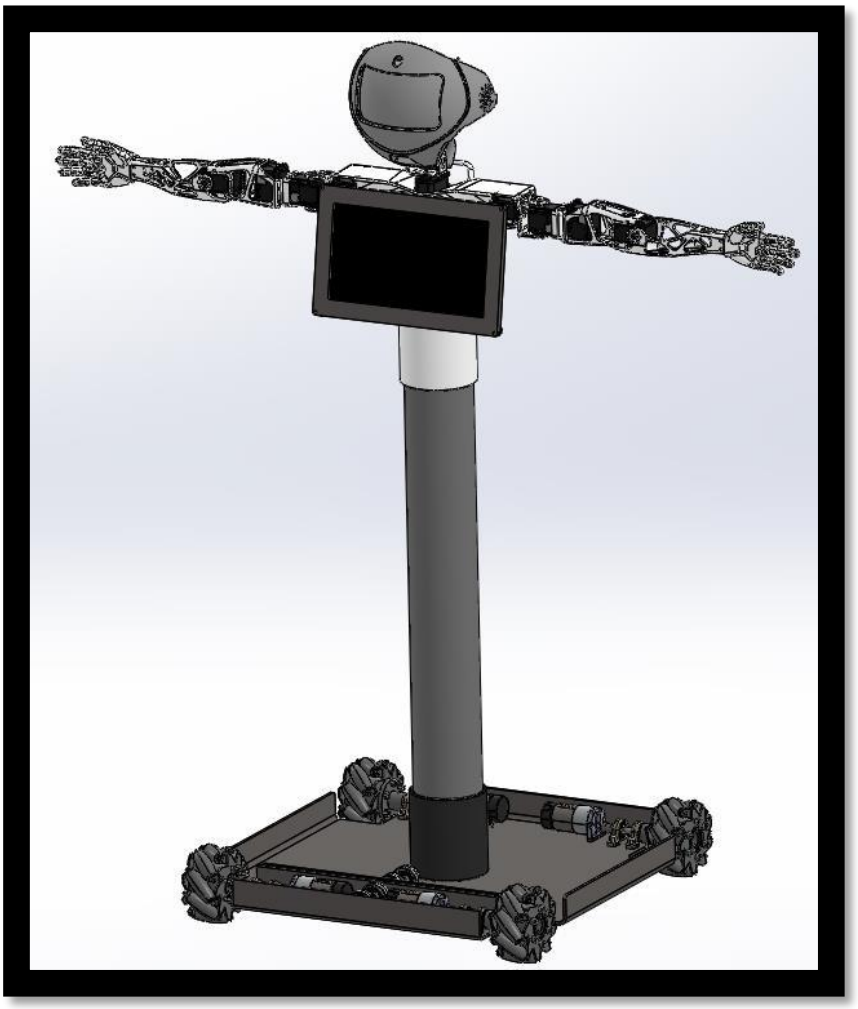
TECHNISCHE HOCHSCHULE NÜRNBERG
GEORG SIMON OHM

Projektarbeit-Bericht



Interaktionskonzept für einen humanoiden Roboter

Betreuer: Prof. Dr. Stefan May

Dong Trung Son

TECHNISCHE HOCHSCHULE NÜRNBERG
GEORG SIMON OHM

Selbständigkeitserklärung:

Hiermit erklären wir, dass ich die vorliegende Projektarbeit selbständig verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe.

Die Stellen der Projektarbeit, die anderen Quellen im Wortlaut oder dem Sinn nach entnommen wurden, sind durch Angaben der Herkunft kenntlich gemacht. Dies gilt auch für Zeichnungen, Skizzen, bildliche Darstellungen sowie für Quellen aus dem Internet.

Dong Trung Son MSY2 - Matrikelnummer: 2973939

Technische Hochschule Georg Simon Ohm

Nürnberg den 23.08.2021

Inhalt

I Einführung

Im Rahmen dieser Projektarbeit wurden einen Begrüßungsroboter aufgebaut. Der sollte im Eingangsbereich des neuen Empfangsgebäude im Einsatz angebracht werden. Bei der Bewältigung der verschiedenen Aufgaben wurden mehrere integrierte Künstliche-Intelligenz-Module für einen humanoiden Roboter geschaffen. Letztendlich werden die Varianten ausgewählt, die technisch auf einem Raspberry Pi4 realisierbar sind.

II 3D gedruckter humanoider Roboter

Der Roboter im Projekt wurde weiter entwickelt aus dem Poppy Projekt, spezifisch Poppy Torso. Das Poppy-Projekt wurde 2012 im Blumenlabor von Inria Bordeaux Sud-Ouest geboren. Es wurde während der von Pierre Yves Oudeyer betreuten Doktorarbeit von Matthieu Lapeyre initiiert. Zu Beginn bestand das Entwicklungsteam aus Matthieu Lapeyre (Mechanik & Design), Pierre Rouanet (Software) und Jonathan Grizou (Elektronik).
Dieses Projekt ist zunächst ein vom ERC Grant Explorer finanziertes Grundlagenforschungsprojekt, um die Rolle von Verkörperungs- und Eigenschaften auf die Kognition und insbesondere auf das Erlernen sensomotorischer Aufgaben zu untersuchen.

Abbildung 1: Poppy Torso

Der Poppy Torso Roboter wird hauptsächlich mit MX-28AT Dynamixel Servomotoren gebaut (MX-28T ist die Vorgängerversion und kann problemlos verwendet werden). Die anderen Servomotoren sind zwei AX-12A (oder AX-18), die kleiner sind und nur für den Kopf verwendet werden.
Jeder Dynamixel-Servomotor enthält eine elektronische Platine, die es ihm ermöglicht, verschiedene Arten von Befehlen (über Tor, Drehmoment...) zu empfangen und mit anderen Dynamixel-Servos zu kommunizieren. Daher können Sie mehrere Dynamixel-Servomotoren (jeder mit einer anderen ID) verketten und alle von einem Ende der Kette aussteuern: Jeder Servomotor gibt die Befehle an den nächsten weiter.

III Künstliche Intelligenz Technologie

1 Bekannte Plattform und Werkzeug in KI

1.1 OpenCV

OpenCV (Open Source Computer Vision Library) ist eine Open-Source-Softwarebibliothek für Computer Vision und maschinelles Lernen. OpenCV wurde entwickelt, um eine gemeinsame Infrastruktur für Computer-Vision-Anwendungen bereitzustellen und den Einsatz der maschinellen Wahrnehmung in kommerziellen Produkten zu beschleunigen. OpenCV ist eine Art "Muss" für eine Person, die in die Computer Vision einsteigt. Wir werden die Videoaufnahmefunktion von OpenCV verwenden, um die Webcam des lokalen Computers zu verwenden, um den Video-Feed für das Beispiel zu erhalten.

