2022

Fingerpose Functioneel Ontwerp

TADRAŁA,PIOTR P.P.

VERSIE 1.1

Inhoud

Versiebeheer

VERSIE	DATUM	OPMERKING
1.0	13-12-22	Eerste concept van het document.
1.1	08-01-23	Lijst met requirements uitgebreid.
1.1	08-01-23	Toelichting over Mediapipe en Fingerpose uitgebreid.
1.1	08-01-23	Toelichting over UNIX Sockets toegevoegd.
1.1	08-01-23	Flowchart toelichting toegevoegd.
1.1	08-01-23	Use case diagram context uitgebreid
1.1	08-01-23	Conclusie toegevoegd.

Inleiding

Het doel van dit algoritme is om als universele backend te functioneren voor applicaties die gebarentaalherkenning willen toepassen. Door middel van een interface kan het algoritme door verschillende soorten frontends gebruikt worden. Het algoritme zal aangeroepen kunnen worden via de command line interface (CLI), met het aantal frames als parameter. Het zal anchor points genereren op basis van de webcamfeed en deze anchor points converteren naar menselijk leesbare parameters die vervolgens herkend worden als gebaren. De response zal vervolgens via een Unix socket teruggestuurd worden naar de frontend waar die vervolgens verwerkt kan worden.

Requirements

- Het algoritme moet aangeroepen kunnen worden via command line interface CLI (*Met aantal frames als parameter*)
- Het moet in staat zijn om te werken met verschillende soorten gebarentalen (*Mogelijkheid hebben om meerdere datasets te importeren / genereren*).
- Het algoritme moet dynamische gebaren kunnen onderscheiden.
- Anchor points genereren op basis van webcam feed.
- Anchor points converteren naar human readable parameters (*Fingerpose input*) en die vervolgens herkennen als gebaren.
- Response returnen via Unix socket.
- Het algoritme moet in staat zijn om te integreren met verschillende soorten front-end applicaties (Door middel van API) .

Toegepaste technieken

Mediapipe_Hands

MediaPipe Hands is een framework ontwikkeld door Google voor het bouwen van pipelines voor hand- en vingerbewegingen. Het maakt gebruik van machine learning om handen en vingers in real-time nauwkeurig te detecteren en te volgen. Dit kan gebruikt worden in verschillende toepassingen, zoals virtual- en augmented reality, gaming en video-analyse. Het framework is onderdeel van het MediaPipe-framework dat een verzameling modulaire componenten biedt voor het bouwen van machine learning pipelines. MediaPipe Hands is een krachtig hulpmiddel voor het ontwikkelen van applicaties die gebruik maken van hand- en vingerbewegingen, omdat het de nauwkeurigheid en snelheid van de herkenning verhoogt dankzij de hoeveelheid data dat gegenereerd wordt bij het gebruik.

Fingerpose

Fingerpose is een open-source Node JavaScript/Python library die gebruik maakt van het MediaPipe framework. Door middel van een simpele RGB-webcam kan Fingerpose gebaren herkennen en deze omzetten naar parameters met behulp van MediaPipe. Om de parameters om te zetten naar letters, cijfers of woorden is een dataset nodig.

Fingerpose maakt gebruik van een statische dataset waarbij aan elk teken alle mogelijke parameters en 'Confidence' punten zijn gekoppeld. De library kijkt naar de parameters van elke vinger en vergelijkt deze met alle items uit de geselecteerde dataset in real-time. Het teken met de meeste confidence punten wordt geselecteerd als het juiste teken. Om user error te voorkomen en de nauwkeurigheid te verbeteren kan een dataset item meerdere parameters toegewezen hebben aan een vinger. Fingerpose is een handige tool voor het omzetten van gebaren naar parameters en het herkennen van deze gebaren in real-time, dankzij de combinatie van het MediaPipe framework en de beschikbare datasets.

UNIX Sockets

Een Unix socket is een IPC *(inter-process communicatiemechanisme)* dat wordt gebruikt om te communiceren tussen processen op hetzelfde systeem. Het maakt gebruik van een bestand in de system files als endpoint voor het versturen en ontvangen van data tussen processen.

Wanneer een Unix socket wordt gebruikt voor IPC, wordt er een bestand aangemaakt in het systeem met een specifiek pad. Dit bestand functioneert als een endpoint voor de communicatie tussen de processen. Wanneer een proces gegevens naar het bestand schrijft, worden de gegevens doorgestuurd naar het andere proces dat verbonden is met het Unix socket via het bestand. Op deze manier kunnen processen op het systeem met elkaar communiceren door middel van het bestand als een soort intermediair.

