

---

# **Yoga Trainer mit Pose Estimation**

**Stephan Seliner**

**Alex Koller**

---



## **Bericht Fachmodul**

Systemtechnik Fachrichtung Ingenieurinformatik  
OST – Ostschweizer Fachhochschule

Referent: Prof. Dr. Christoph Würsch  
Korreferent: Nicola Notari

St. Gallen, den 07.01.2022

---

# Inhaltsverzeichnis

<b>Abbildungsverzeichnis</b>	<b>III</b>
<b>Tabellenverzeichnis</b>	<b>IV</b>
<b>1 Einleitung</b>	<b>1</b>
1.1 Motivation . . . . .	1
1.2 Aufgabenstellung . . . . .	1
<b>2 Pose Estimation</b>	<b>2</b>
2.1 Klassische Ansätze . . . . .	2
2.2 Ansätze mit Deep Learning . . . . .	4
2.2.1 DeepPose . . . . .	5
2.2.2 Coarse Heatmap Regression Model . . . . .	6
2.2.3 Convolutional Pose Machine . . . . .	6
<b>3 Framework</b>	<b>8</b>
3.1 MediaPipe . . . . .	8
3.1.1 BazePose . . . . .	9
3.2 TensorFlow . . . . .	10
3.2.1 MoveNet . . . . .	11
3.3 TensorFlowJS Pose-Detection API . . . . .	12
3.4 Entscheid . . . . .	13
<b>4 Trainer</b>	<b>15</b>
4.1 Anwendung . . . . .	15
4.2 Anforderung an die Anwendung . . . . .	15
4.3 Benutzerfunktionen der Anwendung . . . . .	15
4.3.1 Learn und Train . . . . .	15
4.3.2 Change Trainingsets . . . . .	16
4.3.3 View History . . . . .	16
4.4 Administratorfunktionen der Anwendung . . . . .	17
4.5 Erkennung der Yoga Asanas . . . . .	17
<b>5 Arbeiten mit Mediapipe</b>	<b>18</b>
5.1 Berechnung des Winkels . . . . .	18
5.2 Statistische Auswertung . . . . .	19
<b>6 Meilensteine der Bachelorarbeit</b>	<b>21</b>
6.1 Vorarbeit Programmierung . . . . .	21
6.2 GUI implementiert . . . . .	22
6.3 Klassifizierung . . . . .	22

---

6.4	Datenspeicherung . . . . .	22
6.5	Metriken . . . . .	22
6.6	Feedback . . . . .	22
6.7	Learn Funktion . . . . .	22
6.8	Trainingssets . . . . .	23
6.9	Auswertung der Daten . . . . .	23
6.10	Abgabe Bachelorarbeit . . . . .	23
	<b>Literaturverzeichnis</b>	<b>24</b>
	<b>Abkürzungsverzeichnis</b>	<b>24</b>
	<b>Anhang A: Fachmodulauftrag</b>	<b>25</b>

---

## Abbildungsverzeichnis

1	Zwei Beispiele mit klassischem Ansatz, Strukturmodelle [MingGuangYong.2019]	3
2	Heatmap-Regression, a) Originalbild b) erzeugte Heatmap c) Erkennungsergebnis [.11.12.2021] . . . . .	5
3	Links: Darstellung der Deep Neural Network (DNN)-basierten Posenregression. Rechts: im Stadium wird ein verfeinernder Regressor auf ein Teilbild angewendet. [deepPoseToshev] . . . . .	5
4	Übersicht der kaskadierten Architektur des Coarse Heatmap Regression Models [Tompson.16.11.2014] . . . . .	6
5	Architektur einer Convolutional Pose Machine (CPM) [Wei.30.01.2016] . . . . .	7
6	MediaPipe Komponenten, Beispielgrafik für Objekterkennung [MingGuangYong.2019]	9
7	Übersicht der Pipeline [PoseGoogleBlog] . . . . .	10
8	Vitruvianischer Mann, der anhand von zwei virtuellen Schlüsselpunkten ausgerichtet wird. [PoseGoogleBlog] . . . . .	10
9	Die 33 Keypoints von BlazePose [PoseGoogleBlog] . . . . .	11
10	Prozessablauf von MoveNet [MoveNetTFJS] . . . . .	12
11	Use-Case Diagramm . . . . .	16
12	Asanas im Yoga82 Datensatz [verma2020yoga] . . . . .	17
13	Winkelberechnung direkt im Videoinput eingeblendet . . . . .	18
14	Funktion in Python für die Winkelberechnung . . . . .	19
15	Frequenz der Auf- und Ab-Bewegung . . . . .	20
16	Duschbiegung bei Liegestützen . . . . .	20

---

## **Tabellenverzeichnis**

1	Auflistung der bewerteten Frameworks . . . . .	14
---	--	----

## **1 Einleitung**

---

# **1 Einleitung**

## **1.1 Motivation**

Die Schätzung der menschlichen Pose ist ein anspruchsvolles Problem in der Disziplin der Computer Vision. Dabei geht es um die Lokalisierung menschlicher Gelenke in einem Bild oder Video, um eine Skelettdarstellung zu erstellen. Die automatische Erkennung der Pose einer Person in einem Bild ist eine schwierige Aufgabe, da sie von einer Reihe von Aspekten wie Massstab und Auflösung des Bildes, Beleuchtungsvariationen, Hintergrundstörungen, Kleidungsvariationen, Umgebung und Interaktion des Menschen mit der Umgebung abhängt. Eine Anwendung der Pose-Schätzung, auch als Pose Estimation bekannt, ist Sport und Fitness. Eine Form der Übung mit komplizierten Körperhaltungen ist Yoga, eine uralte Trainingsmethode, die in Indien ihren Ursprung hat, heute aber wegen ihrer vielen spirituellen, körperlichen und geistigen Vorteile weltweit bekannt ist.

Das Problem beim Yoga ist jedoch, dass es, wie bei jeder anderen Sportart auch, von grösster Wichtigkeit ist, sie richtig zu praktizieren, da jede falsche Haltung während einer Yoga-Sitzung unproduktiv und möglicherweise schädlich sein kann. Daher ist es notwendig, dass ein Lehrer die Sitzung überwacht und die Haltung des Einzelnen korrigiert.

Da nicht alle Zugang zu einem Yogalehrer haben oder über entsprechende Ressourcen verfügen, könnte eine auf künstlicher Intelligenz basierende Anwendung eingesetzt werden, um Yogastellungen zu erkennen und personalisiertes Feedback zu geben, damit der Einzelne seine Form verbessern kann.

## **1.2 Aufgabenstellung**

Das Ziel der Arbeit ist es, eine detaillierte Aufgabenstellung für die Bachelorarbeit zu erarbeiten. Mit Hilfe einer Literaturrecherche soll der aktuelle Stand der Technik der Pose Estimation aufgezeigt werden. Dabei sollen mögliche Schwierigkeiten erkannt und sofern möglich, vorab bereinigt werden. Um diese möglichen Probleme aufzudecken, wird ein Demonstrator mit unterschiedlichen Frameworks, auf verschiedenen Betriebssystemen programmiert.

## 2 Pose Estimation

Pose Estimation ist die Aufgabe, mithilfe eines Machine Learning (ML) Modells die Pose einer Person aus einem Bild oder einem Video zu schätzen, indem die räumlichen Positionen der wichtigsten Körpergelenke (Keypoints) geschätzt werden.

Pose Estimation bezieht sich auf Computer-Vision-Techniken, die menschliche Figuren in Bildern und Videos erkennen, um beispielsweise festzustellen, wo der Ellenbogen einer Person in einem Bild zu sehen ist. Es ist wichtig, sich der Tatsache bewusst zu sein, dass die Posenschätzung lediglich schätzt, wo sich die wichtigsten Körpergelenke befinden, und nicht erkennt, wer sich in einem Bild oder Video befindet.

Die Modelle zur Posenschätzung verwenden ein verarbeitetes Kamerabild als Eingabe und geben Informationen über Keypoints aus. Die erkannten Keypoints werden, abhängig ob 2D oder 3D Pose Estimation angewendet wird, mit den entsprechenden Koordinaten und mit einem Konfidenzwert zwischen 0 und 1 ausgegeben. Der Konfidenzwert gibt die Wahrscheinlichkeit an, dass an dieser Position der Keypoint vorhanden ist.

Diese Ansätze zur Schätzung der Körperhaltung lassen sich in Bottom-up- und Top-down-Methoden unterteilen [[Odemakinde.08.01.2021](#)].

Bei Bottom-up-Methoden wird zunächst jedes einzelne Körpergelenk geschätzt und dann zu einer eindeutigen Pose gruppiert. Top-down-Methoden führen zunächst einen Personendetektor aus und schätzen die Körpergelenke innerhalb der erkannten Bounding Boxes.

### 2.1 Klassische Ansätze

Klassische Ansätze beziehen sich in der Regel auf Techniken und Methoden, die Algorithmen des maschinellen Lernens verwenden. Die meisten dieser Modelle funktionieren gut, wenn das Eingabebild eindeutige und sichtbare Körperteile aufweist. Sie versagen jedoch bei der Erfassung und Modellierung von Körperteilen, die verborgen oder aus einem bestimmten Winkel nicht sichtbar sind. Um diese Probleme zu überwinden, wurden Methoden zur Merkmalsbildung wie Histogrammorientierte Gradienten (HOG), Konturen, Histogramme usw. verwendet [[.2021](#)]. Trotz des Einsatzes dieser Methoden mangelt es den klassischen Modellen an Genauigkeit, Korrelation und Verallgemeinerungsfähigkeit.

Die klassischen Ansätze lassen sich in die Kategorien Erscheinungs- und Strukturmodelle einteilen.

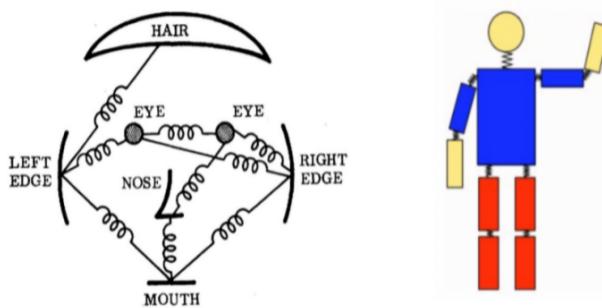
Bei den Erscheinungsmodellen spielen diese erwähnten Methoden zu Merkmalsbildung eine zentrale Rolle. Poselets nutzen genau diesen Ansatz [[Lubomir D. Bourdev.2011](#)]. Sie sind darauf trainiert, auf einen gewissen Ausschnitt des Objekts unter einem bestimmten Blickwinkel und Pose zu reagieren. Es gibt eine Vielzahl von Poselets – ein Frontalgesicht, ein Profilgesicht, eine Kopf-Schulter-Konfiguration und viele Weitere.

## 2 Pose Estimation

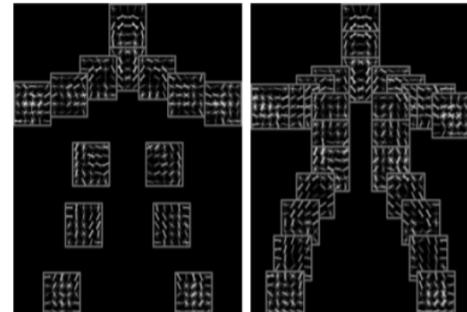
---

Bei den Strukturmodellen ist das „Pictorial Structures Model“ ein weit verbreitetes Modell [Josyula.13.10.2021]. Dieses modelliert die räumlichen Korrelationen starrer Körperteile, indem diese in Form eines baumstrukturierten grafischen Modells ausdrückt werden, um die Lage der Körpergelenke vorherzusagen. Diese räumlichen Verbindungen werden durch die Verwendung von Federn dargestellt, und die Körperteile sind Erscheinungsbildvorlagen, die auf dem Bild basieren. Durch die Parametrisierung der Körperteile mittels Pixelposition und -ausrichtung, kann die resultierende Struktur die Artikulation modellieren.

