TECHNICKÁ UNIVERZITA V KOŠICIACH
FAKULTA ELEKTROTECHNIKY A INFORMATIKY
Katedra počítačov a informatiky

SPRACOVANIE POSUNKOVEJ REČI

Diplomová práca

2021 **Bc. Miriama Mattová**

TECHNICKÁ UNIVERZITA V KOŠICIACH
FAKULTA ELEKTROTECHNIKY A INFORMATIKY
Katedra počítačov a informatiky

SPRACOVANIE POSUNKOVEJ REČI

Diplomová práca

Študijný program:	Informatika
Študijný odbor:	9.2.1. Informatika
Školiace pracovisko:	Katedra počítačov a informatiky (KPI)
Školiteľ:	doc. Ing. Branislav Sobota PhD.
Konzultant:	doc. Ing. Branislav Sobota PhD

2021 Košice **Bc. Miriama Mattová**

Abstrakt v SJ

Diplomová práca je zameraná na systémy spracovania posunkovej reči, implementované pre mobilné a desktop riešenia. Práca obsahuje analýzu možných technológií spracovania posunkovej reči s vyhodnotením. Návrh pozostáva z dvoch systémov. Systém prekladu posunku z kamery zariadenia a systém prekladu textu do posunku má v návrhu detailný opis architektúry a komunikačného procesu pre oba systémy. V práci sa implementujú navrhnuté systémy a algoritmy, opisuje sa zber vstupov a korigovanie výstupov. V overení sa vyhodnocujú výsledky implementovaných systémov na základe grafov úspešnosti alebo záťažových testov pre implementované algoritmy.

Kľúčové slova v SJ

Posunková reč, Slovenský jazyk, Analýza, Rozoznanie objektu, Animácie, Gestá, Prekladač

Abstrakt v AJ

Diploma thesis is focused on processing sign language systems, implemented for mobile and desktop solutions. Thesis contains analysis of possible technologies for processing sign language with an evaluation. Design consist of two systems. System of sign language translation via device camera and system of text translation into sign langauge has detailed description of architecture and communication process for both systems in design chapter. In thesis are implemented these designed systems and algorithms, described the collection of inputs and outputs correction. In verification are results of implemented systems evaluated via success chart or via load tests.

Kľúčové slova v AJ

Sign language, Slovak language, Analysis, Object recognition, Animations, Gestures, Translator

60723

TECHNICKÁ UNIVERZITA V KOŠICIACH

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Katedra počítačov a informatiky

ZADANIE DIPLOMOVEJ PRÁCE

Študijný odbor: **Informatika**
Študijný program: **Informatika**

Názov práce:

Spracovanie posunkovej reči

Sign language processing

Študent: **Bc. Miriama Mattová**
Školiteľ: **doc. Ing. Branislav Sobota, PhD.**
Školiace pracovisko: **Katedra počítačov a informatiky**
Konzultant práce:
Pracovisko konzultanta:

Pokyny na vypracovanie diplomovej práce:

1.Naštudovať problematiku počítačovej grafiky, virtuálnej reality, modelovania a priamej a inverznej kinematiky v spojitosti s najmä humanoidnými kinematickými štruktúrami a naštudovať základy posunkovej reči.
2.Analyzovať spracovanie posunkovej reči obojsmerne ako z pohľadu rozpoznávania tak z pohľadu interpretácie posunkovej reči.
3.Navrhnúť a vytvoriť model tvorby a riadenia kinematickej štruktúry vrátane príslušnej sústavy transformácií za účelom schopnosti interpretovať posunkovú reč.
4.Implementovať systém navrhnutý v bode 2 vrátane vizualizácie modelu s prezentáciou posunkovej reči a jej riadenia pomocou definovaných skriptovacích alebo programových prostriedkov.
5.Experimentálne overiť a zhodnotiť príslušné programové vybavenie na príkladoch, vrátane vytvorenia základného modelu humanoidnej postavy alebo jej potrebnej časti a základnej sady spracovávaných posunkov.
6.Vypracovať dokumentáciu podľa pokynov vedúceho práce.

Jazyk, v ktorom sa práca vypracuje: slovenský
Termín pre odovzdanie práce: 23.04.2021
Dátum zadania diplomovej práce: 30.10.2020

TECHNICKÁ UNIVERZITA V KOŠICIACH FAKULTA ELEKTROTECHNIKY A INFORMATIKY -12-

prof. Ing. Liberios Vokorokos, PhD.
dekan fakulty

Čestné vyhlásenie

Vyhlasujem, že som celú diplomovú prácu vypracoval/a samostatne s použitím uvedenej odbornej literatúry.

Košice, 22. apríla 2021 .....

vlastnoručný podpis

Poďakovanie

Moje poďakovanie patrí predovšetkým vedúcemu a konzultantovi práce doc. Ing. Sobotovi PhD., za skvelé odborné pripomienky a úžasnú schopnosť motivovať.

Ďakujem svojej rodine za vytvorenie skvelého prostredia pre písanie tejto práce, za ich podporu a ochotu počúvať zanietené vysvetľovania problematiky aj v prípade, keď som sama nevedela o čom rozprávam.

Obsah

Zoznam obrázkov

Zoznam diagramov

Zoznam tabuliek

Zoznam symbolov a skratiek

VR,MR, SDK, open CV, ODA, GPU, CPU

Motivácia

Je možné pozorovať, že technológia nabrala rapídne tempo za posledné storočie. Po zdravotnej stránke sa zlepšila kvalita životov, po transportnej stránke sa ľudia dokážu dostať z bodu A do bodu B v pomerne oveľa kratšom čase, po sociálnej stránke sa jedinec môže vďaka technológiám dozvedieť čo mal na obed iný jedinec, z iného kontinentu v reálnom čase. Technológia sa aj naďalej rapídne transformuje a vyvíja. Sociálne technológie napredujú rovnakým trendom, pretože človek je spoločenská bytosť. Človek má preto potrebu sa prejaviť, zapadnúť, inými slovami socializovať. Najdôležitejším krokom socializovania sa je komunikácia. Ľudia komunikujú na dennej báze či už s rodinou, priateľmi alebo domácim maznáčikom. Komunikácia sa prejavuje verbálne ale aj neverbálne, vo všetkých jazykoch sveta. Avšak, žiaľ existuje skupina ľudí, ktorí sú odkázaný jedine na neverbálnu komunikáciu, gestikuláciu pomocou posunkovej reči.

Táto diplomová práca sa preto zameriava na vývoj systému spracovania posunkovej reči. Jedna časť spracovania je zameraná na preklad posunku od posunkujúceho pomocou kamery počítačového alebo mobilného zariadenia na základe analýzy obrazu. Analýza obrazu prebieha implementáciou natrénovaných modelov niekoľkých slov, abecedy a písmen. Druhá časť spracovania je zameraná na vytvorenie humanoidnej postavy – avatara, ktorý na základe vstupu od používateľa, čiže textu, ho preloží tak, aby vytvorený humanoidný avatar gestikuloval sémantický ekvivalent napísaného textu. Táto práca je implementovaná tak, aby bola prístupná z webového prehliadača, nakoľko si užívateľ nemusí sťahovať žiadne aplikácie do telefónu alebo počítača. Tým pádom sa to stáva multiplatformovým riešením a teda by malo byť prístupne pre každého kto má telefón, počítač a internetové pripojenie.

1. Formulácia úlohy a cieľ práce

Prvou úlohou v rámci diplomovej práce bude naštudovanie problematiky počítačovej grafiky, virtuálnej reality, modelovania a priamej a inverznej kinematiky v spojitosti s najmä humanoidnými kinematickými štruktúrami a naštudovať základy posunkovej reči.

Druhá úloha pozostáva z vypracovania analýzy posunkovej reči obojsmerne, ako z pohľadu rozpoznávania, tak z pohľadu interpretácie posunkovej reči.

Nasledovne je potrebné navrhnúť a vytvoriť model tvorby a riadenia kinematickej štruktúry vrátane príslušnej sústavy transformácií za účelom schopnosti interpretovať posunkovú reč.
Po návrhu je potrebné implementovať navrhnutý systém, vrátane vizualizácie modelu s prezentáciou posunkovej reči a jej riadenia pomocou definovaných skriptovacích alebo programových prostriedkov.

Na základe navrhnutého a implementovaného systému bude potrebné experimentálne overiť a zhodnotiť príslušné programové vybavenie na príkladoch, vrátane vytvorenia základného modelu humanoidnej postavy alebo jej potrebnej časti a základnej sady vypracovávaných posunkov.

2. Analýza stavu problematiky

V súčasnosti žijeme v dobe, kde sa technológia vyvíja rýchlym tempom. Pomáha nám uľahčovať život, či už z hľadiska relaxu, komunikácie, medicíny, práce a podobne. Vzhľadom na vyspelú technológiu, je oblasť hluchonemých takmer nepokrytá. Už existujú technológie, ktoré dokážu mať skvelý internetový prenos, rýchle kalkulácie a rozpoznávania. Avšak predpokladá sa, že človek, ktorý je hluchý, nemý alebo kombinácia si v bežnom živote nezakúpi superpočítač a najnovšiu generáciu internetového prenosu. Táto kapitola sa preto zameria na bežne prístupne technológie, pomocou ktorých bude alebo nebude možne vytvoriť systém, ktorý dokáže čítať posunky za bežných podmienok.

2.1. Analýza posunkovej reči

Posunková reč nie je všeobecná pre každú krajinu, čo znamená, že každá krajina má svoj dialekt. Táto diplomová práca sa teda zaoberá Slovenským dialektom. Posunková reč ako taká, ma podobu pohybov jednej alebo dvoch rúk, sprevádzaním výrazom tváre. [1]

Posunky môžeme deliť podľa výrazov na:

- znakové výrazy,
- slovné výrazy,

a podľa počtu aktívnych rúk teda:

- jednoručné
- dvojručné symetrické
- dvojručné asymetrické

Do kategórie aktívnych komponentov, pri zložitejších posunkov, spadá aj mimika alebo postoj. Pre tieto komponenty by bolo potrebné vytvoriť systém na rozoznanie mimiky tváre ako aj postoja.

Oficiálna stránka Posunky.sk opisuje gramatiku slovenských posunkov ako aj oficiálne posunky a hovorové posunky. Opisuje aj ktoré výrazy majú rôznu posunkovú podobu. V takom prípade nastáva problematika, či pre analýzu zvoliť všetky synonymá alebo ako určiť, ktorý posunok je najpoužívanejší. Ďalší problém môže nastať pri posunkoch ktoré sú veľmi podobné a odlišujú sa napríklad len mimikou alebo pohybom. V prípade znakových výrazov ide zväčša o statické držanie rúk a preto sa predpokladá, že posudzovanie týchto dát je oveľa jednoduchšie ako posunky pri ktorých je potrebné zapojiť pohyb aj mimiku. [2]

2.2. Možné zariadenia pre vykonávanie rozpoznávania jednotlivých gest

Pre rozpoznanie gest je možne použiť viacero technologických zariadení. Nasledovná analýza teda opisuje výhody a nevýhody niekoľkých vybraných možných zariadení, ktoré sú schopné riešiť túto problematiku.

2.2.1. Oculus Quest, Oculus Quest 2

Dátové prilby v oblasti virtuálnej reality ako oculus quest a ocululs quest 2 majú vbudovanú technológiu rozoznávania rúk [3] , preto sa daju považovať za jedno z možných riešení.

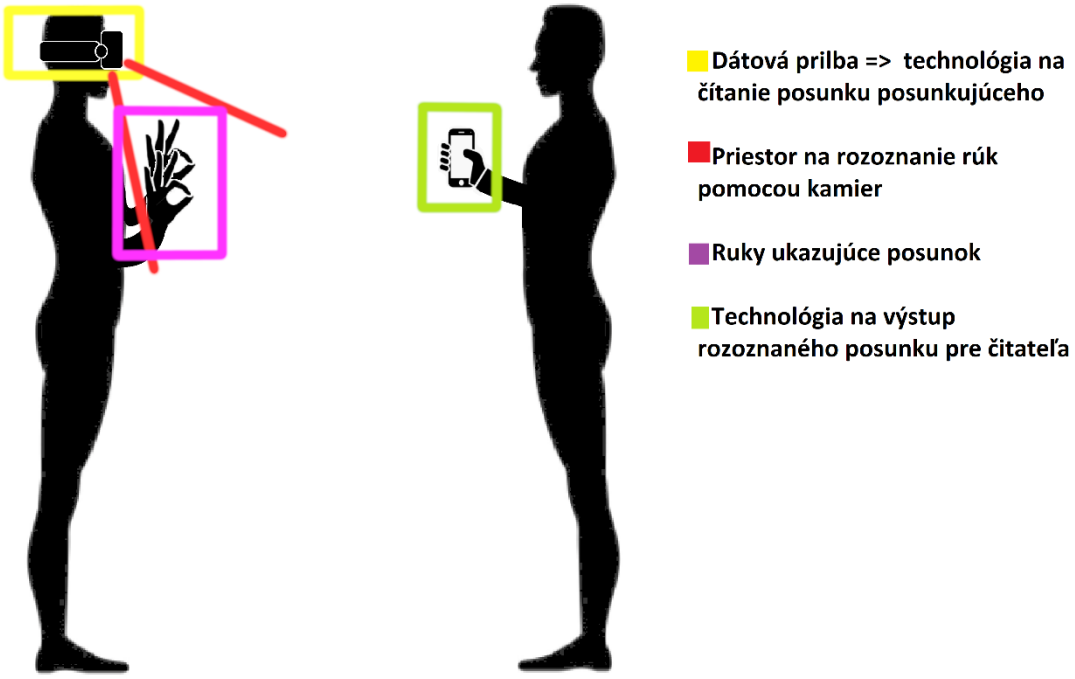
Výhody:

- Priamo implementované rozoznávanie rúk, čím sa rieši problém spracovania vstupu pre ruky,
- existujúce SDK pre prácu s konkrétnymi gestami. [4]

Nevýhody:

- Technológia je v súčasnosti pomerne drahá a nie bežne prístupná,
- posunkujúci nevidí okolitý svet. Ak si aj nastaví pozorovanie cez kamery, nebudí to prirodzený dojem,
- posunkujúci musí mať nasadenú helmu, čo je pomerne robustné riešenie,
- človek čítajúci posunok musí mať dodatočný systém, preposielaný z helmy posunkujúceho, aby videl výstup, čiže rozpoznanú formu daných posunkov.

Obrázok 1 Dátové prilby Samsung Gear VR, Oculus Quest, Oculus Go a ich ovládače



Obrázok 2 Názorný príklad použitia technológie čítania pomocou dátovej prilby

Na základe obrázku je možné tvrdiť, že praktickosť takého to riešenie je minimálna keďže akákoľvek dátová prilba obmedzuje zrakový zmysel posunkujúceho.

2.2.2. Hololens

Táto technológia je tak tiež dátová prilba. Technologický rozdiel od technológie popísanej v kapitole 2.2.1 je ten, že funguje na princípe zmiešanej reality, ďalej len MR, [5] preto sa výhody a nevýhody posudzujú samostatne

Výhody:

- Podobne ako Oculus Quest a Oculus Quest 2 (kapitola 2.2.1.) má priamo implementované rozoznávanie rúk, čím sa rieši problém spracovania vstupu pre ruky,
- existujúce SDK pre prácu s konkrétnymi gestami, [6]
- problematika vizualizácie okolia sa rieši tým, že je to MR systém, čiže človek vidí priestor vôkol seba v reálnom čase vlastným okom, tým pádom to nebudí dojem nereálnosti.

Nevýhody:

- Keďže ide o MR dátovú prilbu, ktorej zameranie je hlavne na vkladanie objektov do reálneho prostredia, má toto riešenie nadbytočné komponenty, ktoré nie sú potrebné pre túto problematiku. Tým pádom je to aj cenovo nevýhodné,
- v súčasnosti sú tieto helmy ťažko dostupné a predávajú sa primárne univerzitám a iným pracoviskám na výskumne účely,
- robustné riešenie z hľadiska toho, že posunkujúci musí mať prilbu na hlave a človek, čítajúci posunok, musí mať dodatočný systém na reprezentáciu preloženého posunku.

Keďže sa jedná o dátovú prilbu, spôsob rozpoznávania a prezentácia posunkov je podobná ako na obrázku 2.

2.2.3. Myo Armband

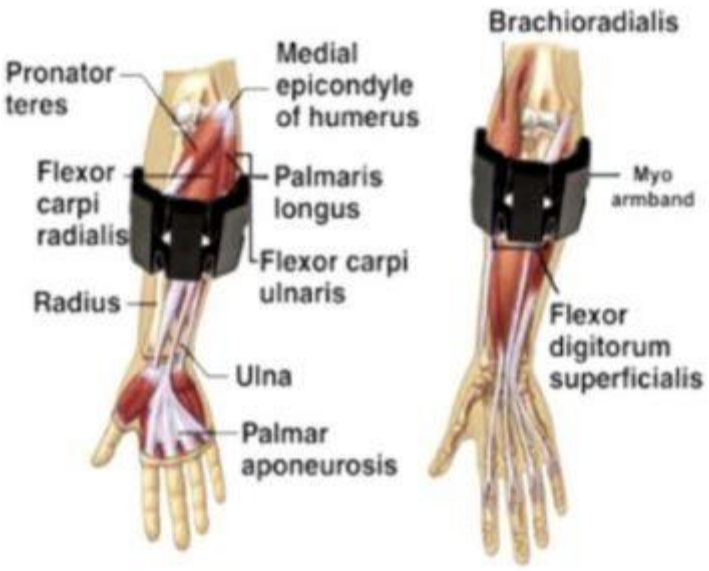
Zariadenie slúži ako náramok, ktorý sa nasadí pod lakeť a monitoruje svalstvo na ruke.

Výhody:

- Zariadenie nie je závisle od kamerového vstupu ale sníma svalové ústrojenstvo ruky,
- nie je potrebný gél ani holenie rúk,
- SDK zariadenia sa zameriava priamo na gestikuláciu a nemá nadbytočné komponenty, [7]
- je to praktické zariadenie keďže neobmedzuje pohyb ani vizualizáciu prostredia
- presnosť gest je na vysokej úrovni, keďže sa sníma svalstvo ruky.

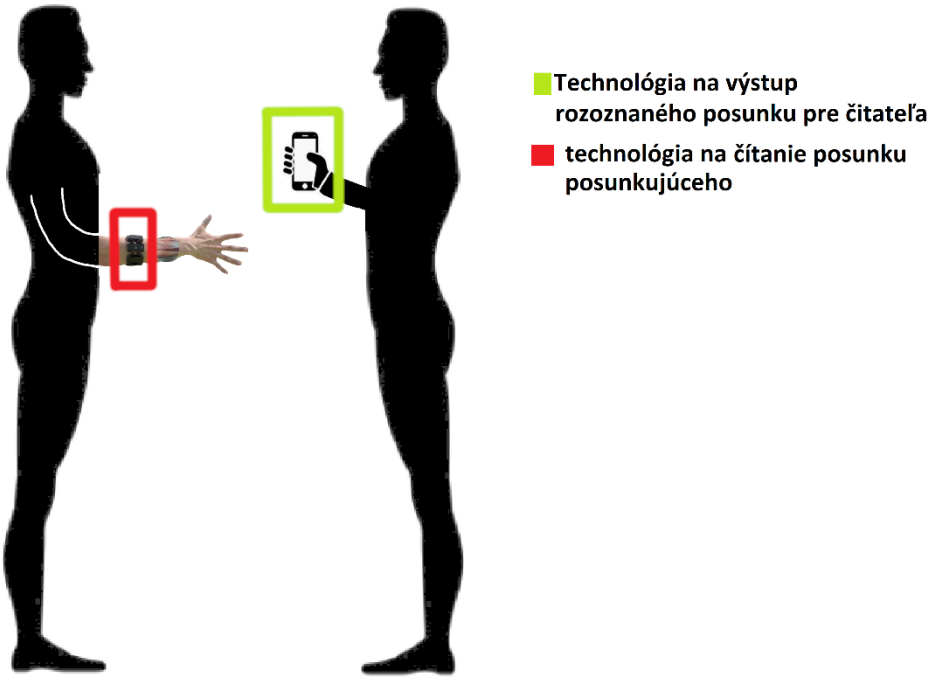
Nevýhody:

- Aktuálne je ťažko dostupný,
- posunkujúci si musí dokúpiť tento komponent, ktorý je momentálne cenovo pomerne vysoko,
- je potrebný dodatočnú technológiu na preklad posunku,
- Obmedzenie rozoznávania len ručných posunkov



Obrázok 3 Myo Armband a jeho umiestnenie na ruke [8]

Obrázok vyššie poukazuje na umiestnenie náramku na ruke tak tiež ako schopnosť získavať impulzy od menovaných svalov.



Obrázok 4 Názorný príklad použitia technológie čítania pomocou Myo Armband

Implementácia takéhoto riešenia neobmedzuje zmysli posunkujúceho a ani obmedzenie priestoru, kde má daný posunok ukazovať.