Unix sockets bieden verschillende voordelen ten opzichte van andere inter-process communicatiemechanismen, zoals lagere overhead *(hoeveelheid aan tijd en middelen die nodig zijn om bepaalde taken te voltooien)* en betere prestaties. Bovendien kunnen Unix sockets gebruikt worden in combinatie met andere IPC-mechanismen, zoals pipes en named pipes, om complexere communicatiestructuren te creëren. In het algoritme zal het Unix socket gebruikt worden om de response van het algoritme terug te sturen naar de aanroepende front-end applicatie.

Flowchart

Toelichting

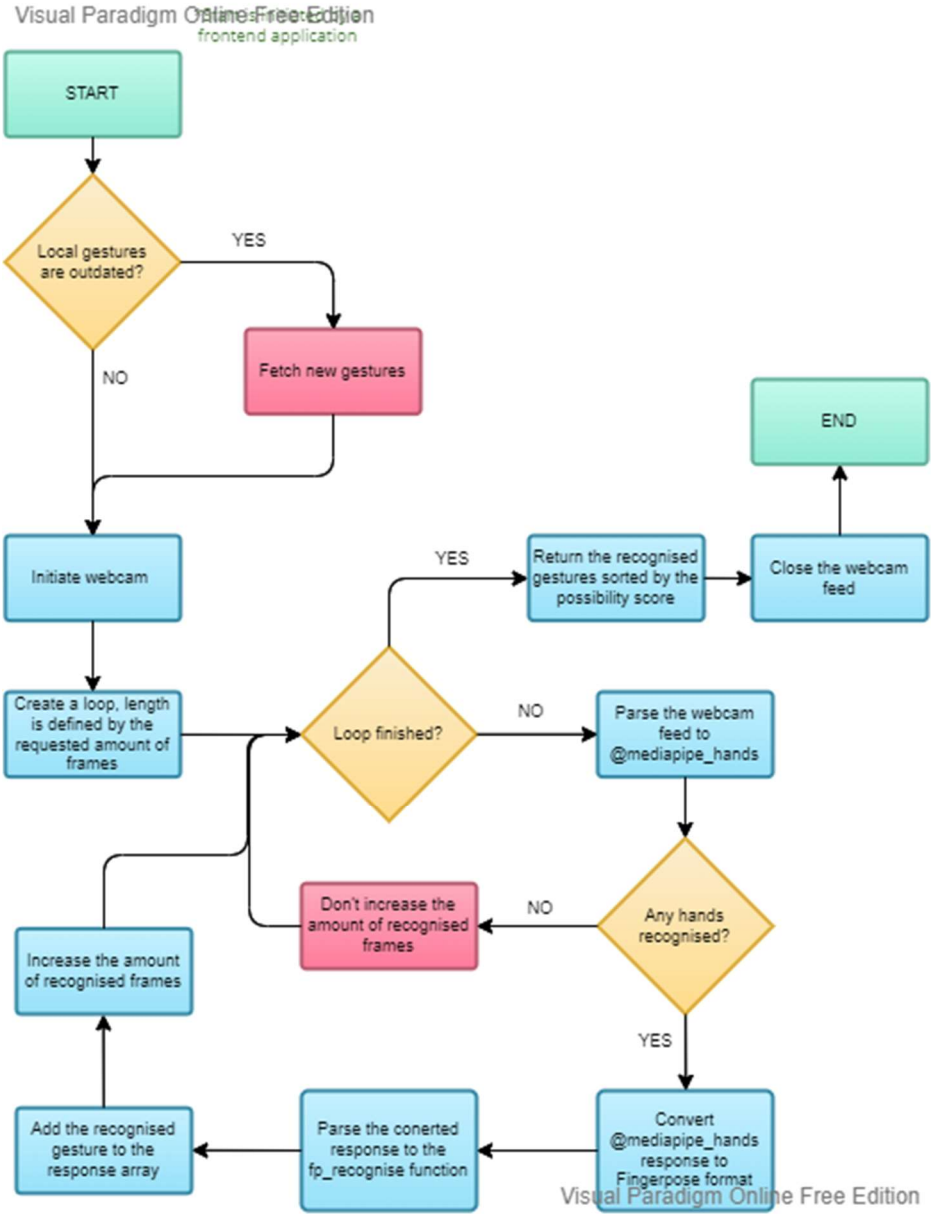
De applicatie wordt geïnitieerd door de API via de command line interface met het aantal frames per seconde als parameter *(bijv. /backend python demo.py 10)*. Vervolgens verifieert het algoritme of de lokale dataset up-to-date is. Indien dat niet het geval is, worden de nieuwste dataset gedownload. De webcam wordt vervolgens geopend en een loop wordt gedefinieerd, waarbij de lengte afhankelijk is van het opgegeven aantal frames.