1.2 Tensorflow

TensorFlow ist eine von Google entwickelte Open-Source-Bibliothek, die beim maschinellen Lernen sehr beliebt geworden ist. TensorFlow bietet APIs, die maschinelles Lernen erleichtern und hat auch eine schnellere Kompilierungszeit als andere Deep-Learning-Bibliotheken wie Keras und Touch. TensorFlow unterstützt sowohl CPU- als auch GPU-Computergeräte.

Für die Mobilgeräte ist TensorFlow zu groß anzuwenden. Weil sein Design nur für mächtige Geräte passt. Deswegen wurde TensorFlow-Lite als die Lösung dafür erstellt.

TensorFlow Lite ist eine Reihe von Tools, die maschinelles Lernen auf dem Gerät ermöglichen, indem sie Entwicklern helfen, ihre Modelle auf mobilen, eingebetteten und IoT-Geräten auszuführen. Er ist obtimiert für maschinelles Lernen auf dem Gerät, indem 5 Hauptbeschränkungen berücksichtigt werden: Latenz (es gibt keinen Roundtrip zu einem Server), Datenschutz (keine persönlichen Daten verlassen das Gerät), Konnektivität (Internetverbindung ist nicht erforderlich), Größe (reduziertes Modell und Binärgröße) und Stromverbrauch (effiziente Inferenz und fehlende Netzwerkverbindungen). Es ist aber nicht möglich, ein Model mit Tensorflow-Lite zu trainieren. Allerding wir können die Tensorflow Modelle in Tensorlow-Lite konvertieren und die Inferenz ausführen.

2 Gesichtserkennung

2.1 DLib

Dlib ist ein modernes C++-Toolkit, das Algorithmen und Tools für maschinelles Lernen zum Erstellen komplexer Software in C++ enthält, um reale Probleme zu lösen. Es ist ein praktisches Werkzeug für einen Computer-Vision-Entwickler, um schnell Computer-Vision-Prototypen zu entwickeln.

Dlib bietet eine mächtige Detektionsmethode HOG (Histogram of Oriented Gradients). HOG-Merkmale werden aus einer großen Anzahl von Gesichtsbildern extrahiert, die als Teil des Erkennungsmechanismus verwendet werden. Diese HOG-Merkmale werden dann zusammen für ein Gesicht/einen Benutzer gekennzeichnet und ein Support Vector Machine (SVM)-Modell wird trainiert, um Gesichter vorherzusagen, die in das System eingegeben werden.

Außerdem wurde Dlib eine modernere Methode MMOD integriert. MMOD (Max-Margin Object Detection) ist ein robuster und zuverlässiger, GPU-beschleunigter Gesichtsdetektor, der ein Convolutional Neural Network (CNN) nutzt und weitaus besser in der Lage ist, Gesichter aus dunklen Winkeln und

unter schwierigen Bedingungen zu erfassen, was sich für die gelegentliche Überwachung und Stadtanalyse eignet.

3 Objekterkennung

3.1 SSD MobileNet v2

MobileNet[1] ist ein Objektdetektor, der 2017 als effiziente CNN-Architektur für mobile und Embedded Vision-Anwendungen veröffentlicht wurde. Diese Architektur verwendet bewährte, in der Tiefe trennbare Faltungen, um leichte tiefe neuronale Netze aufzubauen.

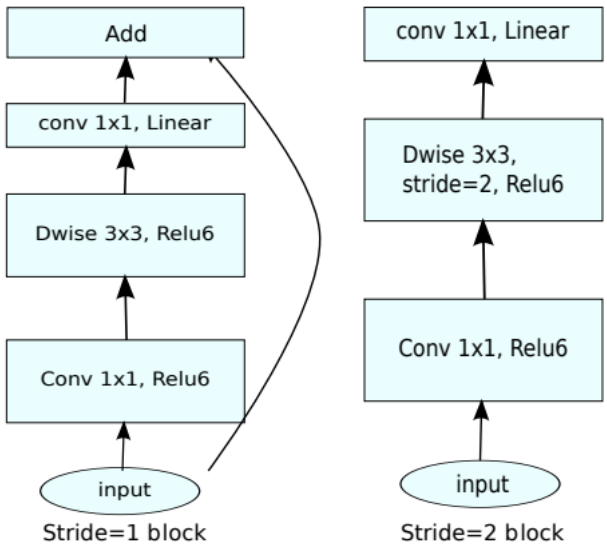


Abbildung 2: Mobilenet v2 Modellstruktur

4 Sprachassistent

4.1 SpeechRecognition

SpeechRecognition ist ein Open-Source-Modul für die Spracherkennung (Sprache zu Text) Sin Python mit Unterstützung für mehrere bekannte Werkzeuge und APIs im Online- und Offline-Modus, z.B. CMU Spinx (offline), Google Speech Recognition usw.

[1] https://paperswithcode.com/method/mobilenetv2#

Ein allgemeines Spracherkennungssystem wurde entwickelt, um die unten genannten Aufgaben auszuführen und kann leicht mit einer Standard-Datenanalysearchitektur korreliert werden:

- ✓ Das Erfassen von Sprache (Wörter, Sätze, Phrasen), die von einem Menschen gegeben werden. Sie können sich dies als den Teil der Datenerfassung in jedem allgemeinen Arbeitsablauf für maschinelles Lernen vorstellen.
- ✓ Transformieren von Audiofrequenzen, um sie maschinenfertig zu machen. Dieser Prozess ist der Teil der Datenvorverarbeitung, bei dem wir die Merkmale der Daten bereinigen, damit die Maschine sie verarbeiten kann.
- ✓ Anwendung von Natural Language Processing (NLP) auf die erfassten Daten, um den Sprachinhalt zu verstehen.
- ✓ Synthese der erkannten Wörter, um der Maschine zu helfen, einen ähnlichen Dialekt zu sprechen

4.2 Google text to speech (gTTS)

Die Text-to-Speech-Konvertierungstechnologie – besteht nicht darin, Maschinen einfach sprechen zu lassen, sondern sie wie Menschen unterschiedlichen Alters und Geschlechts klingen zu lassen. Perspektivisch werden wir in der Lage sein, Hörbücher und Nachrichten mit maschineller Stimme im Fernsehen zu hören oder mit Assistenten zu kommunizieren, ohne den Unterschied zu bemerken.

gTTS ist eine bekannte Text-to-Speech-API von Google. Es kann den eingegebenen Text in Audio umwandelt, das als mp3-/wav-Datei gespeichert werden kann.