2.2.4. Mobilné alebo počítačové zariadenie

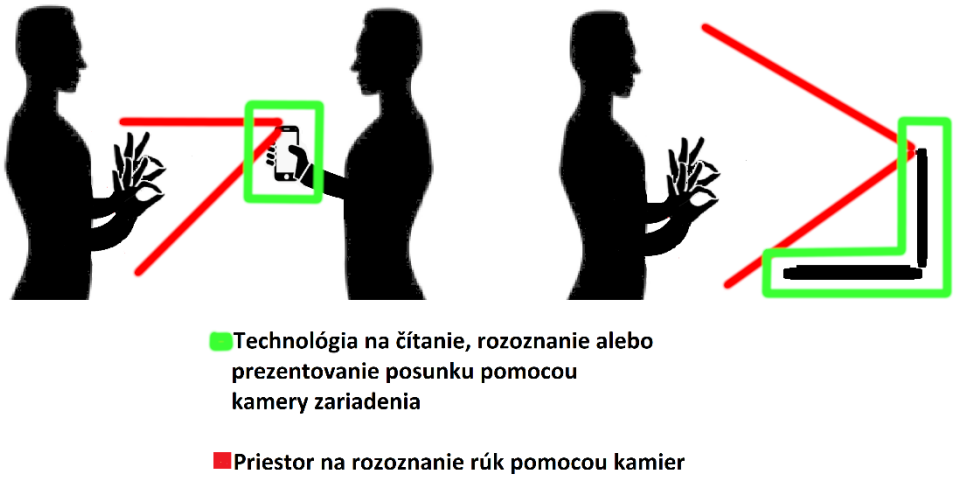
Tieto zariadenia sú závislé od kamerového vstupu posunkov. Výhody a nevýhody sú posudzované na všetkých mobilných a počítačových zariadeniach, ktoré disponujú kamerou a internetovým pripojením.

Výhody:

- Dostupné takmer pre každého,
- predpokladá sa, že skoro každý človek v dnešnej dobe disponuje telefonickým zariadením s kamerou a internetom, preto nie je potrebné dokupovať žiadne zariadenie, čo robí takéto riešenie úplne bezplatné,
- existuje mnoho open CV riešení pre rozoznanie obrazu a rozoznávanie gest,
- nie je potrebný dodatočný systém pre človeka čítajúci posunok, keďže výstup sa mu zobrazí priamo na zariadení
- posunkujúci nie je nijako obmedzený doplňujúcim zariadením pre čítanie posunku.

Nevýhody:

- Takýto systém potrebuje mať zariadenie s vysokým pripojením na internet, prípadne silné kalkulačné jadro,
- ak by šlo o systém, ktorý sa deje niekde na externom serveri a pripojenie by bolo cez mobilné dáta, je to veľký objem prenosu dát a je potrebné vysoké a stabilne pripojenie,
- rozoznávanie obrazu je závisle od okolitých faktorov,
- obraz sa rozpoznáva na základe 2D vstupu, čo tak tiež znižuje presnosť rozpoznania gesta



Obrázok 5 Názorný príklad použitia technológie čítania pomocou kamier mobilných či počítačových zariadení

2.2.5. Smart rukavice so senzorom monitorovania polohy a rotácie ruky

Takéto riešenie vyžaduje vlastný zostavený systém z existujúcich senzorov. Smart rukavica by teda obsahovala senzory pre určenie rotácie prsta a dlane.

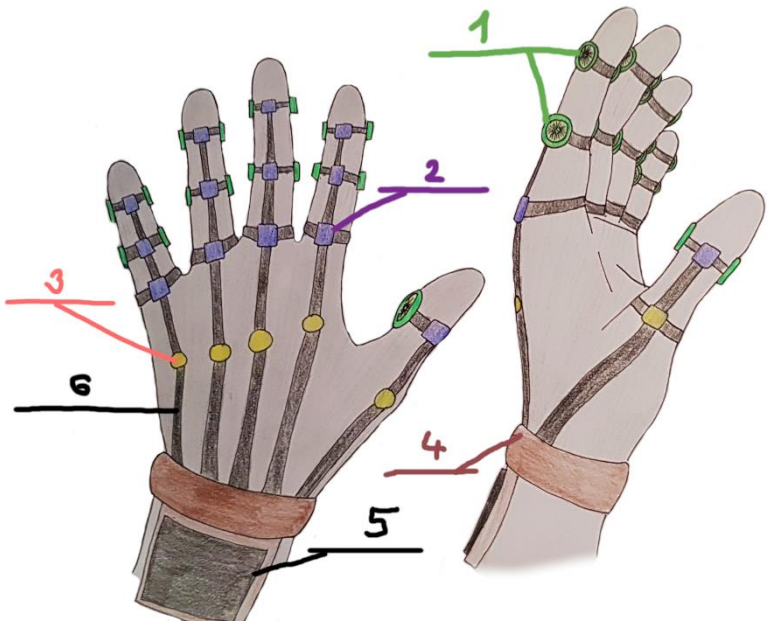
Výhody:

- Zariadenie nie je závislé od kamerového vstupu,,
- nijak neobmedzuje posunkujúceho,
- systém nemá nadbytočné komponenty a sústredí sa priamo na gestikuláciu ruky.

Nevýhody:

- Systém treba zostaviť,
- je potrebné implementovať vlastné SDK,
- je potrebný dodatočný systém pre výstup posunkov
- posunkujúci si musí toto zariadenie dokúpiť,
- obmedzenie posunkov len na zápästnú časť rúk.

Nasledujúci obrázok znázorňuje jeden z možných návrhov smart rukavice.



Obrázok 6 Kreslený prototyp smart rukavice

1- Pomocné rotačné kolieska na určenie uhla rotácie prsta

2- Napínavé senzory určujúce zohnutie alebo prehnutie prsta

3- Pomocné vodivé komponenty

4- Upevňovač rukavice a puzdro pre organizáciu káblov

5- Potrebný hardware na preklad posunku (Matičná doska, wifi, display a pod.)

6- Dátové prenášače s pevnou konštrukciou na hornej časti prsta.

2.3. Programovacie jazyky

Podobne ako zariadenia, programovacie jazyky majú svoje výhody a nevýhody pre implementáciu tejto problematiky. Programovacie jazyky sú závislé od konkrétneho zariadenia. Preto sa táto kapitola zameriava na jazykovú implementáciu v zariadeniach zanalyzovaných v kapitole 2.2.

2.3.1. Oculus Quest, Unity, C#

Pokiaľ sa zvažuje vývoj v dátových prilbách ako je napríklad oculus, unity sa preukázalo ako dobré vývojové prostredie, vzhľadom na podporu pre Oculus, Windows mixed reality, Google vr a OpenVr. [9] [10]

Platformy vymenované vyššie, majú pripravené vlastné balíčky, ktoré je možné importovať do prostredia. Zväčša obsahujú základnú funkcionalitu a podporu pre avatara a ovládače. Tým pádom programátor nemusí skriptovať vlastnú komunikáciu a mapovanie kamery s helmou alebo mapovanie ovládača a tak sa predchádza zdĺhavému programovaniu a chybám v kóde.

Balíčky na podporu heliem sú primárne programované v jazyku C#. Avšak unity poskytuje možnosť výberu vlastného skriptovacieho jazyka podľa preferencie.

Výhody jazyka C# sú tie, že je to objektovo orientovaný jazyk, má automatický „garbage-collection“, je to multiplatformový jazyk a má spätnú kompatibilitu. Jazyk sa nevyužíva len na vývoj pre VR ale tak tiež sa dá použiť napríklad aj ako jazyk na backend podporu pre webové aplikácie. [11] Pri zvažovaní riešenia tejto problematiky prostredníctvom dátovej helmy, je možné považovať unity a C# ako vhodné riešenie.

Nevýhoda implementácie tohto riešenia pre túto problematiku je tá, že účelom tejto problematiky nie je práca s objektami, jeho skriptovaním a vývojom virtuálneho sveta, ale preklad posunkovej reči do textového výstupu a textového vstupu do posunkovej reči.

2.3.2. Lua, javascript – myo.js

Lua, je programovací jazyk, ktorý sa často využíva v hernom priemysle. [12] Keďže jedno z možných riešení tejto problematiky je Myo Armband, technológia popísana v kapitole 2.2.3. , jazyk Lua podporuje skriptovanie tejto technológie. Náramok oficiálne prezentuje jazyk Lua ako odporúčaný jazyk pre vývoj, hlavne na základe toho, že skriptá vytvorené pre Myo, sú písane práve v tomto jazyku. [13] Jedna z alternatív môže byť aj javascript. Existuje knižnica s názvom myo.js, [14] ktorá ma implementovanú základnú funkcionalitu na narábanie s komponentami tejto technológie ako je napríklad pripojenie a odchytávanie dát so senzorov. Jazyk javascript je všeobecné známy tým, že je možne v ňom urobiť takmer všetko, od webových služieb, serverových pripojení až po ovládanie animácií a správanie sa iných objektov. Ak sa zvažuje riešenie problematiky pomocou Myo

Armband, je možné tvrdiť, že výber jazyku závisí od preferencie programátora, či už na základe naštudovaných možných akcií v daných jazykoch a ich pripravených balíčkoch alebo skúseností s daným jazykom

2.3.3. Možnosti jazykovej implementácie pre mobilné a počítačové zariadenia

Problematika riešená pomocou mobilných, či počítačových zariadení ponúka pestrý zoznam programovacích jazykov. Otázka je, na akej platforme zariadenia by sa to implementovalo. Preto je potrebné zanalyzovať možnosti najpoužívanejších platforiem a jazykov.

Pre stand-alone mobilne aplikácie:

1. Pokiaľ by sa práca zamerala na Android zariadenia, mali by sa zvažovať jazyky java alebo kotlin.
2. Pre IOS zariadenia sú zase najpopulárnejšie jazyky swift alebo objective-c. [15]
3. Windows mobilne zariadenia ponúkajú možnosť C#, VB.NET, C/C++ a podobne.

Pre stand-alone počítačové aplikácie:

1. Ak ide o windows, z najpopularnejších jazykov sú znova C#, java, python.
2. Pre linux sú to C/C++, java, python.
3. Mac OS najznámejšie sú swift, ruby, je však možné využiť aj multiplatformové jazyky ako python, C/C++.

Pre webové aplikácie

Keď že v podstate netreba riešiť, či ide o mobilné alebo počítačové zariadenie, sú na výber rôzne webové jazyky ako Angular, Javascript, Typescript, HTML, CSS, PHP a tak ďalej. Týchto jazykov je v dnešnej dobe veľa.

Pri spätnom pohľade na možné riešenia pomocou mobilných a počítačových technológií, je možné tvrdiť, že sa dá použiť takmer každý programovací jazyk pre konkrétnu platformu na problematiku tejto práce. Preto je vhodné sa zamerať na konkrétne príkladné riešenie s čo najväčšou možnosťou univerzality. Ak je teda cieľom dosiahnuť, aby technológia na preklad posunku a preklad do posunku bola pokrytá na čo najviac platformách, je možné analyzovať napríklad jazyky ako sú Python, Java, Angular, Typescript a php.

Analýza frontend-ových riešení

Ako sa spomína na predchádzajúcej strane, možnosť zobrazovania požadovaného výstupu a možnosti vstupu vedia zabezpečiť mnohé webové jazyky. Pre logické umiestnenie komponentov na obrazovke pomocou webového klienta sa môže zvažovať jazyk HTML. V spolupráci s logickým chovaním a zobrazovaním je možné zvažovať aj framework Angular. Výhoda tohto frameworku spočíva hlavne v tom, že je možné dopisovať do HTML kódu napríklad podmienky alebo cykly. Tento framework spolupracuje s typescriptom, ktorý je nadstavba javascriptu. [16]

Výhody tejto alternatívy spočívajú v tom, že typescript obsahuje množstvo variant práce s elementami na obrazovke, je reaktívny, dá sa kombinovať s jazykom javascript a v dnešnej dobe obsahuje mnoho podporných modulov. Preto sa predpokladá, že pokiaľ by bolo potrebné vykonávať analýzu aj na klientovej strane, napríklad rozdeľovanie textu a priradenie správnej animácie na posunok, bolo by to výhodné riešenie.

Nevýhoda takéhoto riešenia je v základe taká, že aplikácie stavané pomocou angularu a typescriptu sú zväčša pre väčšie webové aplikácie, keď že pre spojazdnenie a fungovanie celého systému je potrebné mať implementované rôzne podporne knižnice a moduly, ktoré môžu byť kapacitne náročne viac ako keby to bolo riešené napríklad cez javascript prípadne PHP.

Analýza backend-ových riešení

Pre analýzu a spracovanie údajov, čiže posunku, existuje znova mnoho variant, čo sa týka jazykovej implementácie. Predpokladajme teda, že klient prepošle obraz zo svojej kamery. Úlohou backend-ového procesu je prekalkulovať aké gesto sa ukazuje na danom médiu. Ak by bolo potrebné jednoduché ukladanie dát do tabuliek, vytváranie tabuliek, entít alebo iné databázové a objektové operácie, mohli by sa zvažovať napríklad jazyky C# alebo Java. Táto problematika sa však zameriava aj na analýzu obrazu a z pravidla, komunita developerov formovali postupne jazyk python presne pre takéto účely. V jazyku java aj C# je možné aplikovať neurónové učenie, avšak jazyk Python sa zameriava hlavne na neurónové učenie. Veľkou výhodou tohto jazyka je aj množstvo existujúcich riešení na podobnú problematiku. Tieto verejne dostupné a pravidelne testované riešenia sa vedia aplikovať do tejto problematiky s minimálnymi úpravami pre prispôsobené cieľu tejto práce. Tým pádom sa eliminuje potreba vytvoriť vlastný systém na rozoznávanie 2d obrazu, ktorý sa dá považovať ako samostatná problematika.

2.3.4. Smart rukavice a možné programové implementácie

Existuje mnoho variant ako zostaviť smart rukavicu na čítanie posunkov. Komponenty na vyskladanie takejto rukavice následne určia možné programovacie jazyky. Táto analýza sa zaoberá na momentálne najpopulárnejšie matičné dosky ako sú napríklad arduino a raspberry pi. Senzory sú pomocné komponenty na čítanie posunku, no pre analýzu možného programovacieho jazyka stačí vedieť, matičnú dosku.

Arduino technológia má vlastné IDE, ktoré ma zadefinované vlastné dátové typy, funkcie a podobne. V podstate sa jedná o nadstavbu jazyka C. [17] V prípade zvolenia tejto alternatívy je možné použiť iba tento jazyk.

Raspberry pi sa odlišuje tým, že táto doska ma aj vlastný operačný systém, ďalej len OS. Celý tento systém je postavený v jazyku python. [18] Keď že ide o OS, je možné využiť akýkoľvek multi-platformový jazyk. Avšak ako sa spomína v kapitole 2.3.3. , jazyk Python sa primárne využíva a sústredí presne na túto problematiku a preto by bolo vhodne zvážiť, či je potrebné zaplniť priestor s podporou pre iný programovací jazyk

Obe tieto varianty sú však jednosmerná neúplná cesta, a to rozoznanie posunku. Pre prezentovanie posunku sa môže zvoliť alternatíva pripojenia napríklad malej obrazovky, ktorá bude ukazovať výstup, avšak takto by bolo obmedzenie jednosmernej cesty. Pokiaľ by sa chcel prekladať aj text do posunku, bolo by potrebné dodať ďalší komponent. Ďalšia alternatíva teda môže byť preposielanie rozoznaného posunku niekde na server. Znova môže existovať stand- alone aplikácia alebo webová aplikácia na komunikáciu so serverom, kde by neposunkujúci vedel zobrazovať preložený posunok a písať text, ktorý by sa následne preložil a zobrazil. Možnosti jazykov pre tento spôsob sú v princípe zanalyzované v kapitole 2.3.3.

2.4. Preklad textu do posunkovej formy

Ďalšia otázka sa vyskytuje v oblasti transformácie slovnej hodnoty do posunku. Tu vzniká viacero dilem, ako to dosiahnuť. Je potrebné sa zamyslieť, aká forma komunikácie od neposunkujúceho bude prekladaná, tak tiež ako sa požadovaná slovná hodnota preloží a následne zobrazí ako posunok. Keďže posunok ako taký je čisto vizuálna záležitosť, je potrebné mať zobrazovaciu plochu, na ktorej je možné daný posunok vizualizovať. Avšak spôsob komunikácie používateľa so zariadením nie je taký jednotný. Nasledovná podkapitola analyzuje možné spôsoby komunikácie používateľa so zariadením.

2.4.1. Spracovanie a rozoznanie slovnej hodnoty do textu

Človek komunikuje primárne cez hovorené slovo. Každé hovorené slovo sa vie transformovať do textu. Avšak v tejto technologickej dobe, kde každý človek využíva nejaké sociálne služby sa vytvoril nový spôsob komunikácie - emotikoni, gify, meme a podobne. Keďže ide o vizuálnu reprezentáciu komunikácie, tento spôsob netreba posudzovať. Vzniká viacero dilem, ako dosiahnuť slovný preklad do posunku

1. Priamy preklad písaného textu
2. Transformácia audia do textu
3. Iné alternatívy komunikácie do textu

Treba sa zamyslieť nad problémom ako rozdeliť text na jednotlivé slová, čo ak slová nie sú rozpoznateľne, ako ošetriť gramatickú bariéru medzi slovenskými posunkami a písaným textom, ako sú napríklad pomocné slovesá.

Priamy preklad písaného textu

Takéto riešenie pozostáva s čítania vstupu, teda textu od používateľa, jeho rozdelenia a rozoznania jednotlivých slov a znakov. To znamená, že ak používateľ zadá text, riešenie by malo vedieť rozdeliť tento text na vety, potom na samostatné slová. Pokiaľ navrhnutý systém nemá v slovníku dané slovo, prerozdelí ho na písmena.

Transformácia audia do textu

Takýto spôsob je doplnený, od obyčajného písania textu o rozoznanie audio stopy. To znamená, že používateľ by musel disponovať mimo klávesnice aj mikrofónom. V princípe by systém mal byť schopný zachytiť hovorené slovo používateľa, vedieť ho rozpoznať a preložiť do textu. Nasledovná práca s textom je už popísaná v predchádzajúcom odseku. Takýto spôsob je pohodlnejší z hľadiska rýchlejšieho vstupu. Používateľ tak môže jednoducho nadiktovať slová a systém by ich mal vedieť rozpoznať. Nevýhoda audio doplnku spočíva v tom, že systémy na rozpoznania slovenskej reči nie sú také dokonalé a môže dôjsť k chybám v preklade. Táto nevýhoda sa dá ošetriť tak, že text ešte pred posunutím na zobrazenie posunku by sa zobrazil používateľovi a ten by si ho vedel editovať, pokiaľ by došlo k zlému prekladu.

Iné alternatívy komunikácie do textu

Pod pojmom iných alternatívnych komunikácií sa dá chápať napríklad neverbálna komunikácia, meme, gif, emotikoni a podobne. Keďže jedným z cieľov tejto práce je zobraziť posunok pre nepočujúceho, predpokladá sa, že formy takejto komunikácie chápe aj mimo posunkového

prekladu. Ak by sa ale zvažoval preklad aj takejto formy, bolo by potrebné vymyslieť samostatné systémy na interpretáciu týchto alternatív a v konečnom dôsledku by šlo aj tak o grafický výstup, čo by bolo vlastne zbytočne robustné a redundantné riešenie.

2.4.2. Zobrazovanie posunku na základe spracovaného textu

V princípe, každý spôsob komunikácie je potrebné transformovať do nejakého textu, aby systém bol schopný pomocou neho posúdiť, ktoré slovo ma akú sémantickú hodnotu. Na základe sémantickej hodnoty sa potom vie priradiť posunok s rovnakou alebo aspoň podobnou sémantickou hodnotou. Prezentácia posunku je grafická záležitosť. Je možné ho zobrazovať napríklad reálnou postavou alebo 3D modelom - avatarom. Spomenuté spôsoby majú svoje výhody a nevýhody.

3D model

Pod pojmom 3D model sa dá predstaviť namodelovaný avatar postavy s kostrou. Kostra by mala slúžiť ako prvok pre animovanie konkrétnych posunkov.

Výhody:

- Ľahká udržateľnosť, to znamená, že pri pridávaní nových posunkov stačí len pridať animáciu,
- na rozdiel od videa reálneho človeka sa eliminuje potreba nastavovať rovnaké osvietenie, šaty, účes a podobne,
- animáciu je možné začať stále z rovnakej polohy rúk a tým pádom sa vie docieliť plynulý prechod medzi jednotlivými gestami,

Nevýhody:

- Animovaný objekt nemusí pôsobiť prirodzeným dojmom,
- zle spracovaná animácia môže viesť k zlej interpretácií posunku,
- programovo náročnejšie,
- treba si dávať pozor na kapacitu objektu.

Spomenuté nevýhody sa v dnešnej technológii už dajú riešiť pomocou cloud-ových služieb. Existuje tzv. model viewer technológia, ktorá dokáže zobraziť 3D objekt aj pomocou linku. [19] Tým pádom sa zamedzuje potreba stiahnutia modelu priamo do systému používateľa.

Je potrebné tak tiež zvážiť aký modelovací program je vhodný pre tento účel ale v princípe ide o preferenciu.

Reálna postava človeka

Táto možnosť grafického výstupu posunku znamená, že konkrétny posunok bude prezentovaný reálnym človekom nahraným cez kameru. Výsledne médium sa môže prezentovať rôznymi spôsobmi, napríklad video, gif alebo aj statická fotka.

Výhody:

- Nepôsobí neprirodzeným dojmom,
- reprezentácia je ľahko prezentovateľná cez couldové služby, tým pádom je to výhodne riešenie z hľadiska priestoru,
- jednoduchá programová implementácia,
- predchádza sa softvérovým chybám, ktoré by mohli nastať pri animovanom objekte.

Nevýhody:

- Ťažšie na udržiavanie,
- Väčší problém pri dosiahnutí plynulého prechodu medzi posunkami,
- Pri pridávaní nového posunku je potrebné nasimulovať osvetlenie, človeka a celkové prostredie na základe starých posunkov, čo môže byť obťažné až nemožné.

