



**Entwicklung und Umsetzung einer intuitiven
Steuerung für eine Roboterhand durch Erfassen der
Geste einer menschlichen Hand**

Kinematik Labor

des Studienganges Mechatronik und Robotik
an der Frankfurt University of Applied Sciences

von

Peter Abt 1400337
Felix Girke 1386888

4. Juli 2022

Bearbeitungszeitraum: 9 Wochen
Betreuer Prof. Dr. Enno Wagner

Selbstständigkeitserklärung

Wir versichern hiermit, dass wir die Projektarbeit mit dem Thema: „Entwicklung und Umsetzung einer intuitiven Steuerung für eine Roboterhand durch Erfassen der Geste einer menschlichen Hand“ , selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt haben.

Frankfurt a. M., 4. Juli 2022

Ort, Datum

Unterschrift (Abt)

Frankfurt a. M., 4. Juli 2022

Ort, Datum

Unterschrift (Girke)

Inhaltsverzeichnis

Selbstständigkeitserklärung	I
Abbildungsverzeichnis und Tabellenverzeichnis	III
1 Einleitung	1
2 Stand der Technik	2
3 Mögliche Konzepte	3
3.1 Bowdenzug über Finger	3
3.2 Biegesensoren DMS	4
3.3 Image Processing	5
3.4 Entscheidung	5
4 Umsetzung des Konzepts	6
4.1 Code zur Handgestenerkennung und Nutzeroberfläche	6
4.2 LabVIEW	10
4.3 Demonstration	13
5 Diskussion	15
6 Ausblick	16
Literaturverzeichnis	A
Anhang	B

Abbildungsverzeichnis

1.1	Die zu steuernde „Frankfurter“ Roboter Hand	1
2.1	PowerGlove von Nintendo [8]	2
3.1	Bautenzug über die Hand für die Finger	3
3.2	Bautenzug über die Hand für den Daumen	4
3.3	auf Leitfähigkeit basierender Biegesensor, lizenziert unter CC BY 2.0 . .	5
4.1	Übersicht der Dateien des LabVIEW Projektes	10
4.2	Aufbau des User Panels in der GET-Methode des LabVIEW Servers . .	11
4.3	Aufbau des Block Diagrams in der GET-Methode des LabVIEW Servers	11
4.4	Aufbau des Connector Panes in der GET-Methode des LabVIEW Servers	12
4.5	Nötige Einstellungen in der NI-Webserver-Konfiguration	12
4.6	URL des HTTP-GET Servers	13
4.7	Screenshot aus dem Demovideo im Anhang	13
6.1	VIVE-Tracker für VR der Firma HTC [13]	16

1 Einleitung

Endeffektoren sind „gewöhnlich das letzte Glied der kinematischen Kette einer Handhabungseinrichtung“ [1, S.302]. Schon früh wurde versucht mit Endeffektoren die menschlichen Hand nachzubilden. Lange Zeit war diese Aufgabe zu komplex und es wurden Endeffektoren auf eine Aufgabe spezialisiert [2, S.9]. Aber aufgrund der technischen Fortschritte der letzten Jahrzehnte gelingt es immer besser diese Komplexität zu beherrschen. So wurde 2013 mit der „SVH 5-Finger-Hand“ die erste serienreife 5-Finger-Hand mit 9 Motoren von der Schunk GmbH & Co. KG vorgestellt [3, S.57]. Durch die Arbeit vorheriger Gruppen, ist auch an der University of Applied Sciences in Frankfurt eine Roboterhand entwickelt und gebaut worden (Abbildung 1.1), welche sich über einen Mikrocontroller steuern lässt.

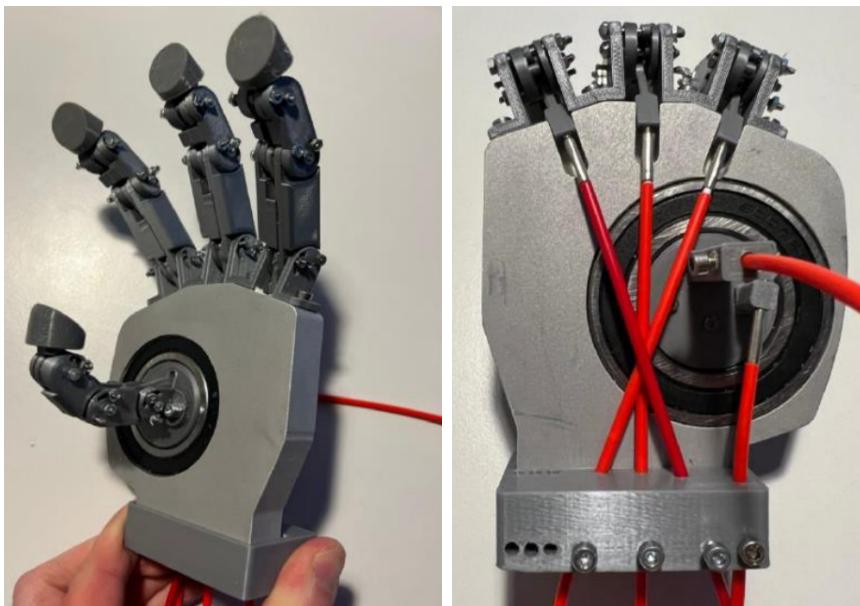


Abbildung 1.1: Die zu steuernde „Frankfurter“ Roboter Hand

Diese Roboterhand besteht aus drei Fingern und einem Daumen. Die einzelnen Finger werden über Bowdenzüge und Linearmotoren bewegt. Der Daumen kann rotieren, so dass ein Pinzettengriff mit jedem einzelnen der Finger möglich ist. Die Ansteuerung geschieht über einen Arduino Mikrocontroller und vier verschiedene Knöpfe. Dies erfordert ein hohes Maß an Einarbeitung und Konzentration der bedienenden Person. Deshalb wird versucht mit dieser Arbeit eine Ansteuerung zu schaffen welche intuitiver und einfacher ist. Hierzu soll versucht werden die Bewegung der Hand der bedienenden Person aufzuzeichnen und als Eingangssignal zu verwenden. Besonderen Wert soll hierbei auf die Stärke der Hand – den Pinzettengriff – gelegt werden.