Die Hauptmerkmale von gTTS:

- ✓ Anpassbarer sprachspezifischer Satz-Tokenizer, der das Lesen von Texten in unbegrenzter Länge ermöglicht, während gleichzeitig die richtige Intonation, Abkürzungen, Dezimalzahlen und mehr beibehalten werden.
- ✓ Anpassbare Textvorprozessoren, die beispielsweise Aussprachekorrekturen vornehmen können.

- ✓ Automatischer Abruf der unterstützten Sprachen.

IV Ellie – ein humanoider Roboter

1 Ellie – Anfangszustand

Im Labor standen bereits ein schönen grob gefertigten Ellie-Kopf und ein Ellie-Arme zur Verfügung.

Abbildung 3: Ellie-Arme

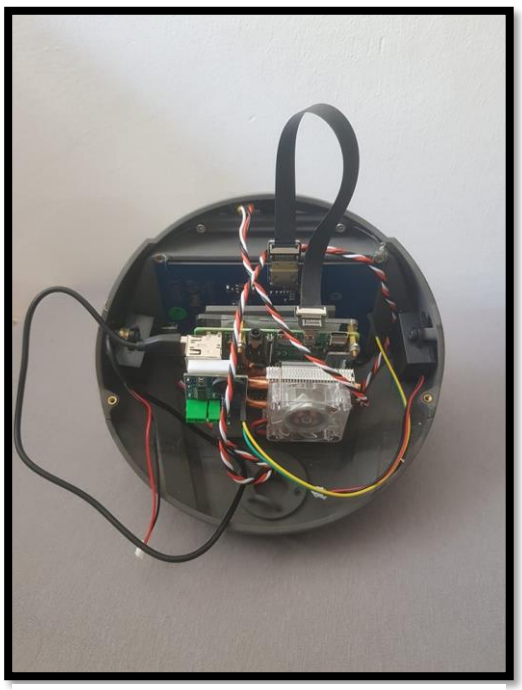


Abbildung 4: Ellie-Kopf

2 Herausforderungen

Die beide funktionierten noch nicht, deswegen wurden wir im Rahmen dieser Projektarbeit beauftragt, die Roboterfunktionen, soweit zu implementieren, dass Ellie sich als eine Assistentin verhalten könnte.

Die Arbeit selbst können in folgende Hauptkategorien, Module unterteilt werden:

- ➢ Gesichtserkennung: Ellie sollte sich ein paar Leute, Mitarbeiter erkennen.
- ➢ Objektserkennung: Sie sollte auch fähig sein, ein paar Dinge, Objekte zu unterschieden.
- ➢ Sprachassistent: Es ist für sie möglich, manche Frage auf Deutsch zu antworten.

- Augen-Animation: Die Augen-animation hilft ihnen, lebhaft auszusehen, und ihre Gefühle zu äußern.
- Motionsimplementierung: Nachdem sie etwas sieht, oder anhört, sollte sie eine bestimmte Motion implementieren. Die integrierte Inverse-Kinematik (IK) Motion, Motoren sollten auch von außen angesteuert werden.
- Informationsbildschirm: Hier sollen die Informationen oder eine Website angezeigt werden.

Alle Module sollen, wenn möglich mit ROS2 verbunden, damit sie mit anderen IOT Sachen kommunizieren kann.

V Projektdurchführung

Es ist zu erwähnen, dass die Module mit ROS2 kommunikationsfähig sein soll. Deswegen haben wir ein Konzept für das ganze Project zu planen. Für jedes Modul wurde ein ROS2 Paket erstellt. Es funktioniert unter der ROS2 Paketstruktur. Die ist vielleicht für weitere Entwicklung nachvollziehbar.

1 EllieEyes

1.1 Gesichtserkennung

Diese Module verwendet Dlib als Kern für Facical-Detektion[2] und OpenCV für Vor- und Nachverarbeitung von Bildern.
Spezifisch, OpenCv ändert die Größen von dem Eingangsbild, der direkt aus Kamera aufgenommen wird. Die eingestellte Größe ist 640px für Breite und 480px für die Höhe. Dilib wird das vorverarbeitete Bild analysieren und anerkannte Leute mit bestimmtem Schwellwert (hier ist 0.6) und ihre Gesichtsposition ausgeben. OpenCv zeichnet ein Begrenzungsrahmen anhand der Positionen und ein Label dafür.
Dlib markiert die Punkte auf dem Gesicht und vergleicht sie mit den Punkten auf dem registrierten Gesicht. Indem sie Abstand zwischen Punkte rechnet.

[2] https://en.wikipedia.org/wiki/Face_detection

Das beste Ergebnis wird ausgeben, wenn seine Wertung größer als der Schwellwert ist, sonst wird keines zurückgeliefert.

Um ein Gesicht zu registrieren, erstellen wir einfach einen Ordner, in dem die Gesichtsbilder liegen, Dieser Ordner muss im „data“ Ordner liegen. Der Name ist die Identity von Gesichte.

