UNIVERZITA KOMENSKÉHO V BRATISLAVE

FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

Nahrávanie animácií v Unity získaných modelmi hlbokého učenia

Bakalárska práca

2022 **Ladislav Suchý**

UNIVERZITA KOMENSKÉHO V BRATISLAVE

FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

Nahrávanie animácií v Unity získaných modelmi hlbokého učenia

Bakalárska práca

Študijný program: Aplikovaná informatika
Študijný odbor: Informatika
Školiace pracovisko: Katedra aplikovanej informatiky
Školiteľ: RNDr. Andrej Lúčny, PhD.

2022 **Ladislav Suchý**

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Ladislav Suchý
Študijný program: aplikovaná informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Nahrávanie animácií v Unity získaných modelmi hlbokého učenia
Recording animations in Unity provided by the deep learning models

Anotácia: Táto implementačná práca využíva existujúce modely hlbokého učenia určené na získanie reprezentácie hlavy (alternatívne je možné riešiť celú postavu, stačí riešiť jeden typ úlohy). Túto reprezentáciu premieta do pohybu avatara vo virtuálnom prostredí vytvorenom herným enginom Unity. Vďaka tomu sa avatar hýbe podľa pohybu postavy užívateľa, ktorého sníma obyčajná kamera. Úlohou práce je takýto systém rozbehať a dorobiť do neho nahrávanie tohto pohybu do podoby animácie. Tú možno neskôr použiť na animovanie avatara nezávislé od systému nahrávania. Úlohou práce nie je zaoberať ako použité modely hlbokého učenia pracujú, avšak popíše ich vstup, výstup a analyzuje dostupné možnosti v tejto oblasti.

Cieľ: Cieľom práce je vytvoriť plugin alebo asset do Unity, ktorý umožní nahrávanie animácii získaných modelmi hlbokého učenia.

Literatúra: Tomáš Holan: Unity, CZ NIC, 2021
Developerská web stránka Unity www.unity.com
Dokumentácia k projektu https://github.com/TianxingWu/OpenVHead.git

Poznámka: Platforma: Unity. C#, C++.

Kľúčové slová:

Vedúci: RNDr. Andrej Lúčny, PhD.
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: prof. Ing. Igor Farkaš, Dr.

Dátum zadania: 07.10.2021

Dátum schválenia: doc. RNDr. Damas Gruska, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Poďakovanie

Abstrakt

Abstract

Úvod

1. Východiská

Táto kapitola obsahuje teoretické základy, ktoré sú využívané v mojej bakalárskej práci. Kapitola začína vysvetlením strojového učenia, hlbokého učenia a počítačového videnia. Nasleduje opis knižnice OpenCV a orientačných bodov tváre. Ďalej vysvetľuje fungovanie Kalmanovho filtra a animácií v Unity. Na záver opisuje existujúce riešenie, na ktoré moja bakalárska práca nadväzuje.

1.1 Strojové učenie

Strojové učenie je systém, ktorý za pomoci množstva príkladov relevantných k úlohe dokáže vytvoriť štatistickú štruktúru a na základe nej určiť pravidlá pre automatizovanie úlohy. Ak by našou úlohou bolo označiť všetky obrázky vytvorené na dovolenke pri mori, poskytli by sme systému množstvo obrázkov, ktoré boli ľuďmi označené ako obrázky na dovolenke. Systém by vytvoril pravidlá ako: na obrázku sa nachádza more alebo na obrázku sa nachádza pláž. Pomocou týchto pravidiel by systém dokázal označiť všetky obrázky zo vstupu vytvorené na dovolenke. [3]

Pre fungovanie strojového učenia sú potrebné tri veci:

- Vstupné dáta – tie sa líšia na základe úlohy, v prípade rozpoznávania obrázkov to sú napríklad obrázky.
- Príklady očakávaného výstupu – v prípade rozpoznávania obrázkov to sú značky, ktoré označujú veci na obrázku napríklad mačka, pes a podobne.
- Spôsob merania úspešnosti algoritmu – toto meranie sa využíva na určenie veľkosti rozdielu medzi aktuálnym výstupom algoritmu a očakávaným výstupom. Na základe tohto rozdielu sa následne upravuje spôsob, akým algoritmus funguje. Tento krok prispôsobenia sa volá učenie. [3]

