesten verwendet werden. Es wurde festgestellt, dass bei 40 Benutzern, die die Technik verwendeten, die Falschakzeptanzrate bei 4.66% und die Falschrückweisungsrate bei 0.13% lag. [61] entwickelten ebenfalls eine Authentifikationsmethode, bei der allerdings unbewusste Gesten des Benutzers zur Authentifikation verwendet werden. [48] haben eine ähnliche Technik entwickelt.

[35] entwickelten eine neue Art von Gesten, die sich Whack-Gesten nennen. Dies sind Gesten, die sehr ungenau sind und für unaufmerksame Interaktionen mit mobilen Geräten gedacht sind. Benutzer können mit Hilfe von Whack-Gesten interagieren, indem sie ein Gerät mit der offenen Handfläche oder dem Handballen schlagen.

[38] nutzen Blickgesten als Eingabemethode für mobile Geräte und vibrotaktiles Feedback (Vibration) als alternative Methode zur Bestätigung von Interaktionseventen. Sie kamen zum Schluss, dass Vibration als Feedback für Blickgesten sehr nützlich ist, da Aufgaben dadurch schneller, leichter und bequemer abgeschlossen werden konnten.

[41] entwarfen ein PaperPhone. Dies ist ein mobiles Gerät mit flexiblem elektronischen Papierdisplay, bei dem Biegegesten erkannt werden können.sind

[70] erfanden LensGesture, eine Gestenerkennung bei der Gesten auf der Rückseite des Smartphones erkannt werden. Dies funktioniert, indem volle und halbe Verdeckungen sowie dynamische Wischbewegungen der Finger auf der Kameralinse erkannt wurden. Dazu wurden durch die eingebaute Kamera Bildersequenzen in Echtzeit analysiert. [64] ermöglichten das Erkennung von Gesten an der Oberfläche von Geräten mit Hilfe von akustischen Signalen.

[33] haben analysiert, ob sich Tritt-Gesten (Fuß) für mobile Anwendungen eignen. Sie kamen zum Schluss, dass Benutzer ihre Tritte am Besten zielen konnten, wenn Tritt-Gesten(Fuß) der Bewegungsbereich in Segmente von mindestens 24° unterteilt wurde. Sie kamen auch zum Schluss, dass Benutzer mit mindestens zweit verschiedenen Geschwindigkeiten treten können.

[54] untersuchten die Möglichkeit, vom Benutzer eigene Gesten als Shortcut für grundlegende Funktionen des Smartphones zu verwenden.

6 Vorteile und Nachteile einer Gestensteuerung

Bei Mausgesten wurde festgestellt, dass bei normalen Elementen der Benutzeroberfläche wie Menüs und Buttons Gesten die Aktivität des Benutzers nicht stören, indem er die Hand zur Position des Befehls bewegen muss. Stattdessen können die Gesten an der Position des Cursors ausgeführt werden [11]. Dies gilt auch bei Gesten im mobilen Anwendungsbereich, da hier Gesten entweder an der Fingerposition oder an einer beliebigen Position auf den Bildschirm ausgeführt werden. Sie benötigen auch meist keine zusätzlichen Geräte. Alle notwendigen Sensoren sind bereits in modernen Smartphones eingebaut. Normale Eingabemethoden wie Touchscreens schränken den Benutzer auf wenige Interaktionen ein, wenn keine Gesten verwendet werden. Durch Einsatz von Gesten, die speziell von Sensoren erkannt werden, gibt es theoretisch beliebig

viele Gesten, die erkannt werden können.

[15] verglichen alternativer Techniken zur Präsentationssteuerung und kamen zum Schluss, dass Gesten gesteuerte Presentation nicht vom Redner als angenehmer, sondern auch vom Publikum als attraktiver wahrgenommen wurde. Die Präsentatoren konnten häufiger Blickkontakt herstellen und nonverbale Kommunikation zur Informationsvermittlung einsetzen. Aus dieser Studie kann man schließen, dass durch Gesten der Benutzer nicht so sorgfältig und aufmerksam sein muss um die gewollte Aktionen ausführen zu können.