De webcam feed wordt vervolgens doorgevoerd naar MediaPipe Hands, dat bepaalt of er handen gedetecteerd zijn. Indien dat niet het geval is, begint de loop opnieuw. Zo ja, worden de anchor points omgezet naar het Fingerpose formaat (*[{x:1, y:2, z:3}] => [[1, 2 ,3]]*) en wordt de functie

fp_recognise aangeroepen met de anchor points als parameter. De response van fp_recognise (*een lijst met gebaren*) wordt toegevoegd aan het response en de loop begint opnieuw.

Wanneer de loop beëindigd is, wordt het response array gesorteerd op basis van het possibility score. Vervolgens wordt de webcam afgesloten en stuurt de applicatie de response via een Unix socket naar de API. Tot slot sluit de applicatie zichzelf af.

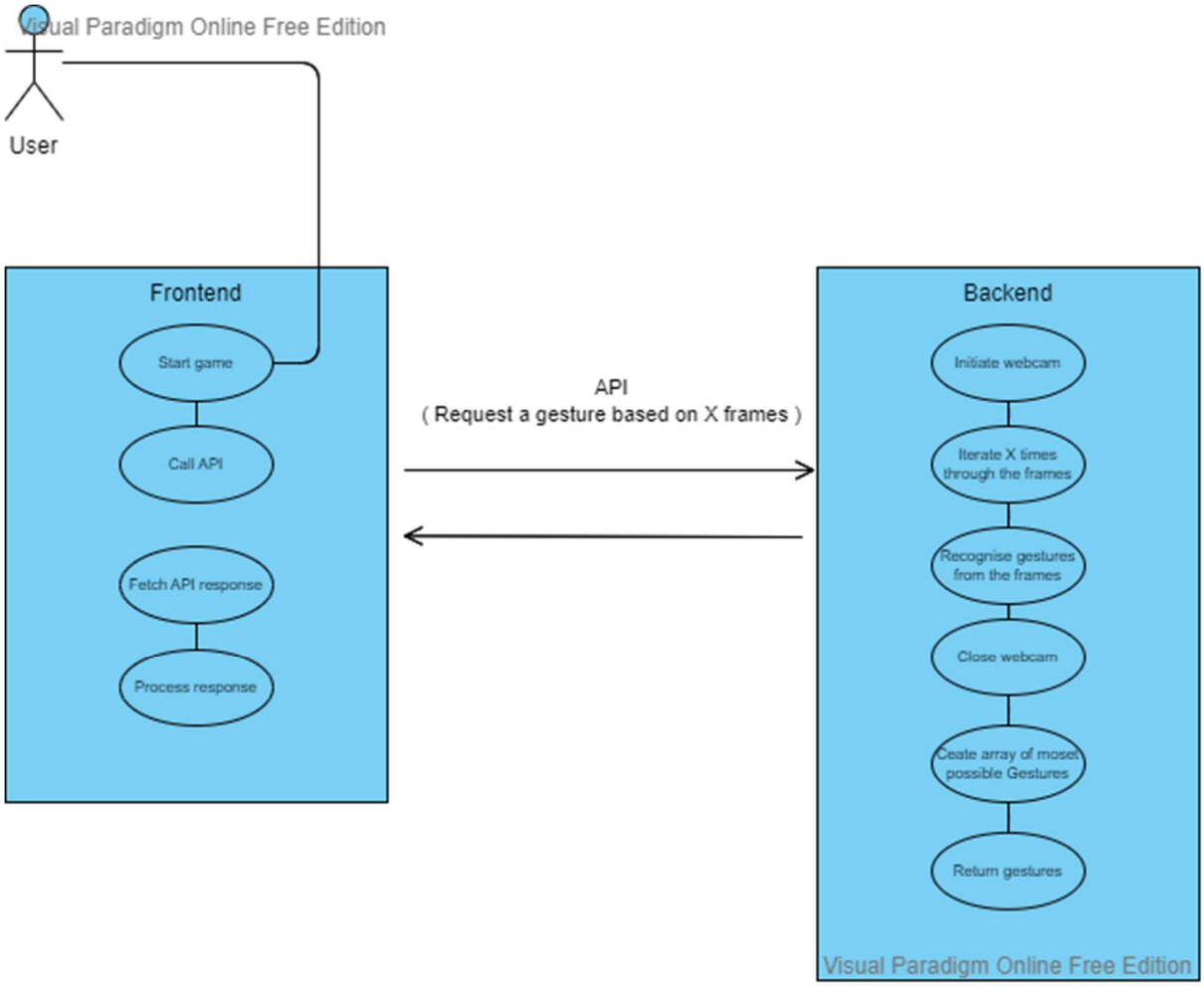
Diagram



Use Case Diagram

Het algoritme zal ontwikkeld worden in Python en is daarom noodzakelijk dat elke front-end die gebruik wil maken van het algoritme de mogelijkheid heeft om Python te interpreteren. Devices met Windows, MacOS en Linux ondersteunen Python, dus het zal mogelijk zijn om het algoritme op deze apparaten te draaien.

Bij een volledige stack (front-end, API en back-end) zal de applicatie als volgt werken: een gebruiker start de game/applicatie. Zodra de game een gebaar moet herkennen, wordt de API aangeroepen. De back-end van het algoritme wordt vervolgens geïnitieerd door de API en wordt het algoritme flow uitgevoerd (*Zie Flowchart diagram*). De API ontvangt vervolgens een array met gebaren als response, die doorgevoerd wordt naar de front-end. De front-end beslist vervolgens zelf wat er met de lijst gebeurt, zoals het gebruiken van het eerste gebaar of het laten kiezen van een gebaar door de gebruiker.



Acceptatietest

Prefix	The test can be recognized by the **prefix**.
Index	The iteration can be recognized by the **Index.**
Afterfix	The exceptional flow test can be recognized by the **afterfix.**

Code	Instructie	Verwacht resultaat
ALGORITHM		
A_0	1. Call /python main.py 10 2. Webcam feed should start recoding. 3. Make a gesture in front of the webcam	After 10 Frames the algorithm will return a positive response of an array with the recognized gestures.