2 Stand der Technik

Die Echtzeit-Erkennung von Handbewegungen ist für die Steuerung von humanoiden Händen von Essenz. Die komplexen Bewegungsabläufe der menschlichen Hand lassen sich aufgrund der großen Anzahl an Fingersegmenten und Freiheitsgraden nur mit hohem Aufwand erfassen. Und eine Steuerung mittels Joysticks o.ä. ist ungeeignet, da es wenig intuitiv ist. Erste Versuche die Bewegungsabläufe der Hand aufzunehmen wurden mithilfe von in Handschuhen eingebauten Biegesensoren [4] und Lagesensoren durchgeführt. Die zu dieser Zeit boomende Computerspielindustrie griff die Idee schnell auf und brachte den, technisch vereinfachten, PowerGlove [5] (Abbildung 2.1) auf den Markt. Heute sind verschiedene Firmen im Markt die professionelle Systeme vertreiben wie CyberGlove Systems [6] oder Cobra Glove [7]. Diese bedienen sich meist der Erfassung der Fingerpositionen durch eine Kombination von mehreren an den Fingern angebrachten Inertial Measurement Units (IMUs) und Biegesensoren.



Abbildung 2.1: PowerGlove von Nintendo [8]

Handschuhe haben im allgemeinen einige Nachteile die sie mit sich bringen. Der an und Abziehvorgan ist umständlich, die Größe des Handschuhs muss stimmen, Desinfektionsmaßnahmen sind kompliziert.

Alternativ werden Handbewegungen auch mit Bewegungserkennungssystemen durch Marker und IR-Kamerasystemen aufgezeichnet. Über Triangulation die Position der einzelnen Markerpunkte berechnet. Hier ist die Firma VICON ein Vorreiter auf dem Markt. Auch markerlose Kamerasysteme zur Bewegungserkennung existieren wie durch z.B. die Kinect Kamera ermöglicht.

3 Mögliche Konzepte

Ziel ist es die Bewegungen der Hand aufzunehmen. Um eine ideale Lösung zu finden, werden verschiedene Lösungsmöglichkeiten skizziert und anschließend bewertet. Wichtig für die Bewertung sind die Genauigkeit des Systems, die Flexibilität zwischen verschiedenen Anwendern und die entstehenden Kosten.

3.1 Bowdenzug über Finger

Die komplexe, mehrdimensionale Bewegung der Finger kann über einen Bowdenzug abgegriffen werden und in eine lineare Bewegung verwandelt werden. Inspiration hierzu ist die bestehende Roboter Hand (Abbildung 1.1), welche die einzelnen Finger durch solche Bowdenzüge steuert. Wird diese lineare Bewegung gemessen, so kann bestimmt werden wie stark ein Finger gekrümmmt wird.

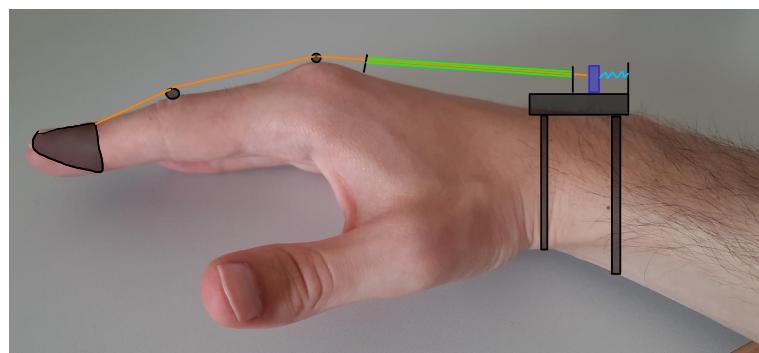


Abbildung 3.1: Bautenzug über die Hand für die Finger

In Abbildung 3.1 ist zu erkennen wie ein solcher Bowdenzug (orange) für die Finger angebracht werden könnte. Das schwarze Objekt an der Fingerspitze ist der Befestigungspunkt des Bowdenzugs, die schwarzen Kreise auf den Gelenken sind Führungen, damit der Bowdenzug nicht von den Fingern rutscht. Hellgrün ist die druckfeste Hülle des Bowdenzugs welche verhindert, dass die Bewegungen des Handgelenks in die Messung des Bowdenzugs eingeht. Am rechten Ende des Bowdenzugs befindet sich für jeden Finger ein lineares Potentiometer (dunkel Blau), welches von einer Feder in die Nullstellung zurückgezogen wird. Es muss für jeden zu messenden Finger ein Potentiometer angebracht werden. Durch messen des Widerstandes des Potentiometers kann die Krümmung des Fingers in ein elektrisches Signal umgewandelt werden. Hierfür würde sich zum Beispiel ein Mikrocontroller eignen. Um die Punkte alle auf einer Hand zu befestigen eignet sich ein Handschuh.