Ein weiteres Modell dieser Kategorie ist das „Flexible Mixture-of Parts“. Bei diesem Ansatz werden verformbare Teilmodelle verwendet, die eine Sammlung von Vorlagen sind, welche in einem Bild gefunden werden und in einer verformbaren Konfiguration angeordnet sind. Außerdem verfügt jedes Modell über globale und partielle Vorlagen. Die Hauptidee ist, die Verwendung einer Mischung aus kleinen, nicht orientierten Körperteilen zu verwenden, anstelle von verformbaren Vorlagen. Die Gründe dafür haben mit der unterschiedlichen Darstellung von Gliedmassen und Änderungen des Blickwinkels zu tun.



(a) Pictorial Structures Model



(b) Flexible Mixture-of Parts

Abbildung 1: Zwei Beispiele mit klassischem Ansatz, Strukturmodelle  
[MingGuangYong.2019]

### 2.2 Ansätze mit Deep Learning

Durch die enorme Entwicklung von Deep Learning Lösungsansätzen in den letzten Jahren, wurden die klassischen Methoden in verschiedenen Gebieten, darunter auch Objekterkennung oder Bildersegmentierung, übertroffen [**Odemakinde.08.01.2021**]. Diese Deep Learning Techniken brachten signifikante Vorteile und Performancesteigerung in Pose Estimation Aufgaben.

Die klassische Pipeline hat ihre Grenzen, und die Posenschätzung wurde durch Convolutional Neural Networks (CNN)s stark verändert. Mit der Einführung von DeepPose von Toshev et al. [**deepPoseToshev**], begann sich die Forschung auf dem Gebiet der menschlichen Posenschätzung von klassischen Ansätzen auf Deep Learning zu verlagern. Die meisten neueren Systeme zur Posenschätzung haben durchgängig CNN als Hauptbaustein verwendet und damit handgefertigte Merkmale und grafische Modelle weitgehend ersetzt. Diese Strategie hat zu drastischen Verbesserungen bei Standard-Benchmarks geführt [**Babu.12.04.2019**].

Bevor einige dieser Deep-Learning Modelle etwas genauer betrachtet werden, wird nachfolgend das Prinzip der Keypoint- und Heatmap-Regression erläutert [**.11.12.2021**]. Bei der Keypoint-Regression extrahiert das Modell die Keypoints direkt aus den feature maps (gefiltertes Eingangsbild durch CNN), weshalb sie in einigen Referenzen als direkte Regression bezeichnet wird. Wenn mit dieser Methode 17 Schlüsselpunkte in 2D für eine Person geschätzt werden möchten, ist die Ausgabe des Modells ein  $17 \times 2$ -Vektor, der die x- und y-Koordinaten jedes vorhergesagten Keypoints enthält. Viele verschiedene Modelle wurden auf der Grundlage des Keypoint-Regressionsansatzes vorgeschlagen. Eine grosse Schwierigkeit ist jedoch die Empfindlichkeit des Modells, welches zu Instabilität beim Training führt. Diese entsteht, wenn im Modell ein bestimmter Keypoint mit einer Abweichung von wenigen Pixeln von der Grundwahrheit vorhersagt wird. Dies stört den Trainingsprozess und hindert die Konvergenz des Modells zu einer optimalen Lösung.

Um das Problem der Empfindlichkeit und Instabilität zu lösen, wurde die Heatmap-Regression als alternativer Ansatz eingeführt. Im Gegensatz zur vorherigen Methode, bei der die genaue Position jedes Keypoints direkt ermittelt wird, schätzt dieser Ansatz die Wahrscheinlichkeit der Existenz eines Keypoints in jedem Pixel des Bildes. Dieses Modell ist sehr erfolgreich, und viele Arbeiten verwenden Heatmaps anstelle einer direkten Regression. Bei der Umsetzung dieser Methode werden geringfügige Unterschiede bei der Vorhersage von Keypoints ausser Acht gelassen und das Modell kann entspannter trainiert werden.

## 2 Pose Estimation

---

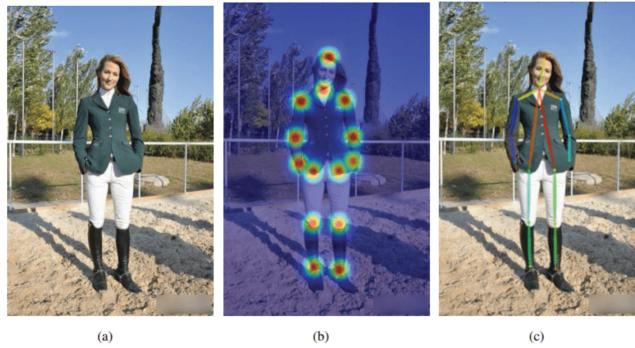


Abbildung 2: Heatmap-Regression, a) Originalbild b) erzeugte Heatmap c)  
Erkennungsergebnis [11.12.2021]

### 2.2.1 DeepPose

DeepPose [[deepPoseToshev](#)] war die erste grössere Arbeit, in der Deep Learning auf die Schätzung der menschlichen Körperhaltung, angewendet wurde. Es erreichte eine solide Performance und übertraf bestehende Modelle. Bei diesem Ansatz wird die Posenschätzung als CNN-basiertes Regressionsproblem für Körpergelenke formuliert. Es wird eine Kaskade solcher Regressoren verwendet, um die Posenschätzungen zu verfeinern und bessere Schätzungen zu erhalten. Ein wichtiger Aspekt dieses Ansatzes ist die ganzheitliche Betrachtung der Pose. Dies ermöglicht es selbst verborgene Gelenke zu schätzen. Die Verwendung von CNNs erlaubt es diese Art von Schlussfolgerungen auf natürliche Weise zu erhalten und überzeugende Ergebnisse zeigen.

Eine interessante Idee, die dieses Modell umsetzt, ist die Verfeinerung der Vorhersagen durch kaskadierte Regressoren. Die anfängliche grobe Pose wird verfeinert und eine bessere Schätzung wird erreicht. Die Bilder werden um das vorhergesagte Gelenk herum beschnitten und der nächsten Stufe zugeführt. Auf diese Weise sehen die nachfolgenden Posenregressoren Bilder mit höherer Auflösung und lernen so Merkmale für feinere Massstäbe, was letztlich zu einer höheren Präzision führt. Dieses Modell wird mit der Verlustfunktion  $\mathcal{L}2$  für die Regression trainiert.



Abbildung 3: Links: Darstellung der DNN-basierten Posenregression. Rechts: im Stadium s wird ein verfeinernder Regressor auf ein Teilbild angewendet.

[[deepPoseToshev](#)]

## 2 Pose Estimation

---

### 2.2.2 Coarse Heatmap Regression Model

Beim Coarse Heatmap Regression Model [Tompson.16.11.2014] werden Heatmaps erzeugt, indem ein Bild parallel durch mehrere Auflösungsblocks läuft, um gleichzeitig Merkmale in verschiedenen Massstäben zu erfassen. Das Ergebnis ist eine diskrete Heatmap anstelle einer kontinuierlichen Regression.

Bei diesem Modell wird eine CNN-Architektur mit mehreren Auflösungen zur Implementierung eines Schiebefenster-Detektors verwendet, um eine grobe Heatmap-Ausgabe zu erzeugen. Die Hauptmotivation besteht darin, die räumliche Genauigkeit wiederherzustellen, die durch das Pooling im Ausgangsmodell verloren gegangen ist. Zu diesem Zweck wird ein zusätzliches CNN zur Verfeinerung der Pose verwendet, dass das Lokalisierungsergebnis der groben Heatmap verfeinert. Im Gegensatz zu einer Standardkaskade von Modellen verwenden diese Modell jedoch vorhandene Faltungsmerkmale wieder. Dies reduziert nicht nur die Anzahl der trainierbaren Parameter in der Kaskade, sondern wirkt auch als Regularisierer für das grobe Heatmap-Modell, da das grobe und das feine Modell gemeinsam trainiert werden.

Im Wesentlichen besteht das Modell aus dem Heatmap basierten Teilmodell für die grobe Lokalisierung, einem Modul zum Abtasten und Zuschneiden der Faltungsmerkmale bei einem bestimmten ( $x, y$ ) für jedes Gelenk, sowie ein zusätzliches Faltungsmodell für die Feinabstimmung. Ein entscheidendes Merkmal dieser Methode ist die gemeinsame Verwendung eines CNN und eines grafischen Modells. Das grafische Modell lernt typische räumliche Beziehungen zwischen Gelenken.

Dieses Modell wird trainiert, indem der Mean Squared Error (MSE) zwischen der vorhergesagten Heatmap und einer Ausgangs-Heatmap minimiert wird.

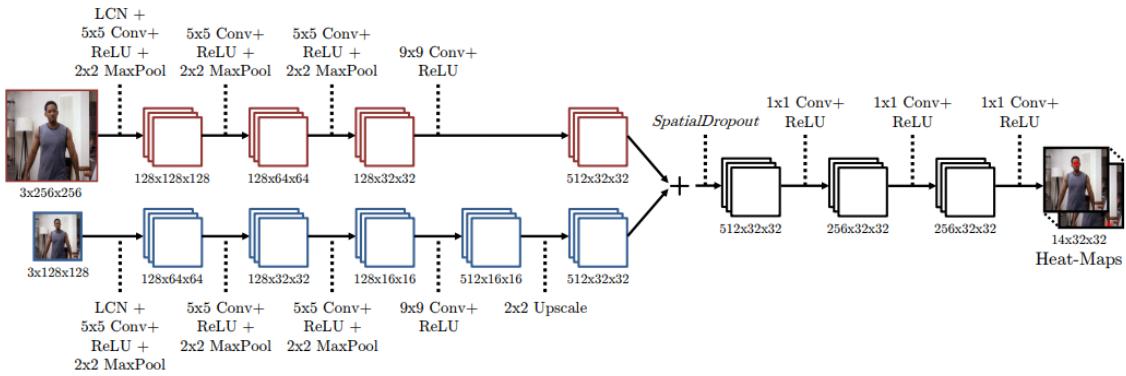


Abbildung 4: Übersicht der kaskadierten Architektur des Coarse Heatmap Regression Models [Tompson.16.11.2014]

### 2.2.3 Convolutional Pose Machine

Eine Pose-Maschine [Wei.30.01.2016] besteht aus einer Folge von Mehrklassen-Prädiktoren, die trainiert werden, um die Lage jedes Körperteils in jeder Hierarchiestufe vorherzu-

## 2 Pose Estimation

---

sagen. Dieses Modell besteht aus einem Modul zur Berechnung von Bildmerkmalen, gefolgt von einem Vorhersagemodul. Diese CPMs sind vollständig differenzierbar und ihre mehrstufige Architektur kann durchgängig trainiert werden. Sie bieten einen sequenziellen Vorhersagerahmen für das Erlernen umfangreicher impliziter räumlicher Modelle und funktionieren sehr gut für die menschliche Pose.

Stufe 1 ist das Modul zur Berechnung von Bildmerkmalen, und Stufe 2 ist das Modul zur Vorhersage. Die Anzahl der Stufen ist ein Hyperparameter, normalerweise drei, und ab der zweiten Phase sind dies Wiederholung von zwei. Dieses sequentielle Vorhersagekonzept, das aus Faltungsnetzen besteht und implizite räumliche Modelle erlernt, nutzt grössere rezeptive Felder auf den belief maps (eine Art Heatmap) der vorherigen Phasen, was das Erlernen der räumlichen Beziehungen zwischen den Körperteilen unterstützt und zu einer verbesserten Genauigkeit, durch zunehmend verfeinerte Schätzungen der Positionen in den späteren Phasen, führt [Josyula.13.10.2021]. Nach jeder Stufe wird eine Zwischenüberwachung verwendet, um die Problematik der verschwindenden Gradienten zu vermeiden, das ein häufiges Problem bei tiefen mehrstufigen Netzen ist.