Demgegenüber ist es schwierig für Benutzer, Gesten zu lernen, sich zu merken und korrekt auszuführen. Der Entwickler muss ein System bereitstellen, dass Gesten richtig erkennt. [24] fanden heraus, dass die Beobachtung von Gesten nicht ausreicht um sie zu lernen, da der Beobachter wichtige und unwichtige Bewegungen nicht unterschieden kann. Der Entwickler muss daher nicht nur für eine schnelle und korrekte Erkennung sorgen, sondern auch eine gute Anleitung erstellen, die das schnelle und einfach Erlernen der Gesten ermöglicht.

Das Lernen von Multi-Touch-Gesten, Gesten in der Luft und Sensor basierenden Gesten ist schwieriger als von einfachen Berührungsgesten, da die Handhaltung und/oder auch die Körperhaltung bei Letzteren unbedeutend sind. Bei der anderen Form der Gesten sind unter Umständen komplexe Bewegungen notwendig.

[12] erkannte schon 1993, dass Gesten weder sich selbst sichtbar machen noch selbsterklärend sind. Eine Schaltfläche mit einem Namen auf einer Symbolleisten hat einen expliziten Zweck und ist leicht zu finden, Gesten hingegen können willkürlich sein und sind meist schwieriger zu entdecken. Google und Apple haben daher für ihre Plattformen ein Menü bzw. ein App entworfen, dass dem Benutzer nützliche Tipps zur Benutzung des Betriebsystems gibt. Unter diesem Tipps befinden sich auch Tipps zur Nutzung von Gesten.

[11] erkannte, dass normale Befehle nur erkannt werden müssen, Gesten allerdings auch gemerkt werden müssen, bevor sie ausgeführt werden können. [67] erkannten, dass leicht merkbare Gesten erstellt werden können, wenn sie so intuitiv wie möglich gestaltetet werden. Durch werden sie leichter gemerkt. Auch ist die Intuitivität einer Geste abhängig von der Kultur und Erfahrung. Eine Geste, wie zum Beispiel das Nicken wird in unterschiedlichen Länder anders interpretiert. Eine Zoom-Geste, bei der mit zwei Finger der Bildschirm auseinander gezogen wird, ist z.B. auch nur natürlich für häufige Benutzer von Smartphones und nicht für Personen, die noch nie einen Touchscreen gesehen haben.

Laut [12] sind bei Gesten mehr Muskeln beteiligt als bei anderen Interak-

tionstechniken. Daher sollten Entwickler Gesten entwickeln, die einfach und bequem ausgeführt werden können.

Auch Erkennungsfehler dürfen nicht außer Acht gelassen werden. Obwohl im privaten Anwendungsbereich ein Erkennungsfehler meist keine große Auswirkung hat, muss bei sicherheitskritischen Anwendung die Fehlerrate beachtet werden. Die Falschakzeptanzrate muss dabei minimiert werden.

7 Software- und Hardwarelösungen zur Gestenerkennung

In diesem Kapitel werden zusätzlich zu den bereits erwähnten Methoden in Android und iOS Softwarebibliotheken vorgestellt, die in der Open Source Community entwickelt wurden. Bibliotheken wurden aufgrund der Anzahl an Sternen auf der populären Entwicklerseite Github[14] gewählt, die die Popularität einer Software widerspiegeln. Zusätzlich werden einige kommerzielle Produkte vorgestellt.

Sensey
: Sensey ist einer der populärsten Gestenbibliothek für Android. Sie unterstützt die folgenden Gesten: Flip, Light, Orientation, PinchScale, Proximity, Shake, Wave, Chop, WristTwist, Movement, SoundLevel, RotationAngle, TiltDirection, Scoop, PickupDevice, Steps, TouchType. Zu den einzelnen Gesten gibt es jeweils mehrere Events, für die eine Anwendung einen Listener registrieren muss.

Android Gesture Detectors Framework[1]
: Dieses Framework enthält einige Basisklassen zum Erweitern des Frameworks sowie einige nützliche Klassen zur Gestenerkennung: *RotateGestureDetector* erkennt den Rotationswinkel der Linie zwischen zwei Fingern. *MoveGestureDector* ist eine Gestenerkennung, die hilft, Objekte mit ein oder mehreren Finger zu bewegen. *ShowGestureDetector* erkennt eine vertikale Stubs-Bewegung mit zwei Fingern.