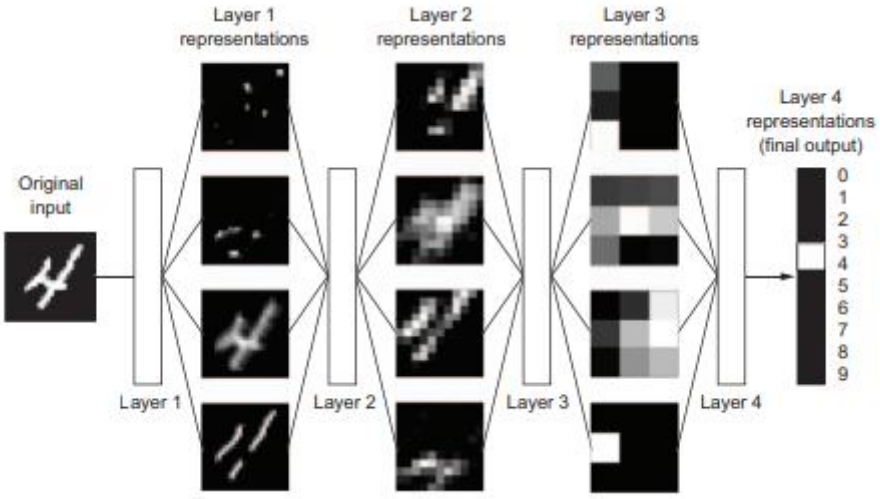
Jedným z hlavných problémov strojového a hlbokého učenia je zvolenie rozumnej reprezentácií údajov, na ktorú bude algoritmus transformovať vstupné údaje, aby sa čo najviac priblížil očakávanému výstupu. Reprezentáciu si môžeme predstaviť ako iný spôsob pohľadu na údaje, ako sa budú dáta kódovať. Farebný obrázok môže byť zakódovaný vo

formáte RGB alebo HSV. Sú to dve rôzne reprezentácie rovnakých údajov. Ak by sme mali za úlohu nájsť na obrázku všetky pixely určitej farby, jednoduchším spôsobom by bolo použiť reprezentáciu vo formáte RGB. Naopak, ak by sme chceli urobiť farby obrázka menej nasýtenými, využili by sme reprezentáciu vo formáte HSV. Preto je výber reprezentácie veľmi dôležitý. [3]

1.2 Hlboké učenie

Je to podoblasťou strojového učenia. Proces hlbokého učenia sa skladá z mnoho postupných vrstiev čoraz zmysluplnejších reprezentácií. To, koľko vrstiev reprezentácií model hlbokého učenia obsahuje sa nazýva hĺbka modelu. Modely hlbokého učenia často obsahujú desiatky až stovky po sebe idúcich reprezentácií, ktoré sa automaticky učia z tréningových údajov. Tieto vrstvy reprezentácií sa učia pomocou modelov nazývaných neurónové siete. [3]

Na obr. 1 vidíme transformáciu obrázku s číslom na vrstvy reprezentácií. Tie sa po každej vrstve čoraz menej podobajú na pôvodný obraz ale poskytujú čoraz viac informácií o konečnom výsledku. [3]



Obr. 1: Vrstvy reprezentácií hlbokého učenia, prevzaté z [3].

1.3 Počítačové videnie

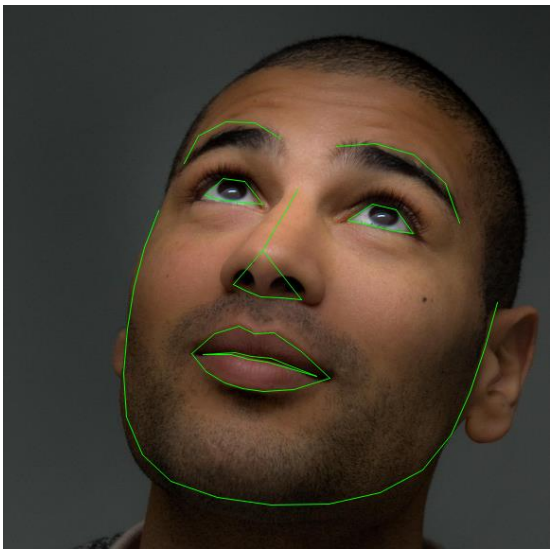
Je transformácia údajov z fotoaparátu alebo videokamery za účelom vytvorenia novej reprezentácie alebo získania informácií z údajov. Nová reprezentácia môže znamenať napríklad premena farebného obrázka na čiernobiely alebo odstránenie pohybu kamery zo sekvencie obrázkov. Pomocou počítačového videnia sa dajú zo vstupných údajov získať informácie o objektoch a pozadí. Príkladom takýchto informácie je nájdenie polohy tváre na obrázku, zistenie vzdialenosti fotoaparátu od určitého objektu alebo rozhodnutie o tom, či sa fotoaparát nachádza vo vozidle. [1]