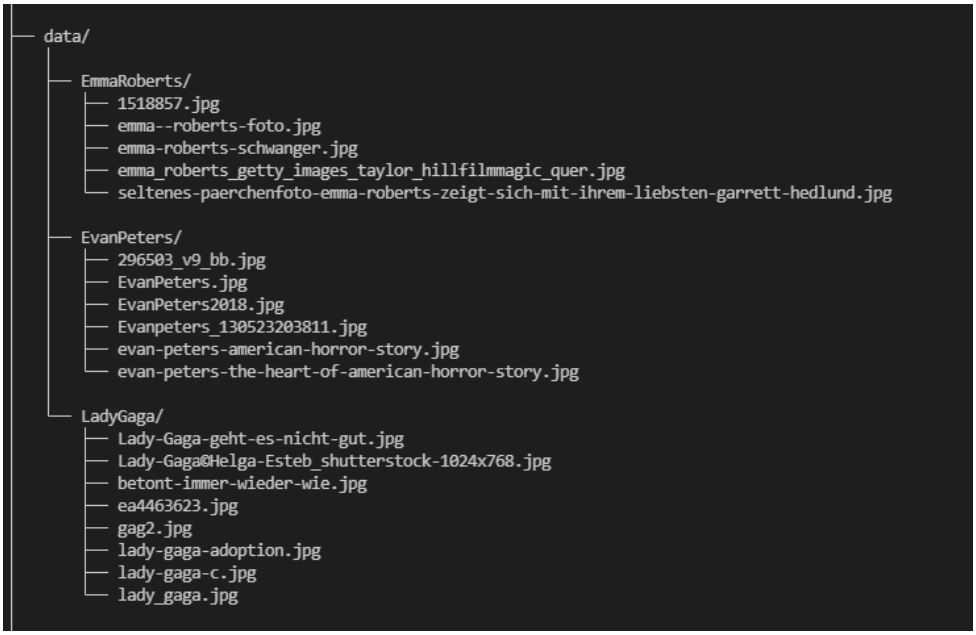


Abbildung 5: Beispiel für Data-Ordner

Hier in diesem Beispiel wurde drei sehr bekannte Leute registriert. Jeder Ordner erhält die eigene Gesichtsbilder. Die Anzahl von Bildern soll wegen dir Einschränkung von Hardware nicht so groß sein. Zum ersten Mal, nach dem ein neues Gesicht registriert wurde, dauert das Prozess ein bisschen länger, weil die neuen Bilder verschlüsselt und für das nächste Mal speichert werden müssen.

1.2 Objektserkennung

Für Objektsdetektieren wir haben Mobilenet v2 Modell angewandt. Das Trainieren von dem eigenen Model ist aufwändig, Es benötigt ein mächtigen GPU integrierten Rechner, um das Modell zu trainieren, sonst kann es bis Tage lang dauern. Mobilenet v2 wurde mit COCO Dataset[3] trainiert und kann auf Raspberry funktionieren.

[3] https://cocodataset.org/#home (2021)

Die Bilder aus Kamera werden auch mit OpenCV in Vorverarbeitung durchführen, bevor sie vom Modell bearbeiten werden. Die Ausgaben sind auch die detektiere Objekte in Rahmen und ihre Positionen.
Die Objekte, die Ellie unterscheiden kann, liegt in der File „ellie_eyes/object_detection_lite/tmp/coco:labels.txt“ z.B. „person“, „bicyle“, „car“ usw.

1.3 ROS2 Integration

Objects-, Gesichtserkennung benutzen beide vorverarbeitete Bilder, deswegen wurde ein ROS2 Node erstellt, die das Kamerarahmen nimmt, verarbeitet und es nach beiden Modulen liefert. „Ellie_eyes“ Node ermöglicht sowohl die originalen Bilder als auch die prognostizierten Bilder in ROS2_Image.msg zu veröffentlichen. Andere Sachen z.B. anerkannte Leute, Objekte finden wir auch in eigene Topics. Parameters können wir auch über ROS2 einstellen oder modifizieren.

Führen Sie bitte den Befehl, um ellie_node zu starten

```
ros2 run ellie_eyes start
```

1.4 Ergebnis

Das Ergebnis für die Objekts- und Gesichtserkennung ist ziemlich gut. Die Verarbeitungszeit auf Raspberry ist ungefähr 1 Frame pro Sekunde.

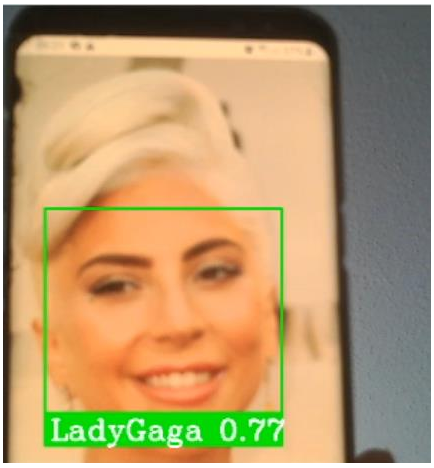


Abbildung 7: Gesichtserkennung

Abbildung 6: Objekterkennung

In diesem Beispiel wurde ein Gesicht mit der Genauigkeit von 77% und ein Objekt mit der Genauigkeit von 63 % erkannt.

2 EllieVoice

2.1 EllieVoice-Modell

Für EllieVoice wurde es den folgenden Arbeitsablauf aufgebaut.

Da wir für ChatBot ein auf Deep Learning basierendes Modell entwickeln werden, benötigen wir Daten, um unser Modell zu trainieren. Wir werden jedoch keine großen Datensätze sammeln oder herunterladen, da es sich um einen einfachen Chatbot handelt. Wir können einfach unseren eigenen Datensatz erstellen, um das Modell zu trainieren. Um diesen Datensatz zu erstellen, müssen wir verstehen, welche Absichten wir trainieren werden. Eine „Absicht“ ist die Absicht des Benutzers, mit einem Chatbot zu interagieren, oder die Absicht hinter jeder Nachricht, die der Chatbot von einem bestimmten Benutzer erhält.
Eine JSON-Datei namens „intents.json“ wurde erstellt, die diese Daten wie folgt enthält.

```
{"intents": [      You, 3 months ago • Initiate
    {"tag": "greeting",
     "patterns": ["Grüß Gott!","Guten Tag","Guten Morgen", "Hallo", "Hi"
     "responses": ["Grüß Gott!","Guten Tag", "Hallo", "Hi", "Guten Morge
     "context_set": ""
    },
    {"tag": "goodbye",
     "patterns": ["Schönen Abend noch", "auf Wiederschauen", "auf Wieder
     "responses": ["Schönen Abend noch", "auf Wiederschauen", "auf Wiede
    },
    {"tag": "thanks",
     "patterns": ["Danke schön", "Danke", "vielen Dank", "ich bedanke mi
     "responses": ["nichts zu danken", "kein Thema", "kein Problem", "ge
    },
    {"tag": "name",
     "patterns": ["wer bist du", "Wie heißt du", "was ist deine name"],
     "responses": ["Ich bin Ellie, der Informationsroboter der Technisch
    },
    {"tag": "story",
     "patterns": ["Warum heißt du Elli", "Was bedeutet Elli","Warum Elli
     "responses": ["Die Frau von Georg Simon Ohm hieß Elisabeth"]
    },
    {"tag": "library",
     "patterns": ["Wo ist hier die Bibliothek", "wo kann ich die Bibliot
     "responses": ["Gleich hier im Gebäude!"]
```