Die Abbildung 5 zeigt die Architektur einer CPM mit beliebigen T Stufen. Die Pose-Maschine ist in den Einschüben (a) und (b) dargestellt, und die entsprechenden Faltungsnetzwerke sind in den Einschüben (c) und (d) zu sehen. Einschübe (a) und (c) zeigen die Architektur, die in der ersten Stufe nur mit Bildinformationen arbeitet. Die Einschübe (b) und (d) zeigen die Architektur für die nachfolgenden Stufen, die sowohl auf Bildevidenz als auch auf belief maps aus den vorangegangenen Stufen arbeiten. Die Architekturen in (b) und (d) werden für alle nachfolgenden Stufen (2 bis T) wiederholt. Von den drei vorgestellten Modellen hat dieses die höchste Genauigkeit.

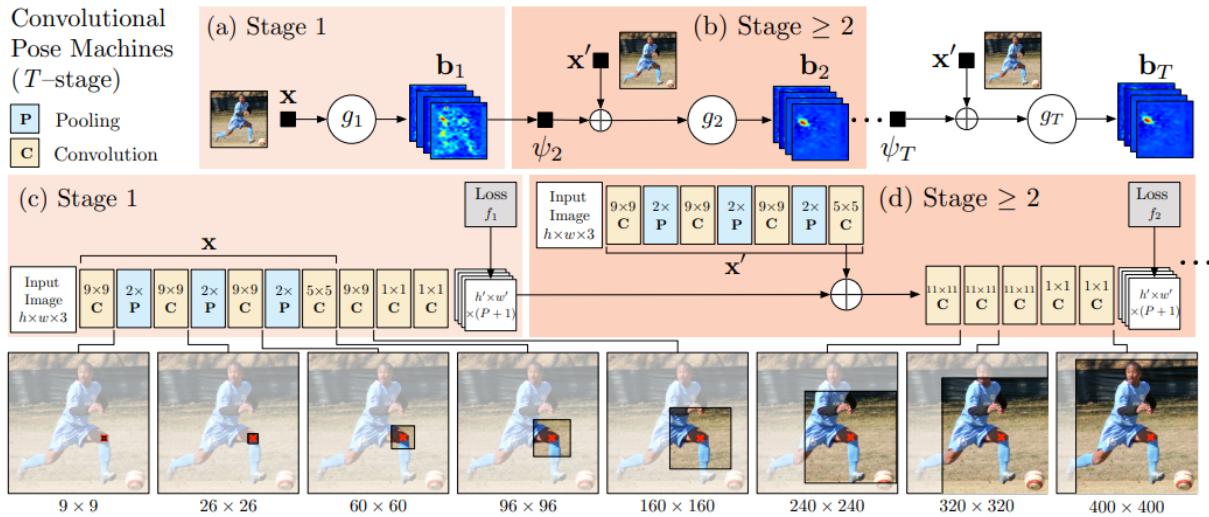


Abbildung 5: Architektur einer CPM [Wei.30.01.2016]

## 3 Framework

Ein geeignetes Framework für die Pose Estimation zu finden war ein zentraler Aspekt dieses Fachmoduls. Zum einen sollte die Pipeline-Verarbeitung in Quasi-Echtzeit erfolgen können, und ausserdem muss dies auch auf einem gewöhnlichen Windows-Rechner ausführbar sein. Dafür wurden die Frameworks von Mediapipe und Tensor-Flow genauer untersucht.

### 3.1 MediaPipe

MediaPipe ist ein Open-Source-Framework von Google zum Aufbau von Anwendungen im Bereich der Pose Estimation. Der Code ist in C++ geschrieben und kann leicht auf den meisten Plattform eingesetzt werden [[Zhang.18.06.2020](#)]. Die grosse Stärke ist die Geschwindigkeit, welche durch Graphic Processing Unit (GPU)-Beschleunigung, jedoch nicht auf allen Plattformen unterstützt, und Multithreading erreicht wird [[Alavi.01.10.2020](#)]. Dies zahlt sich vor allem bei neuen Smartphones aus, welche hohe Frames Per Second (FPS)-Werte erreichen.

Jedoch kann der Mangel an Dokumentation oftmals viel Zeit in Anspruch nehmen. Die “Dokumentation” von MediaPipe besteht aus einer Website, auf der die Konzepte auf hohem Niveau erläutert werden, und einfachen Codebeispielen. Um MediaPipe wirklich zu verstehen, muss man schon fast in den Quellcode eintauchen. Dort hat es am Anfang vieler .cc-Dateien Kommentare die erklären was im Skript genau passiert. MediaPipe ist momentan noch in der Alpha (V0.7), was die fehlende Dokumentation erklärt. Viele Funktionen können noch geändert werden und deshalb ist wahrscheinlich auch noch keine endgültige Dokumentation vorhanden. Laut der GitHub-Seite [[GitHub.06.01.2022b](#)] erwartet man noch Änderungen der API und versucht deshalb mit der v1.0 zu einer stabilen API zu kommen.

Die Blackbox MediaPipe nutzt die Komponenten Pakete, Graphen, Rechner und Untergraphen zur Verarbeitung des Inputs [[GitHub.06.01.2022](#)]. Pakete sind einfach jede Datenstruktur mit einem Zeitstempel. In der folgenden Abbildung 6 ist „input\_video“ das Paket, das in den Graphen eingespeist wird.

Ein Graph ist die Struktur des gesamten Programms. Das gesamte Diagramm selbst ist ein Graph namens „Object Detection“.

Rechner können Eingaben und Ausgaben haben. Sie führen Code bei der Erstellung, pro Frame und beim Beenden aus. Ein Beispiel für einen Rechner könnte einer sein, der die Koordinaten von Objekten einliest und diese Koordinaten normalisiert ausgibt.

Ein Untergraph ist eine Reihe von Rechnern, die in einem Graphen gruppiert sind. Untergraphen haben definierte Eingänge und Ausgänge und helfen bei der Abstraktion dessen, was sonst ein zu kompliziertes Diagramm wäre. Die Untergraphen im Diagramm sind blau und lauten in diesem Fall: „ObjectDetection“, „ObjectTracking“ und „Renderer“.

### 3 Framework

---

MediaPipe stellt viele Solutions zur Verfügung, welche diese Komponenten nutzen. Bei intensiver Einarbeit, können dann auch individuelle Änderungen vorgenommen werden. MediaPipe empfiehlt die vorgefertigten Solutions zu verwenden und nur in Ausnahmefällen Änderungen selbst vorzunehmen.

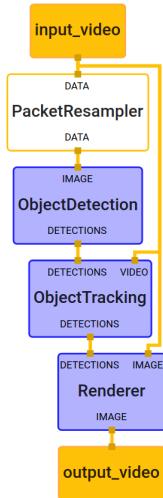


Abbildung 6: MediaPipe Komponenten, Beispielgrafik für Objekterkennung  
[MingGuangYong.2019]

#### 3.1.1 BlazePose

BlazePose ist eine performante Solution zur Pose-Estimation [[PoseGoogleBlog](#)]. Es gibt ein heavy, full und lite Modell von welchen das heavy Model die höchste Genauigkeit erreicht. Mit BlazePose [[PoseGoogleBlog](#)] verbessert MediaPipe den aktuellen Standard, die COCO-Topologie, welche aus 17 menschlichen Körper-Keypunkten besteht, auf 33. Diese Leistung ist gerade für praktische Anwendungen wie Fitness und Tanz von grosser Bedeutung.

Für die Pose Estimation wird eine zwei-Schritt, Detector-Tracker ML-Pipeline verwendet. Der Detektor lokalisiert zuerst die Pose Region of Interest (ROI) innerhalb des Bildes. Anschliessend sagt der Tracker alle 33 Pose-Keypoints aus dieser ROI voraus. Bei Videoanwendungen wird der Detektor nur für das erste Bild ausgeführt. Für nachfolgende Frames werden die ROI aus den Pose-Keypoints des vorherigen Frames abgeleitet.

Um die ML-Pipeline in Echtzeit zu durchlaufen, darf jede Komponente nur wenige Millisekunden pro Frame benötigen. Dies wird erreicht indem das Gesicht einer Person genutzt wird, um die Position des Oberkörpers zu bestimmen. Natürlich bedeutet dies, dass der Kopf einer Person sichtbar sein sollte. Folglich wurde ein Gesichtsdetektor trainiert, der sich am BlazeFace-Modell orientiert, dass im Sub-Millisekundenbereich liegt und als Stellvertreter für einen Pose-Detektor dient. Dieses Modell erkennt die Position

### 3 Framework

---

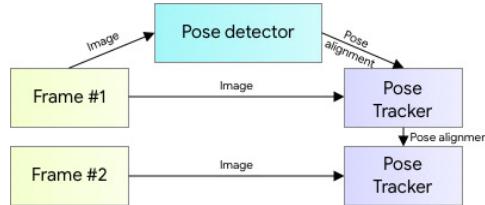


Abbildung 7: Übersicht der Pipeline [PoseGoogleBlog]

einer Person innerhalb eines Bildes, kann aber nicht mehrere Personen identifizieren. Bei diesem Pose-Tracking werden explizit zwei zusätzliche virtuelle Keypoints vorausgesagt, die das Zentrum, die Rotation und den Massstab des menschlichen Körpers als Kreis fest beschreiben. Inspiriert von Leonards Vitruvianischem Mann wird den Mittelpunkt der Hüfte einer Person, den Radius eines Kreises, der die gesamte Person umschreibt, und den Neigungswinkel der Linie, welche die Schulter- und Hüftmittelpunkte verbindet, vorausgesagt. Dies führt zu einer konsistenten Verfolgung selbst bei sehr komplizierten Fällen, wie zum Beispiel bei bestimmten Yoga-Asanas.

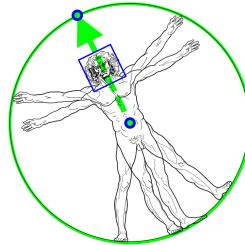


Abbildung 8: Vitruvianischer Mann, der anhand von zwei virtuellen Schlüsselpunkten ausgerichtet wird. [PoseGoogleBlog]

Die Posenschätzungs komponente der Pipeline sagt die Position aller 33 Personen-Keypoints mit jeweils drei Freiheitsgraden ( $x$ -,  $y$ -Position und Sichtbarkeit) sowie der beiden oben beschriebenen virtuellen Ausrichtungs-Keypoints voraus [mediapipe.2020]. Diese Daten sind normalisiert und liegen zwischen 0 und 1. Im Gegensatz zu aktuellen Ansätzen, die eine rechenintensive Heatmap-Vorhersage verwenden, nutzt dieses Modell einen kombinierten Ansatz aus Heatmap, Offset und Regression.

## 3.2 TensorFlow

TensorFlow ist ein maschinelles Lernsystem, das im grossem Massstab und in heterogenen Umgebungen arbeitet [Abadi.27.05.2016]. Mit TensorFlow können Entwickler Datenflussgraphen erstellen. Diese Strukturen beschreiben, wie sich Daten durch einen Graphen oder eine Reihe von Verarbeitungsknoten bewegen. Jeder Knoten im Graphen repräsentiert eine mathematische Operation, und jede Verbindung oder Kante zwischen Knoten ist ein mehrdimensionales Datenfeld oder ein Tensor.