1.4 OpenCV

Ide o knižnicu pre Python, ktorá je navrhnutá pre aplikácie pracujúce s dátami v reálnom čase, napríklad dáta z webkamery. Je napísaná v optimalizovanom jazyku C a dokáže pracovať s výhodami viacjadrových procesorov. OpenCV poskytuje infraštruktúru počítačového videnia, ktorá je často využívaná na rýchle a pomerne jednoduché vytváranie aplikácií využívajúce počítačové videnie. Je užitočná v rôznych oblastiach ako napríklad kontrola produktu v továrňach, kalibrácie kamier, a robotiky. [1]

1.5 Orientačné body tváre

Na určenie orientačných bodov tváre využívame predvídateľ tvaru, ktorý sa pokúša lokalizovať kľúčové body daného tvaru. V prípade tváre je to štruktúra tváre na obr. 2. [2]



Obr. 2: Štruktúra tváre, prevzaté z [2].

Detekcia orientačných bodov tváre sa skladá z dvoch krokov:

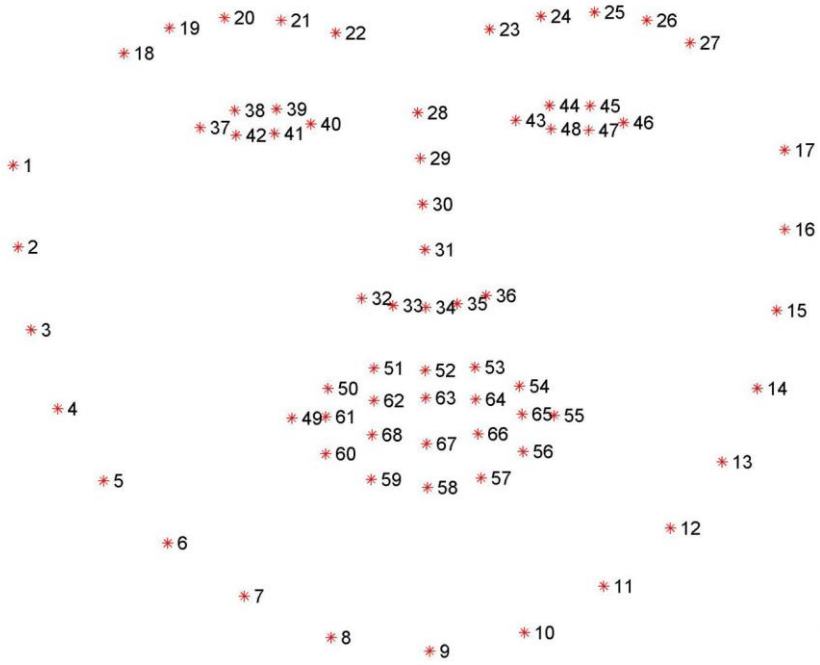
- Lokalizácia tváre na obrázku
- Nájdenie štruktúry tváre.

Na lokalizáciu tváre sa dá využiť viacero spôsobov, napríklad funkcie knižnice OpenCV. Pomocou týchto metód získame ohraničený rám tváre, teda súradnice X a Y lokalizovanej tváre na obrázku. [2]

Následne na nájdenej tvári sa algoritmus snaží lokalizovať a označiť nasledujúce oblasti tváre:

- Ústa
- Pravé obočie
- Ľavé obočie
- Pravé oko
- Ľavé oko
- Nos
- Čeľusť

Na tento účel sa využíva dopredu trénovaný detektor orientačných bodov knižnice dlib. Detektor používa 68 súradníc, ktoré sa mapujú na štruktúru tváre. Poloha všetkých 68 súradníc sa nachádzajú na obr. 3. [2]



Obr. 3: Orientačné body tváre, prevzaté z [2].