Swipe Back[46] ist eine Bibliothek, die es erlaubt, Aktivitäten in Android mit einer Wisch-Geste zu beenden. Es werden dabei die folgenden Richtungen unterstützt: links, oben, rechts und unten.

NewHandwave[65] ist eine Reimplementierung der Bibliothek Handwave, die es erlaubt, mit Hilfe der Frontkamera Gesten zu erkennen, die vor dem Bildschirm ausgeführt werden,

Gesture Recognition Closures[8] benutzt anonyme Funktionen (Closures) um die Arbeit mit Gestenerkennern unter iOS zu erleichtern. Es werden dabei alle gängige Recognizer unterstützt.

Sensitive[42] vereinfacht ebenfalls die Arbeit mit Gesten unter iOS.

Nanogest[50] ist eine kommerzielle Gestenbibliothek für iOS und Android. Sie verwendet die Frontkamera und kann 4 Wischgesten sowie Handwinken erkennen. Sie kann auch in schlechten Licht bei komplexen Hintergründen verwendet werden.

Vision[6] ist ein Framework der Firma Apple, mit dem Algorithmen von Computer Vision auf Bilder und Videos angewandt werden können. Es können z.B. Gesichter und deren Merkmale, Texte oder Barcodes erkannt werden und auch Bildregistrierungen und allgemeine Merkmalsverfolgungen umgesetzt werden. Auf der Apple Worldwide Developers Conference 2020 wurde gezeigt, dass mit diesem Framework auch Körper- und Handstellungen erkannt werden können [7].

MediaPipe[28] ist eine Framework von Google, dass verschiedene Lösungen im Bereich des maschinellen Lernens anbietet. Auf der Conference on Computer Vision and Pattern Recognition wurde 2019 ihre eigene Gestenerkennungstechnologie für die Open-Source-Community veröffentlicht und in MediaPipe integriert. Das Framework kann unter anderem Gesichter und deren Strukturen, die Iris im Auge, Handposten und holistische Erkennungen durchführen.

Ultrahaptics[66] bietet eine Palette von Produkten zum Hand-Tracking sowie haptischen Technologien an. Seit 2019 bieten sie unter den Namen Ultraleap ein Hand-Tracking-Produkt an, dass früher von der Firma LeapMotion produziert wurde. Ihr Hauptprodukt ist der Leap Motion Controller, der optisch Handbewegungen fast ohne Latenz millimetergenau erkennen kann. Sie bieten auch ein zweites Produkt an, dass sich Stereo IR 170 (früher: Rigel) nennt, um das ein größeres Sichtfeld, eine größere Tracking-Distanz sowie eine smallere Form gegenüber dem Leap Motion Controller besitzt.

Kinect[31] war eine Hardware zur Steigerung von Videospielkonsolen der Firma Microsoft. Sie wurde zusammen mit der Firma PrimaSense entwickelt und wurde 2017 eingestellt. In diversen Arbeiten [13] [56] [31] wurde gezeigt, dass sich diese Hardware auch zur Erkennung von Gesten eignet.

8 Beispielprojekt für Gestenerkennung

Es soll ein einfaches App für Android entwickelt werden, dass als Abschreckung für fremde Personen dienen soll. Wenn das Smartphone bewegt wird, soll eine Warnmeldung angezeigt werden und das Smartphone fängt an zu vibrieren. Dieses App soll nur als Demonstration dienen und nicht als Sicherheitsanwendung gesehen werden. Es gibt einige Vorraussetzungen für dieses Projekt:

- Java
- Android Studio
- Android SDK Tools
- Android Emulator oder ein echtes Smartphone mit Android als Betriebsystem

Für die Gestenerkennung wird die Bibliothek Sensey verwendet, die zuerst wie in Listing 3 importiert werden muss.

Listing 3: Import von Sensey

```
implementation 'com.github.nisrulz:sensey:{latest version}'
```

Wie in Listing 4 gezeigt wird, wird Sensey in onCreate() in der Aktivität initialisiert, sowie der *MovementListener* registriert.