V konečnom dôsledku sú obe možnosti grafického zobrazenia posunku kapacitne výhodné na základe toho, že existujú metódy, ako nezaťažiť navrhovaný systém. Preto sa dá tvrdiť, že tento prvok v tejto problematike pri takomto prístupe nezohráva rolu. Je stále možné využiť obe varianty zobrazenia a ponechať rozhodnutie na preferencii používateľa.

2.5. Analytický záver

Podľa analýzy možných zariadení a programovacích technológií, sumarizácie výhod a nevýhod sa táto práca zameria na návrh riešenia v oblasti mobilných a počítačových zariadení. Možnosti riešenia pomocou týchto zariadení je najpestrejší a z hľadiska použiteľnosti najpraktickejší.

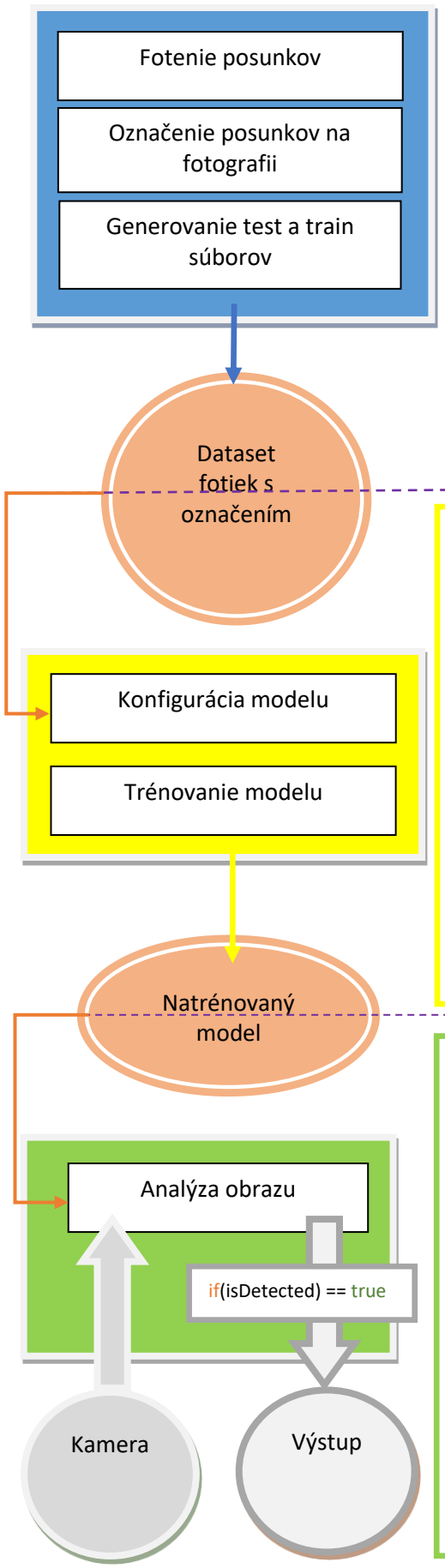
3. Návrh spracovania systému rozoznávania posunkovej reči

Návrh technológie pre počítačové a mobilné zariadenia na rozoznávanie posunkovej reči pomocou kamery je z hľadiska dostupných technológií veľmi pestrý. Počas procesu hľadania vhodného modelu pre analýzu sa zistilo, že pravdepodobne neexistuje slovenský dataset posunkovej reči, ktorý by bol použiteľný už priamo na analýzu obrazu a preto sa tato kapitola zameria na návrh technológie nie len samostatného rozoznávania posunkovej reči, ale aj na návrh vlastného datasetu fotiek s ich označením ako aj generovanie modelu.

3.1. Všeobecná architektúra systému

Z bežného života je možné tvrdiť, že kamery na mobilných a počítačových zariadeniach snímajú okolie podobne ako ľudské oko. Výhodou týchto zariadení je, že majú zobrazovaciu plochu a implementované systémy, vďaka ktorým sa vie nasnímaný obraz zobraziť. To znamená, že samotná implementácia prezentácie obrazu nasnímaného cez kameru je triviálna. Otázkou teda ostáva ako analyzovať priaty obraz.

Jedným z cieľov tejto práce je implementovať systém, ktorý bude vedieť čítať ľubovoľný frame, napríklad každý tretí alebo hoci aj každý jeden, z priateho obrazu. Tento systém potrebuje natrénovaný model, vďaka ktorému bude vedieť určiť, aký posunok ukazuje. Ako sa spomína v kapitole 3, takýto model sa nepodarilo nájsť a preto je potrebné ho vytvoriť, čo vytvára potrebu vytvoriť ďalší systém tréningu dát. Na konfiguráciu a trénovanie takého to modelu je potrebné mať dané dáta, čiže prvotný krok je tieto dáta vytvoriť. V konečnom dôsledku je potrebné začať vytvorením systému na zber dát. Nasledujúci diagram na ďalšej strane znázorňuje spomínané systémy, ako aj ich postupnosť a komunikačný proces:



Systém zberu dát

1. Tento systém pozostáva z nasnímania fotiek daných posunkov. Každý posunok sa nafotí niekoľko krát pre lepšie tréningové výsledky.
2. Pokiaľ sú posunky nafotené, je potrebné v každej fotografii označiť kde sa posunok nachádza, aby pri trénovaní systém vedel, ktorý objekt ma z danej fotografie trénovať.
3. Nakoniec sa fotografie s označením prerozdelia na trénovanie a testovanie a vygenerujú sa trénovacie a testovacie súbory vo formáte potrebnom na konfiguráciu .

(Bližšie v kapitole 3.2.)

Systém tréningu dát

1. Konfigurácia modelu pozostáva z nastavenia tried, čiže počtu druhov posunkov, ktoré boli nafotené, priradení značiek a informácie o spôsobe tréningu napríklad nastavení farieb, náhodne prevrátenie obrazu, počet krokov učenia a podobne.
2. V súčasnosti existujú rôzne trénovacie algoritmy. V prípade riešenia pre túto prácu sa použije Object Detection API
[20] , z dôvodu funkcionalít korešpondujúcimi s týmto riešením.

(Bližšie v kapitole 3.3.)

Systém rozpoznávania

Posledným krokom ostáva natrénovaný model načítať a postaviť podľa pokynov Object Detection API a vytvoriť algoritmus, ktorý bude cyklicky analyzovať obraz.
Pseudokód pre algoritmus môže vyzerať nasledovne:

```
while(true){
        frame = Kamera.getActualFrame();
        if(frame.isGestureDetected){
                Výstup.notifyUser();
        }else{
                continue;
        }
}
```

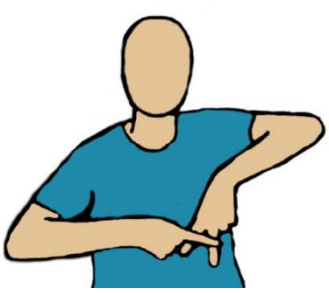
(Bližšie v kapitole 3.4.)

Diagram 1 Všeobecný systém rozoznávania posunkovej reči.

3.2. Systém zberu dát

Dnešná spoločnosť je skvelo vytrénovaná na vytváranie selfie fotiek. V tomto systéme namiesto sledovania tvárovej oblasti je potrebné presmerovať pozornosť na ruky, pretože od toho závisí, ako sa model v konečnom dôsledku natrénuje.

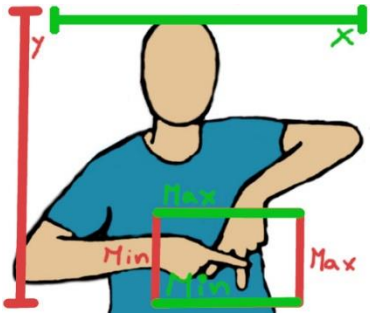
Fotenie posunkov



Aby bol model natrénovaný čo najlepšie, je potrebné odfotiť pár krát gesto priamo na kameru. Ďalšie fotky by mali obsahovať aj mierne potočenie gesta. Zmena pozadia je tak tiež dobrý krok, pretože sa môže stať, ak je za gestom výrazný prvok ktorý bude v označení, že ho bude model vyžadovať ako súčasť objektu, čiže gesta. Obrázok 7 znázorňuje príklad pozície na kamere pre písmeno ‘a’.

Obrázok 7 Názorný príklad fotenia datasetu

Označovanie posunkov na fotografii – labelling



Aby sa model dokázal naučiť požadované gesto, potrebuje mať označené minimá a maximá pre y-ovú a x-ovú os. Na základe toho označenia bude hľadať a porovnávať opakujúci sa vzor. Ďalší dôležitý údaj je konkrétna značka napríklad ako podľa obrázka 8 je ‘a’. Vďaka tomu model bude vedieť názov zobrazeného gesta čo je kľúčová hodnota pre rozoznanie aj samotný výstup.

Obrázok 8 Názorný príklad označenia fotografie

Generovanie test a train

V poslednom kroku tohto systému sa prerozdelí väčšinová časť fotiek jedného gesta do train súboru a zvyšok do test súboru. To platí aj pre ostatné gestá. Príkladový diagram:

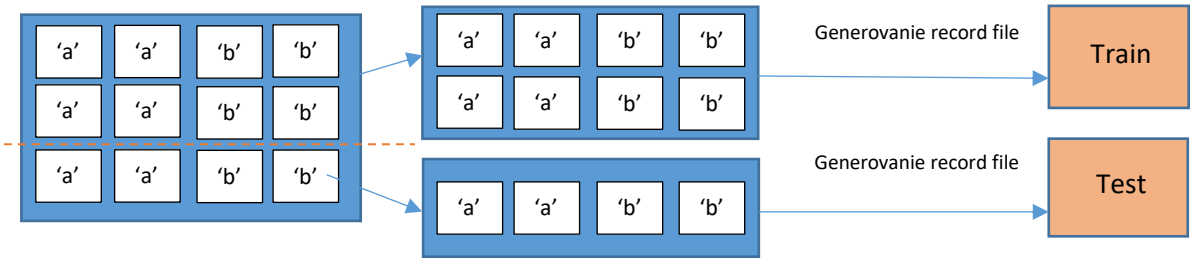


Diagram 2 Príkladový diagram rozdelenia fotiek do datasetu so značkou ‘a’ a ‘b’.

3.3. Systém tréningu dát

Trénovanie dát je kľúčový bod k tomu, ako sa bude chovať výsledný výstup rozoznaných posunkov. Ako sa spomína v kapitole 3.1. v časti o systéme tréningu dát v bode 2., existuje mnoho modelov na trénovanie. Je dôležité zvoliť si model, ktorý korešponduje s potrebami pre túto problematiku. Táto práca sa teda zameria na možnosti poskytované od Object Detection API, ďalej len ODA. Tieto API fungujú na princípe učenia sa rozoznávania objektov podľa ich predpokladaného tvaru. [20] V tomto prípade ide stále o ruky, avšak v rôznych pozíciách a tvaroch čo v konečnom dôsledku vytvára každé nove gesto nový objekt.

Konfigurácia modelu

Konfigurácia modelu pozostáva z nastavení parametrov pre model. To znamená že treba synchronizovať vybraný model od ODA s vygenerovanými súbormi podľa návrhu v kapitole 3.2. Nasledujúce body ukazujú kľúčové parametre, na ktoré sa bude sústrediť táto práca.

To zahŕňa napríklad:

- Nastavenie počtu tried, čiže koľko druhov objektov sa plánuje učiť, čiže koľko druhov gest
- Ktorý typ modelu sa využije od ODA
- Cestu k uloženým značkám
- Či sa má model učiť aj zrkadlovú verziu ukázaného gesta [21]

Modely ODA, o ktoré sa táto práca zaujíma sa nazývajú Tensorflow 2 Detection Model Zoo, ktorý majú zverejnený na webe ako aj popis každého modelu napríklad aká je jeho rýchlosť a čí požaduje označovanie v tvare kocky alebo je možné objekt označiť pomocou bodov v jeho obrysoch. Priamy link je možné nájsť v kapitole zoznam použitej literatúry viď. [22]

Trénovanie modelu

Pokiaľ je všetko správne nastavené, táto časť vyžaduje už iba spustenie skriptu od ODA. Rýchlosť trénovania datasetu závisí už len od toho, či počítač na ktorom sa tento návrh implementuje má viac jadrový procesor a ak áno tak koľko, či sa to plánuje učiť cez grafický procesor (GPU) alebo cez centrálny procesor (CPU). Tieto možnosti sa budú ďalej rozoberať v priamej implementácií.

3.4. Systém rozpoznávania

Konečný krok celého rozpoznávacieho systému je vytvoriť skript, ktorý:

- načíta naučený model,

- načíta kameru používateľa,

- cyklicky bude prezerať po jednom frame-e obraz a detekovať,

- v prípade detekcie vykreslí výsledok za pomoci značiek.

Nasledovný diagram zobrazuje postupnosť operácií ako aj príkladný pseudokód skriptu:

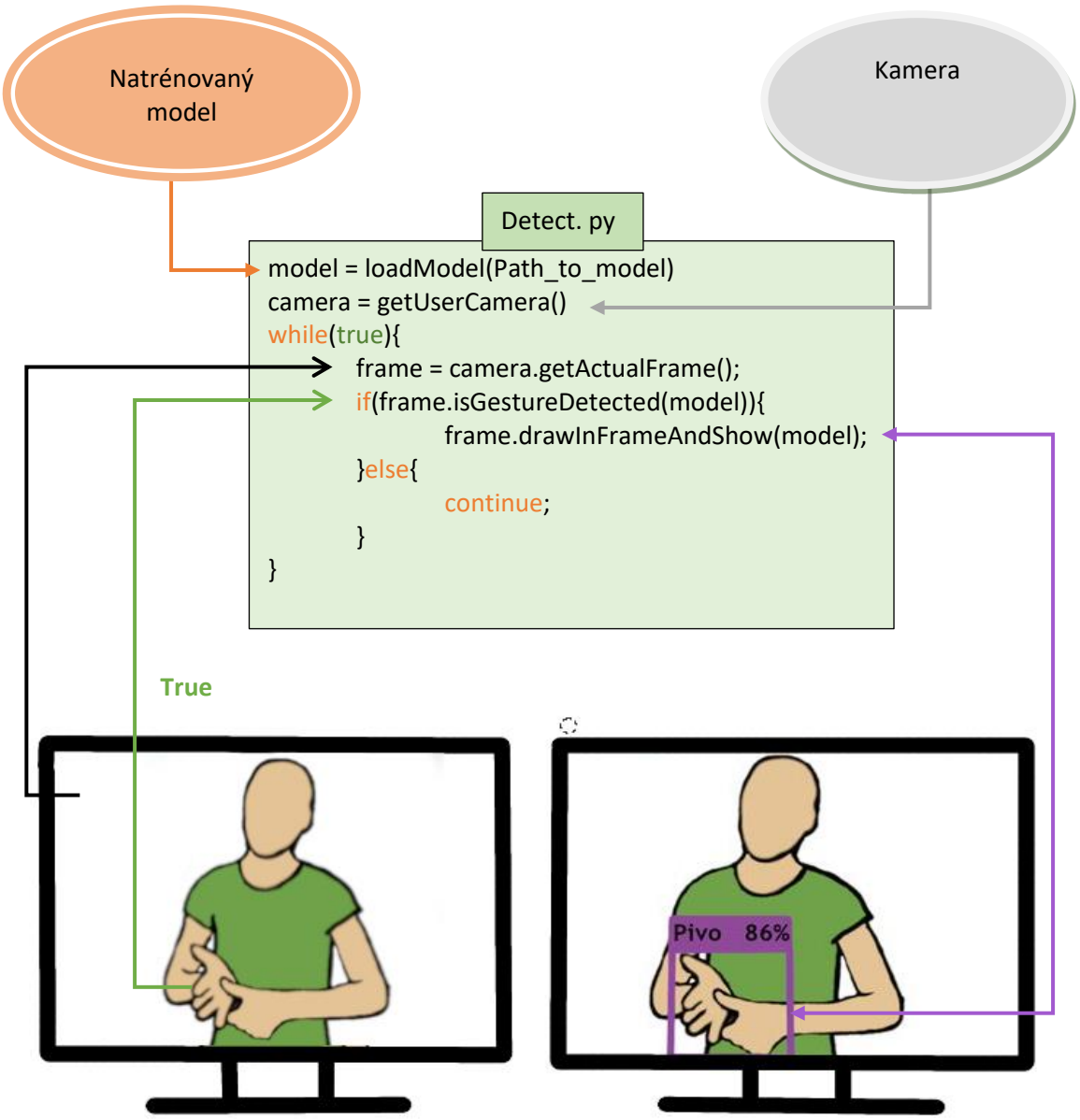


Diagram 3 Návrh systému rozpoznávania posunkovej reči

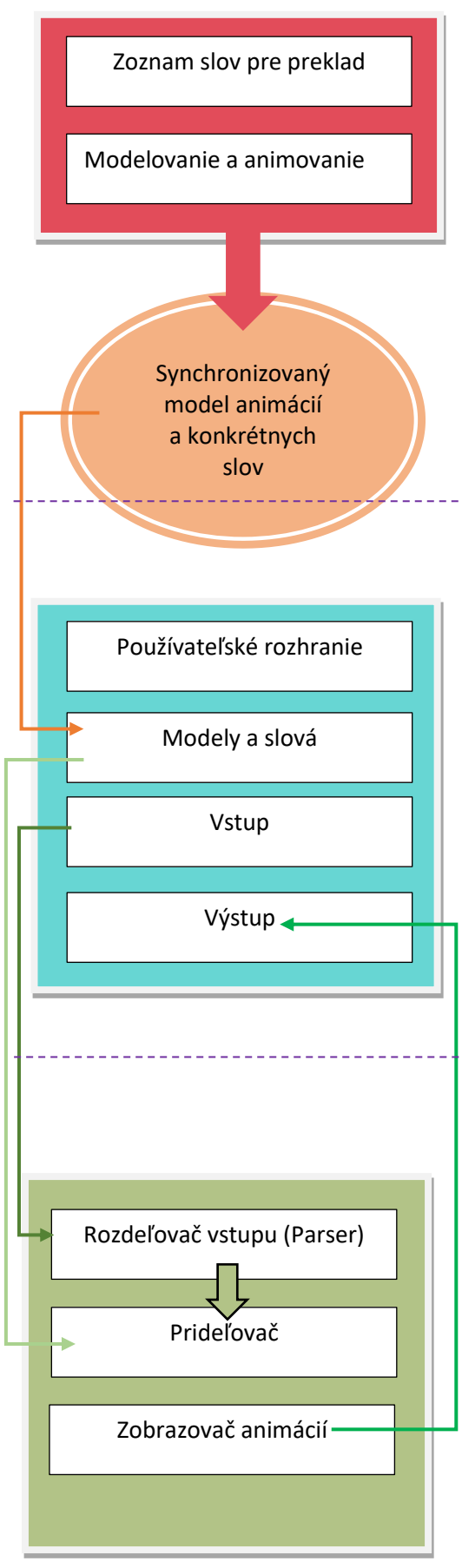
4. Návrh spracovania systému pre preklad do posunkovej reči

Pri prehodnotení analýzy v kapitole 2.4., sa táto práca bude sústrediť na preklad priamo z textu sprevádzane vizualizáciou pomocou 3D modelu avatara. Princíp takého to fungovania môže byť zdanlivo jednoduchý a to napísať text a následne ukázať animáciu posunku. Avšak je potrebné vymyslieť algoritmy pre spracovanie vstupu pretože ako sa spomína v kapitole 2.4. Preklad textu do posunku nie je taký jednoduchý. Je potrebné ošetriť jazykové výnimky, skloňovania, priradzovanie osoby a podobne. Všetky riešenia pre tieto jazykové bariéry budú navrhnuté v tejto kapitole.

4.1. Všeobecná architektúra systému

Vo všeobecnosti je možné tvrdiť, že pri preklade z textu do posunku je potrebné nájsť sémantický význam napísaného slova alebo postupnosti slov a nájsť vhodné gesto ako jeho sémantické synonymum. V princípe je možne vnímať tento systém ako prekladač z jedného jazyka do iného. Národnostné jazyky majú primárne zvukovú formu, programovacie jazyky majú primárnu, ak nie len iba, textovú formu. V princípe zvukové výrazy v akomkoľvek národnostnom jazyku majú svoj presný ekvivalent pre textovú formu. Jediný posunkový jazyk je jazyk grafického charakteru. Preto, hoci sa jedná o slovenské posunky, nenájdeme ku každému textovému slovu presný a jediný ekvivalent gesta. Preto architektúra takého to prekladu je trochu komplexnejšia.

Architektúra tejto problematiky by mala byť navrhnutá čo najuniverzálnejšie z dôvodu prípadnej budúcej potreby pridávať nové slová a modely pre preklad z textu do posunkovej reči. V princípe sa dá prerozdeliť na postupnosť krokov podobne ako pri návrhu systému rozoznávania posunkovej reči z diagramu 1 v kapitole 3. Postupnosť krokov sa však nemusí striktne dodržať. V tejto práci však bude postupovať podľa nasledovného diagramu 4 :



Systém tvorby modelov a algoritmizácie slov

1. Prvý krok bude logicky výber slov aby finálny systém vedel zložiť základné vety. Tak tiež bude obsahovať algoritmy dopomáhajúce stohovaniu rôznych verzií týchto slov, aby sa nestalo, že program vie preložiť napríklad 'chlapec', ale nie 'chlapcovi', 'chlapcom' a podobne.
(Bližšie v kapitole 4.2.)