1.6 Kvaterióny

V unity sa kvaterióny používajú na reprezentáciu rotácie. Sú kompaktné a dajú sa ľahko interpretovať. Unity interne používa kvaterióny na reprezentáciu všetkých rotácií. Sú založené na komplexných číslach. Jednotlivým komponenty kvaterniónu (x, y, z, w) sa takmer nikdy priamo neupravujú. Väčšinou sa na vytvorenie novej rotácie používajú už dopredu definované rotácie v Unity. Používajú sa napríklad na otočenie jednej rotácie za druhou alebo na otočenie vektora o rotáciu. Najčastejšie využívané funkcie kvaterióna sú:

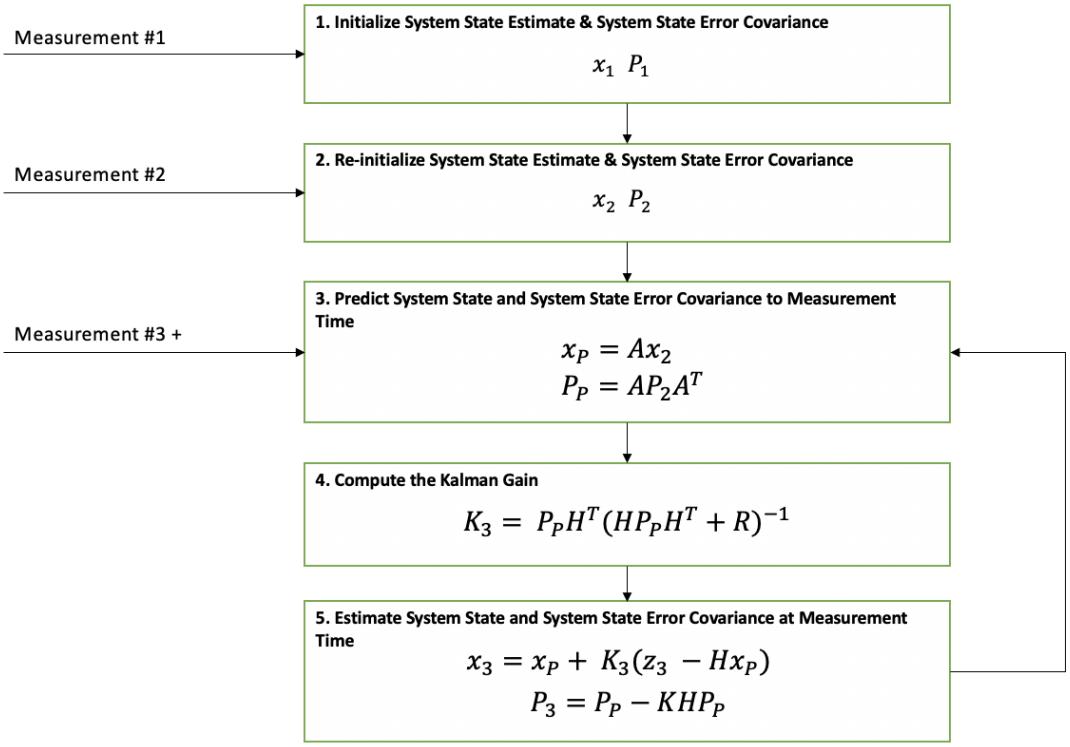
- Quaternion.LookRotation – vytvorí rotáciu s určitým smerom dopredu a nahor.
- Quaternion.Angle – vráti uhol v stupňoch medzi dvoma otočeniami *a* a *b.*
- Quaternion.Euler – vráti rotáciu, ktorá otočí vektor o zvolený počet stupňov okolo osi *z*, osi *x* a osi *y*.
- Quaternion.Slerp – sféricky interpoluje medzi kvaterniónmi *a* a *b* pomerom *t*. Parameter *t* je v rozsahu od 0 do 1.
- Quaternion.FromToRotation – vytvorí rotáciu, ktorá sa otáča od *fromdirection* do *toDirection.*

- Quaternion.identity – vráti identitu rotácie. [5]

1.7 Kalmanov filter

Je to algoritmus používaný na odhadnutie systémových parametrov. Z nepresných alebo šumivých meraní odhadne stav tejto premennej alebo inej nerozpoznateľnej premennej s väčšou presnosťou. Používa sa napríklad na sledovanie objektu. Z nameranej polohy objektu dokáže presnejšie odhadnúť polohu a rýchlosť tohto objektu. [7]

Na obr. 4 sa nachádza diagram, ktorý ukazuje ako funguje Kalmanov filter. Premenné z diagramu sú vysvetlené pod obrázkom.