Abbildung 8: EllieVoice-Intents

```
net = tflearn.input_data(shape=[None, len(train_x[0])])
net = tflearn.fully_connected(net, 8)
net = tflearn.fully_connected(net, 16)
net = tflearn.dropout(net,0.5)
net = tflearn.fully_connected(net, len(train_y[0]), activation="softmax")
net = tflearn.regression(net)
```

Abbildung 9: Eine einfache Neuronale Netzwerkarchitektur

Die Eingangsdaten müssen vektorisieren werden, indem wir die Klasse „Tokenizer“ verwenden, und es ermöglicht uns, die Größe unseres Vokabulars auf eine bestimmte Anzahl zu begrenzen. Wenn wir diese Klasse für die Textvorverarbeitungsaufgabe verwenden, werden standardmäßig alle Satzzeichen entfernt, wodurch die Texte in durch Leerzeichen getrennte Wortfolgen umgewandelt werden, und diese Folgen werden dann in Token-Listen aufgeteilt. Sie werden dann indiziert oder vektorisiert.

Die Vorverarbeitung von Sprache zu Text wurde von „speech_recognition“ verantwortet. Eine eigene Struktur wurde auch aufgebaut, damit es in Multithreading problemlos arbeiten kann. Die Textausgabe oder die Antworte von Chatbot wurde mit „pydub“- AudioSegment bearbeitet, damit die Sprache mehr natürlicher anhört.

2.2 ROS2 Integration

ExllieVocie wurde auch mit ROS2 integriert. Was sie gehört hat und ihre Antwort werde in Topic „ellie/communication“ veröffentlicht. Mit dem ROS2 Service „speak“ kann Ellie einen Text lesen. Der Befehl ist wie folgender:

```
ros2 service call /ellie/speak  ellie_msgs/srv/String {"request: 'Hallo'}
```

```
(venv) trungson@ubuntu:~/EllieHumanoid$ ros2 topic echo /ellie/ellie/communication
listen: wie heißt du
response: Ich bin Ellie, der Informationsroboter der Technischen Hochschule Nürnberg
---
```

Abbildung 10: Communication-Topic

2.3 Ergebnis

Obwohl das Chatbot Modell ziemlich kein ist, funktioniert es ganz gut, besonders für Raspberry, das niedrige Rechenleistung hat. Die Reaktionszeit liegt unter eine Sekunde für da ganze Prozess. Wir können in Zukunft ein Chatbot mit „Wake-word“ wie Alexa entwickeln.

3 EllieScreens

In diesem Paket wurde die Funktionen für die Darstellung des Kopf- und Brustbildschirm implementiert.

3.1 Augenanimation

Die Augenanimationen wurde mit Hilfe von After Effect erstellt. Es beschreibt ein paar Motion von den Augen z.B glücklich, wütend usw.

Die Animationen wurde in .gif Daten gespeichert und durch Qt5 wiedergespielt. Sie werden zufällig angezeigt, wenn keinen besonderen Befehl zugeschickt wird.

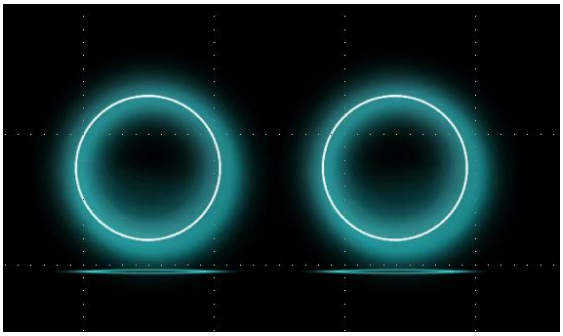


Abbildung 11: Augenanimation - Idle

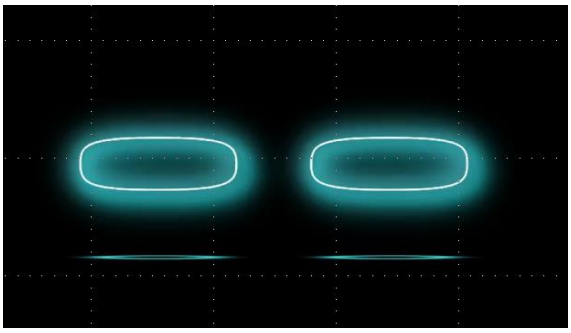


Abbildung 12: Augenanimation - Blinken

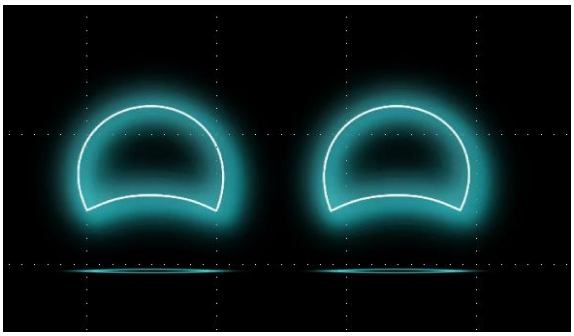


Abbildung 14: Augenanimation - glücklich

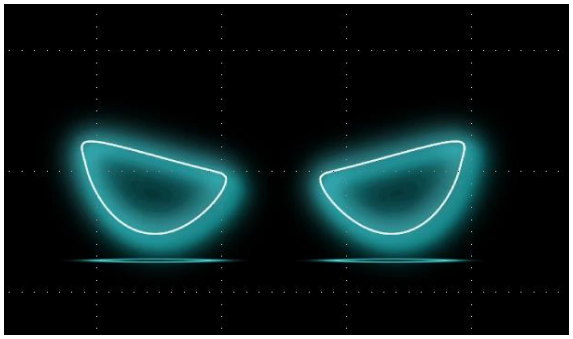


Abbildung 13: Augenanimation - wütend

3.2 Informationsbildschirm

Der Informationsbildschirm stellt standardmäßig die Webseite von der TH Nürnberg https://www.th-nuernberg.de/ dar. Mit dem Touchscreen können Benutzer die Information suchen. Der Monitor kehrt automatisch nach einem Zeitintervall auf die standardmäßige Webseite wieder, wenn er keine Interaktion erkannt. Diese Funktion wurde mit QT5 Web-Engine implementiert.