A_1_E	1. Call /python main.py	Algorithm will return a negative response as no valid parameters were provided.
A_2_E	2. Call /python main.py -10	Algorithm will return a negative response as no valid parameters were provided.
Convert Mediapipe to Fingerpose		
CMF_0	1. Call convertLandmarks() function 2. Use this variable as parameter: [{x: 5, y: 5, z: 2}, …]	Function will return a nested array without defined indexes: [[5, 5, 2] …]
CMF_1_E	1. Call convertLandmarks() function 2. Use this variable as parameter: [{x: 5, y: 5 }, …]	Function will return a negative response, all parameters must be provided.
Is finger curled (Part of the recognition alogirthm)		
IFC_0	1. Call is_finger_curled() function 2. Use these variables as parameters: (1, 2, 3)	Function will return: NoCurl.
IFC_1	1. Call is_finger_curled() function 2. Use these variables as parameters: (1, 2, 2)	Function will return: HalfCurl.
IFC_2	1. Call is_finger_curled() function 2. Use these variables as parameters: (1, 2, 3)	Function will return: FullCurl.
IFC_3_E	1. Call is_finger_curled() function 2. Use these variables as parameters: (1, 2)	Function will return a negative response.

Conclusie

In conclusie biedt het algoritme dat we ontwikkelen een universele back-end oplossing voor applicaties die gebarentaal herkenning willen toepassen. Door gebruik te maken van het MediaPipe Hands framework en de Fingerpose library, kunnen we hand- en vingerbewegingen in real-time nauwkeurig detecteren en omzetten naar leesbare parameters. Deze parameters worden vervolgens doorgegeven aan de front-end applicatie via een Unix socket, die verantwoordelijk is voor het interpreteren van de gebaren en het verwerken van de response.

Dankzij de compatibiliteit met verschillende devices en het gemak waarmee het algoritme kan worden geïntegreerd in een breed scala aan toepassingen, biedt het een veelzijdige oplossing voor gebarentaal herkenning. Het algoritme is eenvoudig te implementeren en kan worden gebruikt in een groot aantal situaties waarbij gebarentaal herkenning nodig is.