Listing 4: Initialisierung von Seney

```
Sensey.getInstance().init(context);
MovementListener listener = new MovementListener {
        void onMovement() {
            showWarning();
        }
        void onStationary() {}
    }
Sensey.getInstance().startMovementDetection(listener)
```

Die Warnung wird dabei wie in Listing 5 erzeugt.

Listing 5: Warndialog

```
void showWarning() {
new AlertDialog.Builder(this)
    .setTitle("Warnung")
    .setIcon(android.R.drawable.ic_dialog_alert)
    .setMessage("Unerlaubter Zugriff")Die Warnung wi
    .setPositiveButton("Ok", new
        DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int
                which) {

            }
     }).show();
}
```

Damit das Smartphone vibriert, muss die Vibration mit dem Vibrationsservice wie in Listing 6 gestartet werden.

Listing 6: Vibration

```
void vibrate() {
    Vibrator v = (Vibrator)
        getSystemService(Context.VIBRATOR_SERVICE);
    // Vibrate for 500 milliseconds
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
        v.vibrate(VibrationEffect.createOneShot(500,
            VibrationEffect.DEFAULT_AMPLITUDE));
    } else {
        //deprecated in API 26
        v.vibrate(500);
    }
}
```

Es muss zusätzliche eine Erlaubnis für das Service in der Manifest-Datei wie in Listing 7 gesetzt werden.

Listing 7: Berechtigung für Vibration

```
<uses-permission android:name="android.permission.VIBRATE"/>
```

Sensey wird in onDestroy() gestoppt und der Listener wird wie in Listing 8 entfernt.

Listing 8: onDestroy-Methode

```
Sensey.getInstance().stopMovementDetection(listener)
Sensey.getInstance().stop();
```

Es wird empfohlen, eine Permission für den verwendeten Sensor in der Manifest-Datei anzugeben, damit im Google Play Store bei Smartphones, die diesen Sensor nicht unterstützen, dieses App nicht angeboten wird. Der Code ist in Listing 9 dargestellt.

Listing 9: Berechtigung für Beschleuningssensor

```
<uses-feature
    android:name="android.hardware.sensor.accelerometer"
    android:required="true" />
```

9 Fazit

Gesten sind aus dem mobilen Bereich seit dem Einsatz in Smartphones schon lange nicht mehr wegzudenken. Anfangs gab es nur einfache Gesten wie Wisch-Gesten, die am Bildschirm ausgeführt wurden, heutzutage können Benutzer schon selbst Gesten aufnehmen, die sie selbst durch Körperbewegungen definieren.

Um Gesten zu erkennen zu können, müssen Sensoren verwendet werden, deren Daten dann durch Software ausgewertet werden. Früher wurde die Erkennung von Gesten mit Hilfe heuristische Techniken gelöst. In den letzten Jahren, in denen künstliche Intelligenz immer populärer wurde, werden vermehrt auch Techniken des maschinellen Lernens eingesetzt, um Gesten genauer zu erkennen.

Berührungsgesten haben Eingang in sämtliche mobile Betriebssysteme gefunden, darunter besonders Android und iOS. Bei Smartphones wurden mit der Zeit die Anzahl der physischen Tasten reduziert und/oder durch Softwaretasten ersetzt. In den letzten Jahren wurde die Gesten-Navigation immer stärker eingesetzt, sodass viele Funktionen bei Smartphones bereits ohne Tasten aufgerufen werden können.

Ein eng verwandtes Thema ist Proximic Interactions, bei der es um Interaktionen zwischen Mensch und Maschine geht, bei der der räumliche Abstand eine Rolle spielt. Speziell Gesten, die die Annäherung betreffen spielen sind hier sehr wichtig.

Auch andere Formen von Gesten wurden in den letzten Jahren erforscht wie zum Beispiel Trittgesten oder Blickgesten, die in der Zukunft womöglich bei mobilen Anwendungen eingesetzt werden.

Wie sich herausgestellt hat, hat eine Gestensteuerung Vorteile und Nachteile, wobei die Vorteile überwiegen. Trotzdem ist auf dem Gebiet noch einiges an Forschungsbedarf, da sich die meisten Studien nur auf Berührungsgesten beziehen.