2. Podľa logického výberu slov sa namodelujú ekvivalenty v gestikulačnej podobe
(Bližšie v kapitole 4.3.)

Systém webového rozhrania

1. Vizuálny návrh používateľského rozhrania
2. V tomto systéme by už mala byť ucelená forma modelov a slov, záleží už len od návrhu repozitárov pre nájdenie cesty.
3. Vstupy sú textového charakteru, čiže slovo alebo veta od používateľa ktorá ma byť preložená.
4. Výstup je zobrazenie animácie alebo postupnosti animácií na základe vstupu.
(Bližšie v kapitole 4.4.)

Systém spracovania vstupu a výstupu

1. Je potrebné vedieť rozdeliť vstup ak obsahuje viac slov na jednotlivé slová a kategorizovať ich do slovných druhov.
2. Prideľovač by mal triediť slová k príslušnej animácií pomocou vygenerovaných polí všetkých možných tvarov slov ktoré sa rozoberajú v kapitole 4.2
3. Posledným krokom celého procesu jedného alebo viacerých slov, ak prešli všetkými kritériami, by malo byť pole upravených slov korešpondujúcim s názvami animácií, čiže sa budú postupne premietať.
(Bližšie v kapitole 4.5.)

Diagram 4 Všeobecný systém prekladu textu do posunkovej reči.

4.2. Systém výberu a generovania slov pre preklad

Prvým krokom pri celom systéme prekladu je vybrať gestá, ktoré sa v konečnom kroku budú zobrazovať pomocou modelu a súčasne vybrať slová k daným gestám. Táto kapitola sa zameria na správny výber slov k príslušnému gestu, preložených pomocou slovníka zo stránky posunky.sk [2]. Na začiatku sa stanovil približný počet gest, ktoré ma systém vedieť rozoznávať. Po dohode zo školiteľom sa zhodlo na čísle 200.

4.2.1. Výber a formulácia podstatných mien

Na základných školách sa učí slovenský jazyk. Tam bolo možné nadobudnúť vedomosť o podstatných menách a ich formách. Na základe toho je možné tvrdiť, že jedno podstatné meno má skoro stále plus mínus 12 podôb, pokiaľ nejde o pomnožné podstatné meno alebo slová s dvojtvarmi.

Príklad1: Uhorka, vzor žena

Pády	**Singulár**	**Plurál**
Nominatív (kto/čo)	1. Uhorka	1. Uhorky
Genitív (od koho/od čoho)	1. Uhorky	1. Uhoriek
Datív (komu/čomu)	1. Uhorke	1. Uhorkám
Akuzatív (koho/čo)	1. Uhorku	1. Uhorky
Lokál (o kom/o čom)	1. Uhorke	1. Uhorkách
Inštrumentál (s kým/s čím)	1. Uhorkou	1. Uhorkami

Tabuľka 1 Skloňovanie slová uhorka

Na základe tabuľky 1. si je možné všimnúť istý opakujúci sa vzorec. Uhorka ma základ slova ‘uhork’ jedine v pluráli ma výnimku ‘uhor’. Nie všetky ženské rody sa povedia v genitíve plurálu s koncovkou ‘iek’ napríklad mama sa povie mamám.

Na základe tohto príkladu je možné tvrdiť, že všetky podstatné mená vzoru žena podliehajú tomuto skloňovaniu, tým pádom stačí vedieť základ slova a či sa jedná o výnimku v genitíve plurálu, aby sa vytvoril automat pre generovanie zvyšných foriem tohto slova.

Príklad2: Otec, vzor chlap

Pády	Singulár	Plurál
Nominatív (kto/čo)	1. Otec	1. Otcovia
Genitív (od koho/od čoho)	1. Otca	1. Otcov
Datív (komu/čomu)	1. Otcovi/Otcu	1. Otcom
Akuzatív (koho/čo)	1. Otca	1. Otcov
Lokál (o kom/o čom)	1. Otcovi	1. Otcoch
Inštrumentál (s kým/s čím)	1. Otcom	1. Otcami

Tabuľka 2 Skloňovanie slová otec

Rovnako ako príklad 1 v tejto kapitole, je možné si všimnúť opakujúci sa vzor. Slovo je Otec, základ slova je ‘otc’ a v genitíve singuláru je dvojtvar, čiže ide o špeciálnu výnimku. Slovo brat napríklad nemá tento dvojtvar.

Čiže znova, na základe príkladu 2. je možné tvrdiť, že všetky podstatné mená vzoru chlap podliehajú tomuto skloňovaniu, pokiaľ sú špeciálne a majú dvojtvar v genitíve singuláru.

Rovnaký princíp platí aj pre všetky ostatne vzory v slovenskom jazyku. V slovenčine sú nasledujúce vzory:

Mužské:	chlap	hrdina	dub	stroj		
Ženské:	žena	ulica	kosť	dlaň	gazdiná	idea
Stredné:	mesto	srdce	vysvedčenie	dievča		

Čiže je 14 spôsobov, pokiaľ nepočítame pomnožné a výnimky, ako sa môže podstatné meno v slovenčine formovať.

Všetky fakty spomenuté vyššie umožňujú vytvoriť automat pre stohovanie slov aby sa nemuseli písať ručne v každom tvare a aby sa vedeli priradiť k názvu animácie, jediné čo treba vedieť je:

1. názov animácie
2. slovo
3. vzor
4. základ slova
5. či sa jedná o špeciálnu výnimku pre daný vzor

Syntax zápisu podstatných mien

Ako sa spomína na predchádzajúcej strane, syntax na stohovanie slova do poľa s potrebnými parametrami môže byť teda nasledovná:

[a] b #V (c1 c2) d#

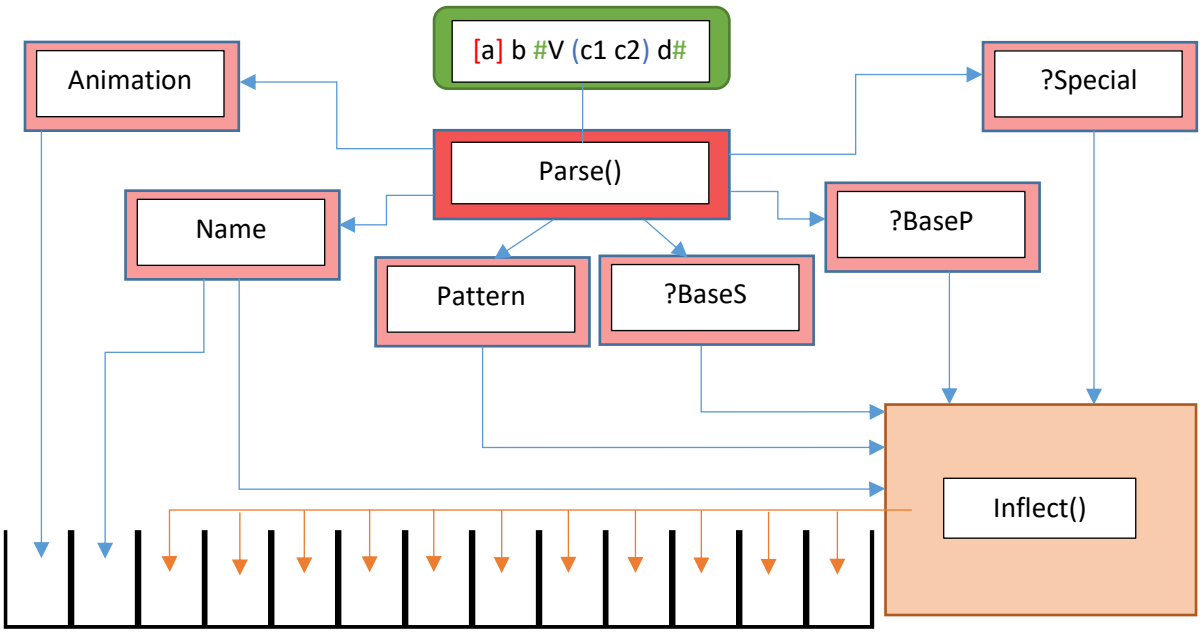
[] – separačné značky pre rozpoznanie, že obsah v nich je názov animácie,
'a' - názov animácie,
'b' – názov podstatného mena v nominatíve singulári,
'##' – separačné značky pre rozpoznanie, že obsah v nich je informácia o slove
'V' – skratka vzoru,
'()' – separačné značky pre rozpoznanie, že obsah v nich je základ slova v singulári alebo pluráli, ak sa nenachádza ani jeden, skloňuje sa podľa nominatívu, ak sa nachádza jeden, použije sa aj pre plurál aj singulár,
'c1' – nepovinný parameter základu slova v singulári (ak sa základ slova nezhoduje s nominatívom singuláru),
'c2' – nepovinný parameter základu slova v pluráli(ak sa základ slova nezhoduje s nominatívom singuláru),
'd' – nepovinný parameter skratky, ak ide o špeciálnu výnimku pre konkrétny vzor,

Príklady zápisu:
[uhorka] uhorka #Z (uhork)#
[hruška] hruška #Z (hruš)#
[hruška] paradajka #Z (paradajk)#

Na príklade s hruškou a paradajkou je tiež vidno, že takto je možné zapisovať aj synonymá alebo prípadné gestá, ktoré sa ukážu rovnako, ale v slovnej/textovej podobe majú iný lexikálny význam.

Nasledovný jednoduchý diagram poukazuje na postupnosť krokov pri navrhnutej syntaxi:

Diagram 5 Jednoduchý diagram stohovania podstatných mien



4.2.2. Výber a formulácia slovies, zámen a nekategorizovaných slov

Pre slovesá, zámená a nekategorizované slová sa navrhne všeobecný algoritmus stohovania. Dalo by sa polemizovať nad automatom pre slovesá, keďže sa v nich nachádza opakujúci sa vzorec kto tu činnosť vykonáva, ale pre účely tejto práce postačí jednoduchší stohovací algoritmus.

V princípe stačí napísať všetky verzie slova do jedného riadku, kde prvé bude názov animácie, napríklad pre slovesá:

nevedieť neviem nevieš nevie nevieme neviete nevedia , ...

Keďže sa to bude stohovať postupne, na prvej pozícií bude stále sloveso v neurčitku, čo bude značiť aj názov animácie, a pokiaľ sa zachová syntax osôb, je možné určiť, ktorá osoba činnosť vykonáva.

Príklad pre zámená:

ja mňa mne mňa mnou , ...

Tu taktiež platí pravidlo zachovania syntaxe, pokiaľ sa dodrží, je možné určiť pád .

Príklad pre nekategorizované slová:

ahoj serus nazdar sevas čauko

V tomto prípade možno ide aj o rôzne verzie gesta, systém však chce dosiahnuť čo najväčšie obsadenie súvisiacich slov pre jedno gesto. Nasledujúci jednoduchý diagram poukazuje na stohovanie slov po poradí ako boli napísané:

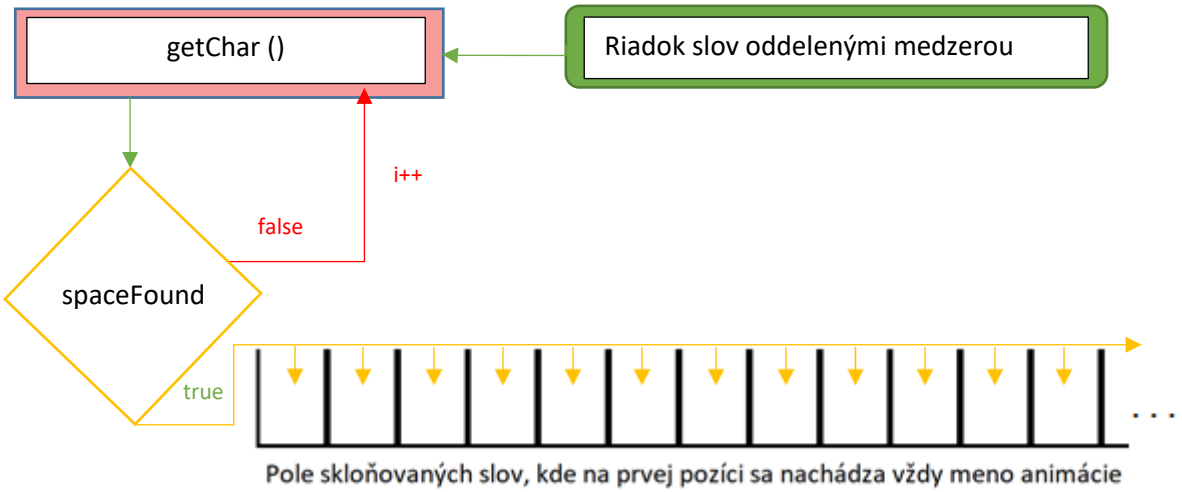
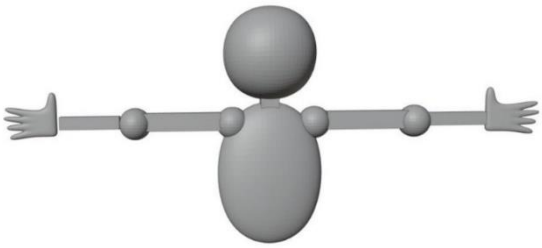


Diagram 6 Jednoduchý diagram stohovania slovo po slove

4.3. Modelovanie a animovanie

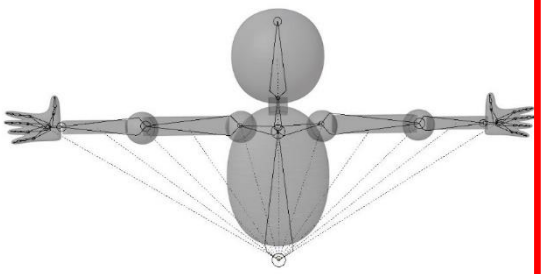
V kapitole 2.4.2. sa spomína, že modelovací program pre túto problematiku závisí primárne od preferencie. Pre účely tejto práce sa využije modelovací program Blender 2.8 [23], nakoľko obsahuje všetky potrebné komponenty pre tvorbu modelov a animácií.



1. Modelovanie

V princípe je potrebné zrobiť jeden model pre všetky animácie. Model môže byť ľubovoľného charakteru, základom však je humanoidná štruktúra. To znamená štruktúru tela človeka. Obrázok 9 znázorňuje základný vizuálny princíp takého to modelu. Model je výhodne modelovať v póze zobrazenej na obrázku 9 kvôli ľahšiemu priradeniu kostry v nasledujúcom kroku.

Obrázok 9 Návrh modelovania avatara



2. Priradenie kostry

Podobne ako u ľudí, pre animované modely sa tvorí takzvaná kostra. Táto kostra napomáha modelu pohýnať sa smerom akým sa hýbe jednotlivá kosť. Keďže v tejto práci sa chce dosiahnuť animovanie gest, treba si obzvlášť dávať pozor na rozloženie kosti pri rukách aby výsledne zohnutie prstov alebo rúk nepôsobilo nesymetricky. Viď obrázok 10.

Obrázok 10 Návrh pridelenia kostry pre model avatara

3. Animovanie

Animovanie pozostáva už len primárne z rotácií kostí. Takto je možné simulovať gestikuláciu rovnako ako u ľudí, viď. obrázok 11. Animácie sa budú vytvárať na základe vybraných slov, ktoré budú vymenované už priamo v implementácií.

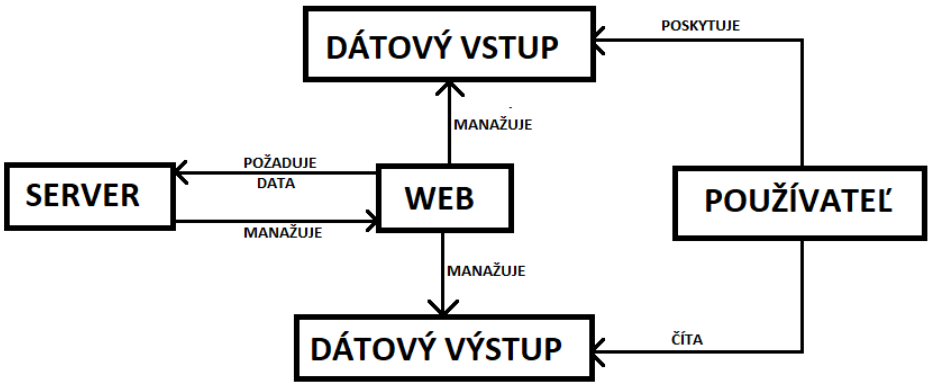
Obrázok 11 Názorný príklad pózy avatara

4.4. Systém webového rozhrania

Používateľské rozhranie predstavuje premietanie kamerového záznamu na obrazovku zariadenia, tak tiež premietanie animácií a notifikácií. Rozhranie obsahuje aj tlačidlo potvrdenia pre odoslanie vstupu.

Konceptuálny model

Nasledujúci obrázok znázorňuje konceptuálny model procesov navrhovaného systému.



Obrázok 12 Jednoduchý konceptuálny model vstupu a výstupu dát

1. Server - prijímateľ komplexnejších výstupov od webového rozhrania a odosielateľ rozpoznaných výsledkov.
2. Web – konkrétny webový klient v zariadení používateľa, manažuje zobrazovanie výsledkov čitateľných pre používateľa, prijímanie obrazových vstupov z webovej kamery zariadenia a rozoznávanie gest.
3. Dátový vstup – kamerový záznam alebo textový záznam zo zariadenia používateľa .
4. Dátový výstup – textový výstup rozoznaných posunkov alebo animácia gesta.
5. Používateľ – Čitateľ výstupu, poskytovateľ a koordinátor správnych obrazových výstupov a textov.

Schematické rozdelenie obrazoviek

Webové riešenie tejto problematiky vytvára problém zobrazovania systému na zariadeniach, ktoré môžu mať rôznu veľkosť. Na to, aby to bolo responzívne riešenie, je potrebné navrhnúť používateľské rozhranie pre malé aj veľké obrazovky samostatne.

Zariadenia s veľkosťou šírky do 800 pixelov

Do tejto kategórie spadajú:

- Smartfóny

- Malé tablety do šírky 800px

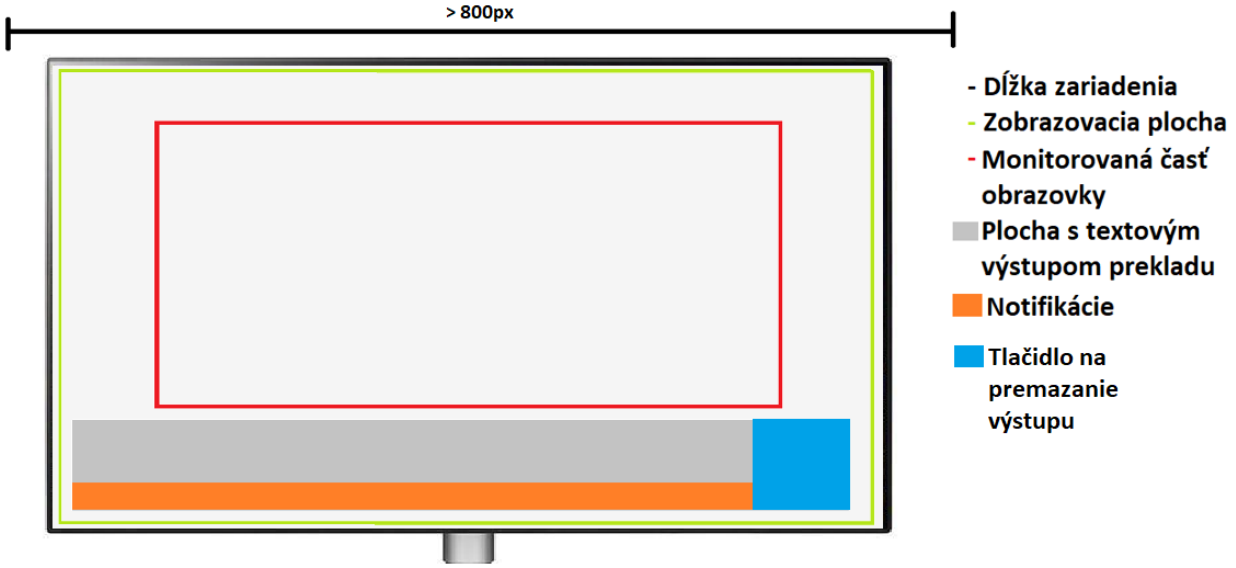
Obrázok 13 Príkladné rozloženie obrazovky pre mobily so štandardnou výškou – obrazovka pri analýze posunkov

Obrázok 14 Príkladné rozloženie obrazovky pre mobily so štandardnou výškou a šírkou – obrazovka pri reprezentácií posunkov z textu

Zariadenia s veľkosťou šírky nad 800 pixelov

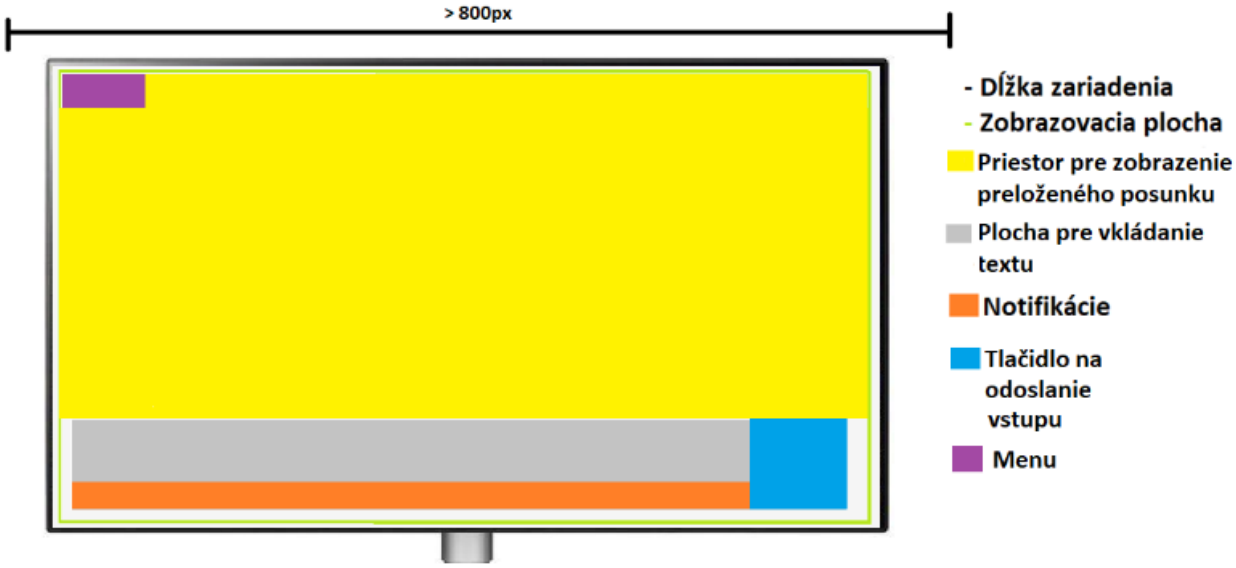
Do tejto kategórie spadajú:

- Zobrazovacie plochy o šírke a výške podľa štandardov monitorových zariadení.