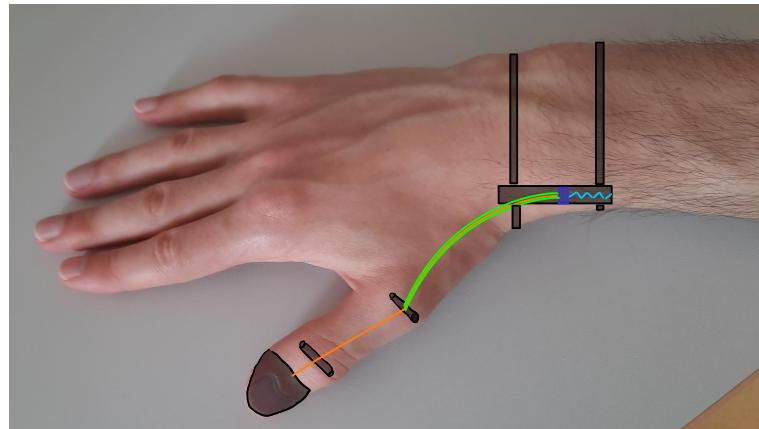


Abbildung 3.2: Bautenzug über die Hand für den Daumen

Wird für die Messung des Daumens ähnlich vorgegangen (Abbildung 3.2), so werden die Messdaten wahrscheinlich wenig Aussagekraft haben. Geht man mit dem Daumen die einzelnen Fingerspitzen des Pinzettengriffs durch so ist die Bewegung zwischen den einzelnen Fingerknochen sehr gering. Die Bewegung findet lediglich zwischen den Handwurzelknochen und dem ersten Fingerknochen statt. Diese Bewegung liegt direkt neben der Bewegung des Handgelenks und ist schwieriger voneinander zu trennen. Des Weiteren ist die Bewegung des Daumens beim Pinzettengriff nur sehr gering weshalb die Unterscheidung zu welcher Fingerspitze der Daumen sich bewegt schwer ist.

Sind alle Bowdenzüge perfekt angebracht, so ist es theoretisch möglich in einer zukünftigen Arbeit einen Servomotor parallel zu den Potentiometern anzubringen und somit die Bowdenzüge zu spannen. Hierdurch könnte dem Nutzer simuliert werden, es würde sich ein Gegenstand zwischen den Fingern befinden, da eine Kraft auf die sich schließenden Finger ausgewirkt werden kann.

3.2 Biegesensoren DMS

Die heutzutage erhältlichen Biegesensoren sind wesentlich preiswerter als die ersten ihrer Art. Sie basieren nicht mehr auf dem Prinzip eines Lichtwellenleiters sondern auf der Änderung der Leitfähigkeit von Materialien durch deren Biegung (vgl. Abb. 3.3). Somit ermöglichen sie eine einfache Möglichkeit den Gebogenheitsgrad eines einzelnen Fingers zu bestimmen. Als schwieriger erweist sich jedoch die Positionsbestimmung des Daumens. Dieser kann sich auch unabhängig seines Biegegrades auf dem unteren Sattelgelenk in zwei Freiheitsgraden bewegen. Ein Biegesensor reicht nicht um z.B. den Pinzettengriff zwischen Daumen und Zeigefinger und zwischen Daumen und Mittelfinger zu unterscheiden.



Abbildung 3.3: auf Leitfähigkeit basierender Biegesensor, lizenziert unter CC BY 2.0

3.3 Image Processing

Mithilfe des Einsatzes von moderner Bildverarbeitung und künstlicher Intelligenz lassen sich viele Probleme der vorherigen Methoden vermeiden. So sind allen voran die benötigten Investitionen nahe Null. Lediglich ein PC sowie eine passende Webcam sind bereits aussreichen um die Positionserkennung zu ermöglichen. Außerdem fallen umständliches an und abziehen eines sensiblen Handschuhs so wie der damit verbundene verkabelungs Aufwand weg. Dank frei verfügbarer, vor-trainierter Machine Learning-Modelle lassen sich selbst komplexe Objekte mit geringem Rechenaufwand erkennen [9].

3.4 Entscheidung

Die erste mechanische Idee mit dem Bowdenzug besticht zwar durch die Möglichkeit Feedback in Form eines Widerstandgefühls zu simulieren. Allerdings gestaltet sich der Aufbau mithilfe eines Handschuhs schwierig und würde bei nicht exakt passender Hand(-schuh)-größe sehr leicht verrutschen. Ein weiteres Problem die Positionsbestimmung des Daumens. Aufgrund des Sattelgelenks kann eine Unterscheidung zwischen Pinzettengriffen für verschiedene Finger nicht erfolgen. Selbiges gilt für den Ansatz mit Biegesensoren. Der Aufbau wäre zwar leichter und weniger anfällig zu Verrutschen, die genaue Erkennung der Handgesten ist aber aufgrund der vielen Freiheitsgrade nicht möglich. Daher fällt die Entscheidung auf den Ansatz mit moderner Bildverarbeitung und Maschine-Learning. Die Unterscheidung zwischen verschiedenen Pinzettengriffen ist möglich. Hinzu kommt dass das System unabhängig von der Handgröße für jeden Nutzer funktioniert und durch den Verzicht auf Handschuhe keine besonderen Reinigungsmaßnahmen erforderlich sind. Fertig trainierte Modelle sind frei verfügbar und die Genauigkeit der Erkennung sehr hoch.