Obr. 4: Diagram Kalmanovho filtra, prevzaté z [7].

Vysvetlenie premenných diagramu:

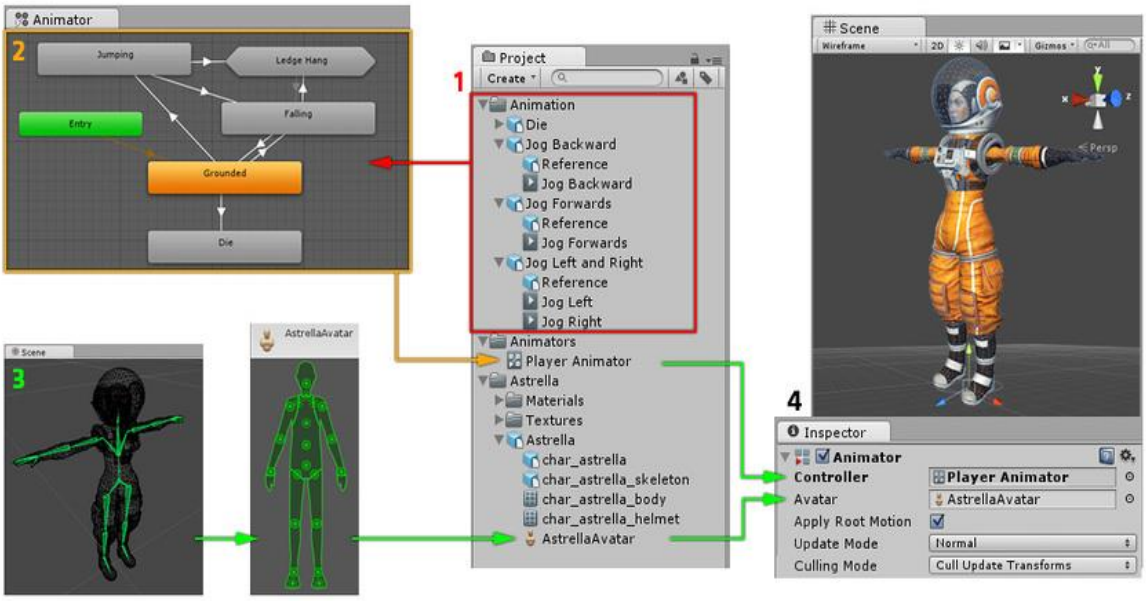
- x – stavová premenná
- P – stavová kovariačná matica

- z – meranie
- A – matica prechodného stavu
- H – matica stavu od merania
- R – meranie kovariačnej matice
- Q – kovariačná matica šumu
- K – zisk Kalmanovho filtra [7]

1.8 Animácie v Unity

Animačný systém v Unity je založený na koncepte Animovaných klipov, ktoré obsahujú informáciu o tom, ako by mal objekt v určitom časovom rozsahu meniť svoju polohu, rotáciu a podobne. Každý klip si je možno predstaviť ako jeden lineárny záznam. Animované klipy sú usporiadané do štruktúrovaného systému nazývaného Animačný kontrolér. Ten určuje, ktorý klip by sa mal práve prehrávať a kedy by sa mali animácie meniť alebo spájať. Jednoduchý Animačný kontrolér môže obsahovať iba jeden alebo 2 klipy, napríklad na pohyb alebo skákanie postavy. Zložitejšie Animačné kontroléri obsahujú desiatky humanoidných klipov na ovládanie všetkých akcií postavy (Avatara) a prelína jednotlivé klipy medzi sebou aby docielil plynulý pohyb postavy. [6]

Na obr. 5 sa nachádza diagram, ktorý poukazuje na komponenty animácií v Unity.



Obr. 5: Orientačné body tváre, prevzaté z [6].