### 3 Framework

---

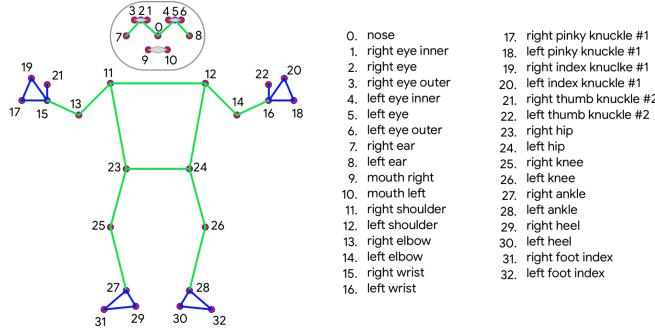


Abbildung 9: Die 33 Keypoints von BlazePose [PoseGoogleBlog]

TensorFlow stellt all dies dem Programmierer mit Hilfe der Sprache Python zur Verfügung. Python ist einfach zu erlernen und zu verwenden, ausserdem bietet es bequeme Wege, um auszudrücken, wie High-Level-Abstraktionen miteinander verbunden werden können. Knoten und Tensoren in TensorFlow sind Python-Objekte, und TensorFlow-Anwendungen sind selbst Python-Anwendungen.

Die eigentlichen mathematischen Operationen werden jedoch nicht in Python durchgeführt. Die Bibliotheken von Transformationen, die durch TensorFlow verfügbar sind, sind als hochleistungsfähige C++-Binärprogramme geschrieben. Python leitet nur den Verkehr zwischen den Teilen und bietet High-Level-Programmierabstraktionen, um sie miteinander zu verbinden.

TensorFlow-Anwendungen können auf fast jedem Ziel ausgeführt werden, das sich anbietet: ein lokaler Rechner, ein Cluster in der Cloud, iOS- und Android-Geräte, Central Processing Unit (CPU) oder GPUs. Auf Googles eigener Cloud, kann TensorFlow auf den benutzerdefiniertem TensorFlow Processing Unit (TPU) zur weiteren Beschleunigung ausgeführt werden.

#### 3.2.1 MoveNet

Bei MoveNet handelt es sich um einen Hochgeschwindigkeits-Positionstracker von Google. Das Modell ist vor-trainiert und daher nach der Einrichtung sofort einsatzbereit. Es erfasst insgesamt 17 Keypoints. Diese Keypoints sind mit (x, y)-Koordinaten verknüpft und werden jedes Mal aktualisiert, wenn der Detektor aufgerufen wird. Jeder zurückgegebene (x, y) Keypoint ist mit einer Punktzahl verknüpft, die das Vertrauen von MoveNet in die Genauigkeit der Messung darstellt. Auch bei diesem Modell sind die Werte normalisiert und liegen zwischen 0 und 1.

MoveNet gibt es in zwei Versionen, die unterschiedliche Leistungsmerkmale aufweisen. Lightning ist schneller, kann aber weniger genaue Ergebnisse liefern. Thunder ist etwas langsamer, aber genauer. Laut TensorFlow können beide mit 30+ FPS laufen, welche wir in unseren Tests jedoch auf keinem Gerät erreicht haben. MoveNet ist ein Bottom-up-Schätzungsmodell, das Heatmaps zur genauen Lokalisierung menschlicher Schlüsselpunkte verwendet. Die Architektur besteht aus zwei Komponenten, einem Merkmal-

### 3 Framework

---

sextraktor und einem Satz von Vorhersagekomponenten. Obwohl diese Vorhersagen parallel berechnet werden, kann man einen Einblick in die Funktionsweise des Modells gewinnen, wenn man die folgende Abfolge von Vorgängen betrachtet [MoveNetTFJS]:

**Schritt 1:** Die Personenzentrum-Heatmap wird verwendet, um die Zentren aller Individuen im Frame zu identifizieren, definiert als das arithmetische Mittel aller zu einer Person gehörenden Keypoints. Der Standort mit der höchsten Punktzahl (gewichtet nach dem inversen Abstand zum Bildzentrum) wird ausgewählt.

**Schritt 2:** Ein anfänglicher Satz von Keypoints für die Person wird erstellt, indem die Keypoint-Regressionsausgabe von dem Pixel geschnitten wird, das dem Objektzentrum entspricht. Da es sich hierbei um eine Vorhersage aus der Mitte heraus handelt, die über verschiedene Skalen hinweg funktionieren muss, ist die Qualität der regressierten Keypoints nicht sehr genau.

**Schritt 3:** Jedes Pixel in der Keypoint-Heatmap wird mit einem Gewicht multipliziert, das umgekehrt proportional zum Abstand zum entsprechenden regressierten Keypoint ist. Auf diese Weise wird sichergestellt, dass keine Keypoints von Personen im Hintergrund akzeptiert werden, da sie sich in der Regel nicht in der Nähe der dieser Keypoints befinden und daher eine niedrige Punktzahl haben werden.

**Schritt 4:** Der endgültige Satz von Keypoint-Vorhersagen wird durch Abrufen der Koordinaten der maximalen Heatmap-Werte in jedem Keypoint-Kanal ausgewählt. Die lokalen 2D-Offset-Vorhersagen werden dann zu diesen Koordinaten hinzugefügt, um verfeinerte Schätzungen zu erhalten. Die folgende Abbildung veranschaulicht diese vier Schritte.

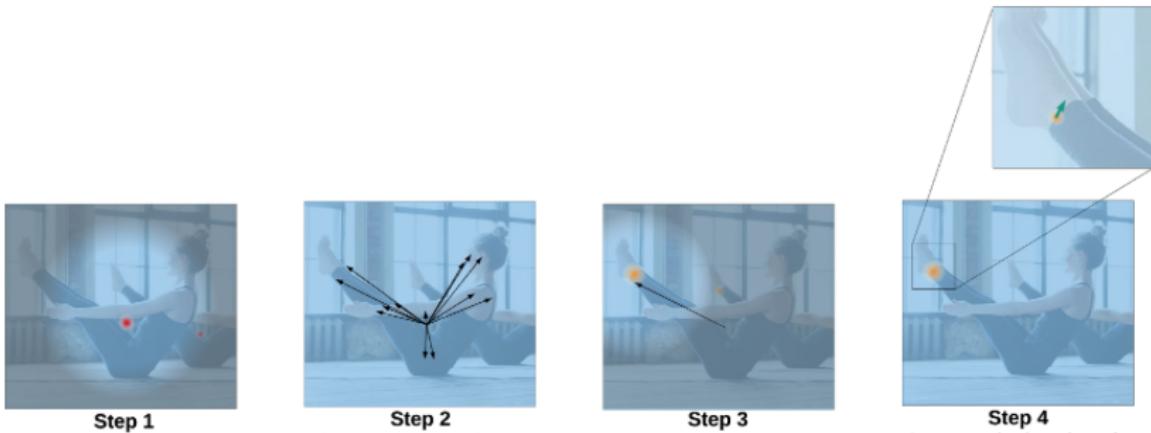


Abbildung 10: Prozessablauf von MoveNet [MoveNetTFJS]

### 3.3 TensorFlowJS Pose-Detection API

Ein entscheidender Faktor um möglichst viele FPS zu erreichen, ist die parallele Nutzung der CPU und GPU für die Abarbeitung einer Pipeline. Die Nutzung der GPU

### 3 Framework

---

ist jedoch bei MediaPipe unter Verwendung von Windows eher experimentell und bei TensorFlow ist die Beschleunigung dadurch nur klein. Eine weitere Möglichkeit diese GPU-Beschleunigung zu erreichen, besteht durch das verschieben der Modelle in den Browser. Somit kümmert sich dann der Browser um die richtige Konfiguration von Treibern und Bibliotheken. Da JavaScript auch Clientseitig läuft, können diese Modelle unabhängig der Bandbreite, also direkt beim Anwender laufen. Schliesslich verbleiben alle Daten auf dem Client, was die TensorFlowJS API für Inferenzen mit geringer Latenzzeit sowie für Anwendungen zur Wahrung der Privatsphäre nützlich macht.

Die neue TensorFlowJS Pose-Detection API macht dies möglich. Sie unterstützt die zwei Laufzeiten TensorFlowJS und MediaPipe sowie die beiden Modelle BlazePose und MoveNet. TensorFlowJS bietet die Flexibilität und eine breitere Akzeptanz von JavaScript, optimiert für mehrere Backends, einschliesslich Web Graphics Library (WebGL) (GPU), WebAssembly (WASM) (CPU) und Node. MediaPipe macht sich WASM mit GPU-beschleunigter Verarbeitung zunutze und bietet eine schnellere Inferenzgeschwindigkeit. Der MediaPipe-Laufzeitumgebung fehlt derzeit noch die Unterstützung für Node und iOS Safari [**MoveNetTFJS**].

Die Nutzung TensorFlowJS Pose-Detection Application Programming Interface (API) würde jedoch auch bedeuten das die gesamte Arbeit sich stark mit JavaScript beschäftigt. Weiter wird auch durch den Einsatz einer Webanwendung, das Vertrauen in den Datenschutz gefährdet.

#### 3.4 Entscheid

Damit die geeignete Plattform für dieses Projekt gefunden werden kann, wurden die beiden Frameworks auf verschiedenen Betriebssystemen mit verschiedenen Backends und Programmiersprachen getestet. Die folgenden drei Punkte galten dabei als Bewertungskriterien:

- **Aufwand Einrichtung:** Aufwand für Endbenutzer bis Programm benutzbar ist
- **Frames Per Second (FPS):** Wie flüssig erscheint die Posenschätzung
- **Modell Schlüsselpunkte:** Wie viele Schlüsselpunkte werden vom Modell erkannt

Da der Einsatz von Linux unseren potentiellen Marktanteil auf nur 2.4% reduzieren würde [**linuxAnteil**] und uns die Verwendung von Python in der Datenanalyse vieles vereinfachen wird, fiel die Entscheidung auf MediaPipe mit dem Python Framework.

### 3 Framework

---

	Aufwand	FPS	Keypoints	Bemerkungen
<b>MediaPipe Python Win10</b>	gering	25	33	Einfachste Implementierung
<b>MediaPipe Python Linux</b>	sehr hoch	–	33	GPU Unterstützung schwierig
<b>MediaPipe JavaScript</b>	–	25	33	JavaScript Kenntnisse
<b>TensorFlowJS</b>	–	120	33	JavaScript Kenntnisse
<b>TensorFlow Python CPU</b>	gering	20	17	FPS identisch zu MediaPipe
<b>TensorFlow Python GPU</b>	mittel	20	17	GPU ohne Vorteil

Tabelle 1: Auflistung der bewerteten Frameworks

## 4 Trainer

### 4.1 Anwendung

Unser Ansatz zielt darauf ab, die Yogastellungen des Benutzers automatisch aus Echtzeitvideos, die von einer Rot-Grün-Blau (RGB)-Kamera aufgezeichnet werden, zu erkennen. Der Ablauf kann in vier Schritte unterteilt werden. Zunächst werden Daten gesammelt, die parallel zur Erkennung in Echtzeit stammen. Zweitens wird MediaPipe Pose verwendet, um alle 33 Schlüsselpunkte zu identifizieren. Die erkannten Schlüsselpunkte werden an unser Modell weitergeleitet, in dem unser ML-Modell Muster erkennt und diese analysiert. Abschliessend wird das Bild bearbeitet, so dass die Schlüsselpunkte für den Benutzer ersichtlich sind und an das Graphical User Interface (GUI) als Feedback ausgegeben.