4 Umsetzung des Konzepts

Die Wahl der Programmiersprache für den Bildbearbeitungsansatz von Handgesten fiel auf die Sprache Javascript. Es ermöglicht die Verwendung des Programms auf allen Betriebssystemen mit aktuellem Browser. So könnten auch mobile Endgeräte zur Erfassung der Handgesten genutzt werden. Außerdem lässt sich der Zugriff auf die Webcam und das Senden von Informationen über die Netzwerkschnittstelle leicht realisieren. Als System zur Erkennung von Handgesten wurde „MediaPipe Hands“ gewählt, dieses ist auch für Javascript verfügbar [10]. Zum erleichterten Zugriff auf die Funktionalitäten des MediaPipe-Modells wird die Bibliothek Handsfree eingesetzt [11].

4.1 Code zur Handgestenerkennung und Nutzeroberfläche

Der Javascriptcode wird über ein HTML-Dokument im Browser geöffnet. Im HTML-Code werden zuerst, im Header-Element die benötigten JavaScript-Bibliotheken zur Handgestenerkennung geladen und CSS-Style Vorschriften für das Design der Seite definiert:

```

3 | <head>
4 |   <!-- Include Handsfree.js -->
5 |   <link rel="stylesheet" href="https://unpkg.com/
6 |     handsfree@8.5.1/build/lib/assets/handsfree.css" />
7 |   <script src="https://unpkg.com/handsfree@8.5.1/build/lib/
8 |     handsfree.js"></script>
9 |   <!-- Styles -->
10 |   ...
11 |
12 | </head>

```

Anschließend wird die Seitenüberschrift sowie der Webcamfeed eingebettet:

```

30 | <div> <!--Title-->
31 |   <h1>Finger tracking</h1>
32 | </div>
33 | <!-- Holds the Video -->
34 | <div id="video-holder" style="margin: 1cm; width: 1280px;
35 |   height: 720px"></div>

```

Das letzte Element der Nutzeroberfläche bildet eine Tabelle zur Übersicht der aktuell erkannten Gesten. Jede Zelle enthält ein span-Element welches basierend auf den im Header definierten Styles als Kreis sichtbar wird. Zusätzlich ist jedes span-Element mit einer ID ausgestattet sodass die Farbe des Kreises vom folgenden Javascript aus geändert werden kann.

4 Umsetzung des Konzepts

```

38 <div> <!--Status Table-->
39 <table id="coll">
40   <thead>
41     <tr>
42       <th></th>
43       <th>Zeigefinger [0]</th>
44       <th>Mittelfinger [1]</th>
45       <th>Ringfinger [2]</th>
46       <th>kleiner Finger [3]</th>
47     </tr>
48   </thead>
49   <tbody>
50     <tr>
51       <th>linke Hand [0]</th>
52       <td><center><span class="dot" id="F00"></span></td>
53       <td><center><span class="dot" id="F01"></span></td>
54       <td><center><span class="dot" id="F02"></span></td>
55       <td><center><span class="dot" id="F03"></span></td>
56     </tr>
57     <tr>
58       <th>rechte Hand [1]</th>
59       <td><center><span class="dot" id="F10"></span></td>
60       <td><center><span class="dot" id="F11"></span></td>
61       <td><center><span class="dot" id="F12"></span></td>
62       <td><center><span class="dot" id="F13"></span></td>
63     </tr>
64   </tbody>
65 </table>
66 </div>

```

Anschließend folgt der Javascript Code. Zuerst wird eine Funktion definiert die das Senden von Get-Requests mit Informationen über die Handgeste erlaubt. In Zeile 32 wird die Adresse, sowie der Port, des LabView-Servers oder localhost, falls Client und Server auf dem gleichen PC laufen, angegeben.

```

67 <script>
68   function httpGet(theUrl) {
69     let xmlhttpReq = new XMLHttpRequest();
70     let url = 'http://localhost:80/WebServerHand/
71       GetPinchState'+theUrl;
72     xmlhttpReq.open("GET", url, false);

```

4 Umsetzung des Konzepts

```

72     xmlhttpReq.send(null);
73     return xmlhttpReq.responseText;
74   }
75 </script>
```

Es wurde auch die schnellere Übermittlung der Handgestendaten mittels Websocket getestet. Dazu muss ein Websocket angelegt werden.

```
99 var websocket = new WebSocket("ws://localhost:2323");
```

Dies hat jedoch den Nachteil dass eine dauerhafte Verbindung benötigt wird und die Implementierung eines Websocket-Server in Lab-View komplizierter ist. Daher wird im folgenden die robustere Variante mittels Get-Request verwendet.

Um Gebrauch der Handsfree-Bibliothek zu machen wird ein Handsfree Element angelegt und entsprechend initialisiert. Um den Camerafeed für den Nutzer sichtbar zu machen wird `showDebug` auf `true` gesetzt und die ID des Video-Elements übergeben.

