2022

FINGERPOSE RELATIVE MOVEMENT RECOGNITION

CUSTOM EXTENSION OP BESTAANDE GESTURE RECOGNITION LIBRARY

TADRALA,PIOTR P.P. 487080

Versie 1.1

Inhoud

Versiebeheer

VERSIE	DATUM	OPMERKING
1.0	15-10-22	Eerste concept het document.
1.1	10-01-23	Duidelijke probleemstelling in het voorwoord verwerkt.
1.1	10-01-23	@mediapipe/face_detection en Fingerpose toelichting toegevoegd aan 'Toegepaste technieken'
1.1	10-01-23	'Techinisch uitleg' duidelijker gemaakt.
1.1	11-01-23	'voorbeeld' duidelijker gemaakt
1.1	11-01-23	'Selectie' toegevoegd.
1.1	11-01-23	'Beeprkingen' toegevoegd.
1.1	11-01-23	'Conclusie' toegevoegd.

Voorwoord

Fingerpose is een Node.JS / Python Library die gebruikmaakt van het MediaPipe Framework om handgebaren te detecteren en om te zetten naar parameters. Deze parameters zijn belangrijk voor het genereren van datasets en voor het koppelen van bepaalde gebaren aan letters of woorden. Hoewel Fingerpose een praktisch tool is voor het detecteren van statische gebaren, mist het de mogelijkheid om bewegingen te herkennen. In veel gevallen zijn gebaren namelijk niet statisch, waardoor de detectie en herkenning van deze bewegingen cruciaal is.

Daarom stel ik in dit document een oplossing voor om een library genaamd FrMR (*Fingerpose (Relative) Movement Recognition*) te ontwikkelen en toe te voegen aan Fingerpose. Hierdoor zal het mogelijk zijn om een breder scala van gebaren te herkennen en wordt het probleem van dynamische gebaren opgelost. In de volgende paragrafen zal ik ingaan op de toegepaste technieken, het proces van bewegingsherkenning en een voorbeeld van de implementatie in de praktijk.

Toegepaste techniek

Gezichtsherkenning

Om te beginnen hebben we een gezichtsherkenning library nodig die ons gaat helpen de coördinaten van het gezicht te bepalen. Er zijn verschillende ready-to-go bibliotheken waar we gebruik van kunnen maken, en aangezien we geen uitgebreide gezichtsherkenning nodig hebben, maar alleen de positie van het gezicht willen bepalen, maakt dit het gemakkelijker. Mijn voorkeur gaat uit naar @mediapipe/face_detection (demo) aangezien mediapipe ook vergelijkbare libraries heeft die we kunnen implementeren:

- @mediapipe/face_mesh
- @mediapipe/hands
- @mediapipe/holistic
- @mediapipe/objectron
- @mediapipe/pose
- @mediapipe/selfie_segmentation

@mediapipe/face_detection

Een specifieke library van Mediapipe die nuttig zou kunnen zijn voor ons doel is @mediapipe/face_detection. Deze library maakt gebruik van CV (*computer vision*) om gezichten te detecteren in real-time en deze te vertalen naar een set van eenvoudige gezichtsfeatures zoals de gezichtspositie, de gezichtsvorm en de oogpositie. Deze informatie zal dienen als referentiepunt voor onze bewegingsherkenning en helpt om ervoor te zorgen dat we een precieze relatie kunnen vaststellen tussen de handgebaren en het gezicht.

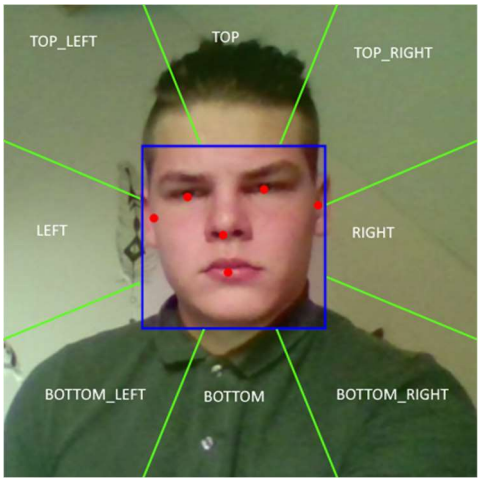
Fingerpose

Tot slot, Fingerpose wordt gebruikt als basis voor ons project. We zullen de bestaande functionaliteiten van Fingerpose gebruiken en hier onze bewegingsherkenning aan toevoegen. Dit zal ons in staat stellen om een breder scala van gebaren te herkennen en de nauwkeurigheid van de detectie te verhogen.