Abbildung 15: EllieScreens - Informationsmonitor

3.3 ROS2 Integration

Wie andere Pakete wurde EllieScreens auch für ROS2 umstrukturiert. Der Benutzer kann die Augenanimationen oder darzustellende Webseite von außen durch ROS2 Services einstellen.

Öffnen eine Website mit dem Befehl:

```
(venv) trungson@ubuntu:~/EllieHumanoid$ ros2 service call /ellie/open_url ellie_msgs/srv/String "{request: 'https://www.google.com/'}"
requester: making request: ellie_msgs.srv.String_Request(request='https://www.google.com/')

response:
ellie_msgs.srv.String_Response(response='open : https://www.google.com/')
```

Ändern die Augenanimation:

```
(venv) trungson@ubuntu:~/EllieHumanoid$ ros2 service call /ellie/change_eyes_motion ellie_msgs/srv/String "{request: 'smile'}"
waiting for service to become available...
requester: making request: ellie_msgs.srv.String_Request(request='smile')

response:
ellie_msgs.srv.String_Response(response='eyes motion changed !')
```

3.4 Ergebnis

Nur auf eine Sache müssen wir beachten, dass die QT5 WebEngine mit Google Chrome arbeitet, die für das Problem von hohen RAM-Verbrauch bekannt. Aber in diesem Fall ist alles gut geläuft.

4 EllieArms

EllieArms ist ein Paket für die Kommunikation mit den Dynamixel Motoren, Wir können den „low-level“ Befehl an Motoren schicken oder ihren aktuellen Zustand lesen. Ellie enthält insgesamt 13 Motoren, jeder hat eigene Identifiziernummer.

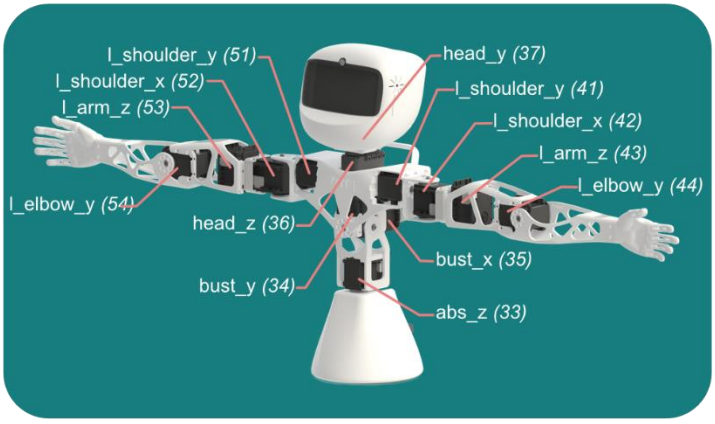


Abbildung 16: Ellie - Motoren

Innerhalb von 13 Motoren gibt es 11 Servomotoren Dynamixel MX-28 AT und 2 Dynamixel AX-12A. Mit Hilfe vom Dynamixel SDK wurde eine Kommunikationsschnittstelle erstellt. Zur Steuerung von DYNAMIXEL sollte die Kommunikation gemäß dem Protokoll von DYNAMIXEL aufgebaut werden. Es gibt die Versionen 1.0 und 2.0 des DYNAMIXEL-Protokolls. Das DYNAMIXEL SDK unterstützt beides, und der Benutzer kann mit dem DYNAMIXEL SDK beide Protokolle gleichzeitig verwenden. Obwohl die Version 2.0 mit der Synchronisierung-Lesen Funktion mächtiger ist, wurde die Version 1.0 angewandt. Weil die Version 2.0 die AX-12A nicht unterstützt. Um die Lesegeschwindigkeit zu beschleunigen, sollte die USB-Latenz immer minimiert werden. Ansonsten wird es ein Problem beim Werte Aufzeichnen sein.

```
# cat /sys/bus/usb-serial/devices/ttyUSB0/latency_timer
 16
 # echo 1 > /sys/bus/usb-serial/devices/ttyUSB0/latency_timer
 # cat /sys/bus/usb-serial/devices/ttyUSB0/latency_timer
 1
```

Für den Motor ist es nötig, dass wir eine Winkelbegrenzung haben, damit die falsche Bewegung den Aufbau von dem Roboter nicht zerstört. Die Ellie-Arms wurden auch mit einem Versatz kalibriert, um zu sicher, dass es mit den logischen Funktionen richtig funktioniert. Die Drehrichtung muss auch berücksichtigt werden.

```
self.head_y = Motor(
    name="head_y",
    offset=-20,
    model=MotorType.AX_12,
    id=37,
    angle_limit=[-80, 30],
    direct=False
)
```

Abbildung 17: Motor-Klasse

Die Variable „direct“ heißt, ob der Motor im Uhrzeigersinn oder gegen den Uhrzeigersinn gebaut wurde. Da folgende Bild ist die Anfangsposition, bei der alle Motorpositionen gleich 0 sind.

Abbildung 18: Ellie -Anfangsposition

4.1 Inverse-Kinematic

Es wurde auch ein Inverser-Kinematik – Ikpy[4] Modul integriert. Mit Hilfe von diesem Modul wir können die Position der Gelenke berechnen, wenn wie ein Zielposition eingeben. Es kann automatisch die kinematische Kette aus einer URDF-Datei für die Bewegungsplanung einlesen.

4.2 Trajectory Planning

Die Trajektories können einfach mit dem linearen Verhalten laufen, aber es ist nicht optimal. Deswegen haben wir die MinJerk Trajectory[5] angewandt. Die Positionswerte werden optimal nach der Zeit genegiert, damit eine Bewegung von A nach B glatter ausgeführt werden kann.

4.3 Ellie Behavior

Die Motion von Elli-Arm ist kompliziert, dass die Ansteuerung nicht einfach durch Inverse-Kinematik Methode implementiert werden kann. Die Zielposition ist zu schwer zu identifizieren und es ist auch sehr zufällig, weil das Inverse-Kinematik Problem nicht nur eine Lösung, sondern mehrere Lösungen ergibt. Aus diesem Grund ist es meiner Meinung nach in diesem Fall am besten, die Bewegung zuerst aufzunehmen und dann erneut abzuspielen. Die Motion wird in der Json-Datei speichern, die Position und die Geschwindigkeit von Gelenken mit Zeitstempel erhält.


```
"positions": {
    "0.0": {
        "head_z": [
            -4.55,
            0.0
        ],
        "head_y": [
            18.09,
            0.0
        ],
        "l_shoulder_y": [
            1.0300000000000011,
            0.0
        ],
        "l_shoulder_x": [
            -2.0400000000000063,
            0.0
        ],
```


Abbildung 19: Behavior Json-Datei

[4] https://github.com/Phylliade/ikpy (2021)
[5] https://courses.shadmehrlab.org/Shortcourse/minimumjerk.pdf (2021)

Der Schlüssel dieser Methode ist das KD-Dictionary[6]. Wenn wir einen Schlüssel finden, gibt das Dictionary den Wert mit dem kleinsten Schlüsselabstand an.