Da nur eine Roboterhand gesteuert werden soll wird die Maximale Anzahl an Händen auf 1 gesetzt. Um die Möglichkeit zu haben mehrere Hände zu steuern, wurde der restliche Code auf die Möglichkeit von 2 Händen angepasst. Dabei wird immer die Hand mit der höchsten Sicherheit gewählt. Um falsche Detektionen von Hand-ähnlichen Objekten möglichst gering zu halten wird die minimale Sicherheit auf 90 % gesetzt. Anschließend wird die Handerkennung gestartet (Zeile 96).

```

79 const handsfree = new Handsfree({
80   showDebug: true,
81   setup: {
82     wrap: {
83       $parent: document.querySelector('#video-holder')
84     }
85   },
86   hands: {
87     enabled: true,
88     // The maximum number of hands to detect [0 - 4]
89     maxNumHands: 1,
90     // Minimum confidence [0 - 1] for a hand
91     minDetectionConfidence: 0.9,
92     // Minimum confidence [0 - 1] for the landmark tracker
93     minTrackingConfidence: 0.9
94   }
95 })
96 handsfree.start()
```

4 Umsetzung des Konzepts

in der Variable `FState` werden die Zustände der einzelnen Finger gespeichert. Dabei steht 0 für offen und 1 für Pinzettengriff zum Daumen.

```
101 | let FState = [0,0,0,0];
```

Im folgenden werden über eine doppelte for-Schleife einmalig die Callbacks für alle Finger, beider Hände definiert. Das bedeutet sobald die aktive `handsfree`-Instanz den Zangengriff für einen Finger erkennt wird das entsprechend definierte Callback aufgerufen. In den Callbacks wird die Farbe des entsprechenden Punktes in der oben definierten Tabelle geändert sowie der Status des Fingers in der globalen Variable `FState` angepasst. Anschließend wird ein Get-Request, mit den Finger Zuständen, an den Lab-View Server gesendet. Alternativ kann durch auskommentieren der Zeilen 112 oder 120 auch ein WebSocket an einen geeigneten Empfänger gesendet werden.

```
104 | for (let hand = 0; hand < 2; hand++) {
105 |   for (let finger = 0; finger < 4; finger++) {
106 |     handsfree.on(`finger-pinched-start-${hand}-${finger}`, () =>
107 |       {
108 |         const Finger = document.getElementById('F'+hand+finger)
109 |         Finger.classList.add('dotRed')
110 |         if(hand==0){return;} // send only right hand to server
111 |         FState[finger]=1
112 |         httpGet("?finger3="+FState[3]+"&finger2="+FState[2]+"&
113 |             finger1="+FState[1]+"&finger0="+FState[0])
114 |         //websocket.send("finger"+finger+"=pinched");
115 |       })
116 |     handsfree.on(`finger-pinched-released-${hand}-${finger}`, () =>
117 |       {
118 |         const Finger = document.getElementById('F'+hand+finger)
119 |         Finger.classList.remove('dotRed')
120 |         if(hand==0){return;} // send only right hand to server
121 |         FingerState[finger]=0
122 |         httpGet("?finger3="+FState[3]+"&finger2="+FState[2]+"&
123 |             finger1="+FState[1]+"&finger0="+FState[0])
124 |         //websocket.send("finger"+finger+"=released");
125 |       })
126 |   }
127 | }
```

4.2 LabVIEW

Für die Roboterhand wird derzeit eine Steuerung in LabVIEW von einem anderen Team entwickelt. Deshalb müssen die erfassten Daten nach LabVIEW übertragen werden. Hierfür wird mit LabVIEW ein lokaler Server Programmiert, welcher von der Weboberfläche per GET-Request angesprochen wird und die entsprechenden Daten übertragen bekommt [12]. Das angelegte LabVIEW Projekt besteht aus dem WebServerHand und den

SharedVariables.lvlib (Abbildung 4.1).

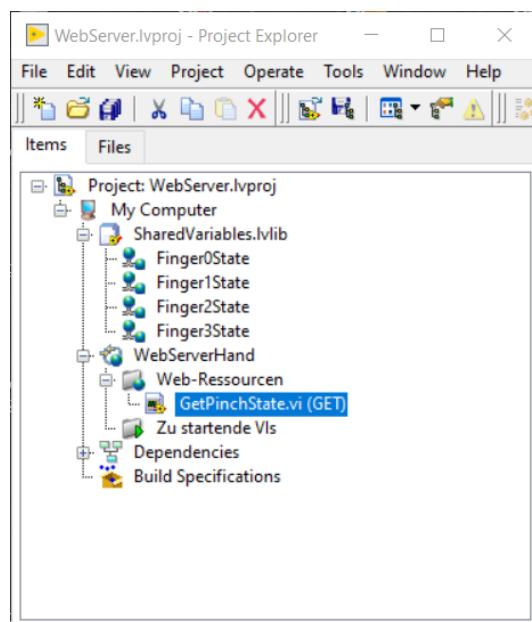


Abbildung 4.1: Übersicht der Dateien des LabVIEW Projektes

In dem Server ist eine GET-Methode angelegt namens GetPinchState.vi. Diese besteht aus einem User Panel (Abbildung 4.2), einem Block Diagram (Abbildung 4.3) und einem Connector Pane (Abbildung 4.4).