Auf der Softwareseite gibt es unter Android einige freie Bibliotheken, die weitere Gesten erkennen können zusätzlich zu den Gesten, die bereits von der Android-Plattform selbst erkannt werden können. Unter iOS ist man mehr an die offizielle Plattform gebunden, Bibliotheken können den Umgang mit Gesten nur vereinfachen.

Anhand vom einem Beispielprojekt konnte man schlussendlich erkennen, dass der Umgang mit Gesten unter Android sehr leicht ist.

10 Zukünftige Entwicklungen

Mit dem Aufschwung von virtueller und erweiterter Realität ist auch das Erkennen von Gesten sehr wichtig. Damit Gesten gut erkannt werden, müssen auch Bewegungen in den virtuellen Welt genau erfasst werden. [32] haben deswegen ein System zum Hand-Tracking in Echtzeit entwickelt, bei dem vier Fischaugenkameras verwendet werden. Auch für andere Körperteile müssen bessere Tracking-Methoden entwickelt werden.

Auch das Internet der Dinge, bei dem physische und virtuelle Gegenstände miteinander vernetzt werden und miteinander kommunizieren oder Befehle entgegennehmen, ist ein zukünftiges Anwendungsgebiet für Gesten, da dann mit den Objekten intuitiv und einfach interagiert werden kann.

Das Erkennen von Gesten wird auch weiterhin ein Forschungsthema sein. Es gibt bereits seit Längerem kommerzielle Handverfolgungsmodule wie Leap Motion[1]. Eine Herausforderung ist es, diese Daten in virtuelle Steuerungen umzuwandeln. Google [43] arbeitet bereits seit einigen Jahren an so einem Projekt, bei dem ein kleiner Chip schnell und auf Millimeter genau die Bewegungen verfolgt.

Der Einsatz von neuen Gesten wie z.B. Whack-Gesten, Blick-Gesten etc. muss ebenfalls in echten Anwendungen erprobt werden um herauszufinden ob diese Gesten von den Benutzern angenommen werden.

Schlussendlich stellt sich auch noch die Frage: Wenn es in der Zukunft so viele neue Gesten geben wird, wie werden die Benutzer all diese Gesten richtig lernen und sich merken? Diese Frage ist bereits ein Problem, da ein normaler Smartphonebenutzer schon jetzt nicht alle Gesten seines Smartphones kennt.

[1] www.ultraleap.com

Listings

Tabellenverzeichnis

Literatur

[1] User "Almeros". Android gesture detectors framework. https://github.com/Almeros/android-gesture-detectors.

[2] Apple. Core motion - apple developer documentation, . URL https://developer.apple.com/documentation/coremotion.

[3] Apple. Srsensor - apple developer documentation, . URL https://developer.apple.com/documentation/sensorkit/srsensor#3681604.

[4] Apple. Use gestures to navigate your iphone with face id - apple support, . URL https://support.apple.com/en-us/HT208204.

[5] Apple. Gestures - user interaction - ios - human interface guidelines, . URL https://developer.apple.com/design/human-interface-guidelines/ios/user-interaction/gestures/.

[6] Apple. Vision | apple developer documentation. https://developer.apple.com/documentation/vision, .

[7] Apple. Detect body and hand pose with vision - wwdc 2020 - videos - apple developer. https://developer.apple.com/videos/play/wwdc2020/10653/, .

[8] Marc Baldwin. Gesture recognizer closures. https://github.com/marcbaldwin/GestureRecognizerClosures.

[9] Till Ballendat, Nicolai Marquardt, and Saul Greenberg. Proxemic interaction: designing for a proximity and orientation-aware environment. In *ACM International Conference on Interactive Tabletops and Surfaces*, pages 121–130, 2010.

[10] Igor LO Bastos, Victor HC Melo, and William Robson Schwartz. Multi-loss recurrent residual networks for gesture detection and recognition. In *2019 32nd SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, pages 170–177. IEEE, 2019.

[11] Olivier Bau and Wendy E Mackay. Octopocus: a dynamic guide for learning gesture-based command sets. In *Proceedings of the 21st annual ACM symposium on User interface software and technology*, pages 37–46, 2008.

[12] Thomas Baudel and Michel Beaudouin-Lafon. Charade: remote control of objects using free-hand gestures. *Communications of the ACM*, 36(7): 28–35, 1993.