4.4 ROS2 Integration

EllieArms unterstützt ein paar Services, um die Motion aufzuzeichnen oder ein Behavior abzuspielen.

Ausführungsverhalten:

```
(venv) trungson@ubuntu:~/EllieHumanoid$ ros2 service call /ellie/execute_behavior ellie_msgs/srv/String "{request: 'behave_hello'}"
requester: making request: ellie_msgs.srv.String_Request(request='behave_hello')

response:
ellie_msgs.srv.String_Response(response='Executed behavior behave_hello')
```

das Aufzeichnen starten:

```
ros2 service call /ellie/start ellie_msgs/srv/String "{request: 'behavior_name' }"
```

das Aufzeichnen anhalten:

```
ros2 service call /ellie/stop ellie_msgs/srv/String "{request: ''}"
```

das Aufzeichnen fortsetzen:

```
ros2 service call /ellie/resume ellie_msgs/srv/String "{request: ''}"
```

das Aufzeichnen speichern:

```
ros2 service call /ellie/resume ellie_msgs/srv/String "{request: 'name_of_directory or none'}"
```

das Aufzeichnen abspielen:

```
ros2 service call /ellie/replay ellie_msgs/srv/String "{request: ''}"
```

4.5 Ergebnis

Das gespeicherte Aufzeichnen können richtig abspielen. Die ROS2 Services wurde auch getestet, und konnte problemlos durchgeführt werden.

5 Ellie und EllieMessage

EllieMessage ist ein ROS2 Paket, das alle nötige „message“, „action“, „service“ -Type erhält. Ellie ist ein anderes Paket für alle vorgestellt Module zu kombinieren.

[6] https://stackoverflow.com/questions/29094458/find-integer-nearest-neighbour-in-a-dict (2021)

Wir können so einstellen, wenn Ellie ein registriertes Gesicht sieht, soll sie eine bestimmte Motion implementieren, z.B. ändert die Augenanimation, öffnet andere Webseite oder implementiert ein Behavior oder sagt etwas. Alles können wir in behaviors.json-Datei hinzufügen.

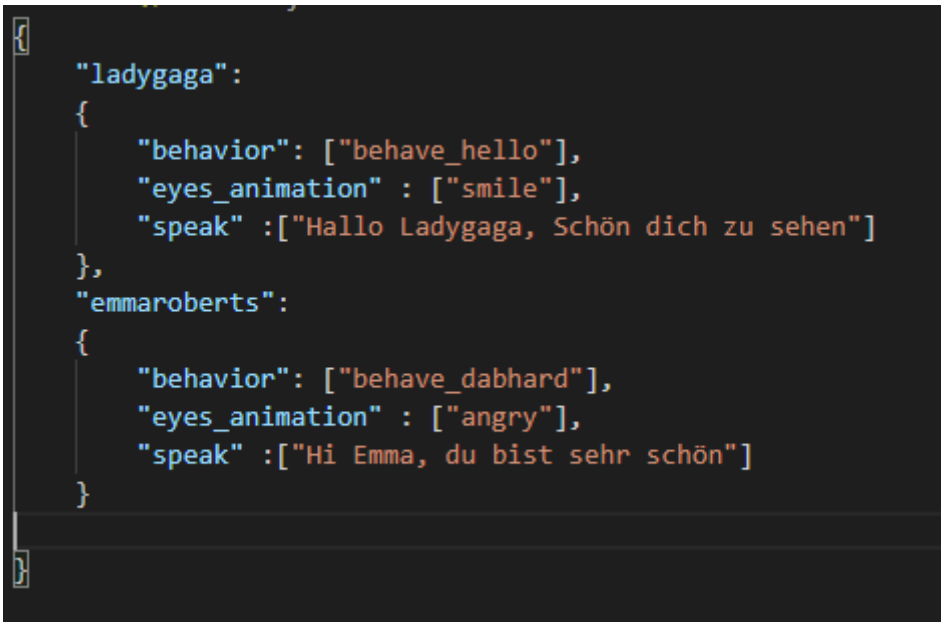
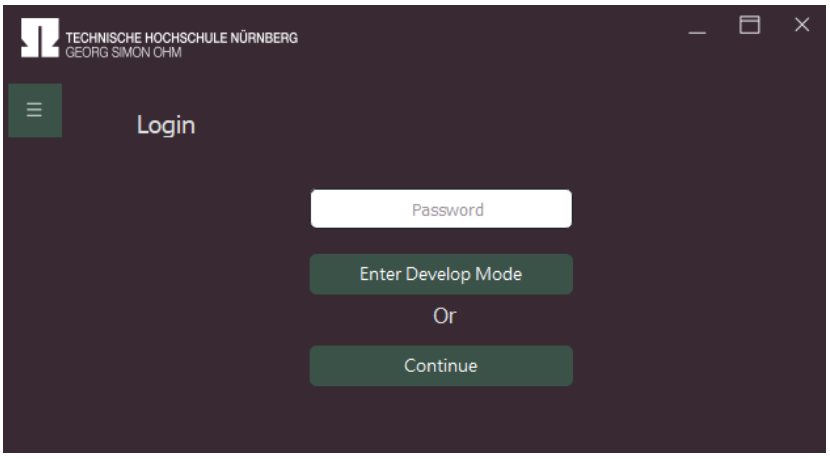


Abbildung 20: Behaviors.json

Das Bild da oben heißt, wenn ellie „Ladygaga“ sieht, tut sie die Motion „behave_hello“ , wechselt die Augenanimation zu “smile“ und sagt „Hallo Ladygaga, Schön dich zu sehen“. Das ist einer einfache Arbeitsablauf, wie Ellie sich reagiert, wenn sie eine bestimmte Person sieht. Nach der Begrüßung, diese Person wird in eine „Blacklist“ hinzugefügt und wird wieder verfügbar nach einer bestimmten Zeit, damit Ellie diese Person nicht vielmals gegrüßt.