4 Umsetzung des Konzepts

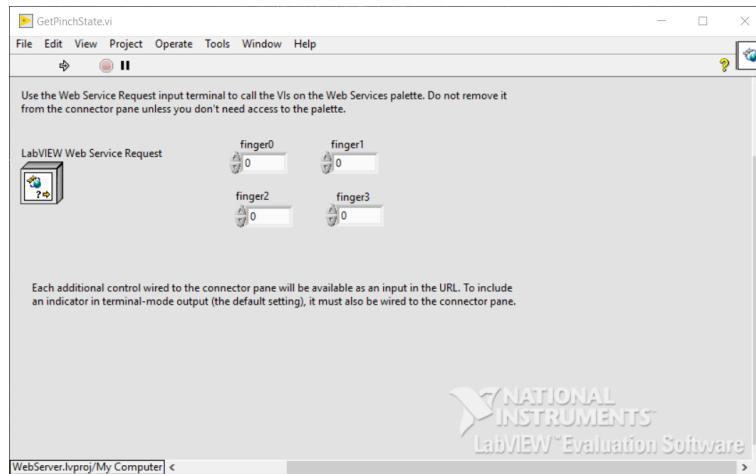


Abbildung 4.2: Aufbau des User Panels in der GET-Methode des LabVIEW Servers

In dem User Panel ist der Block LabVIEW Web Service Request angelegt über diesen kommt der GET Request in LabVIEW an. Des Weiteren sind für jeden der vier gemessenen Finger ein Numeric Control Panel angelegt. Diese sollen den aktuellen Stand der Finger anzeigen.

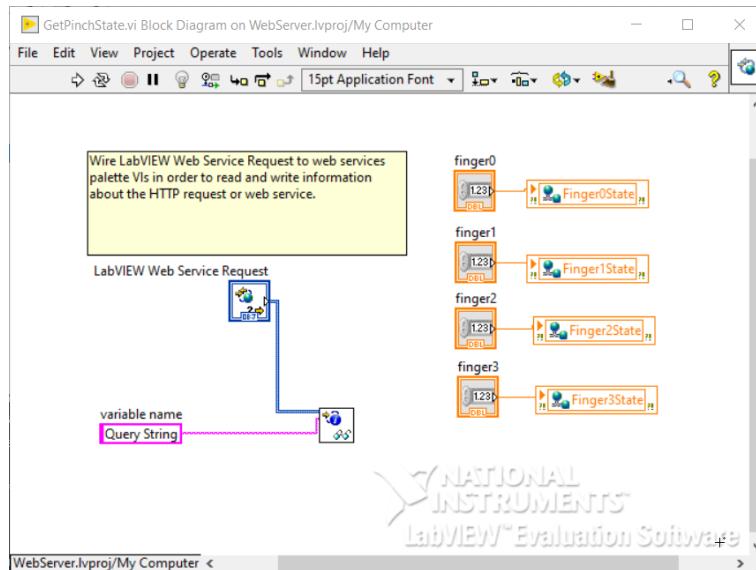


Abbildung 4.3: Aufbau des Block Diagrams in der GET-Methode des LabVIEW Servers

Die beschriebenen Blöcke im User Panel werden im Block Diagram miteinander logisch verbunden. Der Block LabVIEW Web Service Request und ein Query String sind mit dem eigentlichen Server verbunden. Die einzelnen Finger müssen nicht mit dem Server verbunden werden, sie erhalten Ihre Daten durch die Verknüpfung im Connector Pane.

4 Umsetzung des Konzepts

Damit die Werte der Finger außerhalb des Servers verwendet werden können, sind diese mit einer Shared Variable Node verbunden. Somit wird der jeweilige Wert in die globale Variable übernommen und kann von der Ansteuerung der Roboterhand genutzt werden.



Abbildung 4.4: Aufbau des Connector Panes in der GET-Methode des LabVIEW Servers

Im Connector Pane sind von oben nach unten die Finger 0 bis 3 als Input verknüpft (Abbildung 4.4, orange) und der LabVIEW Web Service Request (blau). Dadurch gelangen die Werte an die richtigen Variablen.

Mit einem Rechtsklick auf den WebServerHand im Explorer (Abbildung 4.1) kann der Server gestartet werden. Vorher sollten allerdings diese Einstellungen in der NI-Webserver-Konfiguration vorgenommen werden (Abbildung 4.5).

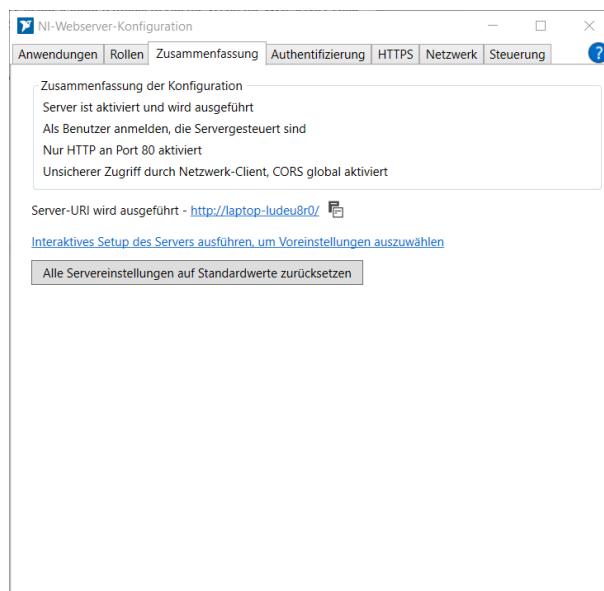


Abbildung 4.5: Nötige Einstellungen in der NI-Webserver-Konfiguration

Sind diese Einstellungen vorgenommen und der Server gestartet, so kann über einen Rechtsklick auf die GetPinchState.vi (Abbildung 4.1) die Methoden-URL angezeigt werden (Abbildung 4.6)