[13] Kanad K Biswas and Saurav Kumar Basu. Gesture recognition using microsoft kinect®. In *The 5th international conference on automation, robotics and applications*, pages 100–103. IEEE, 2011.

[14] Hudson Borges and Marco Tulio Valente. What's in a github star? understanding repository starring practices in a social coding platform. *Journal of Systems and Software*, 146:112–129, 2018.

[15] Xiang Cao, Eyal Ofek, and David Vronay. Evaluation of alternative presentation control techniques. In *CHI'05 Extended Abstracts on Human Factors in Computing Systems*, pages 1248–1251, 2005.

[16] Corinna Cortes and Vladimir Vapnik. Support vector machine. *Machine learning*, 20(3):273–297, 1995.

[17] Nasser H Dardas and Nicolas D Georganas. Real-time hand gesture detection and recognition using bag-of-features and support vector machine techniques. *IEEE Transactions on Instrumentation and measurement*, 60 (11):3592–3607, 2011.

[18] Nasser H Dardas and Emil M Petriu. Hand gesture detection and recognition using principal component analysis. In *2011 IEEE International Conference on Computational Intelligence for Measurement Systems and Applications (CIMSA) Proceedings*, pages 1–6. IEEE, 2011.

[19] Paul Dourish. *Where the Action is: 24510Where the Action is: The Foundations of Embodied Interaction*. MIT Press, 2001.

[20] Hans-Friedrich Eckey, Reinhold Kosfeld, and Martina Rengers. Diskriminanzanalyse. In *Multivariate Statistik*, pages 289–390. Springer, 2002.

[21] Tao Feng, Ziyi Liu, Kyeong-An Kwon, Weidong Shi, Bogdan Carbunar, Yifei Jiang, and Nhung Nguyen. Continuous mobile authentication using touchscreen gestures. In *2012 IEEE conference on technologies for homeland security (HST)*, pages 451–456. IEEE, 2012.

[22] Simon Fong, Justin Liang, Iztok Fister, and Sabah Mohammed. Gesture recognition from data streams of human motion sensor using accelerated pso swarm search feature selection algorithm. *Journal of Sensors*, 2015, 2015.

[23] Hardy Francke, Javier Ruiz-del Solar, and Rodrigo Verschae. Real-time hand gesture detection and recognition using boosted classifiers and active

learning. In *Pacific-Rim Symposium on Image and Video Technology*, pages 533–547. Springer, 2007.

[24] Dustin Freeman, Hrvoje Benko, Meredith Ringel Morris, and Daniel Wigdor. Shadowguides: visualizations for in-situ learning of multi-touch and whole-hand gestures. In *Proceedings of the ACM international conference on interactive tabletops and surfaces*, pages 165–172, 2009.

[25] Google. Sensors overview - android developers, . URL `https://developer.android.com/guide/topics/sensors/sensors_overview`.

[26] Google. Get around on your android phone - android help, . URL `https://support.google.com/android/answer/9079644`.

[27] Google. Use touch gestures - android developers, . URL `https://developer.android.com/training/gestures`.

[28] Google. Mediapipe. `https://github.com/google/mediapipe`, .

[29] Saul Greenberg. Opportunities for proxemic interactions in ubicomp (keynote). In *IFIP Conference on Human-Computer Interaction*, pages 3–10. Springer, 2011.

[30] Edward T Hall. A system for the notation of proxemic behavior 1. *American anthropologist*, 65(5):1003–1026, 1963.

[31] Jungong Han, Ling Shao, Dong Xu, and Jamie Shotton. Enhanced computer vision with microsoft kinect sensor: A review. *IEEE transactions on cybernetics*, 43(5):1318–1334, 2013.

[32] Shangchen Han, Beibei Liu, Randi Cabezas, Christopher D Twigg, Peizhao Zhang, Jeff Petkau, Tsz-Ho Yu, Chun-Jung Tai, Muzaffer Akbay, Zheng Wang, et al. Megatrack: monochrome egocentric articulated hand-tracking for virtual reality. *ACM Transactions on Graphics (TOG)*, 39 (4):87–1, 2020.