Obrázok 15 Príkladné rozloženie obrazovky pre monitory so štandardnou výškou a šírkou - obrazovka pri analýze posunkov

Obrázok 16 Príkladné rozloženie obrazovky pre monitory so štandardnou výškou a šírkou -



obrazovka pri analýze posunkov

4.5. Systém spracovania vstupu a výstupu

Ako sa spomína v kapitole 4.1., cieľom spracovania vstupu, je v prípade tejto problematiky nájsť sémanticky ekvivalent k modelu z napísaného textu a zobraziť ho/ich. Návrh tohto systému v tejto práci pozostáva z troch krokov k dosiahnutiu stanoveného cieľa:

Rozdeľovač vstupu (parser):

Pre rozdeľovač platí nasledujúca postupnosť:

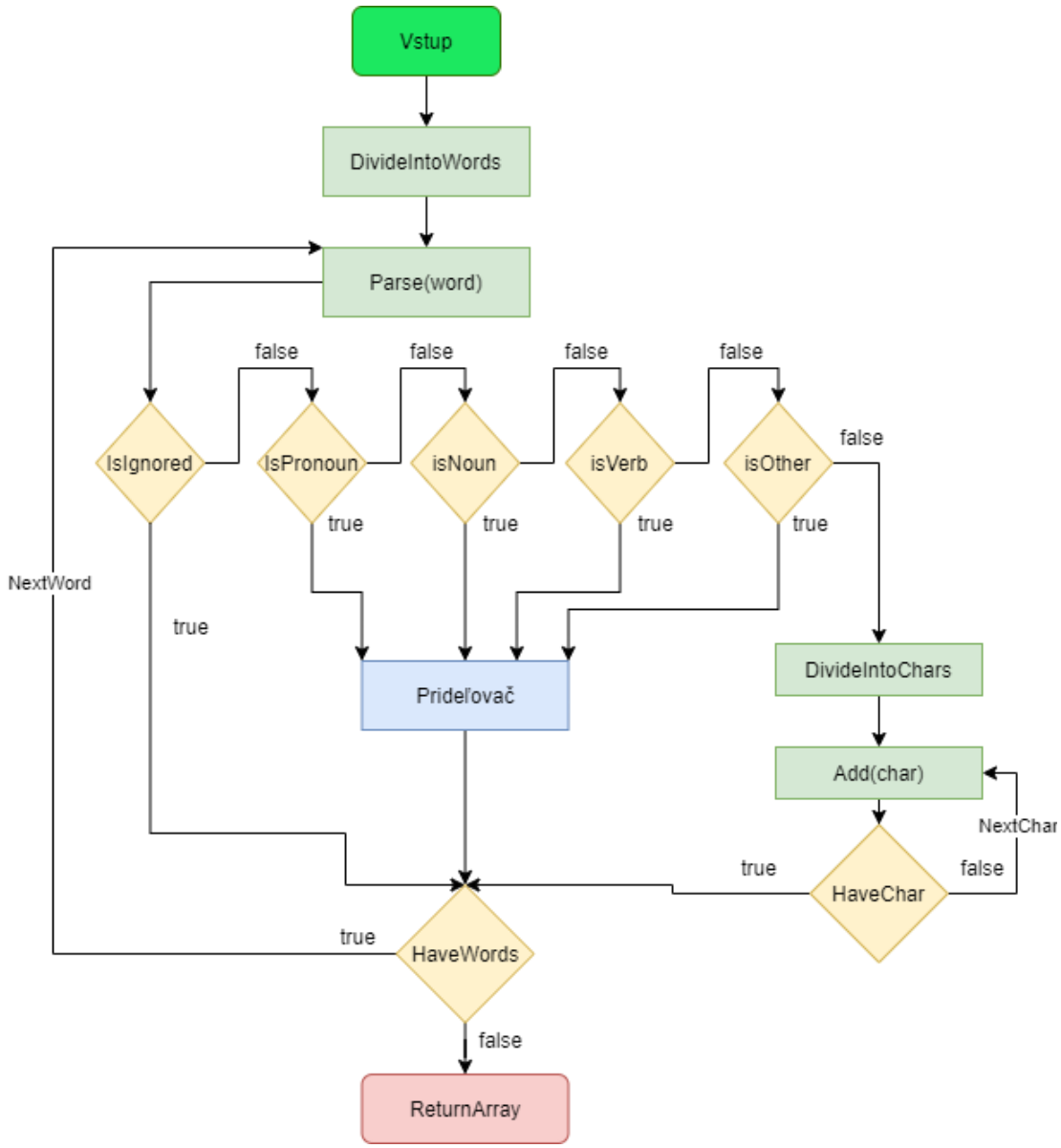


Diagram 7 Algoritmus rozdeľovača vstupu (parser) .

Znázornený diagram 7. v princípe opisuje, že vstup sa prerozdelí na slová v kroku DivideIntoWords. Každé slovo sa potom samostatné hodnotí o aký slovný druh ide a ak sa zhoduje so slovným druhom prechádza do fázy prideľovača, ktorý bude opísaný ďalej v tejto kapitole. Ak

sa nezhoduje ani s jedným, jednotlivé slovo sa prerozdelí na písmená v kroku DivideIntoChars a každé písmeno/číslo sa samostatne uloží do výsledného poľa.

Prideľovač :

Pre rozbalený popis prideľovača platí:

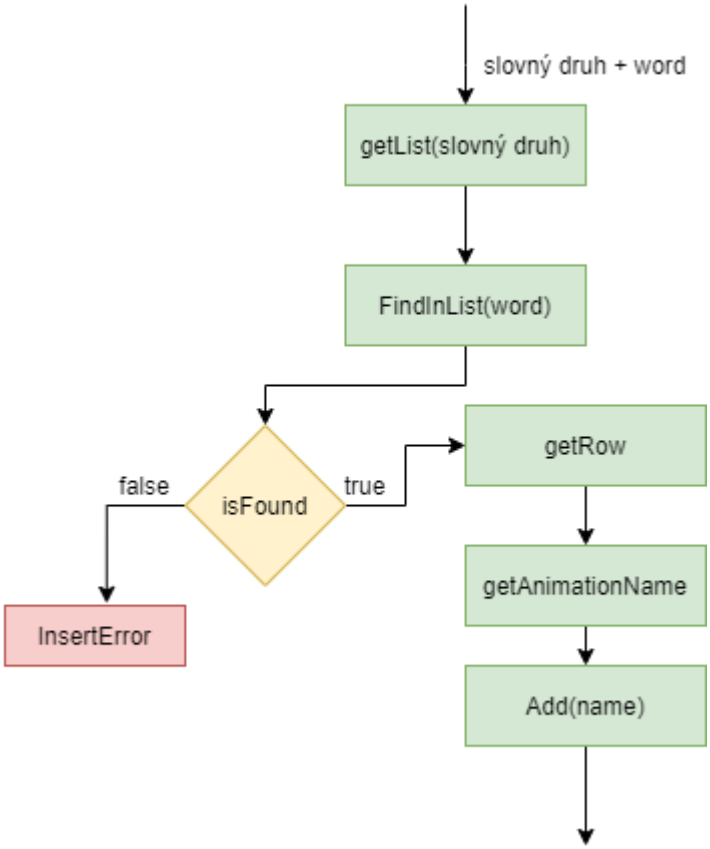


Diagram 8 Algoritmus prideľovača .

Znázornený diagram 8. popisuje detail prideľovača z diagramu 7. Na základe podmienok určenia slovného druhu prideľovač vie, ktorý zoznam slov, generovaný na základe návrhu v kapitole 4.2., má vybrať. Prehľadá sa celý zoznam, či sa tam nachádza slovo zo vstupu od používateľa. Ak nie, namiesto názvu animácie sa vloží chybná hláška. Ak áno, vezme sa riadok daného zoznamu a z prvej pozície, kde sa podľa návrhu má nachádzať názov animácie, sa tento názov pridá do poľa.

Výsledné pole by malo teda obsahovať názvy postupnosti animácií. V prípade slov to koriguje prideľovač, v prípade písmen a čísel to je priama cesta, keďže animácia s názvom a obsahom písmená/čísla sa bude volať rovnako ako jednotlivý vstup po korekcii diakritiky.

Zobrazovač animácií :

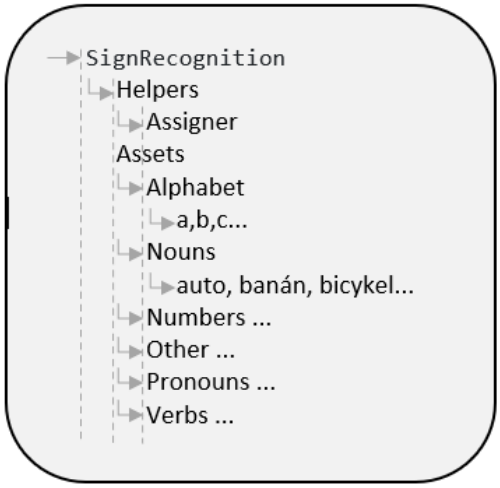
Ako sa spomína na predchádzajúcej strane, výsledkom by malo byť pole názvov animácií logicky idúcimi za sebou podľa vstupu používateľa.

Majme teda príkladný vstup : „Ja chcem ísť spať ihneď. „

Výsledkom by teda mohlo byť: „[ja], [ ], [chciet], [ ], [ist], [ ], [spat], [ ], [i], [h], [n], [e], [d],“

Úlohou zobrazovača je nájsť cestu k správnemu repozitáru a získať cestu pre zobrazenie animácie v html.

Pseudokód môže vyzerať nasledovne pokiaľ sa zachová navrhovaná štruktúra repozitárov:




```
for ( i = 0; i  <  arrayAnimations.lenght ; i++ ){
        type = arrayAnimations [i].getType();
        path = assign(type, arrayAnimations [i]);
        showAnimationFrom(path);
        wait untill animation is done;
}

assigner(type, name){
        Return '.../assets/' + type + '/' + name;
}
```

Prechádzanie poľom názvov.

Získanie typu, teda názvu repozitára, v ktorom je konkrétne slovo

Získanie cesty k animácií, pomocou funkcie assigner, do ktorej sa vkladá názov z poľa a názov repozitára

Zobrazenie animácie na základe cesty pomocou html (napríklad ak by animácia bola vo forme gifu: <img [src] =”path” >)

Alternatívou celého riešenia môže byť aj uloženie fotiek niekde na cloud a získať si cestu cez url link. Avšak takéto linky nemusia mať jednoznačný opakujúci sa vzorec v názve cesty a preto môže nastať problém, že by sa tie cesty museli po jednom napísať manuálne do kódu a priraďovali by sa len premenné. Pri veľkom počte modelov to nemusí byť efektívne riešenie.

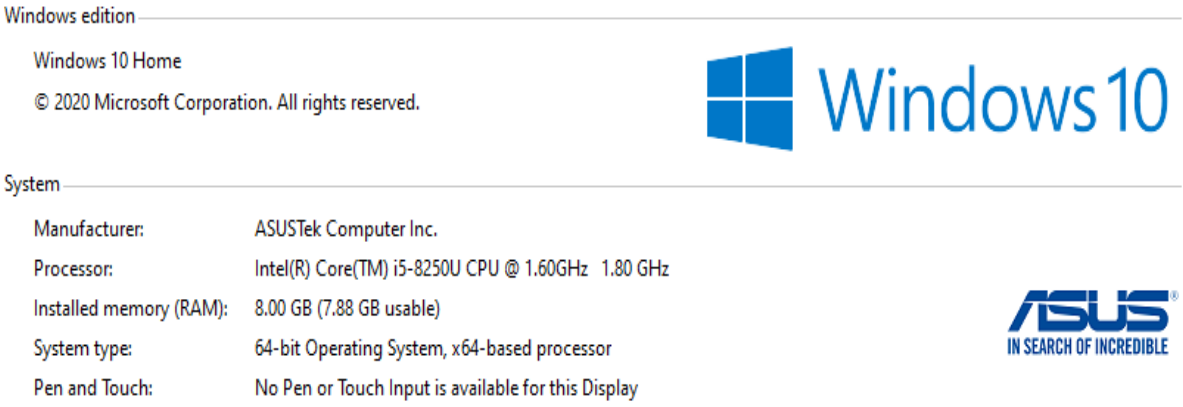
5. Implementácia návrhu systému rozoznávania posunkovej reči

Táto časť práce je implementovaná podľa návrhu postupu v kapitole 3. Na diagrame 1 je znázornený návrh postupnosti vývoju celého systému. Pred tým ako sa začnú implementovať konkrétne systémy zberu dát, tréningu dát a samotného rozpoznávania podľa návrhu, je potrebné opísať operačný systém, vývojové prostredia, potrebné moduly a iné komponenty, na ktorých je táto problematika stavaná.

5.1. Špecifikácia operačného systému a implementácia potrebných komponentov

Detailné špecifikácie a príkazy na inštalácie potrebných modulov sú opísane v systémovej príručke. Táto kapitola sa zameria na základné informácie potrebné k implementácií návrhu. Problematika systému navrhovaná v kapitole 3., je implementovaná na počítači s nasledujúcou charakteristikou.

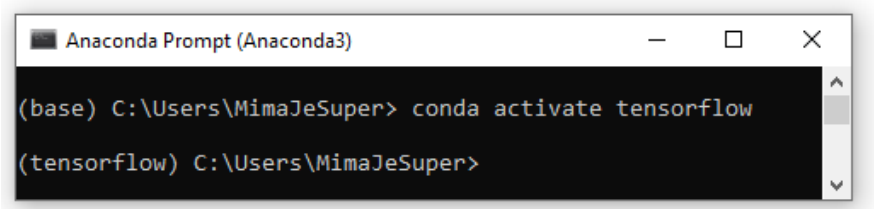
Operačný systém a parametre počítača:



Obrázok 17 Základné informácie počítača na implementáciu

Vývojové prostredie a dodatočné komponenty k vývoju:

1. Táto práca používa na implementáciu systému rozoznávania posunkovej reči Anacondu Python 3.8 [24], s vytvoreným virtuálnym prostredím s názvom tensorflow viď obrázok 18.



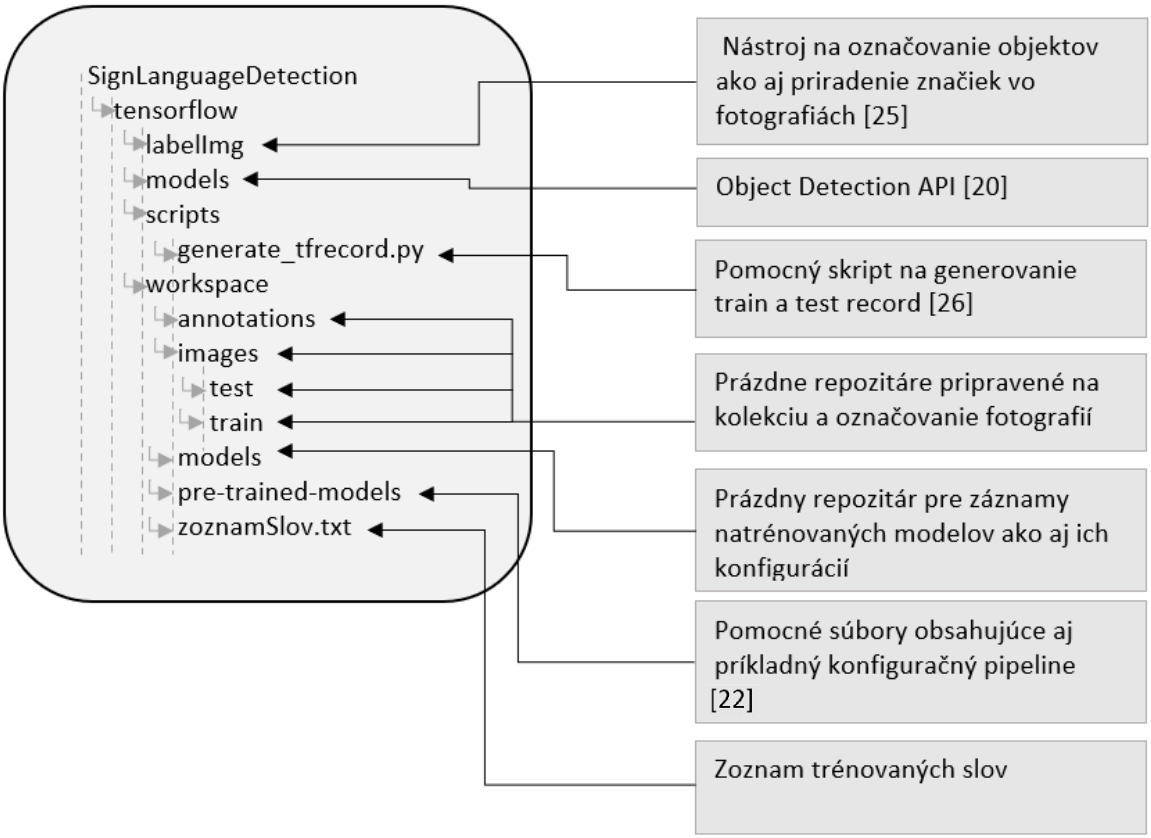
Obrázok 18 Virtuálne prostredie v Anaconde

2. Do tohto virtuálneho prostredia bol pridaný balíček tensorflow vo verzií 2.4.1. Je potrebné mať verziu tensorflowu >= 2, pretože sa budú využívať ďalšie balíčky, ktoré to vyžadujú.

3. Počítač, na ktorom je systém implementovaný obsahuje grafickú podporu NVIDIA, čo otvára možnosť nastavenia GPU, avšak tento krok sa preskočil z nedostatočnej kapacity na úložisku a preto sa bude využívať CPU podľa špecifikácie na obrázku 17.

Počiatočná štruktúra projektu:

Táto štruktúra sa bude odkazovať na nasledovnú implementáciu ako aj na predpripravené moduly a balíky na riešenie tejto problematiky. Na nasledovnom obrázku sa opisuje rozloženie repozitárov pripravené pre implementáciu návrhu ako aj opis predpripravených balíčkov:



Obrázok 19 Počiatočná štruktúra projektu pre rozpoznávanie posunku z obrazu

5.2. Systém zberu dát

Na začiatku kapitoly 3., sa spomína, že pravdepodobne neexistuje dataset so slovenskými posunkami. Preto súčasť tejto problematiky je implementovať vlastný dataset. Postupnosť implementácie bude podľa návrhu v kapitole 3.2. Táto kapitole sa organizuje v repozitároch images a annotations. Využíva sa aj pomocný nástroj labelImg podľa obrázka 19. Tak tiež je potrebné určiť zoznam slov, ktoré budú zbierané.

Zoznam slov:

Základom gest pre rozoznanie je abeceda, avšak vynechali sa znaky s diakritikou. Čísla neobsahujú nulu pretože ekvivalent je znak ‘o’. Zvyšok vybraných slov bol pomerne náhodný, avšak bola snaha o výber základných alebo často používaných slov. Obsah je nasledovný:

Abeceda									
a	b	c	d	e	f	g	h	i	j
k	l	m	n	o	p	q	r	s	t
u	v	w	x	y	z				
Čísla									
1	2	3	4	5	6	7	8	9	
Podstatné mená									
auto	brat	budova	chlapec	dievca	domov	dvere	frajer	hasic	kamera
krava	lekar	mama	pivo	policia	sestra	zena			
Zámená									
ja	ty	on							
Slovesá									
chciet	davat	drzat	jest	lezat	prosit	ucit	usmievat	dakovat	
Iné									
ahoj	ako	alebo	ano	co	kedy	kto	preco		

Tabuľka 3 Zoznam slov fotených pre dataset analýzy obrazu

Pokiaľ sú slová vybrané, je potrebné nájsť sémantický ekvivalent v posunkovej reči, k čomu dopomohol prekladový slovník slovenského jazyka – slovenského posunkového jazyka [2]. Na základe obavy misklasifikácií sa ponechal tento skromný počet slov, pre určenie v kapitole overenia výsledkov ich úspešnosť pri rozoznávaní. Preto tento dataset bude vnímaný iba ako testovaný set navrhovaného riešenia.