1. Animované klipy, ktoré môže byť importované z externého zdroja alebo vytvorené v Unity.
2. Animačný kontrolér v ktorom sú umiestnené a usporiadané klipy. Jednotlivé klipy sú prepojené čiarami, ktoré predstavujú animáciu na prepojenie jednotlivých klipov.
3. Model postavy so špecifickou konfiguráciou kostí, ktoré sú nemapované na na bežný formát Unity. Každým vyznačený bod dokážeme presúvať alebo rotovať.
4. Animátor, ktorý je pripojený k animovanému modelu. Má priradený model avatara a Animačný kontrolér. Animátor riadi animáciu. [6]

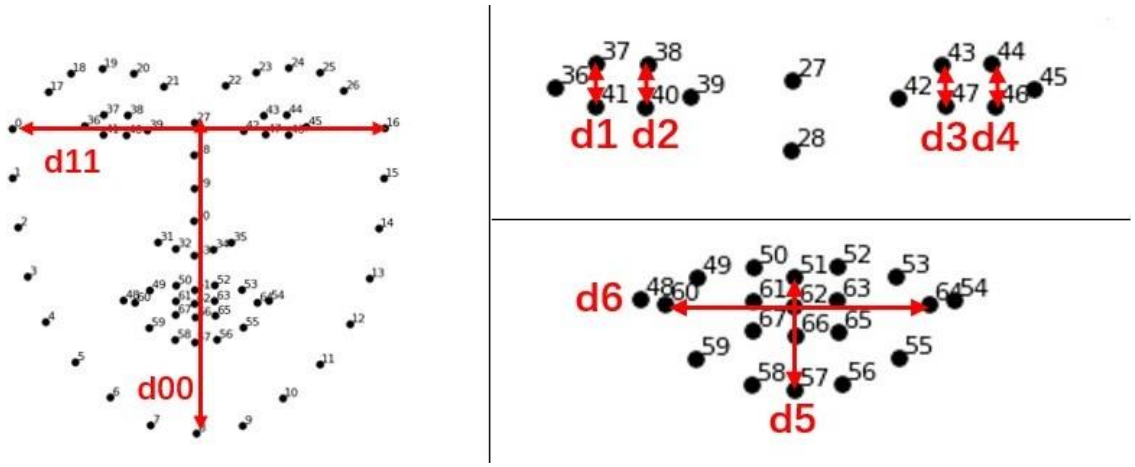
1.9 Existujúce riešenie

V tejto podkapitole sa nachádza opis existujúceho projektu, na ktorý moja bakalárska práca nadväzuje. Ide o snímanie pohybu hlavy založeného na vízii pre VTubers. Pre autentickejší výzor animácie sa používa niekoľko filtrov, metód na vyhladenie. Projekt je implementovaný hlavne v jazyku C# a Python v prostredí Unity3D. [4]

1.9.1 Front – end (Python)

Najprv sa pomocou knižnice Dlib a OpenCV získajú 68 orientačných bodov tváre pre odhadnutie polohy hlavy, orientácie a výrazy tváre. Na získanie presnejších dát sa používa Kalmanov filter, čo znamená, že dáta získané z predchádzajúcich snímkov majú väčšiu váhu pri určovaní aktuálnej polohy hlavy, ako dáta z aktuálneho snímku. [4]

Na extrakciu výrazu tváre z orientačných bodov tváre existuje viacero metód. Jedna z najpoužívanejších metód na detekciu žmurkania sa nazýva Eye-Aspect-Ratio. Avšak pri naklonení hlavy sa prejavujú nechcené odchýlky. Preto autor projektu vytvoril lepšiu metódu merania otvorenosti oči a úst. Na obr. 6 a obr. 7 môžeme vidieť výpočet týchto výrazov tváre. [4]



Obr. 6: Ilustrácia výpočtu výrazu tváre, prevzaté z [4].

$$d_{ref} = \frac{d_{00} + d_{11}}{2} = \frac{\|P_{27} - P_8\| + \|P_0 - P_{16}\|}{2}$$
$$d_1 = \|P_{37} - P_{41}\|$$
$$\vdots$$
$$d_4 = \|P_{44} - P_{46}\|$$