[33] Teng Han, Jason Alexander, Abhijit Karnik, Pourang Irani, and Sriram Subramanian. Kick: investigating the use of kick gestures for mobile interactions. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services*, pages 29–32, 2011.

[34] René Henrion and Günter Henrion. Hauptkomponentenanalyse. In *Multivariate datenanalyse*, pages 10–43. Springer, 1995.

[35] Scott E Hudson, Chris Harrison, Beverly L Harrison, and Anthony LaMarca. Whack gestures: inexact and inattentive interaction with mobile devices. In *Proceedings of the fourth international conference on Tangible, embedded, and embodied interaction*, pages 109–112, 2010.

[36] Incobs. Touch-gesten | incobs. URL `https://www.incobs.de/gesten.html`.

[37] Wendy Ju, Brian A Lee, and Scott R Klemmer. Range: exploring implicit interaction through electronic whiteboard design. In *Proceedings of the 2008 ACM conference on Computer supported cooperative work*, pages 17–26, 2008.

[38] Jari Kangas, Deepak Akkil, Jussi Rantala, Poika Isokoski, Päivi Majaranta, and Roope Raisamo. Gaze gestures and haptic feedback in mobile devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 435–438, 2014.

[39] David Kortenkamp, Eric Huber, R Peter Bonasso, et al. Recognizing and interpreting gestures on a mobile robot. In *Proceedings of the National Conference on Artificial Intelligence*, pages 915–921, 1996.

[40] Christian Kray, Daniel Nesbitt, John Dawson, and Michael Rohs. User-defined gestures for connecting mobile phones, public displays, and tabletops. In *Proceedings of the 12th international conference on Human computer interaction with mobile devices and services*, pages 239–248, 2010.

[41] Byron Lahey, Audrey Girouard, Winslow Burleson, and Roel Vertegaal. Paperphone: understanding the use of bend gestures in mobile devices with flexible electronic paper displays. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1303–1312, 2011.

[42] User "LetsMakeItSwift". Sensitive. `https://github.com/LetsMakeItSwift/Sensitive`.

[43] Jaime Lien, Nicholas Gillian, M Emre Karagozler, Patrick Amihood, Carsten Schwesig, Erik Olson, Hakim Raja, and Ivan Poupyrev. Soli: Ubiquitous gesture sensing with millimeter wave radar. *ACM Transactions on Graphics (TOG)*, 35(4):1–19, 2016.

[44] Aristidis Likas, Nikos Vlassis, and Jakob J Verbeek. The global k-means clustering algorithm. *Pattern recognition*, 36(2):451–461, 2003.

[45] Yanqiu Liu, Xiuhui Wang, and Ke Yan. Hand gesture recognition based on concentric circular scan lines and weighted k-nearest neighbor algorithm. *Multimedia Tools and Applications*, 77(1):209–223, 2018.

[46] User "liuguangqiang". Swipeback. `https://github.com/liuguangqiang/SwipeBack`.

[47] Omar Lopez-Rincon and Oleg Starostenko. Real time gesture recognition with heuristic-based classification. In *Mexican Conference on Pattern Recognition*, pages 135–144. Springer, 2016.

[48] Yuxin Meng, Duncan S Wong, Roman Schlegel, et al. Touch gestures based biometric authentication scheme for touchscreen mobile phones. In *International conference on information security and cryptology*, pages 331–350. Springer, 2012.

[49] Hazem Khaled Mohamed, S Sayed, Hossam Ali, et al. Hand gesture recognition using average background and logical heuristic equations. *INTERNATIONAL JOURNAL OF COMPUTERS & TECHNOLOGY*, 11 (5):2634–2640, 2013.

[50] Nanocritical. Nanogest: Air gestures for ios and android – nanocritical corp. `http://www.nanocritical.com/nanogest`.

[51] PS Neethu, R Suguna, and Divya Sathish. An efficient method for human hand gesture detection and recognition using deep learning convolutional neural networks. *Soft Computing*, pages 1–10, 2020.

[52] Hrishikesh Pardeshi and Chandranath Bhattacharyya. Heuristic-based approach to detect global touch gestures performed on touch devices. *International Journal of Computer Applications*, 82(8), 2013.