Technisch uitleg

Refferentiepunt

Om bewegingen te kunnen detecteren, is het noodzakelijk om een referentiepunt te hebben. In ons geval is dat het gezicht. Zoals eerder genoemd, gebruiken we @mediapipe/face_detection library om de coördinaten van het gezicht te bepalen, en gebruiken die als referentiepunt. Een beweging zal een startpunt hebben in relatie tot het gezicht.



Daarna zullen alle verdere bewegingen worden berekend met de handpositie van het vorige frame als referentiepunt. Resultaat hiervan is een array met locaties in relatie tot het vorige frame. Dit maakt het mogelijk om bewegingen in real-time te herkennen en te vergelijken met onze dataset van gebaren. Voor de beweging van de letter "Z" die in tien frames is opgenomen, ziet het er als volgt uit:

Frame	Locatie
1	BOTTOM_RIGHT (ten opzichte van het gezicht)

2	LEFT
3	LEFT
4	LEFT
5	BOTTOM_RIGHT
6	BOTTOM_RIGHT
7	BOTTOM_RIGHT
8	LEFT
9	LEFT
10	LEFT

Data filtreren

Om fouten met betrekking tot het aantal frames per gebaar te voorkomen, is het noodzakelijk om de gegevens te filteren en distincten. Dit betekent dat we alleen de unieke locaties in onze array van bewegingen behouden, en ook dat we irrelevant data verwijderen. Dit helpt om de precisie van de herkenning te verhogen, door ervoor te zorgen dat we alleen de relevante bewegingen in ons dataset verwerken.

Stap	Locatie	Confidence points
1	BOTTOM_RIGHT (ten opzichte van het gezicht)	1pt
2	LEFT	1pt
3	BOTTOM_RIGHT	1pt
4	LEFT	1pt

Confidence points

In deze stappen zullen we ook werken met "Confidence Points" om user-errors te voorkomen. Dit zal de nauwkeurigheid verhogen door de juiste bewegingen belonen met meer punten. Hierdoor kunnen we meerdere bewegingen aan een letter koppelen door het aantal punten per stap te bepalen, afhankelijk van hoe correct de beweging is. Dit zorgt voor een grotere flexibiliteit in het detecteren van gebaren en een verbetering van de nauwkeurigheid.

Stap	Locatie	Confidence points
1	BOTTOM_RIGHT (ten opzichte van het gezicht)	1pt
2	LEFT	1pt
3	BOTTOM_RIGHT	1pt
4	BOTTOM	.75pt
5	LEFT	1pt

Voorbeeld

In dit voorbeeld wordt beschreven hoe de bewegingsherkenning voor de letter "Z" zou kunnen werken met behulp van een dataset item. Door middel van real-time vergelijking van de huidige beweging met alle beschikbare bewegingen uit de dataset, probeert de code een match te vinden voor de opgenomen beweging. Het kan echter zijn dat een beweging niet in relatie tot het gezicht wordt gemaakt. In dat geval zal de eerste stap worden genegeerd.

Het eerste voorbeeld toont hoe de bewegingen voor de letter "Z" worden herkend door een dataset item. In dit voorbeeld wordt aangenomen dat de beweging van de letter "Z" in vier stappen is opgenomen en wordt vervolgens vergeleken met de dataset. Zoals je kunt zien in de tabel hieronder, krijgt de letter "Z" een probability rating van 2. Dit is een indicatie van hoe nauwkeurig de beweging is herkend.