4 Umsetzung des Konzepts

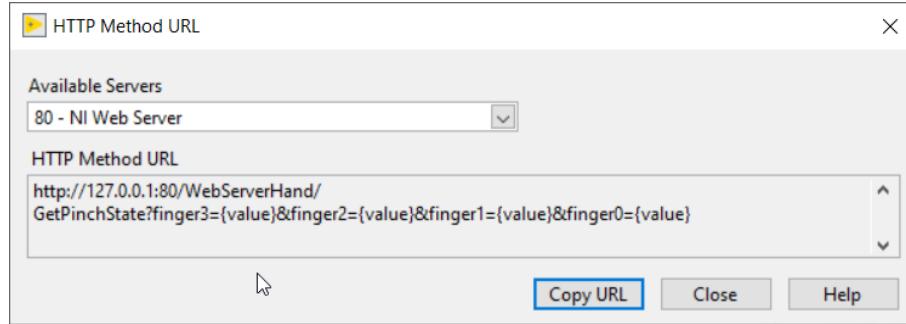


Abbildung 4.6: URL des HTTP-GET Servers

Zur Nutzung des Webservers muss lediglich diese URL in einen Browser eingefügt werden und „{value}“ durch einen Wert ersetzt werden. Anstelle der angegebenen IP-Adresse kann auch „localhost“ angegeben werden, wenn der Browser und der Server sich auf dem selben PC befinden. Möchte man den Server auf einem anderen PC laufen lassen, so muss die IP-Adresse durch die des PC's im Netzwerk ersetzt werden.

4.3 Demonstration

In dem angehängtem Video (Anhang 3, DemoVideo.mkv), ist auf der linken Seite das User Panel des LabVIEW Webserver zu sehen und auf der rechten Seite die Weboberfläche (Abbildung 4.7).

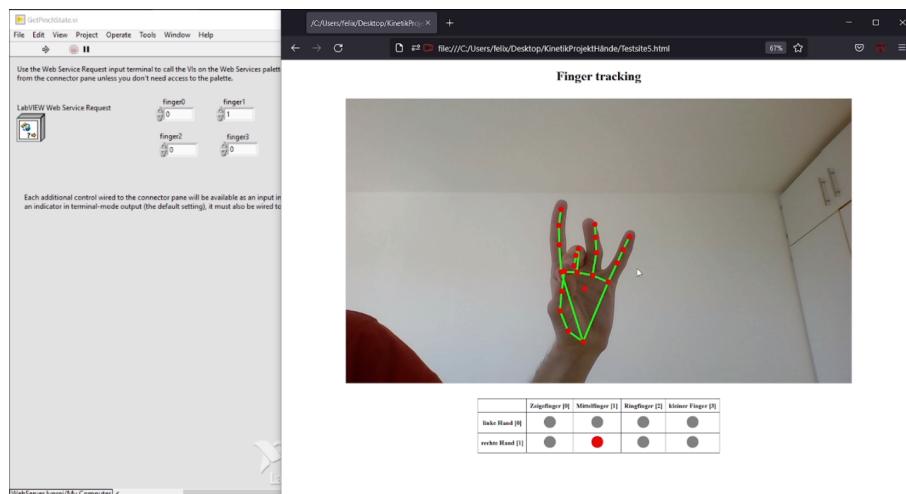


Abbildung 4.7: Screenshot aus dem Demovideo im Anhang

Es ist im Videofeed rechts zu erkennen, dass sich der Mittelfinger und der Daumen berühren. In der Tabelle darunter sieht man das in der Weboberfläche erkannt wurde,

4 Umsetzung des Konzepts

dass sich nur diese Finger berühren. In dem User Panel links ist zu sehen, dass lediglich bei Finger 1 der Wert von 0 auf 1 geändert wurde. Die Berührung wurde also korrekt erkannt und an LabVIEW übermittelt.

Das Demonstrationsvideo lässt den Prozess etwas langsam und verzögert wirken. Dies liegt allerdings nur daran das der Testrechner mit einer schwachen und über 10 Jahren alten CPU ausgestattet ist und gleichzeitig den Bildschirm aufzeichnet. Ohne Bildschirmaufzeichnung läuft der Prozess flüssig und deutlich schneller.

5 Diskussion

Wie in den Demonstrationsvideo zu erkennen ist das entworfene Konzept der Handgestenerkennung mithilfe von Bildverarbeitung und Maschine-Learning ist voll funktionsfähig. Die verschiedenen Pinzettengriffe können frei von jeglicher Verkabelung vom Nutzer durchgeführt werden und werden vom System zuverlässig erkannt. Dank der Implementierung über Javascript lässt sich das System auch auf einfacher Hardware wie Mobiltelefonen, einem SoC o.ä. ausführen.

Die erfassten Daten können direkt intern oder per Lan- oder WLAN-Schnittstelle zu LabVIEW oder einem anderen Server gesendet werden. Dadurch ist eine Anbindung an die Steuerungssoftware der Hand flexibel möglich. Die Übertagung der Daten nach LabVIEW funktioniert zuverlässig und ausreichend schnell. Diese Daten stehen global in dem LabVIEW Projekt zu Verfügung und können somit einfach in die Steuerung eingebunden werden.

Die Kosten für das Projekt sind sehr niedrig bis nicht vorhanden. Sollte die Steuerung der Hand auf einem Laptop laufen, so ist eine Webcam in den meisten Fällen integriert. Wird ein stationärer PC oder ein SoC wie beispielsweise ein Raspberry Pi eingesetzt, so kostet eine Webcam oder PiCam weniger als 20€, vor allem da keine hohe Auflösung nötig ist.