[53] Leif E Peterson. K-nearest neighbor. *Scholarpedia*, 4(2):1883, 2009.

[54] Benjamin Poppinga, Alireza Sahami Shirazi, Niels Henze, Wilko Heuten, and Susanne Boll. Understanding shortcut gestures on mobile touch devices. In *Proceedings of the 16th international conference on Human-computer interaction with mobile devices & services*, pages 173–182, 2014.

[55] Zhang Qiu-yu, Lu Jun-chi, Zhang Mo-Yi, Duan Hong-xiang, and Lv Lu. Hand gesture segmentation method based on ycbcr color space and k-means clustering. *International Journal of Signal Processing, Image Processing and Pattern Recognition*, 8(5):105–116, 2015.

[56] Zhou Ren, Junsong Yuan, Jingjing Meng, and Zhengyou Zhang. Robust part-based hand gesture recognition using kinect sensor. *IEEE transactions on multimedia*, 15(5):1110–1120, 2013.

[57] Jaime Ruiz, Yang Li, and Edward Lank. User-defined motion gestures for mobile interaction. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 197–206, 2011.

[58] O Sangjun, Rammohan Mallipeddi, and Minho Lee. Real time hand gesture recognition using random forest and linear discriminant analysis. In *HAI*, pages 279–282, 2015.

[59] Shahrzad Saremi, Seyedali Mirjalili, and Andrew Lewis. How effective are meta-heuristics for recognising hand gestures. In *2016 IEEE Congress on Evolutionary Computation (CEC)*, pages 104–111. IEEE, 2016.

[60] Robert E Schapire. A brief introduction to boosting. In *Ijcai*, volume 99, pages 1401–1406. Citeseer, 1999.

[61] Muhammad Shahzad, Alex X Liu, and Arjmand Samuel. Secure unlocking of mobile touch screen devices by simple gestures: You can see it but you can not do it. In *Proceedings of the 19th annual international conference on Mobile computing & networking*, pages 39–50, 2013.

[62] Jie Song, Gábor Sörös, Fabrizio Pece, Sean Ryan Fanello, Shahram Izadi, Cem Keskin, and Otmar Hilliges. In-air gestures around unmodified mobile devices. In *Proceedings of the 27th annual ACM symposium on User interface software and technology*, pages 319–329, 2014.

[63] statista.com. Mobile operating systems' market share worldwide from january 2012 to october 2020. URL `https://developer.android.com/guide/topics/sensors/sensors_overview`.

[64] Ke Sun, Ting Zhao, Wei Wang, and Lei Xie. Vskin: Sensing touch gestures on surfaces of mobile devices using acoustic signals. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, pages 591–605, 2018.

[65] Jeremy T. Newhandwave. `https://github.com/jtkac/NewHandwave`.

[66] Ultrahaptics. Digital worlds that feel human | ultraleap. `https://www.ultraleap.com`.

[67] Juan Pablo Wachs, Mathias Kölsch, Helman Stern, and Yael Edan. Vision-based hand-gesture applications. *Communications of the ACM*, 54(2): 60–71, 2011.

[68] Christoph Wick. Deep learning. *Informatik-Spektrum*, 40(1):103–107, 2017.

[69] Wiktionary. Geste — wiktionary, das freie wörterbuch, 2021. URL `https://de.wiktionary.org/w/index.php?title=Geste&oldid=8353376`. [Online; abgerufen am 29. Januar 2021].

[70] Xiang Xiao, Teng Han, and Jingtao Wang. Lensgesture: augmenting mobile interactions with back-of-device finger gestures. In *Proceedings of the 15th ACM on International conference on multimodal interaction*, pages 287–294, 2013.

[71] Chen Zhao, Ke-Yu Chen, Md Tanvir Islam Aumi, Shwetak Patel, and Matthew S Reynolds. Sideswipe: detecting in-air gestures around mobile devices using actual gsm signal. In *Proceedings of the 27th annual ACM symposium on User interface software and technology*, pages 527–534, 2014.

[72] Xian Zhao, Zhan Song, Jian Guo, Yanguo Zhao, and Feng Zheng. Real-time hand gesture detection and recognition by random forest. In *Communications and information processing*, pages 747–755. Springer, 2012.