Automatizované fotenie posunkov:

Pre tréning a nasledovné rozpoznávanie je cieľom mať z každého slova spomenutého v tabuľke 3, 20 kusov. Keďže vymenovaných slov je dokopy 72, tvorí to 72x20 = 1440 fotiek dokopy. Pri takomto počte je výhodou zrobiť skript, ktorý automaticky bude fotiť a ukladať fotografiu podľa príslušného názvu gesta na fotografii. Takýto skript bol implementovaný pomocou nástroja jupyter notebook [27] a nachádza sa v hlavičkovom repozitári s názvom PhotoSniping, avšak je možné ho napísať v jazyku a prostredí podľa preferencie programátora. Princíp pozostáva z vytvorenia poľa premenných s názvami gest, ktoré sa plánujú nasnímať a cyklu, ktorý dá dostatočnú prestávku na

premiestnenie rúk rovnakého gesta ako aj prestávku a výpis do konzoly, že sa plánuje fotiť nové gesto s jeho názvom. Nasledovný diagram s pseudokódom poukazuje na postupnosť akcií vykonaných takýmto automatom na fotenie fotografií.

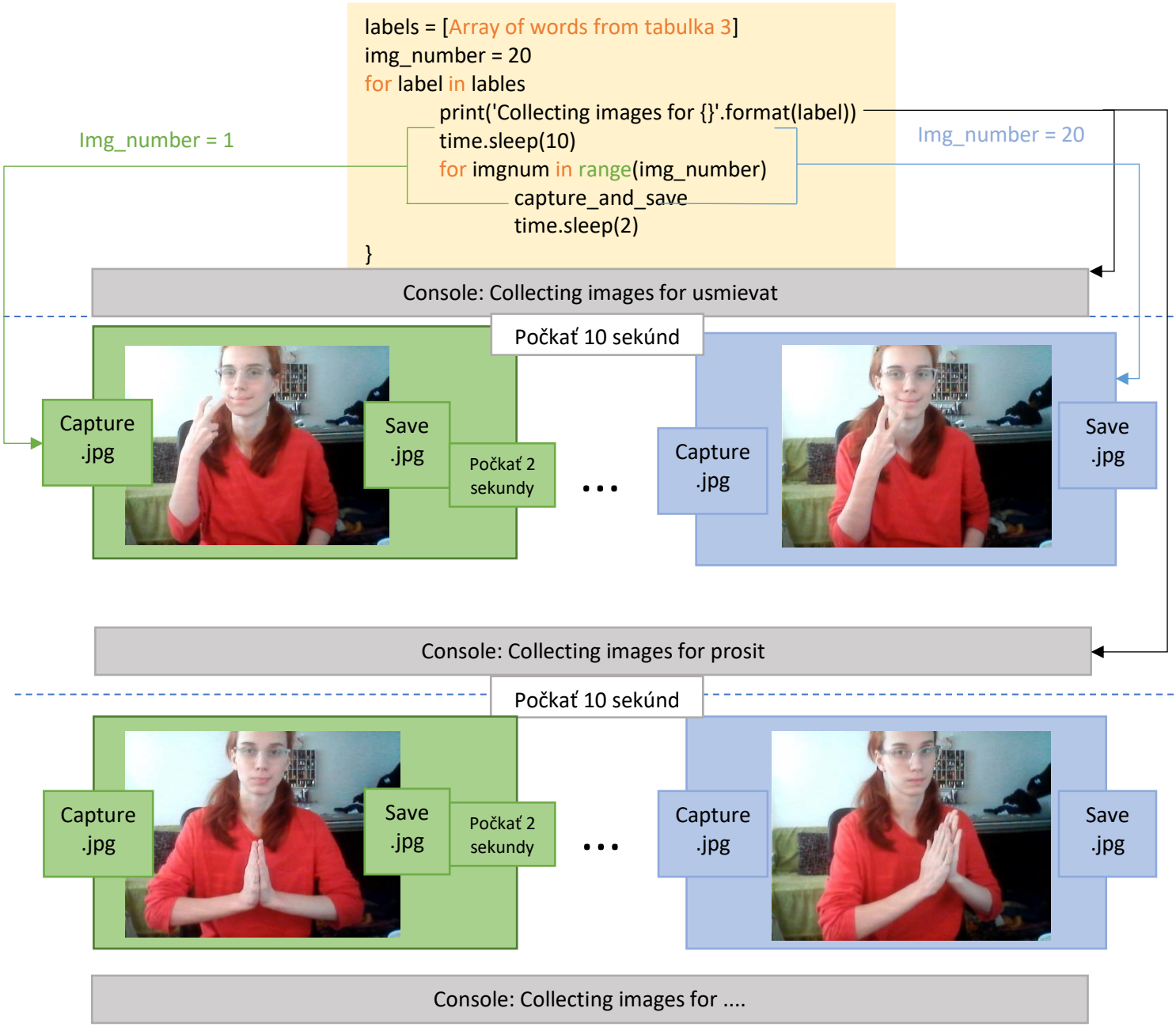
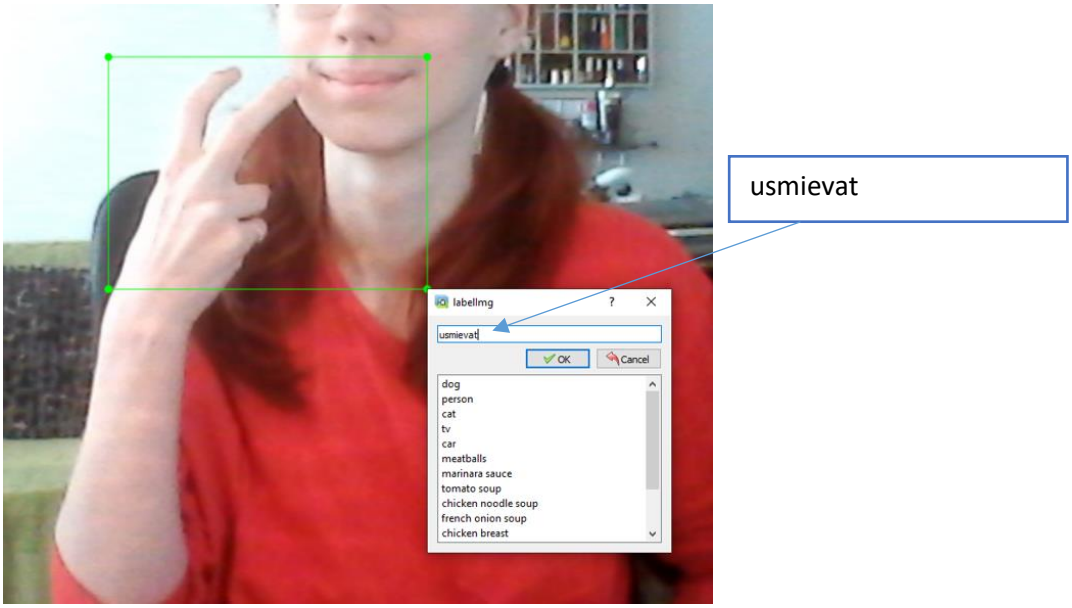


Diagram 9 Postupnosť stohovania fotiek pomocou automatu

V princípe sa vytvoril nový repozitár collectedImages, do ktorého sa stohujú fotky pod názvom ktorý bol zadaný v poli labels. Fotky sa stohujú v 2 sekundových intervaloch. Pokiaľ nastane zmena premennej v lables poli, konzola na to upozorní, vypíše o aký label ide, čiže gesto. Nechá sa 10 sekundový priestor na zmenu pozície a začne sa stohovať pre dané gesto v opakujúcom sa intervale.

Označovanie posunkov na fotografiách:

Ako už bolo spomínane, vytvorila sa séria fotiek okolo 1440 kusov. Pomocou nástroja LabelImg, spomínaného aj na obrázku 19, sa každá fotografia označí so značkou sémantiky zodpovedajúcou ukázaným gestom. Viď. obrázok:



Obrázok 20 Značkovanie fotografie pomocou nástroja LabelImg

Po označení LabelImg vygeneruje .xml súbor, v repozitári kde sa nachádza aj fotografia, ktorý bude obsahovať nasledovné informácie:

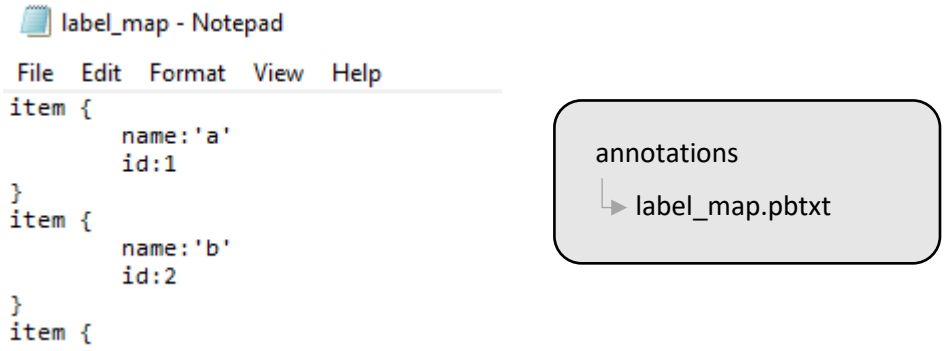
```
<?xml version="1.0"?>
- <annotation>
    <folder>all</folder>
    <filename>usmievat.05e83948-9089-11eb-ac06-7c2a318ad49f.jpg</filename>
    <path> … \Tensorflow\workspace\images\collectedimages\all\usmievat.05e83948-9089-
        11eb-ac06-7c2a318ad49f.jpg</path>
  - <source>
        <database>Unknown</database>
    </source>
  - <size>
        <width>640</width>
        <height>480</height>
        <depth>3</depth>
    </size>
    <segmented>0</segmented>
  - <object>
        <name>usmievat</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
      - <bndbox>
            <xmin>152</xmin>
            <ymin>113</ymin>
            <xmax>317</xmax>
            <ymax>221</ymax>
        </bndbox>
    </object>
</annotation>
```

1. Podľa obrázka to vyzerá, že v tagoch <path> sa nachádza natívna cesta avšak fotka bola poopravená a reálne sa tam generuje absolútna cesta, kde sa označkovaná fotografia v počítači nachádza.
2. Dôležité je pamätať si, aký názov značky bol pridelený, nachádza sa v tagoch <name>
3. A nakoniec najdôležitejší údaj sú minimá a maximá, pretože vďaka tomu sa v procese trénovania bude vedieť, kde na fotografií sa gesto nachádza

Obrázok 21 Vygenerovaný xml súbor pomocou LabelImg pre gesto usmievat

Generovanie súboru label_map:

Súbor label_map je v JSON formáte a obsahuje meno značky priradené pomocou LabelImg ako aj jeho ID. Generovanie je triviálne, stačí to napísať aj ručne. Nasledujúci obrázok znázorňuje čiastočný obsah tohto súboru ako aj umiestnenie v repozitári podľa počiatočnej štruktúry znázornenej na obrázku 19.



Obrázok 22 Súbor label_map a jeho umiestnenie

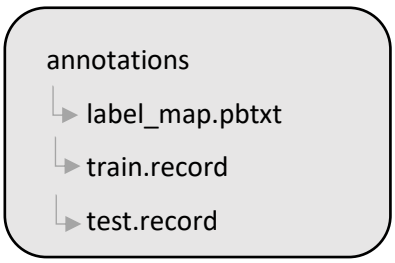
Takto je vypísaná každá značka a priradené ID podľa poradia.

Generovanie súborov test a train:

Pre generovanie je potrebné vybrať fotky z repozitára collectedImages s ich xml súbormi do train a test repozitárov v pomere 17:3, podľa obrázka 19.

Na generovanie train súboru sa použije skript generate_tfrecord.py podľa obrázka 19. spolu so s prihodeným parametrom repozitára train a súboru label_map. Posledný parameter ako -o, čiže output bude train.records. To isté platí aj pre test, kde parameter pre daný skript bude repozitár test a -o bude test.record.

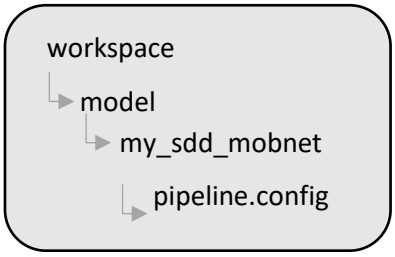
Tento krok v konečnom dôsledku vytvoril 2 súbory, ktoré sa umiestnili do repozitára annotations



5.3. Systém tréningu dát

V tomto kroku je potrebné si vybrať model z ponúkaných od Object Detection API ako sa spomína v kapitole návrhu pre systém tréningu dát 3.3. Pre túto problematiku sa vybral model SDD MobileNet V2 FPNLite 320x320, nakoľko sa dala väčšia priorita menšiemu zaťaženiu pre prenos dát, keďže sa predpokladá, že celý systém bude na webe a ako sa spomína aj v kapitole 3.2. v sekcii zoznam slov na konci, celý dataset je len experimentálny.

Do repozitára pre-trained-models sa stiahne tento model od ODA. Obsahuje pipline.config súbor pre tento stiahnutý model, ktorý je JSON formátu. Tento konfiguračný súbor je kopírovaný do repozitára models podľa obrázka na pravo.



Konfigurácia modelu:

Tento pipeline.config súbor sa prispôsobuje potrebám pre parametre doteraz implementovanými vecami. To zahŕňa:

- Pre *pipeline.model.sdd.num_classes* sa nastaví hodnota 72, pretože to je počet gest, čiže počet classes,
- Pre *pipeline.train_config.fine_tune_checkpoint_type* sa nastaví parameter "detection", pretože sa jedná o detekciu objektu na obraze
- Pre *pipeline.train_config.fine_tune_checkpoint* sa nastaví cesta pre stiahnutý check-point model z miesta, v ktorom sa bude spúšťať skript pre trénovanie, v tejto repozitárovej architektúre to je: "Tensorflow/workspace/pre-trained-models/ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8/checkpoint/ckpt-0"
- Pre *pipeline.train_input_reader.label_map_path* sa nastaví cesta pre label_map.pbtxt z miesta, v ktorom sa bude spúšťať skript pre trénovanie, znova v tejto repozitárovej architektúre to je: "Tensorflow/workspace/annotations/label_map.pbtxt "
- Pre *pipeline.train_input_reader.tf_record_input_reader.input_path* sa nastávi cesta k train súboru rovnakým princípom ako label_map.pbtxt
- A tak isto sa nastavia cesty pre label map a test record v sekcii *eval_input_reader*

Týmito krokmi je nastavený konfiguračný súbor ktorý spája datasety a tým pádom sú súbory pripravené na trénovanie.

Trénovanie modelu:

Vďaka stiahnutému repozitáru signLanguageDetection/tensorflow/models od Object Detection API podľa obrázka 19. po úspešnej inštalácií podľa ich pokynov je možné už len spustiť nasledovný skript z hlavičkového repozitára pomocou ľubovoľného cmd.

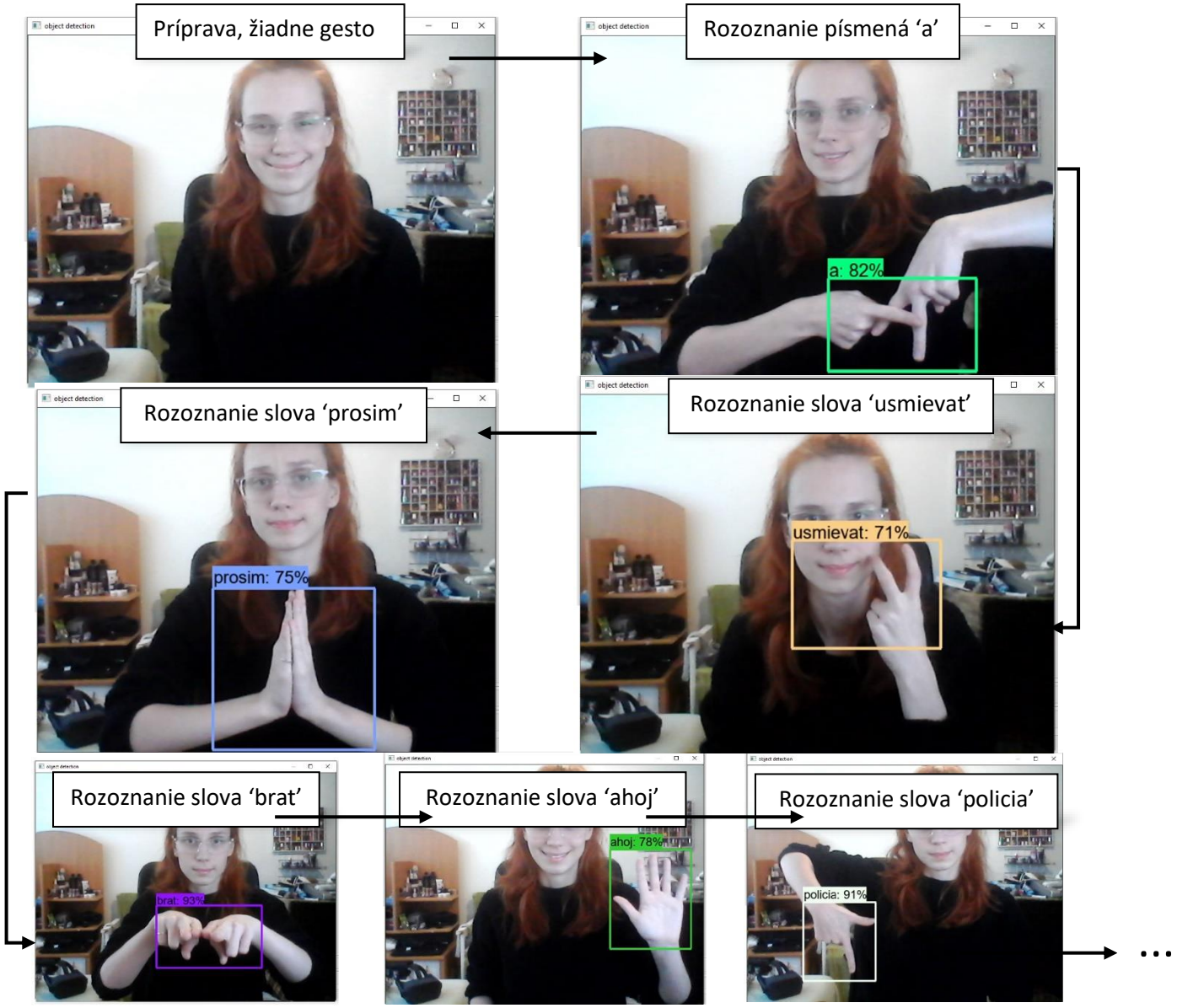
```
python Tensorflow/models/research/object_detection/model_main_tf2.py
--model_dir=Tensorflow/workspace/models/my_ssd_mobnet
--pipeline_config_path=Tensorflow/workspace/models/my_ssd_mobnet/pipeline.config
--num_train_steps=10000
```

Čas trénovania závisí od počítača. Výsledkom je súhrn checkpointov ako aj natrénovaný model.

5.4. Systém rozpoznávania

Pre zistenie funkčnosti natrénovaného modelu, sa naprogramoval skript, ktorý naučený model načíta, ktorý zapne kameru a bude čítať každý frame a nakoniec bude pomocou modelu analyzovať daný frame podľa návrhu kapitoly 3.4. Skript, ktorý pomohol aj na konfiguráciu s názvom Configuration-Detection a umiestnením v hlavičkovom repozitári, obsahuje sekciu "7. Load Train Model From Checkpoint" v ktorom sa načíta naučený model od posledného checkpointu. Sekcia s názvom "8. Detect in Real-Time" obsahuje kód, na princípe pseudokódu z diagramu 3. Stačí sa už len pripraviť a ukázať gestá na kameru pre overenie funkčnosti modelu.

Overenie riešenia je znázornené na obrázku nižšie pomocou postupnosti ukázaných gest.



Obrázok 23 Vizuálny príklad rozpoznania posunkovej reči niektorých naučených gest.

Úspešnosť rozoznania každého naučeného posunku sa preberá v kapitole 7.

6. Implementácia návrhu systému prekladu do posunkovej reči

Táto časť práce je implementovaná podľa návrhu postupu v kapitole 4. Na diagrame 4 je znázornený návrh postupnosti vývoju celého systému. Pred tým ako sa začnú implementovať zoznamy slov a ich automatov na stohovanie, tvorby modelu a animácií, webové rozhrania a systémy spracovania vstupu a výstupu, je potrebne opísať vývojové prostredia, potrebné moduly a iné komponenty, na ktorých je táto problematika stavaná. Operačné systémy ostávajú rovnaké ako v kapitole 5.1. nakoľko sa to implementuje na rovnakom počítači.