Dataset item voor de letter 'Z'

Stap	Locatie	points
1	BOTTOM_RIGHT	1
2	LEFT	1
3	BOTTOM_RIGHT	1
4	LEFT	1

Movement recognition

Stap	Locatie	Points
1	BOTTOM_RIGHT	1
2	LEFT	1
3	BOTTOM_RIGHT	1
4	BOTTOM_LEFT	1

Daarop, om de nauwkeurigheid te verhogen, kan een tweede dataset item voor de letter "Z" worden toegevoegd. Dit zorgt ervoor dat er meer mogelijke matches beschikbaar zijn om de letter "Z" te herkennen. Echter krijgt de laatste semi-correcte beweging niet de totale aantal punten aangezien het geen 'perfecte beweging' is. Met dit trucje hebben we de rating naar 2.75 verhoogt.

Dataset item voor de letter 'Z'

Stap	Locatie	points
1	BOTTOM_RIGHT	1
2	LEFT	1
3	BOTTOM_RIGHT	1
4	BOTTOM_LEFT	.75

Movement recognition

Stap	Locatie	Points
1	BOTTOM_RIGHT	1
2	LEFT	1
3	BOTTOM_RIGHT	1
4	BOTTOM_LEFT	.75

Selectie

Een manier om de score van Fingerpose en FRMR samen te berekenen, is door de scores van beide bibliotheken te combineren en te normaliseren tot een score van 0 tot 10. Hierdoor kunnen we een eenvoudige selectie maken van het resultaat.

Een voorbeeld van hoe we dit kunnen berekenen is door gebruik te maken van een gewichtsfactor voor beide scores. Laten we aannemen dat we een gewichtsfactor van 0,7 toekennen aan de Fingerpose score en een gewichtsfactor van 0,3 toekennen aan de FRMR score. De totale score zou dan worden berekend als (*Fingerpose-score x 0,7*) + (*FRMR-score x 0,3*). Dit zorgt ervoor dat we zowel de sterktes van Fingerpose (*detectie van statische gebaren*) als FRMR (*detectie van bewegingen*) in rekening kunnen nemen bij het maken van een selectie.

Daarnaast kan ook een threshold ingesteld worden voor de beide scores. Wanneer de score boven de threshold komt wordt het gebaar herkent.

Beperkingen

Een van de beperkingen van de huidige implementatie van FrMR is de afhankelijkheid van een beperkt aantal stappen voor het herkennen van een beweging. Dit betekent dat gebaren die uit een groter aantal stappen bestaan of gebaren die grote afstanden tussen de stappen vereisen, minder nauwkeurig herkend kunnen worden. Dit kan een probleem vormen in situaties waar gebaren uitgebreide bewegingen vereisen.

Daarnaast, FrMR is momenteel niet toegerust om bewegingen te herkennen in 3D ruimtes, de huidige implementatie is specifiek gericht op 2D ruimtes. Dit betekent dat gebaren die specifiek in 3D ruimtes worden uitgevoerd, niet herkend kunnen worden. Dit kan een beperking zijn in toepassingen waar 3D ruimtes relevant zijn.

Tot slot, FrMR als een add-on naar Fingerpose, resulteert in een hogere complexiteit van de implementatie, waardoor het moeilijker kan zijn om de systemen te onderhouden of te updaten. Ook is het nodig om het volledige proces van implementatie goed te documenteren.

Conclusie

In conclusie, FrMR is een aanvulling op de bestaande Fingerpose library die de mogelijkheid biedt om bewegingen te herkennen en te koppelen aan letters of woorden. Door de implementatie van FrMR wordt de functionaliteit van Fingerpose uitgebreid waardoor een breder scala van gebaren herkend kan worden en het probleem van dynamische gebaren wordt opgelost.

Hoewel de toepassing van FrMR in theorie een excellente oplossing is voor het herkennen van dynamische gebaren, brengt het enorm veel complexiteit toe aan Fingerpose. Hierdoor kan het moeilijker zijn om de systemen te onderhouden of te updaten. Daarom is het belangrijk om goede documentatie te hebben en een duidelijke structuur voor implementatie en onderhoud.