### 4.2 Anforderung an die Anwendung

Der Yoga-Trainer soll als GUI dem Benutzer zur Verfügung stehen und dabei vollständig in der Python Programmiersprache entwickelt werden. Während der Übung soll in Quasi-Echtzeit, die Yogastellung erkannt, analysiert und mit Feedback an den Benutzer zurückgegeben werden. Nebst den ganzen Funktionen während des Trainings, soll auch eine voll automatische statistische Auswertung der Performance möglich sein. Alles in allem gelten folgende Anforderungen an das Benutzersystem:

- Computer mit Windows 10
- Internetzugang (nur zur Installation)
- Python (Version: 3.9.0)
- RGB-Kamera

### 4.3 Benutzerfunktionen der Anwendung

Wie in Abbildung 11 ersichtlich ist, wird die Anwendung in vier Hauptfunktionen unterteilt. In der ersten Hauptfunktion, View History 4.3.3, kann der Benutzer seine bisher geleisteten Trainings anschauen. Es werden ihm verschiedene Performancemetriken, mit Hilfe von Boxplots und Histogrammen dargestellt. In einer weiteren Hauptfunktion, Change Trainingssets 4.3.2 kann der Benutzer seine Trainings selber zusammenstellen. So kann jeder Benutzer sein Training nach seinen Ansprüchen gestalten. Die letzten beiden Hauptfunktionen Learn und Train 4.3.1, kann der Benutzer die verschiedenen Asanas trainieren. Die Learn-Funktion wird dafür benutzt um die Yogastellungen zu verinnerlichen, ohne das sich mögliche Fehler auf die Statistik auswirken.

#### 4.3.1 Learn und Train

In diesen beiden Funktionen wird nun tatsächlich Yoga angewandt, dabei stehen dem Benutzer zwei Möglichkeiten zur Verfügung. Als erstes empfiehlt sich die „Learn“-Methode. In diesem Bereich kann eine Yoga Pose frei gewählt werden um diese zu

## 4 Trainer

---

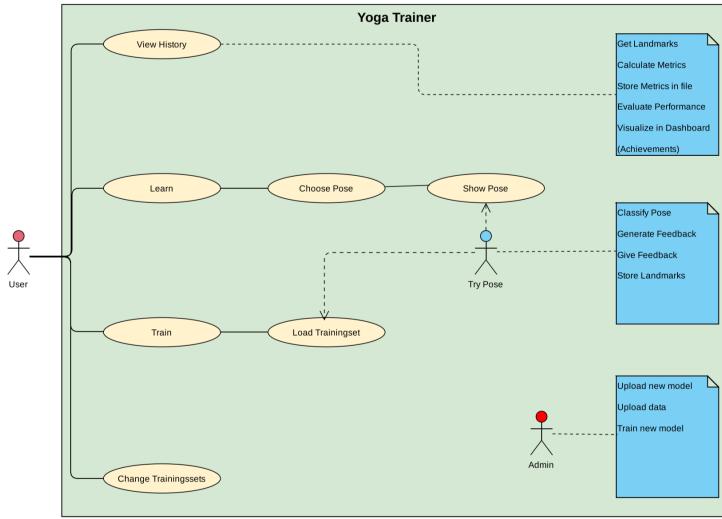


Abbildung 11: Use-Case Diagramm

verinnerlichen. Dazu wird dem Benutzer die Yoga Pose neben dem Kamerabild von sich selbst angezeigt und auf Fehler hingewiesen. Der Trainer versucht dabei mit verschiedenen Hilfsmittel, zum Beispiel optisch oder akustisch, in die richtige Stellung zu führen. Da diese Methode zum lernen dient, werden weder Erfolge noch Fehler aufgezeichnet.

Bei der „Train“-Methode gilt es ernst. Der Benutzer kann ein vorgefertigtes Trainingsset auswählen und dieses schrittweise durcharbeiten. Die Yogastellungen müssen dabei für eine gewisse Zeit korrekt gehalten werden um das Training erfolgreich abzuschließen. Die aufgezeichneten Daten, werden verarbeitet und zum Teil direkt während dem Training angezeigt, oder sie fließen in die Statistik des „View History“ Abschnitts.

### 4.3.2 Change Trainingsets

Zu Beginn sollen einige Trainingssets bereits verfügbar sein. Damit der Benutzer sein Training auf seine Bedürfnisse anpassen kann, gibt es hier die Möglichkeit die bestehenden Sets anzupassen oder neue anzulegen. Die Trainingssets bestehen aus Yogastellungen und einer Zeit, welche angibt wie lange diese Pose gehalten werden soll, sowie Anzahl Wiederholungen und Ruheintervalle.

### 4.3.3 View History

Damit der Benutzer einen Überblick über seine vergangenen Trainings bekommt, kann in diesem Bereich der gesamte Verlauf betrachtet werden. Es können die Trainings der vergangenen Tage betrachtet werden, sowie verschiedene Performance Auswertungen, welche mit Hilfe von statistischen Mitteln einfach verständlich dargestellt werden. Als Motivation für weitere erfolgreiche Trainings, werden auch einige Achievements zur Verfügung stehen.

## 4.4 Administratorfunktionen der Anwendung

Der Administrator Bereich soll dem normalen Benutzer vorenthalten werden, denn in diesem Bereich wird es möglich sein, das ML-Modell zu verändern. Mit Hilfe von Bildmaterial neuer Yogastellungen, soll es hier möglich sein ein neues Modell zu trainieren. Ebenfalls ist eine Option angedacht um einfachere Fitnessübungen zu implementieren. Dies ist jedoch in der kommenden Arbeit als optional zu betrachten.

## 4.5 Erkennung der Yoga Asanas

Zu diesem Zeitpunkt sind nur wenige öffentliche Datensätze für Yogastellungen, sogenannte Asanas, verfügbar. Mit dem Datensatz *Yoga Poses Dataset* von Niharika Pandit [KaggleYoga] können die wohl am bekanntesten fünf abgedeckt werden. Zu den bekanntesten Asanas gehören:

- dog - „Adho Mukha Svanasana“
- goddess - „Utkata Konasana“
- tree - „Vrikshasana“
- plank - „Kumbhakasana“
- warrior - „Virabhadrasana“

Zur Validierung, oder auch um weitere Asanas zu verwenden, kann auf den *Yoga82* Datensatz [verma2020yoga] zugegriffen werden. Da *Yoga82*, wie in Bild 12 zu erkennen, sehr viele Daten umfasst, müssen die Bilder per Web Scraping bezogen werden.

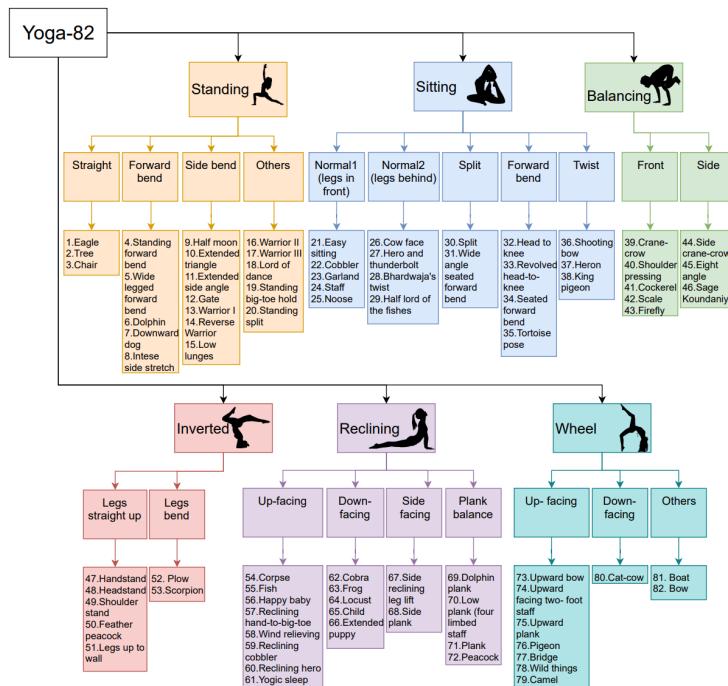


Abbildung 12: Asanas im *Yoga82* Datensatz [verma2020yoga]

## 5 Arbeiten mit Mediapipe

Ein Ziel dieses Fachmoduls bestand darin, erste Aufgaben unter Verwendung von MediaPipe zu lösen. Als Einstieg wurde mit Hilfe dieser Pipeline und mit einem Videoinput durch die Webcam oder als mp4-Datei die Joint-Koordinaten des Skeletons direkt in diesen Inputstream eingefügt. Anschliessend lag der Fokus aus diesen gewonnenen Daten der Keypoints, zum einen in Quasi-Echtzeit Berechnungen durchführen und zusätzlich diese Daten auch abzuspeichern, um auch später noch darauf zugreifen zu können. Die gespeicherten Daten konnten dann mit einem Jupyter-Notebook einer statistischen Auswertung unterzogen werden.

Diese Arbeiten direkt auf den Themenbereich Yoga anzuwenden, hätte umfangreicher Vorarbeit benötigt. Wie z.B. für das definieren geeigneter Metriken für die Beurteilung der Performance. benötigt. Deshalb wurden diese ersten Arbeiten auf eine Gymnastikübung, die Liegestütze, angewendet.

### 5.1 Berechnung des Winkels

Die Berechnung eines Zwischenwinkels erscheint bei einer Gymnastikübung wie Liegestütze nicht von grossen belangen, könnte jedoch bei Yogaübungen als Genauigkeitsmass sehr hilfreich sein. Durch die x- und y-Koordinaten von drei verbundenen Keypoints, kann der eingeschlossene Winkel berechnet werden. Dies ist eine Quasi-Echzeit Berechnung und wird, wie in Abbildung 13 gezeigt, im Inputstream direkt eingeblendet und kann auch als Datenpunkt abgespeichert werden. Die Winkelberechnung kann als Metrik für die Performance eingesetzt werden und auch im Historienverlauf analysiert werden. Bei den Liegestützen wurden der Winkel des linken Ellenbogens berechnet.

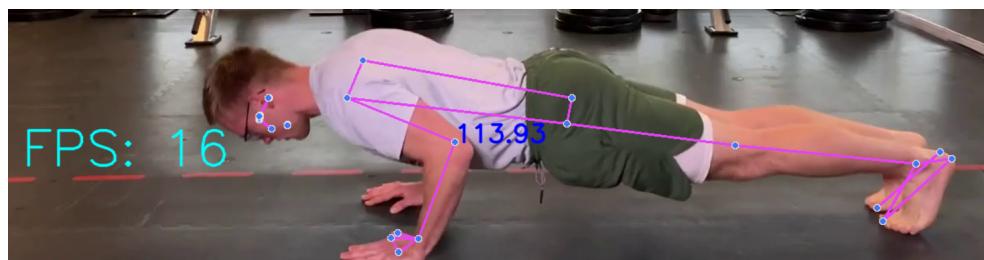


Abbildung 13: Winkelberechnung direkt im Videoinput eingeblendet

## 5 Arbeiten mit Mediapipe

---

Die in Abbildung 14 gezeigte Funktion ist der Hauptteil dieser Ausgabe. Sie benötigt drei Winkel mit X- und Y-Koordinate und berechnet anschliessend über die *arctan2*-Funktion den Winkel in Radianen, welcher abschliessend noch in das hier geeignete Gradmass überführt wird.

```
def calculate_angle(a, b, c):
    a = np.array(a) # First
    b = np.array(b) # Mid
    c = np.array(c) # End

    radians = np.arctan2(c[1]-b[1], c[0]-b[0]) - \
              np.arctan2(a[1]-b[1], a[0]-b[0])
    angle = np.abs(radians*180.0/np.pi)

    if angle > 180.0:
        angle = 360-angle

    return angle
```

Abbildung 14: Funktion in Python für die Winkelberechnung

## 5.2 Statistische Auswertung

Durch die Datenspeicherung der Keypoints als CSV-Datei, kann zu jeder Zeit eine statistische Auswertung dieser Daten gemacht werden. Diese Daten können dann auch über ein Zeitintervall miteinander verglichen werden, um die Veränderungen der Performance zu überwachen. Um eine erste Übersicht von alle Keypoints über den Verlauf der gesamten Übung zu erhalten, wurde ein Boxplot, Histogramm und ein Lineplot auf jeden dieser 33 Keypoints angewendet. Bei dieser Art von Gymnastikübung sind diese Daten jedoch ziemlich redundant, entweder sind sie sehr statisch oder verändern sich abhängig von der Frequenz der Auf- und Abbewegung.