6.1. Implementácia potrebných komponentov

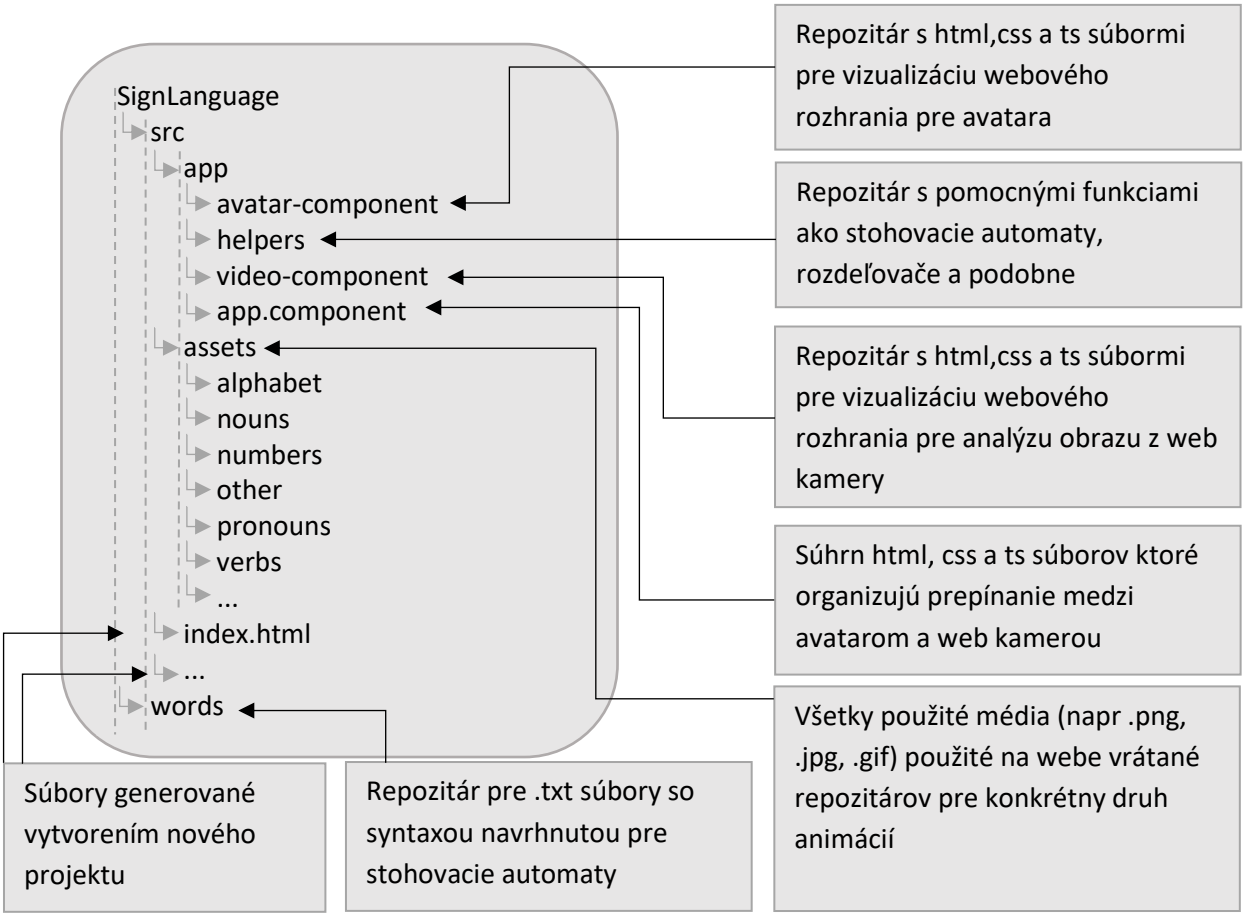
Detailný opis verzii komponentov, ako aj zoznam animácií bude opísaný v systémovej príručke. Je však potrebné opísať vývojové nástroje pre lepšie pochopenie fungovania implementovaných prvkov, ako aj počiatočnú štruktúru projektu, nakoľko sa ďalej v implementácií bude na ňu odkazovať.

Vývojové prostredie a dodatočné komponenty:

1. Ako sa spomína v analýze 2.3.3. v časti “Pre webové aplikácie”, pre túto prácu sa zvolil jazyk typescript a angulár, nakoľko obsahuje výhody korešpondujúce s týmto riešením.
2. Pre stavbu modelov a animácií sa použil voľne dostupný modelovací program Blender [23], ktorý je spomenutý aj v návrhu 4.3. , bol však doplnení o rozšírenie generovania gifov z kamery pomocou stiahnutého doplnku Bligify [28], nakoľko výstup animácií bude v podobe prehrávania gifu.

Počiatočná základná štruktúra projektu:

Podobne ako v implementácií systému rozoznávania posunkovej reči, počiatočná základná štruktúra projektu sa bude ďalej odkazovať na implementáciu všetkých navrhnutých komponentov v kapitole 4. ako aj umiestnenia animácií a zoznamu slov. Umiestnenie týchto vecí je z hľadiska implementácie dôležité na základe toho, že súčasť implementácie tvorí komunikáciu medzi rozmiestnenými pridanými komponentami vytvorenými mimo projektu a skriptami v repozitároch vytvorených v tomto projekte. Nasledovný obrázok 24 opisuje rozloženie repozitárov pripravených pre implementáciu návrhu: (obrázok na ďalšej strane)



Obrázok 24 Počiatočná štruktúra projektu pre preklad do posunkovej reči

6.2. Modelovanie, animovanie a zoznam animácií

Aj keď návrh poukazuje na prvotný krok implementáciu zoznamu slov a stohovacie algoritmy, zistilo sa, že je výhodnejšie spraviť najprv model, potom zoznam slov v neurčitku a nakoniec animácie. Takto sa dosiahne základný počet animácií, čiže slov, a až potom je možné lepšie určiť synonymá a výsledný počet slov, ktorý je systém schopný rozoznať. V prvom rade je však potrebné spraviť humanoidný model.

Modelovanie avatara:

Obrázok 25 Model avatara

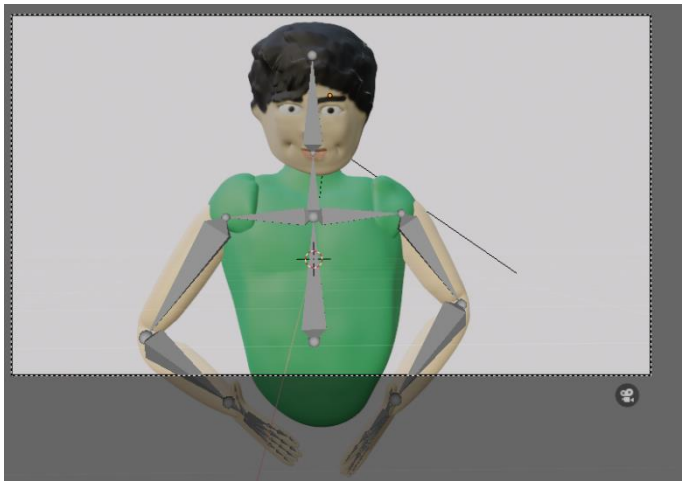
Modelovanie je v princípe najjednoduchšia časť. Závisí už len od času a modelovacích zručností ako bude finálny model vyzerať. Pre túto prácu sa podarilo vyformovať model avatara podľa obrázku 25.

Priradenie kostry:

Ako sa spomína v kapitole 4.3. v časti priradenie kostry, kostra umožní pohybovať modelom v častiach v ktorých sa konkrétna kosť nachádza. Architektúra kostier v rukách teda zodpovedá podobnej architektúre akú ma človek s reálnymi kosťami. Každý prst má teda 4 kosti, podobne ako články prstov na ľudskej ruke vrátane tej v dlani, okrem palca, ten má 3.

Obrázok 26 Priradenie kostry k modelu avatara

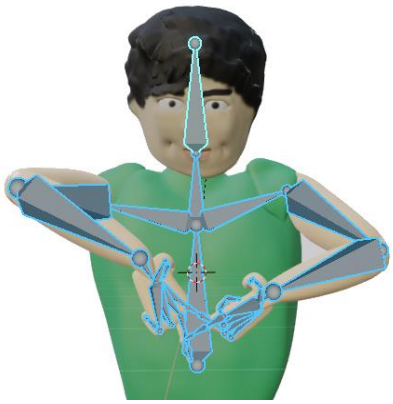
Umiestnenie avatara pred kameru:



Na to aby sa dosiahol plynulý prechod medzi animáciami, je potrebné mať fixný bod a pozíciu avatara aby sa každé gesto začínalo z rovnakej polohy. Posadenie pred kameru je dôležité aj kvôli následnému generovaniu každej animácie do .gif formátu pomocou spomínaného nástroja Bligify.

Obrázok 27 Priradenie kostry k modelu avatara

Animovanie:



Animovanie pozostáva už len z vybratia konkrétnych kosti a rotovať ich dokým avatar nie je v požadovanej pozícii zodpovedajúcemu pre požadované gesto. V princípe sa robia kľúčové body rotácie na časovej osi, ktoré sa plynulo a postupne dostávajú do daných hodnôt.

Obrázok 28 Animovanie modelu avatara.

Generovanie figu a výsledný vzhľad:

Nástroj Bligify si vezme kameru z prostredia a časovú os s kľúčovými bodmi obsahujúce informáciu o polohe a rotácií objektov v scéne, vrátane svetelného zdroja. Z každého frame-u správi fotografiu a nakoniec ich spojí do gif-u. Takto vznikne gif-ová podoba avatara s animáciou, ktorá sa pomenuje podľa sémantického významu ukázaného gesta v neučitku.

Obrázok 29 Výsledný vzhľad avatara

Zoznam animácií:

Kompletný zoznam animácií sa nachádza v systémovej príručke, dokopy ich je 190. Po konzultácií so špecializovanými pedagógmi a nimi odporúčanými knihami, bola snaha vybrať čo najpoužívanejšie a najzakladanejšie slová.

1. Pre podstatné mená je namodelovaných 87 animácií . Vyberalo sa podľa kategórií do ktorých tieto slova spadajú:

Kategórie: Rodina, Ovocie/zelenina, domácnosť, zvieratká, oblečenie, telo, vonkajšok, technológia, neurčité veci (napríklad vec, ráno...) a nakoniec nezaradené (napríklad posunok, skúsenosť)

2. Pre slovesá je zanimovaných 48 animácií a snažilo sa vychádzať primárne z každodenných činností, napríklad behať, ísť, ležať, ale aj jesť, piť, učiť , spievať a podobne.

3. Pre zámená sú to všetky možnosti avšak nekategorizuje sa pohlavie v tretej osobe. To znamená že je zanimované ja, ty, on/ona/ono , my, vy, oni/ony

4. Pre iné, čo je samostatná špeciálna kategória existuje 13 modelov a to zahŕňa napríklad ako, kde, kedy, prečo, aký, nie, alebo, áno, ahoj, vonku, všetko a podobne.

5. Pre čísla to je od 1-9, čiže 9 animácií, nakoľko 0 sa môže ukázať ako ‘o’. Je pravdou, že vyššie čísla sa ukazujú inak, avšak sa predpokladá, že spájaním za sebou idúcich čísel to je domysliteľné.

6. Pre abecedné znaky, čiže samostatné písmena sa zanimovala základná abeceda bez diakritiky čo dokopy tvorí 27 znakov, čiže 27 animácií.

Každá animácia, čiže gif, sa rozdelí do jeho príslušného repozitáru v projekte čiže SignLanguage/src/assets. Viď. obrázok 24.

6.3. Stohovací automat

Ako sa poukázalo v návrhu v kapitole 4.2 a jej podkapitol, stohovacie algoritmy by mali vedieť pre niektoré z animácií vytvoriť rôzne verzie slova rovnakej sémantickej hodnoty. V kapitole 6.2. sa zanimovalo 190 animácií s kategóriami vymenovanými v zozname animácií. Cieľom tejto kapitoly je teda vytvoriť čo najväčšie pokrytie slov pre dané animácie.

6.3.1. Stohovací automat pre podstatné mená

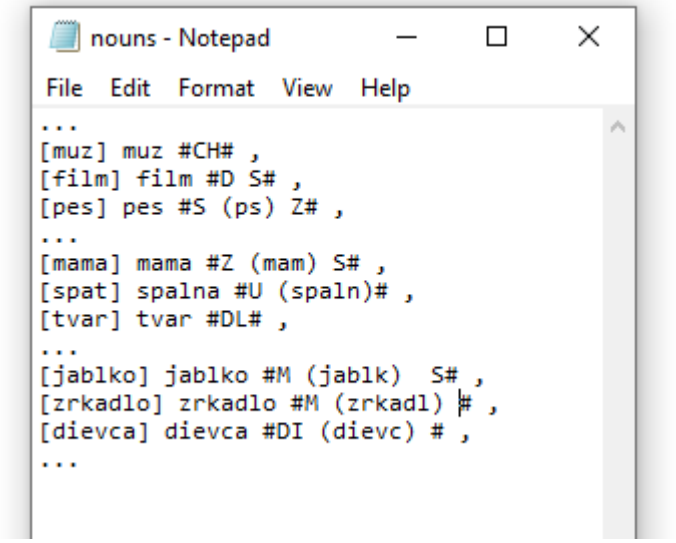
Do repozitára *SignLanguage/words* sa vytvoril textový dokument, ktorý sa naplní slovami podľa sémantiky navrhovanej v kapitole 4.2.1. v časti "Syntax zápisu podstatných mien".

Keďže je zanimovaných 87 podstatných mien, podľa návrhu je teda možné z každého slova mať po skloňovaní 12 ďalších slov, to znamená 87*12 = 1044 slov. Avšak ako sa spomína aj v návrhu, dajú sa ošetriť aj synonymá, to znamená, že textový dokument bol doplnený aj o túto možnosť.

Príklad zapísaného textového dokumentu s každým rodom:
Z textového dokumentu sa vyberie pár príkladných slov so syntaxou:

Pre mužský rod	Pre ženský rod	Pre stredný rod
[muz] muz #CH# [film] film #D S# [pes] pes #S (ps) Z#	[mama] mama #Z (mam) S# [spat] spalna #U (spaln)# [tvar] tvar #DL#	[jablko] jablko #M (jablk) S# [zrkadlo] zrkadlo #M (zrkadl) # [dievca] dievca #DI (dievc) #

V textovom dokumente to môže vyzerá nasledovne, kde 3 bodky značia, že existujú medzi nimi ďalšie slová, zapísané rovnakou syntaxou:



Obrázok 30 Príkladný textový dokument nouns.txt.

V repozitári *SignLanguage/src/app/helpers* sa implementoval nový repozitár s názvom patterns a tento repozitár obsahuje triedy:

1. patterns-manager.ts,
2. male.patterns.ts,
3. woman.patterns.ts.
4. it.patterns.ts

V repozitári *SignLanguage/src/app/helpers* sa nachádza trieda words.ts, ktorá manažuje všetky slová. Táto trieda pred spustením programu obsahuje aj prázdne dvojdimenzionálne pole nounsArray[][], ktoré sa naplní podľa príkladu na diagrame 10.

V princípe existujú 3 vrstvy ktoré sa do seba vnárajú a prechádzajú cyklicky.

Konkrétny príklad postupnosti pre prvý riadok čiže [muz] muz #CH#

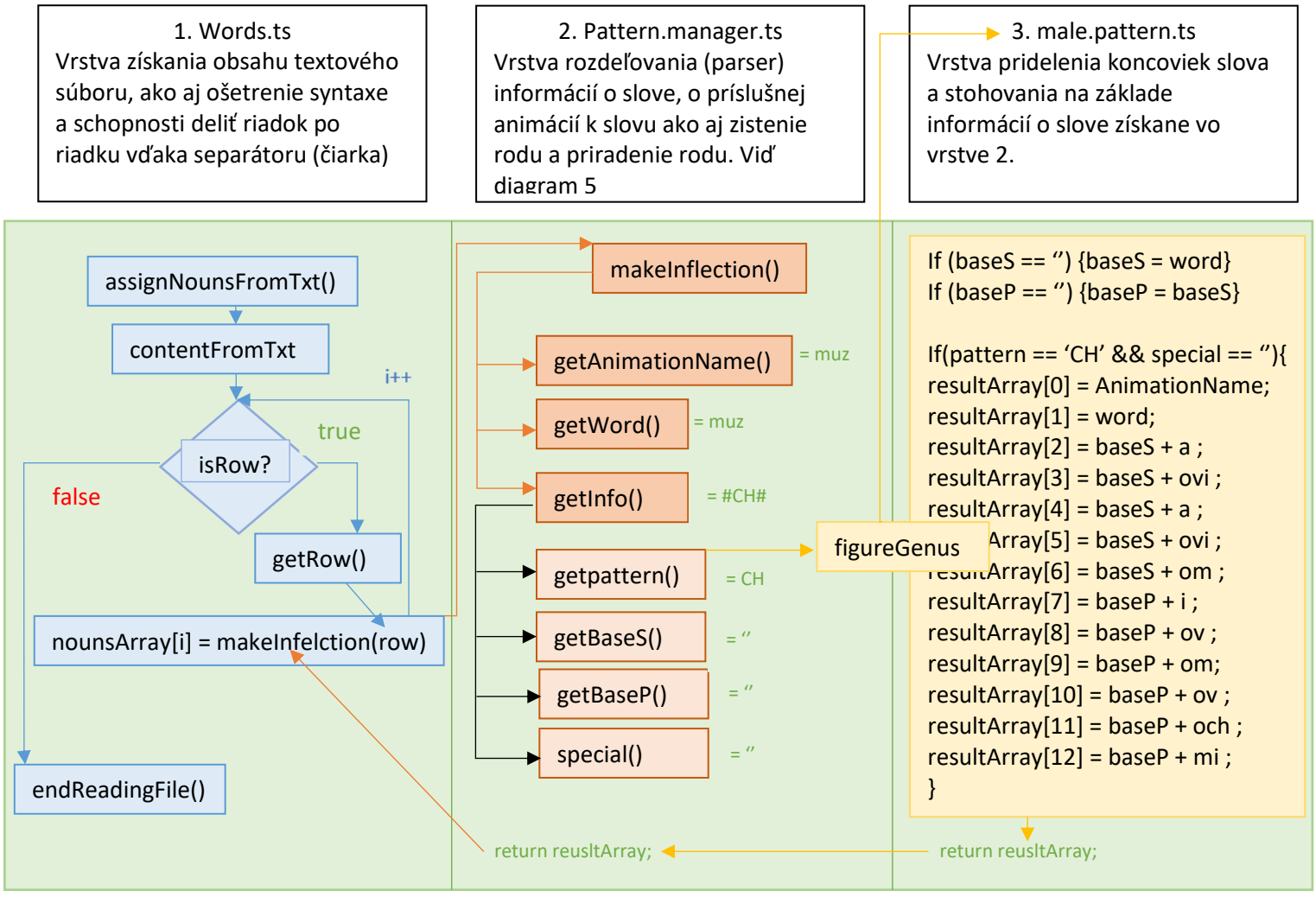


Diagram 10 Príklad stohovania pre riadok [muz] muz #CH#

Na základe dodržania syntaxe riadku podľa návrhu 4.2.1. je možné zautomatizovať skloňovanie namiesto vypisovania všetkých verzií slov.

- Prvá vrstva číta obsah celého textového dokumentu nouns.txt, ktoré obsahuje aj vrátane synoným 113 syntaxných riadkov pre preklad a získava si postupne každý riadok.

- Druhá vrstva funguje ako rozdeľovač (parser) daného riadka na základe syntaxe navrhnutej v kapitole 4.2.1. a aj určí o aký rod ide.

- Tretia vrstva v závislosti o aký rod ide, ako na príkladnom diagrame 10, mužský rod, si nájde pomocou svojho vzoru a iných podmienok postupnosť prideľovania koncoviek a stohuje do poľa.

Tretia vrstva vráti druhej vrstve naplnené pole a druhá vrstva vráti to isté pole do poľa nounsArray na pozíciu iterácie v ktorej sa práve daný spracovaný riadok nachádza. Takto sa získa pole pólov, ktoré dokopy tvoria pokrytie slov, na 84 animácií, v počte 1349. Ak nejde o synonymum a teda názov animácie je zhodný s názvom slova v nominatíve, tak isto ako aj niektoré vyskloňované slová sa svojou formou opakujú, nie je to reálny počet podstatných mien, ktorý je program schopný rozoznať, keďže sa tam nachádzajú aj duplicity.

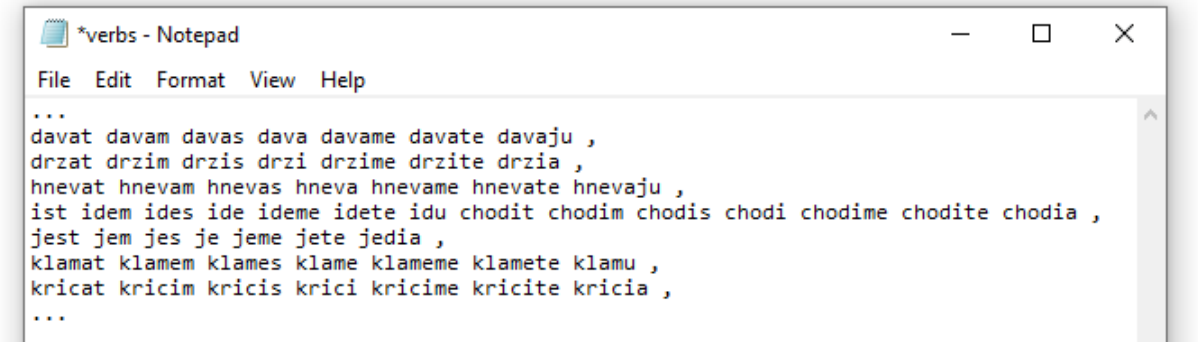
Súčasťou algoritmu je aj vynechanie, prípadne korekcia diakritiky, nakoľko je v dnešnej dobe trend písať práve takýmto štýlom. Takto vie program rozoznať slovo, či už tu diakritiku má alebo nie.

Obrázok 31 Príkladný výpis konzoly niektorých nastohovaných vyskloňovaných slov.