Eines der Hauptziele war es die Steuerung intuitiver zu gestalten als das bisherige System, welches auf vier Knöpfe zur Steuerung setzt. Die neue Steuerung über die Bewegung der eigenen Hand ist für die bedienenden Person sehr einfach. Der gespiegelte Webcamfeed mit dem eingeblendeten Modell und die Tabelle zu den erkannten Pinzettengriffen erleichtert die Bedienung, sodass ein Einweisen in das Systems selbst für neue Nutzer kaum notwendig ist.

Die bedienende Person muss also lediglich ihre Hand vor die Kamera halten und die Roboterhand imitiert sofort den Pinzettengriff.

6 Ausblick

Als anschließende Projekte könnte noch Erweiterungen des Systems folgen. So können theoretisch mehrere Hände in einem Videofeed erkannt werden. Somit könnte beispielsweise eine linke und eine rechte Hand erkannt und gesteuert werden. Dadurch kann eine bedienende Person mehrere Roboter gleichzeitig steuern.

Die Bildverarbeitung erkennt im Moment lediglich den Pinzettengriff als Aktion und sendet diese an die Steuerung. Da die Position jedes Fingergelenkes erfasst wird (Abbildung 4.7, rote Punkte) und abrufbar ist, können theoretisch eigene Algorithmen entwickelt werden. Diese könnten immer die Fingerpositionen erfassen und somit der Roboterhand bereits kleine Fingerbewegungen mitteilen. Dann könnte die Roboterhand nicht nur den Pinzettengriff ausführen sondern fast jegliche Objekte greifen. Schwierigkeiten hierbei sind allerdings, es muss zunächst eine Kalibrierung auf eine Hand stattfinden, da jede Hand anders groß ist, es muss beachtet werden das sich die Hand von der Kamera weg bewegen kann und dadurch kleiner wird und auch das die Roboterhand anders aufgebaut ist als eine menschliche Hand. Hierfür müsste ein System entwickelt werden, welches diese Fälle verarbeitet.



Abbildung 6.1: VIVE-Tracker für VR der Firma HTC [13]

Da die Roboter Hand an das Ende eines Roboterarmes montiert werden muss um ihre volles Potential auszuschöpfen, wäre es ideal wenn nicht nur die Bewegung der Hände sondern auch die Bewegung der Arme erfasst werden könnte. Dann könnte eine Person neben der Roboterzelle stehen und alleine durch ihre Bewegung die Roboter Hand zu einem Gegenstand führen und durch die Fingerbewegungen den Gegenstand greifen. Durch die Dreidimensionalität dieser Bewegung eignet sich hierfür eine Kamera nicht, es muss auf mehrere Kameras oder auf ein System mit auf dem Körper angebrachten Trackern wie den „VIVE Tracker“ (Abbildung 6.1) zurückgegriffen werden.

Literatur

- [1] Stefan Hesse. *Grundlagen der Handhabungstechnik*. 3., neu bearbeitete und erweiterte Auflage. Carl Hanser Verlag GmbH Co KG, 2013. ISBN: 978-3-446-43596-4.
- [2] R. Jansen. „Stand und Perspektiven der Roboter in der Verpackungstechnik“. In: *VDI Berichte 850, Roboter in der Verpackungstechnik* (1990).
- [3] Andreas Wolf und Ralf Steinmann. *Greifer in Bewegung - Faszination der Automatisierung von Handhabungsaufgaben*. München: Carl Hanser Verlag GmbH Co KG, 2016. ISBN: 978-3-446-43993-1.
- [4] Thomas G. Zimmerman. „Optical flex sensor“. In: *US 4542291* (1982).
- [5] Greg Bryant, Russell Eberhart, Erik Frederick, John Gawel, Stephen Turner. *1993 VR Conference Proceedings: abgerufen am 19.06.22*. 1993.
- [6] CyberGlove. *Cyber Glove Systems*. 2017.
- [7] AiQ Synertial. *High-End Gloves for Robotics, Animation, Virtual Reality, Medical and Bio-mechanics Research*. 2022.
- [8] Stephen Reyes. *25 Terrible Video Game Accessories*. online. 2016. URL: <https://guff.com/25-terrible-video-game-accessories>.
- [9] Valentin Bazarevsky und Fan Zhang. *On-Device, Real-Time Hand Tracking with MediaPipe*. online. 2019. URL: <https://ai.googleblog.com/2019/08/on-device-real-time-hand-tracking-with.html>.
- [10] Google LLC. *MediaPipe Hands*. online. 2020. URL: <https://google.github.io/mediapipe/solutions/hands>.
- [11] Oz Ramos. *Handsfree.js*. online. 2021. URL: <https://handsfree.js.org>.
- [12] NATIONAL INSTRUMENTS CORP. *Tutorial: Creating and Publishing a LabVIEW Web Service to the Application Web Server (Real-Time, Windows)*. online. Juni 2022. URL: https://www.ni.com/docs/de-DE/bundle/labview/page/lvhowto/build_web_service.html.
- [13] HTC Corporation. *HTC VIVE Tracker*. online. 2022. URL: https://www.vive.com/media/filer_public/fed-assets/tracker3/images/meta-image.png.

Anhang

1. Code welcher für die Website benötigt wird (Ordner)
2. Code des LabVIEW-Projektes (Ordner)
3. Demonstrations Video DemoVideo.mkv