Aus den verändernden Daten wurden zwei Auswertungsmerkmale genauer betrachtet. Wie in Abbildung 15 die Frequenz der Auf- und Abbewegung der Liegestütze und in Abbildung 16 die Durchbiegung des Rückens und der Beine sowie den Winkel des Ellenbogens.

Um den gesamten Bewegungsumfang bei der Übung zu erfassen, wurden die Y-Komponente des Nasen-Keypoints über die Zeit in einem Lineplot visualisiert. Dies zeigt die Frequenz der Auf- und Ab-Bewegung und durch das herauslesen der peaks kann auch die genaue Zeit pro Durchgang, sowie die Anzahl berechnet werden.

## 5 Arbeiten mit Mediapipe

---

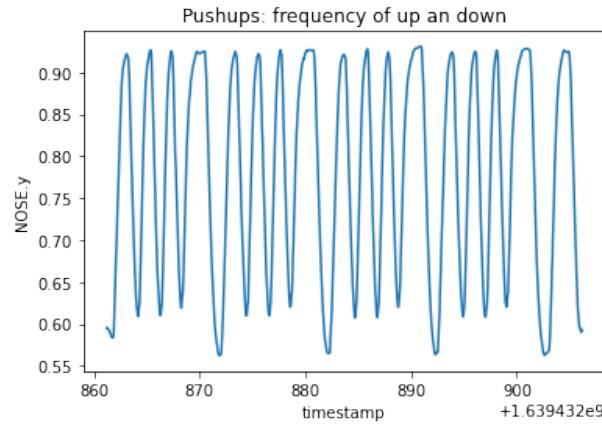


Abbildung 15: Frequenz der Auf- und Ab-Bewegung

Die Durchbiegung des Rückens und der Beine über den Verlauf der Übung darzustellen bedarf insgesamt vier Keypoints entlang dieser Strecke. Die Keypoints an den Enden definieren den Verlauf der Strecke, auf welchen die zwei Keypoints Hüfte und Knie liegen sollten. Die Abweichung dieses Punktes kann dann im Zeitverlauf dargestellt werden.

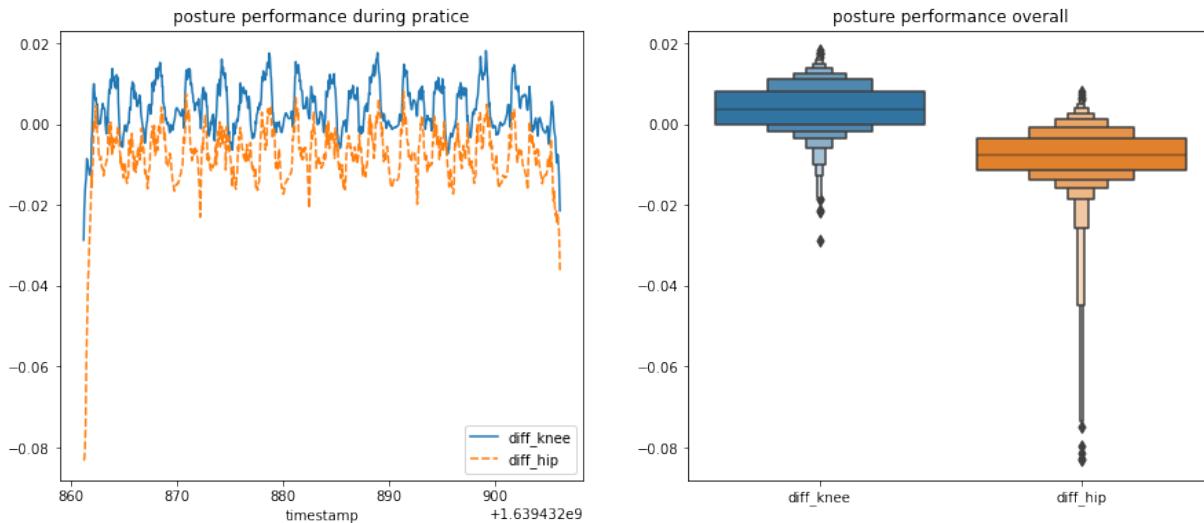


Abbildung 16: Duschbiegung bei Liegestützen

## 6 Meilensteine der Bachelorarbeit

---

# 6 Meilensteine der Bachelorarbeit

Die im Kapitel 4.2 beschriebenen Anforderungen werden in der Bachelorarbeit als Meilensteine verwendet. In Kombination mit einer agilen Software Entwicklungsmethode, entsteht dadurch der Vorteil, in häufigen Releases zu programmieren und so den aktuellen Stand immer demonstrierbar zu haben. Mit Ausnahme von *Vorarbeit Programmierung*, sind alle Meilensteine zeitlich unabhängig voneinander umsetzbar, die Detailplanung kann jedoch erst danach erfolgen.

In den nachfolgenden Kapitel sind die einzelnen Meilensteine im Detail beschrieben der ungefähre zeitliche Ablauf ist in Abbildung 17 ersichtlich.

Pos.	Linie zeigen	Aufgabe/Tätigkeit	Beginn [Datum]	Endet nach [Tagen]	Ende [Datum] keine Eingabe	Status [0 bis 100]	Meilenstein [Datum]
1		Fachmodul	21.10.2021	57	07.01.2022	100%	
2	x	<b>Vorarbeit Programmierung</b>	07.02.2022	6	14.02.2022		14.02.2022
3		Designentwurf GUI	07.02.2022	2	08.02.2022		
4		Use-Case Diagramm	08.02.2022	1	08.02.2022		
5		Klassendiagramm	09.02.2022	2	10.02.2022		
6		ML-Modell (Klassifikation)	09.02.2022	1	09.02.2022		
7		Train/Test Datensatz festlegen	10.02.2022	2	11.02.2022		
8		Detailplanung erstellen	11.02.2022	2	14.02.2022		
9	x	<b>GUI implementiert</b>					
10							
11	x	<b>Klassifizierung</b>					
12							
13	x	<b>Metriken definiert</b>					
14							
15	x	<b>Metriken implementiert</b>					
16							
17	x	<b>Datenspeicherung implementiert</b>					
18							
19	x	<b>Feedback implementiert</b>					
20							
21	x	<b>Learn Funktion implementiert</b>					
22							
23	x	<b>Trainingsset implementiert</b>					
24							
25	x	<b>Statistikauswertungen definiert</b>					
26							
27	x	<b>Statistikauswertungen implementiert</b>					
28							
29	x	<b>Abgabe Bachelorarbeit</b>	25.08.2022	1	25.08.2022		26.08.2022
30							
31	x	<b>Präsentation Bachelorarbeit</b>	29.08.2022	5	02.09.2022		

Abbildung 17: Zeitplan der Meilensteine

## 6.1 Vorarbeit Programmierung

Der erste Meilenstein der Bachelorarbeit befasst sich mit der ganzen Projektstruktur. So wird in dieser Zeit das Design entworfen, Klassendiagramme erarbeitet, sowie Bilddaten für die Klassifikation der Yogastellungen gesammelt. Dieser Meilenstein wird durch das erstellen der Detailplanung, der kommenden Meilensteine, abgeschlossen

## 6 Meilensteine der Bachelorarbeit

---

### 6.2 GUI implementiert

Aus dem Designentwurf wird nun eine graphische Benutzeroberfläche realisiert. Diese wird auch benötigt, um die Applikation an reellen Testpersonen zu testen und um deren Feedback zu erhalten. Alle Funktionen der Applikation sollen im GUI ersichtlich sein, das heisst, der Benutzer benötigt keine Erfahrung in Python um den Trainer zu bedienen. Das Dashboard zur Visualisierung der statistisch ausgewerteten Daten ist nicht im GUI inbegriffen, sondern wird im Meilenstein *Auswertung der Daten* behandelt.

### 6.3 Klassifizierung

Um Yogastellungen zu erkennen, muss ein ML-Modell erstellt werden. Mit dem im *Vorarbeit Programmierung* definierten Verfahren zur Klassifizierung, wird ein aus Yoga82 und anderen Datensätzen trainierter Klassifikator erstellt. Dieser Meilenstein wird durch erfolgreiches erkennen der gängigsten Yoga Asanas abgeschlossen.

### 6.4 Datenspeicherung

Die Daten, welche während eines Trainingssets erfasst werden, müssen auch gespeichert werden können. In diesem Meilenstein wird ein Ablagesystem eingerichtet, welches es ermöglicht die Daten über den gesamten Benutzungszeitraum der Applikation abzuspeichern. Das Layout der Daten muss daher so angepasst sein, dass es mit einfachen Funktionen möglich ist, statistische Berechnungen durchzuführen.

### 6.5 Metriken

Die Yogastellungen werden mit verschiedenen Metriken erweitert, dazu gehören zum Beispiel, die Zeit welche in einer Pose verbracht wurde, die Winkel zwischen verschiedenen Gelenken oder auch der Beugung des Rückens. In diesem Meilenstein werden die Metriken definiert und anschliessend implementiert und bildet somit den Grundstein für den nächsten Meilenstein.

### 6.6 Feedback

Der Trainer soll möglichst ähnliche Aufgaben übernehmen, wie ein echter Instruktor im Fitnesscenter. In diesem Meilenstein werden diese Fähigkeiten implementiert. Die Applikation soll in der Lage sein, Instruktionen zu geben um das Erlernen einer neuen Yogastellung zu vereinfachen. Ebenfalls soll es möglich sein, während der Übung Feedback zu erhalten, welche auf falsche Haltungen hinweist.

### 6.7 Learn Funktion

Die im Kapitel 4.3.1 beschriebene Funktionen werden in diesem Meilenstein realisiert. Dieser Meilenstein kann erst nach erfolgreichem abschliessen der Meilensteine *Klassifizierung* und *Feedback* implementiert werden.

## 6 Meilensteine der Bachelorarbeit

---

### 6.8 Trainingssets

Die Benutzer sollen die Möglichkeit haben ihre eigenen Trainings individuell zusammenzustellen. Ein Trainingsset besteht aus einer Abfolgen von verschiedenen Yogastellungen, welche zu einer benutzerdefinierten Zeit gehalten werden soll. In diesem Meilenstein wird die dazu nötige Datenstruktur implementiert.

### 6.9 Auswertung der Daten

Die Daten aller absolvierten Trainingssets werden gesammelt und statistisch analysiert. Der gesamte Trainingsverlauf soll mit Hilfe von Histogrammen, Boxplots und Zeitreihen, den Benutzern angeboten werden. Die Aufbereitung und Visualisierung der Daten schliessen diesen Meilenstein ab.

### 6.10 Abgabe Bachelorarbeit

Der letzte doch bedeutendste Meilenstein dieser Arbeit, markiert den Abschluss unserer Bachelorarbeit. Geplant ist die Abgabe des Berichts und somit auch der kompletten Applikation am 26. August 2022, bevor dann in Kalenderwoche 35 die Arbeit präsentiert wird.