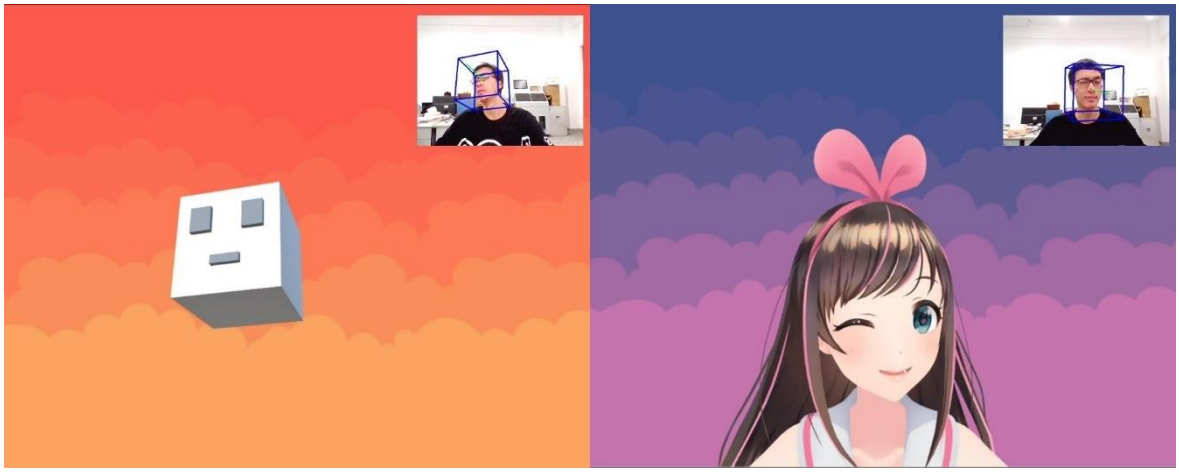
$$leftEyeWid = 6 \times (\frac{d_1 + d_2}{2d_{ref}} - 0.02)$$
$$rightEyeWid = 6 \times (\frac{d_3 + d_4}{2d_{ref}} - 0.02)$$
$$mouthWid = 1.27 \times (\frac{d_5}{d_{ref}} - 0.13) + 0.02$$
$$mouthLen = \frac{d_6}{d_{ref}}$$

Obr. 7: Výpočet výrazu tváre, prevzaté z [4].

1.9.2 Back – end (Unity, C#)

Dáta sa po úspešnom získaní z Pythonu posielajú do Unity pomocou socketu, ktorý je opísaný v časti 1.6.3. V Unity sa ďalej vykonávajú určité kroky, aby sa pohyb hlavy používateľa preniesol na pohyb hlavy modelu.

V projekte sa nachádzajú dva modely hláv, ktoré sa dajú použiť na premietanie pohybu. Na obr. 8 sú znázornené obidva modely. Naľavo sa nachádza jednoduchší model predstavujúci hlavu ako kocku s obdĺžnikmi miesto úst a očí. Napravo je komplexnejší model ľudskej hlavy.



Obr. 8: Modely implementované v projekte, prevzaté z [4].

Na model číslo jedna by sa teoreticky dal uplatniť vektor hlavy a kvaternión rotácie priamo. Keďže model napravo nemá implementovanú inverznú kinematiku, jeho poloha je fixná a preto môžeme ovládať iba jeho rotáciu. Navyše, nastavenie svetového súradnicového systému je v OpenCV odlišné od Unity, preto sa na kvaternióny aplikuje nová, fixná rotácia. Predtým ako sa vypočítané kvaterióny aplikujú na model hlavy, niektoré parametre sa znova odošlú do Kalmanovho filtra, aby sa zabezpečil hladký pohyb. [4].

1.9.3 Socket komunikácia

Komunikácia medzi front – endom a back – endom je zabezpečená pomocou Socketu. Unity C# server a Python klient sú nastavené na prenos údajov cez pripojenie TCP/IP. Koncový bod socketu je nastavený na:

- IP adresa: 127.0.0.1
- Port: 1755

2. Návrh

3. Implementácia

Záver

Použitá literatúra

[1] Gary Bradski and Adrian Kaehler. *Learning OpenCV*. 2008.

[2] Adrian Rosebrock. *Facial landmarks with dlib, OpenCV, and Python:* www.pyimagesearch.com/2017/04/03/facial-landmarks-dlib-opencv-python [Online, 5.1.2021].

[3] François Chollet. *Deep learning with Python.* 2018.

[4] Tianxing Wu. *OpenVHead: https://github.com/TianxingWu/OpenVHead* [Online, 31.1.2022].

[5] Unity Technologies. *Unity Documentation:* https://docs.unity3d.com/ScriptReference/Quaternion.html [Online, 5.1.2022].

[6] Unity Technologies. *Animation System Overview:* https://docs.unity3d.com/Manual/AnimationOverview.html [Online, 31.1.2022].

[7] William Franklin. *Kalman Filter Explained Simply:* https://thekalmanfilter.com/kalman-filter-explained-simply [Online, 31.1.2022].