6 Das erste Konzept

Das erste Konzept wurde nicht mit ROS2 funktioniert. Wir haben mit Hilfe von QT5 eine UI erstellen und der Benutzer kann direkt auf dem Informationsbildschirm arbeiten.



Anmeldungsseite:

Nur Entwickler kann die Änderung durchführen.

Abbildung 21: Anmeldungsseite

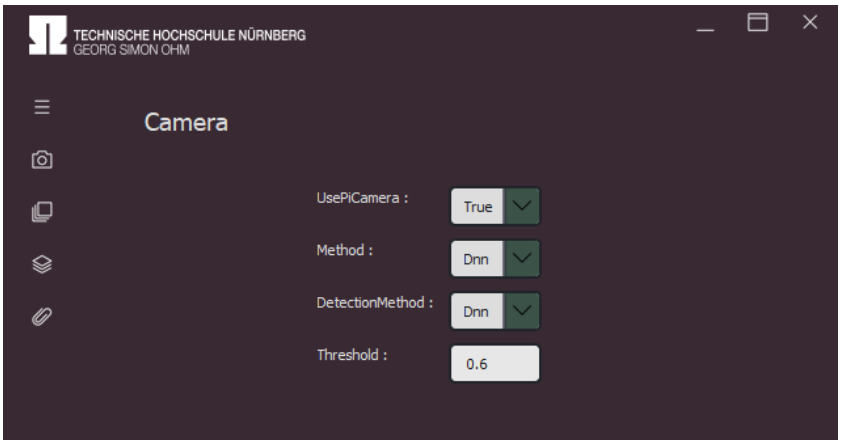


Abbildung 22: Kameraeinstellung

Kameraeinstellung:

Wir können hier wichtige Parameter für die Kamera einstellen.

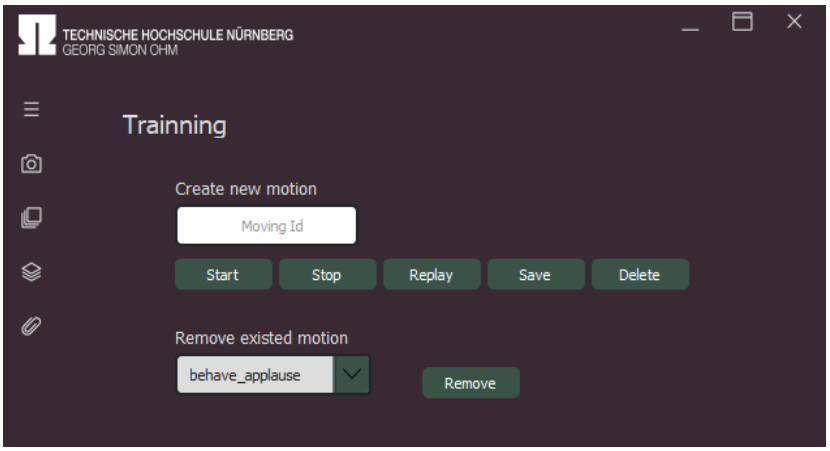


Abbildung 23: Seite für das Aufzeichnen

Hier sind die Funktionen, um eine Motion aufzuzeichnen.

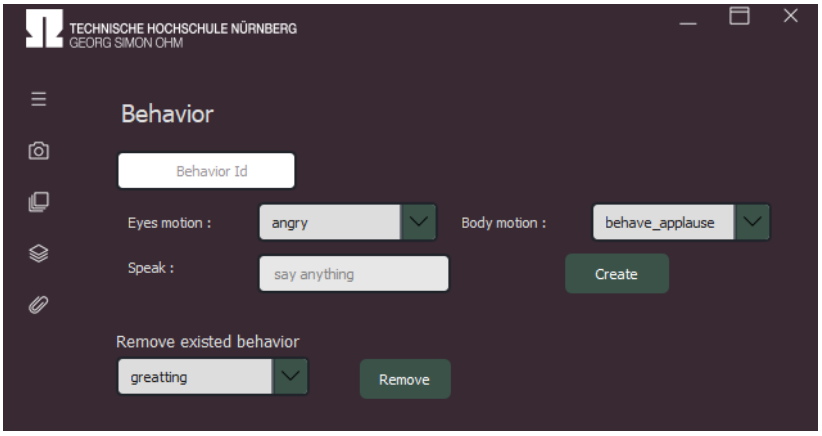


Abbildung 24: Einstellung des Verhaltens

Eine Kombination von der Augenanimation, der Motion können hier erstellt werden.

VI Fazit

In meine Sicht funktioniert die Ellie ziemlich gut. Sie können Gesichte oder Objekte erkennen. Aber steht noch das „On-Board“ Problem. Raspberry ist ziemlich schwach für die KI-Sache. Dazu ist die Implementierung von mehreren ROS“ Nodes sinkt seine Leitung auch runter.

Für die Sprachassistent-Funktion können wir mit „wake-word“ arbeiten oder ein „Alexa“ dazu integrieren. Die Daten für das Trainieren sind noch nicht genug.

Es ist auch möglich, die Lesegeschwindigkeit vom aktuellen Zustand aus den dynamixel Motoren zu erhöhen, indem wir direkt „Serial port“-Paket, also nicht Dynamixel SDk, anwenden. Oder wir können auch zwei AX Motoren ersetzen, und mit Dynamixel Version 2 arbeiten. Je höher die Lesegeschwindigkeit, desto werden mehrere Positionen aufgezeichnet. Es führt zu einer glatten Bewegung.

TECHNISCHE HOCHSCHULE NÜRNBERG
GEORG SIMON OHM

Abbildungsverzeichnis

Literaturverzeichnis

(2021). Von https://stackoverflow.com/questions/29094458/find-integer-nearest-neighbour-in-a-dict abgerufen

(2021). Von https://stackoverflow.com/questions/29094458/find-integer-nearest-neighbour-in-a-dict abgerufen

(2021). Von https://cocodataset.org abgerufen

(2021). Von https://cocodataset.org abgerufen

(2021). Von https://courses.shadmehrlab.org/Shortcourse/minimumjerk.pdf abgerufen

(2021). Von https://github.com/Phylliade/ikpy abgerufen

(2021). Von https://github.com/poppy-project/pypot abgerufen

(2021). Von https://docs.poppy-project.org/en/ abgerufen

Sandler, M. (kein Datum). *MobileNetV2: Inverted Residuals and Linear Bottlenecks.*