---

## **Abkürzungsverzeichnis**

**API** Application Programming Interface

**CNN** Convolutional Neural Networks

**CPM** Convolutional Pose Machine

**CPU** Central Processing Unit

**DNN** Deep Neural Network

**FPS** Frames Per Second

**GPU** Graphic Processing Unit

**GUI** Graphical User Interface

**HOG** Histogrammorientierte Gradienten

**ML** Machine Learning

**MSE** Mean Squared Error

**RGB** Rot-Grün-Blau

**ROI** Region of Interest

**TPU** TensorFlow Processing Unit

**WASM** WebAssembly

**WebGL** Web Graphics Library

## Anhang A

---

### **Anhang A: Fachmodulauftrag**

# Fachmodul HS 2021/2022

## Bachelorarbeit: «Gymnastik Trainer mit Pose Estimation»

<b>Studierende (r)</b>	Alex Koller, alex.koller@ost.ch (mobile: +078 720 9264) Stephan Seliner, stephan.seliner@ost.ch (mobile: + 078 208 06 02)
<b>Referent</b>	Prof. Dr. Christoph Würsch, christoph.wuersch@ost.ch (077 483 33 38), +41 58 257 34 52 (WUCH)
<b>Korreferent</b>	Nicola Notari +41 58 257 31 80 (NONI)
<b>NTB-Betreuer</b>	keiner
<b>Industriepartner</b>	ICE
<b>Version</b>	Entwurf, 21. September 2021
<b>NDA</b>	Nicht notwendig

## Inhalt

<b>1.</b>	<b>Ziele der Bachelorarbeit</b>	<b>2</b>
1.1.	Pose Estimation und Yoga	2
1.2.	Anforderungen an den virtuellen Yoga-Trainer:	4
1.3.	Was soll in dieser Bachelorarbeit erreicht werden?	4
<b>2.</b>	<b>Ziel des Fachmoduls</b>	<b>5</b>
<b>3.</b>	<b>Projektablauf</b>	<b>5</b>
<b>4.</b>	<b>Aufträge im Fachmodul</b>	<b>6</b>
4.1.	Literaturstudium: Pose-Estimation	6
4.2.	Google Mediapipe und erste Messungen	6
4.3.	Klare Aufgaben- und Zieldefinition für die BA	6
<b>5.</b>	<b>Bewertung des Fachmoduls</b>	<b>6</b>
<b>6.</b>	<b>Einverständniserklärung</b>	<b>7</b>
<b>7.</b>	<b>Referenzen</b>	<b>8</b>

# 1. Ziele der Bachelorarbeit

## 1.1. Pose Estimation und Yoga

Die Schätzung der Körperhaltung aus Videos spielt eine entscheidende Rolle bei der Überlagerung digitaler Inhalte und Informationen mit der physischen Welt in der erweiterten Realität, der Erkennung von Gebärdensprache, der Steuerung von Ganzkörpergesten und sogar bei der Quantifizierung körperlicher Übungen, wo sie die Grundlage für Yoga-, Tanz- und Fitnessanwendungen bilden kann.

Die Schätzung von Körperhaltungen für Fitnessanwendungen ist aufgrund der grossen Vielfalt möglicher Haltungen (z. B. Hunderte von Yoga-Asanas), zahlreicher Freiheitsgrade, Verdeckungen (z. B. verdecken der Körper oder andere Objekte die von der Kamera aus gesehenen Gliedmassen) und einer Vielzahl von Erscheinungsbildern oder Outfits eine besondere Herausforderung.



Abbildung 1: Typische Yoga-Posen (Quelle: Yoga ICONIST)

Die Schätzung der menschlichen Körperhaltung ist eine seit langem bestehende Herausforderung. In den Anfängen wurde die Aufgabe der Pose Estimation als Inferenzaufgabe behandelt. Diese Modelle lassen sich grob in zwei Kategorien einteilen.

- **Appearance Models:** Die erste Kategorie der häufig verwendeten Modelle wird als Erscheinungsbildmodelle (*appearance models*) bezeichnet. Die Merkmale von Körperteilen werden zunächst mit Hilfe von Merkmalsdeskriptoren wie dem Histogramm des orientierten Gradienten extrahiert. Dann werden verschiedene Körperteile miteinander kombiniert, wie z.B. Poselets.
- **Structural Models:** Eine weitere Kategorie von Modellen sind deformierbare Modelle oder Strukturmodelle. Bei diesen Modellen werden artikulierte Restriktionen verwendet, um Körperteile zu analysieren. Das *Pictorial Structure Model* verwendet paarweise Terme zur Modellierung des relativen Abstands zwischen zwei Teilen.

**Deep Learning Modelle:** Nachdem tiefe Faltungsnetzwerke (DNN) auf die Pose Estimation angewendet wurden, verbesserte sich die Leistung der Schätzmodelle mit großer Geschwindigkeit.

Zu Beginn konzentrierten sich die Forscher meist auf eine gut beschnittene Aufgabe zur Schätzung der Haltung einer einzelnen Person, die eine vereinfachte Teilaufgabe darstellt. Heutzutage hat die allgemeinere Pose Estimation für mehrere Personen gute Ergebnisse erzielt, und anspruchsvollere Aufgaben wie die Pose Estimation in einer Menschenmenge sind zu einem aktuellen Thema geworden. In Tabelle 4 sind die Bewertungsergebnisse mehrerer repräsentativer, oben vorgestellter Methoden zusammengestellt. Diese Methoden umfassen die Schätzung der Pose von Einzelpersonen und die Schätzung der Pose von mehreren Personen.

### State-of-the-Art Deep-Learning basierte Modelle [27]

Eine auf Deep Learning basierende Methode namens *DeepPose* wird von Toshev und Szegedy [46] vorgeschlagen. Die tiefen neuronalen Netze arbeiten als Regressoren für die Körper-Keypoints. Sie erstellen zunächst eine grobe Vorhersage und extrahieren dann verwandte Bildausschnitte für die künftige Verfeinerung der Position der Keypoints. Später werden einige traditionelle Methoden mit tiefen Netzen kombiniert, um eine bessere Leistung zu erzielen. In [18] werden ein auf einem Faltungsnetzwerk basierender Part-Detector und ein von einem Markov-Random-Field inspiriertes Spatial-Model zu einem einheitlichen Lernrahmen kombiniert. Die Ausgabe des Netzes in [18] ist ein Tensor mit C Kanälen, wobei die Anzahl der Gelenke gemeint ist. Eine solche Ausgabe wird auch als Heatmap bezeichnet, da jedes Gelenk in einem Ausgabekanal kodiert ist. Eine Illustration der Heatmap-Ausgabe ist in Abb. 2 zu sehen. Jain et al. [19] verwenden sowohl Farb- als auch Bewegungsmerkmale zur Schätzung der menschlichen Haltung. Das Farbmerkmal wird aus RGB-Bildern und das Bewegungsmerkmal aus optischen Flusskarten extrahiert. Eine kaskadierte Architektur, die feine und grobe Merkmale kombiniert, wird in [20] für die Schätzung der Körperhaltung einer einzelnen Person vorgeschlagen. Später entwickeln Wei et al. [21] einen Rahmen namens Convolutional Pose Machines, in dem eine Folge von Prädiktoren trainiert wird, um dichte Vorhersagen für jede Bildposition zu treffen. Bei den vorgeschlagenen Convolutional Pose Machines werden die menschlichen Posen iterativ mit verschiedenen Faltungsstufen verfeinert.

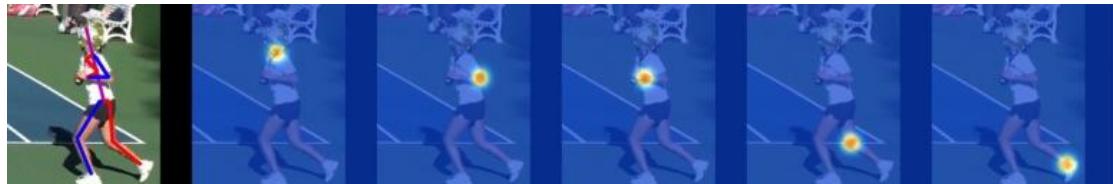


Abbildung 2: Heatmap für Keypoints

Eine weitere Arbeit zur iterativen Verfeinerung der Pose Estimation ist [22], in der ein Netzwerk, das eine Korrektur ausgibt für die Schätzung erlernt wird. In Anlehnung an die kaskadierte Struktur schlagen Newell et al. [23] Stacked Hourglass Networks vor und übertreffen damit andere Methoden deutlich. Die Stacked Hourglass Networks umfassen mehrere wiederholte einzelne Sanduhrräume, bei denen es sich um Down-Sampling- und dann Up-Sampling-Unterstrukturen handelt (siehe Abb. 2). Der grösste Unterschied zwischen Stacked Hourglass Networks und früheren iterativen Methoden besteht darin, dass das Netzwerk das Bild nur einmal als Eingabe benötigt und keine zusätzliche Bildbearbeitung, wie z. B. Zuschneiden, erforderlich ist, wodurch die Methode leichter an andere Szenarien angepasst werden kann. Die Stacked Hourglass Networks haben eine Menge Arbeit inspiriert. In [24] zum Beispiel verbessern die Autoren die Leistung der Stacked Hourglass Networks mit Aufmerksamkeitsmodulen; in [25] wird die Sanduhrstruktur mit der Restverbindung verbessert und ein neues Modul namens Pyramid Residual Module vorgeschlagen; in [26] werden mehrere grobe Detektorzweige mit einem feinen Detektorzweig kombiniert.

## 1.2. Anforderungen an den virtuellen «Yoga»-Trainer:

Um die Idee des virtuellen Yoga-Trainers zu veranschaulichen, schauen wir uns eine konkrete Gymnastik-Übung an: *Liegestützen ausführen*.

- Pose-Estimation in Quasi-Echtzeit möglich: Den Liegestützen werden in Echtzeit getrackt und zusammen mit dem Live-Video als Skeleton dargestellt.
- Automatische und statistische Auswertung der Performance der Gymnastik Übungen:  
Als Performance Metrik können folgende Features verwendet werden:
  - Anzahl Liegestützen
  - Verbrauchte Leistung pro Liegestütze (basierend auf Gewicht und BMI)
  - Verbrauchte Energie
  - Frequenz, Pausen, Rhythmus
  - Durchbiegung des Rückens, Haltung
- Darstellung der Performance für verschiedene Übungen in Form von Zeitreihen, Histogrammen, Box-Plots, Heatmaps etc., Protokollierung und Speicherung der Daten

### Systemanforderungen:

- Kann auf einem Windows-PC (i7, i9) oder einem Windows Laptop installiert und ausgeführt werden, eventuell mit GPU-Grafik-Karte
- Datenerfassung über eine HD-USB-Kamera auf Stativ
- Anschluss eines externen Flachbildschirms als Repräsentant für den virtuellen Trainer
- Code: Erweiterbarkeit durch sauber strukturierte Klassen im Code, klar dokumentiert im Code selbst

## 1.3. Was soll in dieser Bachelorarbeit erreicht werden?

Ziel der Bachelorarbeit ist es, basierend auf der Google-Implementierung MediaPipe BlazePose (<https://ai.googleblog.com/2020/08/on-device-real-time-body-pose-tracking.html>) einen KI-Fitness-Trainier oder einen KI-Yoga-Lehrer zu programmieren. Dieser Trainer soll ähnliche Aufgaben übernehmen, wie ein echter Gymnastik-Trainer im Fitness-Center:

- a. Anleitung und Instruktionen geben, für das Erlernen einer neuen Übung, mit Feedack über die Pose-Estimation
- b. Performance der Ausführung dieser Übungen tracken und Feedback geben
- c. Statistische Auswertungen der Leistungen ermöglichen und diese graphisch darstellen (z.B. über den Verlauf der Trainingsperiode)

Folgende möglichen Arbeitspakete sollen während dieser Bachelorarbeit erarbeitet werden:

1. Konzeption eines virtuellen Gymnastik-Trainers (z.B. Yoga)
2. Definition der *Applikation* und der *Zielgruppe*: Entscheid: Yoga, Taekwon-Do, Gymnastik, etc.
3. Definition der typischen *Use-Cases* und der zugehörigen Test-Suite (Test-Cases)
4. Definition der *Performance Metrik* für die einzelnen Applikationen und Gymnastik-Übungen
5. *Inbetriebnahme* der Media-Pipeline von Google
6. *Programmierung* und *Test* des Virtuellen Gymnastik-Trainers: Agile Software-Entwicklung mit häufigen Releases
7. Test mit realen Testpersonen
8. Diskussion und Fazit

Innerhalb der Bachelorarbeit sind ausserdem folgende Punkte zu bearbeiten:

- Rahmenbedingungen definieren und Pflichtenheft erstellen
- Definieren der notwendigen Daten und Datenerfassung
- Konzipierung des Messaufbaus und Umsetzung in der Anwendung
- Verifikation und Abschätzung Potential für zukünftige Anwendungen
- Zusammenstellen und Niederschrift einer Anleitung für die Verwendung der Applikation

## 2. Ziel des Fachmoduls

Die Einarbeitung in die Bachelorarbeit erfolgt im Fachmodul. Die Studierenden erarbeiten mit dem Referenten die detaillierte Aufgabenstellung und arbeiten sich im begleiteten Selbststudium in die Aufgabenstellung ein.

## 3. Projektablauf

- Dauer Fachmodul 18. Oktober bis 23. Dezember 2021
  - Abgabetermin Bericht Fr. 7.01.2022, bis 10 Uhr
  - Präsentation/Prüfung Mo. 10.01.2020; OST, Buchs
- 
- **Selbststudium**, Hauptanteil des Fachmoduls;  
Umfasst Literaturrecherchen- und Arbeiten, Abklärungen, Planungsarbeiten, Verfassen von Dokumenten etc, (<https://jakevdp.github.io/WhirlwindTourOfPython/>)
  - **Betreute praktische Arbeiten**  
Einarbeitung in die Funktionsweise der benötigten Apparaturen und Verfahren sowie Schulung in der verwendeten Simulationssoftware.

## 4. Aufträge im Fachmodul

### 4.1. Literaturstudium: Pose-Estimation

#### Lernziele:

- a. Die Studierenden können komplexe Applikationen in Python programmieren (objekt-orientiert).
- b. Die Studierenden kennen die wichtigsten Ansätze und neuronalen Netz-Architekturen für Pose Estimation (Review Paper lesen).
- c. Die Studierenden fassen den Stand der Technik für 3D-Pose-Estimation in einer kurzen Übersicht zusammen.

Diese Lernziele werden erreicht durch gemeinsames Durcharbeiten einiger Literaturstellen im Anhang, insbesondere durch Lesen der relevanten Review-Paper.

### 4.2. Google Mediapipe und erste Messungen

- a. Die Studierenden sind in der Lage, mit der Media-Pipeline von Google die Joint-Koordinaten des Skeletts für eine selbst gewählte Gymnastik-Übung zu erfassen und statistisch auszuwerten.
- b. Statistische Auswertung der Messdaten und Visualisierung mit Python.

### 4.3. Klare Aufgaben- und Zieldefinition für die BA

- a) Erstellen eines **Zeitplans** für (FM und) BA mit Deliverables und Milestones
- b) Erstellung eines **Berichtes** (ca. 15 Seiten eigener Text + Bildmaterial) nach vorgegebenen Regeln, der mindestens folgende Elemente enthält:
  - c) Dokumentation der im Fachmodul erledigten Aufträge
  - d) Formulierungsvorschlag für definitive Aufgabenstellung
  - e) Erstellen einer **Präsentation**, welche die relevanten Punkte des Berichtes wiederspiegelt (ca. 15-20 min)

## 5. Bewertung des Fachmoduls

Die Benotung des Fachmoduls setzt sich wie folgt zusammen:

- Bericht (Technischer Gehalt, Berichtsstil, formelle Vollständigkeit); Gewichtung 60%
- Präsentation (20 min) inkl. Beantwortung von Fragen (20-30 min); Gewichtung 40%

## 6. Einverständniserklärung

Der Studierende ist mit diesem Plan einverstanden.

Buchs, Mitte Oktober, 2021

A. Koller

Alex Koller

S. Seliner

Stephan Seliner

## 7. Referenzen

- [1.] Rohit Josyula and Sarah Ostadabbas: A Review on Human Pose Estimation, <https://arxiv.org/abs/2110.06877>, 2021
- [2.] B. Artacho; A. Savakis (2021): OmniPose: A Multi-Scale Framework for Multi-Person Pose Estimation. In ArXiv abs/2103.10180. Available online at <https://www.semanticscholar.org/paper/6b8c7bb60ab6f9f8aaa876b657af6b559bd82cd1>.
- [3.] C. Lugaressi; Jiuqiang Tang; Hadon Nash; C. McClanahan; Esha Uboweja; Michael Hays et al. (2019): MediaPipe: A Framework for Building Perception Pipelines. In ArXiv abs/1906.08172. Available online at <https://www.semanticscholar.org/paper/1c44a3cde07f3f2ffd8880b441449483ed74fc5f>.
- [4.] C. Lugaressi; Jiuqiang Tang; Hadon Nash; C. McClanahan; Esha Uboweja; Michael Hays et al. (2019): MediaPipe: A Framework for Perceiving and Processing Reality. Available online at <https://www.semanticscholar.org/paper/1cd227fb3dacda18ee94d08b04fc1d5b9afb351>.
- [5.] Feng Zhang; Xiatian Zhu; Chen Wang (2021): Single Person Pose Estimation: A Survey. Available online at <https://www.semanticscholar.org/paper/82801042ee2cb9dfbb43344e24944ea0ad95f6c7>.
- [6.] Hongwen Zhang; Jie Cao; Guo Lu; Wanli Ouyang; Zhenan Sun (2020): Learning 3D Human Shape and Pose from Dense Body Parts. In IEEE Transactions on Pattern Analysis and Machine Intelligence PP. DOI: 10.1109/TPAMI.2020.3042341.
- [7.] Jiefeng Li; Siyuan Bian; Ailing Zeng; Can Wang; Bo Pang; Wentao Liu; Cewu Lu (2021): Human Pose Regression with Residual Log-likelihood Estimation. Available online at <https://www.semanticscholar.org/paper/ee894195a8135bd4e189bfb77fe0dc9f844c5e2a>.
- [8.] Jing Zhang; Zhe Chen; D. Tao (2019): Human Keypoint Detection by Progressive Context Refinement. In ArXiv abs/1910.12223. Available online at <https://www.semanticscholar.org/paper/91d74b1d41aa554a7231ef65c21f99bbe2365435>.
- [9.] Ke Sun; Bin Xiao; Dong Liu; Jingdong Wang (2019): Deep High-Resolution Representation Learning for Human Pose Estimation. In 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 5686–5696. DOI: 10.1109/CVPR.2019.00584.
- [10.] Kristijan Bartol; David Bojanić; T. Petković; N. D'Apuzzo; T. Pribanić (2020): A Review of 3D Human Pose Estimation from 2D Images. DOI: 10.15221/20.29.
- [11.] Liangchen Song; Gang Yu; Junsong Yuan; Zicheng Liu (2021): Human pose estimation and its application to action recognition: A survey. In J. Vis. Commun. Image Represent. 76, p. 103055. DOI: 10.1016/J.JVCIR.2021.103055.
- [12.] Shashank Tripathi; Siddhant Ranade; A. Tyagi; Amit Agrawal (2020): PoseNet3D: Unsupervised 3D Human Shape and Pose Estimation. In ArXiv abs/2003.03473. Available online at <https://www.semanticscholar.org/paper/507e65cf5994f012d7732b78006398433a945df5>.
- [13.] Valentin Bazarevsky; I. Grishchenko; Karthik Raveendran; Tyler Lixuan Zhu; Fangfang Zhang; Matthias Grundmann (2020): BlazePose: On-device Real-time Body Pose tracking. In ArXiv abs/2006.10204. Available online at <https://www.semanticscholar.org/paper/c9bcea08fb81c041ed6d2b7576d8f0e47c1c850f>.

- [14.] Ying Huang; Bin Sun; Haipeng Kan; Jiankai Zhuang; Zengchang Qin (2019): FollowMeUp Sports: New Benchmark for 2D Human Keypoint Recognition. In ArXiv abs/1911.08344. DOI: 10.1007/978-3-030-31726-3\_10.
- [15.] Yucheng Chen; Yingli Tian; Mingyi He (2020): Monocular human pose estimation: A survey of deep learning-based methods. In Comput. Vis. Image Underst. 192, p. 102897. DOI: 10.1016/j.cviu.2019.102897.
- [16.] Zhe Cao; Gines Hidalgo; Tomas Simon; Shih-En Wei; Yaser Sheikh (2021): OpenPose: Realtime Multi-Person 2D Pose Estimation Using Part Affinity Fields. In IEEE Transactions on Pattern Analysis and Machine Intelligence 43, pp. 172–186. DOI: 10.1109/TPAMI.2019.2929257.
- [17.] Valentin Bazarevsky and Ivan Grishchenko, Research Engineers, Google Research: On-device, Real-time Body Pose Tracking with MediaPipe BlazePose  
<https://ai.googleblog.com/2020/08/on-device-real-time-body-pose-tracking.html>
- [18.] J. Tompson, A. Jain, Y. LeCun, C. Bregler: Joint training of a convolutional network and a graphical model for human pose estimation, Advances in Neural Information Processing Systems (2014), pp. 1799-1807
- [19.] A. Jain, J. Tompson, Y. LeCun, C. Bregler: Modeep: A deep learning framework using motion features for human pose estimation, Asian Conference on Computer Vision (2014), pp. 302-315
- [20.] J. Tompson, R. Goroshin, A. Jain, Y. LeCun, C. Bregler: Efficient object localization using convolutional networks, IEEE Conference on Computer Vision and Pattern Recognition (2015), pp. 648-656
- [21.] S. Wei, V. Ramakrishna, T. Kanade, Y. Sheikh, Convolutional pose machines, IEEE Conference on Computer Vision and Pattern Recognition (2016), pp. 4724-4732
- [22.] J. Carreira, P. Agrawal, K. Fragiadaki, J. Malik: Human pose estimation with iterative error feedback, IEEE Conference on Computer Vision and Pattern Recognition (2016), pp. 4733-4742
- [23.] A. Newell, K. Yang, J. Deng: Stacked hourglass networks for human pose estimation, European Conference on Computer Vision (2016), pp. 483-499
- [24.] X. Chu, W. Yang, W. Ouyang, C. Ma, A.L. Yuille, X. Wang: Multi-context attention for human pose estimation, IEEE Conference on Computer Vision and Pattern Recognition (2017), pp. 5669-5678
- [25.] W. Yang, S. Li, W. Ouyang, H. Li, X. Wang: Learning feature pyramids for human pose estimation, IEEE International Conference on Computer Vision (2017), pp. 1290-1299
- [26.] X. Nie, J. Feng, J. Xing, S. Xiao, S. Yan: Hierarchical contextual refinement networks for human pose estimation, IEEE Trans. Image Process., 28 (2018), pp. 924-936
- [27.] Liangchen Song, Gang Yu, Junsong Yuan, Zicheng Liu: Human pose estimation and its application to action recognition: A survey, Journal of Visual Communication and Image Representation, Volume 76, 2021, <https://doi.org/10.1016/j.jvcir.2021.103055>.