6.3.2. Rádový stohovací automat

Do repozitára *SignLanguage/words* sa vytvorili nové textové dokumenty pre slovesá, zámená a nekategorizované slová. To znamená verbs.txt, pronouns.txt a other.txt. Pre tento automat je syntax jednoduchá. Ako sa spomína v kapitole 4.2.2. stačí napísať slova za sebou, oddelené iba medzerou. Pre slovesá sa však dodržala syntax postupnosti, ktorá osoba to vykonala, nakoľko je možné zistiť týmto spôsobom danú osobu. Celá implementácia sa podobá implementácií stohovaciemu automatu v kapitole 6.3.1., avšak celý tento proces stohovania ma iba jednu vrstvu, nakoľko účelom automatu je zapísať slova postupne za sebou tak, ako boli napísane aj v textovom dokumente.

Príklad textového dokumentu so spomínanou syntaxou:



Obrázok 32 Príkladný textový dokument verbs.txt.

Pri tvorení animácií, sa spomínalo, že názvy majú byť sémanticky ekvivalent ukázaného gesta v neurčitku. Za neurčitok sa predpokladal pri podstatných menách nominatív singuláru a pri slovesách neurčitok. Ako je možné pozorovať z textového dokumentu, platí rovnaké pravidlo, že na prvej pozícií sa nachádza stále názov animácie, v tomto prípade aj slovo pripravené pre preklad. Rádový stohovací automat má preto nasledujúci algoritmus, ktorý sa deje na jednej vrstve vo word.ts:

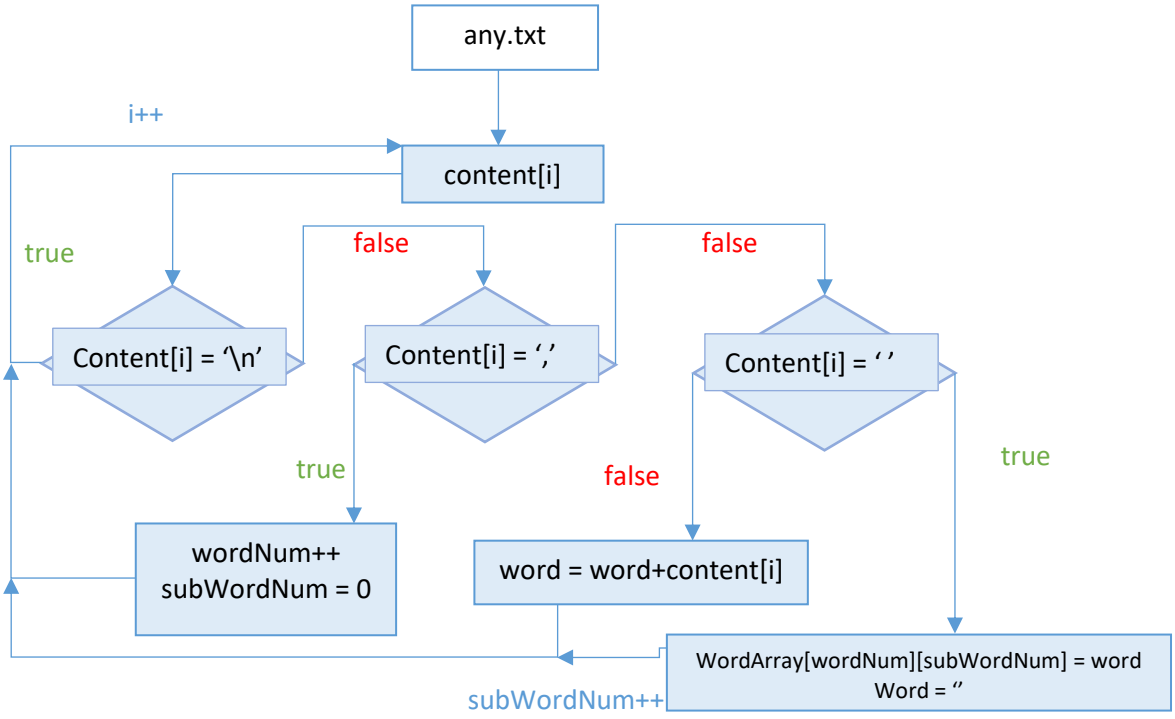


Diagram 11 Postupnosť stohovania fotiek pomocou automatu

Algoritmus si vezme celý obsah textového súboru a v cykle prechádza znak po znaku. Ak narazí na enter, ignoruje ho a prejde na ďalší znak. Ak narazí na čiarku, znamená to, že ide o nové slová pre inú animáciu a vytvorí nové pole v poli pre dané odvetvie slov, a vynuluje počet verzií slov zapísaných pre jednu animáciu. Ak nenarazí na nič predtým ani na medzeru, skladá sa slovo. Ak narazí však na medzeru, znamená to, že slovo je vyskladané, vloží sa do poľa pre danú animáciu

a slovo sa vynuluje. Celý proces trvá, kým sa neprečíta posledný znak nachádzajúci sa v textovom dokumente.

Príkladný konzolový výpis nastohovaných slov pre príkladný textový dokument z obrázka 32.

```
▸ 7: (7) ["davat", "davam", "davas", "dava", "davame", "davate", "davaju"]
▸ 8: (7) ["drzat", "drzim", "drzis", "drzi", "drzime", "drzite", "drzia"]
▸ 9: (7) ["hnevat", "hnevam", "hnevas", "hneva", "hnevame", "hnevate", "hnevaju"]
▸ 10: (14) ["ist", "idem", "ides", "ide", "ideme", "idete", "idu", "chodit", "chodim", "chodis", "chodi", "chodime", "chodite", "chodia"]
▸ 11: (7) ["jest", "jem", "jes", "je", "jeme", "jete", "jedia"]
▸ 12: (7) ["klamat", "klamem", "klames", "klame", "klameme", "klamete", "klamu"]
▸ 13: (7) ["kricat", "kricim", "kricis", "krici", "kricime", "kricite", "kricia"]
```

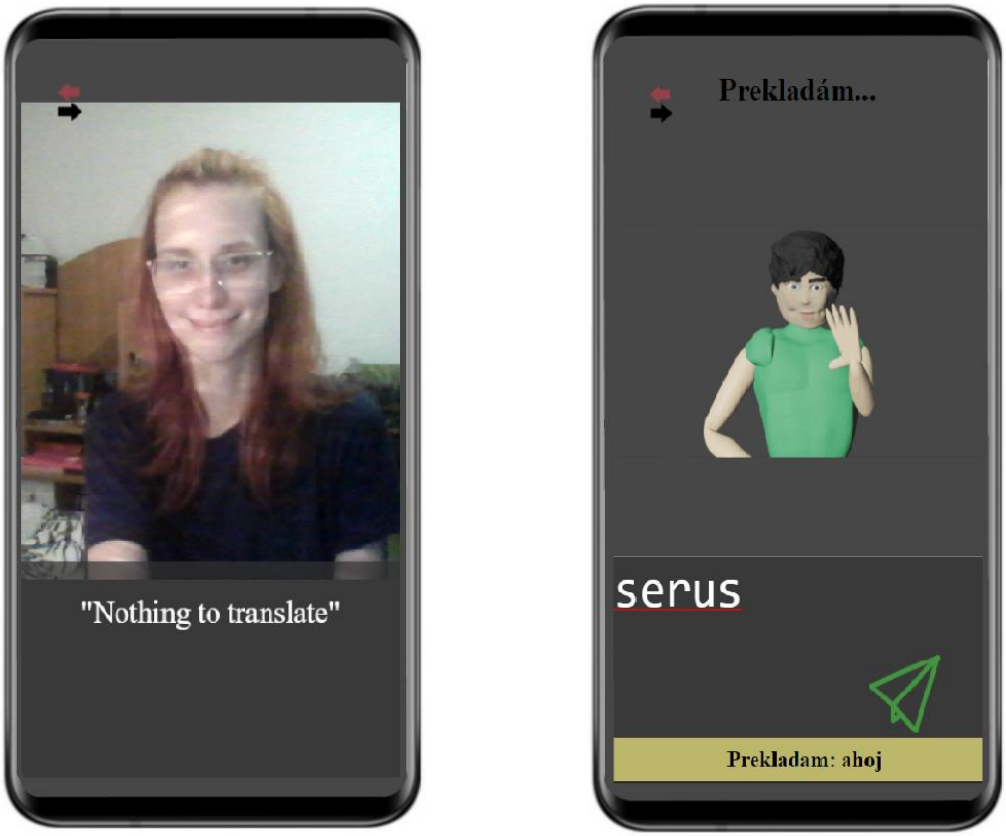
Obrázok 33 Príkladný výpis konzoly niektorých nastohovaných slovies.

Je možné si všimnúť, že pri slovese ísť sú dve varianty a to ísť a chodiť, pri tomto jednoduchom algoritme je možne riešiť synonymá jedine takýmto spôsobom. Vtedy však odpadáva výhoda jednoznačne určiť osobu, ktorá túto činnosť vykonáva.

6.4. Systém webového rozhrania

Webové rozhranie je implementované takmer podľa návrhu v kapitole 4.4. Nasledovné obrázky ukazujú rozloženie implementovaných elementov v prehliadači (popis elementov a ich činností sa nachádza v používateľskej príručke):

Rozloženie webového rozhrania pre mobily :

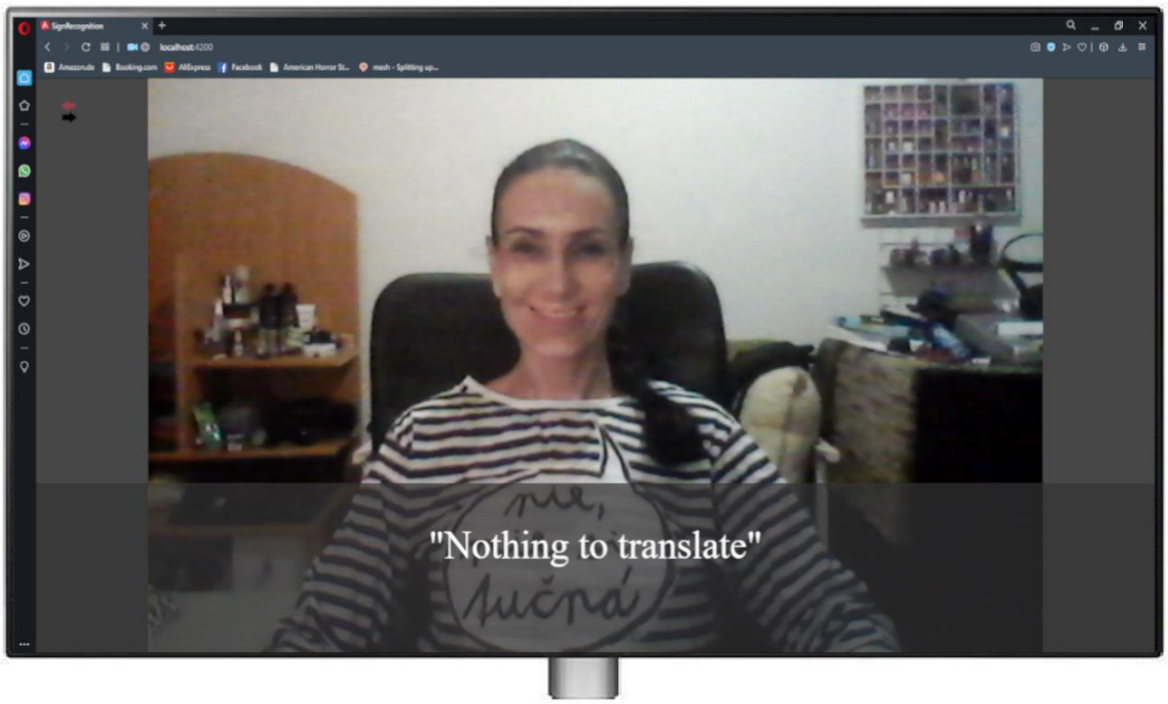


Obrázok 34 Rozloženie obrazovky pre mobily na preklad posunku a preklad do posunku

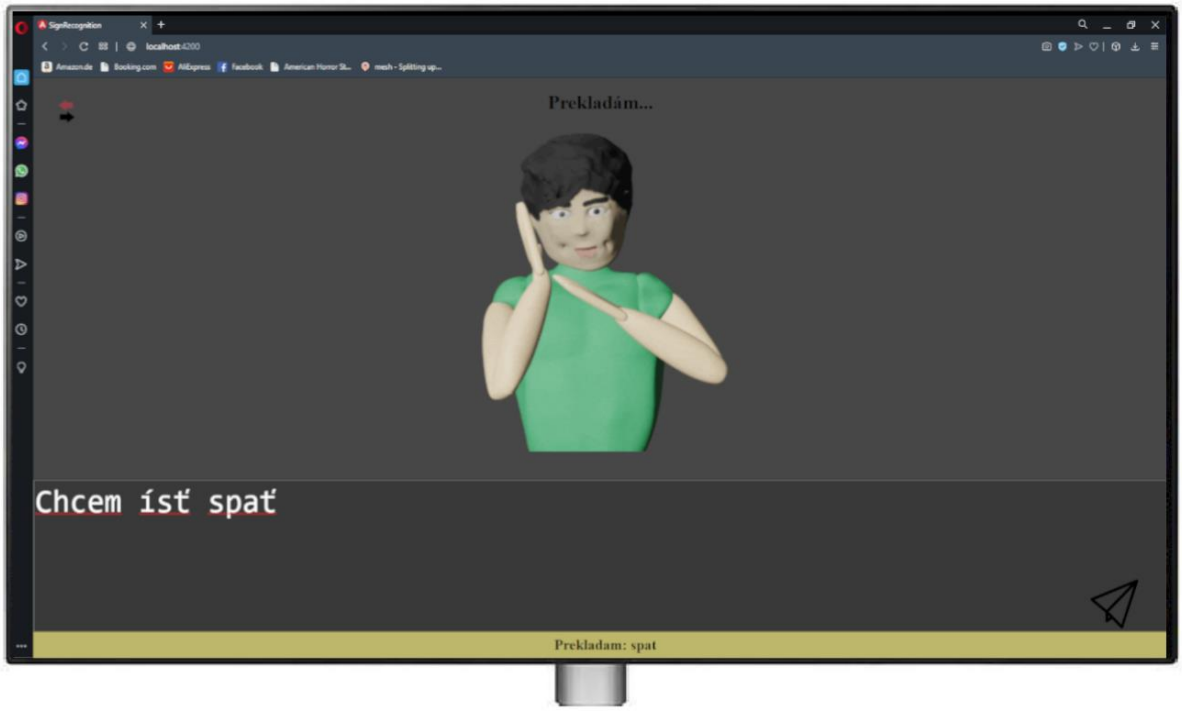
Pri preklade do posunku sa vynechalo tlačidlo na premazanie posunku nakoľko nebolo implementované prepojenie skriptov s rozoznávacím modelom, pretože ako sa spomína v kapitole

5., natrénovaný dataset je len testovací dataset schopnosti rozoznania gesta. Tak tiež sa notifikácie presunuli na dolnú časť obrazovky. Menu nahradilo prepínanie medzi avatarom a kamerou, nakoľko sa zistilo, že menu nie je potrebné.

Rozloženie webového rozhrania pre počítače:



Obrázok 35 Rozloženie obrazovky pre počítače na preklad posunku



Obrázok 36 Rozloženie obrazovky pre počítače na preklad do posunku

Pri preklade do posunkovej reči, text nad avatarom určuje, či užívateľ práve píše, či avatar práve posunkuje, čiže prekladá a tak tiež v nehybnom stave má uvítaciu vetu. Odoslanie vstupu reaguje aj na enter rovnako ako na ikonu v pravom dolnom rohu.

6.5. Systém spracovania vstupu a výstupu

Spracovanie vstupu, čiže zadaného textu od používateľa, a výstupu, čiže zobrazenie animácií sa implementovalo presne podľa navrhnutých algoritmov popísaných v návrhu v kapitole 4.5.

Rozdeľovač (parser):

Diagram 7. je vlastne aj opis implementovaného algoritmu. V princípe prvotný krok, ktorý sa vykoná po potvrdení vstupu od používateľa je to, že sa opraví diakritika. Majme teda vetu: “Chcem ísť spať do postele”.

Program si vezme celý vstup a cyklicky prejde znak po znaku, ak narazí na znak s diakritikou opraví ho na nediakritický znak, tak isto ako aj všetky veľké písmená na malé:

Obrázok 37 Výpis konzoly pred a po oprave diakritiky

Tento text sa nazýva correctedTxt. Ďalej sa posunie aby sa mohlo rozdeliť slovo po slovo a každé slovo sa pridelí do poľa. Separačný znak je medzera:

Obrázok 38 Výpis konzoly po rozdelení slov

Po korekcií vstupu sa získalo pole pripravené na priradenie animácií. Vďaka implementácií stohovacích automatov, program obsahuje premenné naplnené slovami, ktoré vie rozoznať a priradiť im príslušnú animáciu. Tieto premenné sa nastohujú iba raz a to pri spustení programu. A teda sa začína cyklus identický diagramu 7 v kapitole 4.5. Zoberie sa slovo z poľa, spýta sa o aký druh ide. Druh vie zistiť na základe pomocných funkcií, ktoré vratia true alebo false hodnotu, do ktorých keď dáme dané slovo ako parameter, cyklicky prejdú nastohovaním poľom. Ak slovo nemá byť ignorované (napríklad si, sa, som) a nenachádza sa ani v nastohovaných premenných, rozdelí to slovo na písmena a každé písmeno vráti do výsledného poľa animácií. V tomto príklade program nepozná slovo ‘do’ a preto ho rozdelí na písmena.

Prideľovač

Táto časť môže byť braná ako druhá vrstva podobne ako v prípade stohovania podstatných mien, pretože sa program do nej vnára z rozdeľovača, pokiaľ platia podmienky. Z nasledovného konzolového výstupu je jasné, že určovanie slovného druhu je ešte na úrovni rozdeľovača:

Obrázok 39 Výpis konzoly po rozoznania slovného druhu

V tomto štádiu sa teda zoberie dané slovo, pokiaľ mu bol nájdený slovný druh, a vloží sa ako parameter do funckie prideľovača, ktorý sa v programe nazýva getGifName, ako parameter sa do funkcie dá to slovo a aj nastohované pole pre príslušný slovný druh. Funkcia v princípe hľadá riadok kde sa to slovo nachádza a vráti názov animácie. Výsledkom by mal byť teda zoznám názvov animácií. Viď obrázok:

Obrázok 40 Výpis konzoly poľa názvov animácií

Je možné si všimnúť, že po každom slove sa doplnil aj samostatný znak medzery. Je to preto, pretože v prípade ak slovo nerozozná, ako napríklad predložky, animácie sú premietané hneď po sebe, tak aby bol dostatočný priestor, že avatar začal posunkovať jedno slovo po písmenkách. Medzera totiž prinúti avatara aby ostal v nečinnosti necelú sekundu.

Zobrazovač:

V kapitole 4.5. v časti zobrazovač sa navrhovala štruktúra projektu. Keďže štruktúra projektu zodpovedá návrhu, viď obrázok 24, je možné implementovať pseudokód zodpovedajúci v návrhu. Do projektu sa teda implementovala trieda assigner, ktorá obsahuje funkciu assign, do ktorej sa dá parameter už priamo v komponente avatara. To znamená, že komponent avatar obsahuje finálny cyklus, ktorý si berie každé slovo z poľa názvov animácií, nájde mu typ, čiže názov repozitára podobne na princípe zisťovania slovného druhu, a to vygeneruje link, ktorý je relatívna cesta k animácií. Pokiaľ ide o čísla alebo samostatné písmena, to sa zistí vďaka typescriptu pomocou funkcie isNan(). Takto sa nájde stále nový link, keď sa daná animácia prehrá.

7. Overenie systému na rozoznávania posunkovej reči do textu

Tento systém funguje na báze zobrazenia percentuálnej pravdepodobnosti ukázaného objektu, čiže posunku. Táto kapitola zosumarizuje počet nafotených fotografií, počet iterácií učenia sa a priemerná strata (loss) počas toho procesu. Tak tiež sa pozrie hlbšie na počet tried a ako to ovplyvňuje celkové chovanie programu, aký dátový prenos je, ak sa nepreskakuje žiaden frame na webkamere počítača a na konečné výsledky percentuálnej úspešnosti rozoznania posunku. Keďže sa jednalo o prototypný model, táto kapitola obsahuje aj záver odporúčania takého to riešenie pre prípadnú tvorbu modelu rozpoznávania pre obsiahlejšie rozoznávanie posunkov.

7.1. Sumarizácia počtu fotiek pre trénovací model

Zoznamy slov sa nachádzajú v implementačnej časti v tabuľke 3. Boli fotené pomocou webovej kamery z rozlíšením 0.3MP 4:3 (640x480) . Presné čísla nafotených fotografií sú nasledovné:

Abeceda: Keďže z každého kusu sa zrobilo 20 verzii, počet písmen značí 27 čiže 27*20 = 540

Čísla: 9*20 = 180 fotografií

Podstatné mená: 17*20 = 340 fotografií

Zámená: 3*20 = 60 fotografií

Slovesá: 9*20 = 180 fotografií

Čísla: 8*20 = 160 fotografií

Sumár: celkový počet je teda 1440

Ak sa nepočítajú generované súbory .xml, ktoré označujú značku vo fotografií, prerozdelenie bolo v pomore 17:3, čiže:

Train
To train súboru bolo priradených spoločne 1241 fotografií

Test
To train súboru bolo priradených spoločne 219 fotografií

7.2. Trénovací proces modelu.

Trénovanie prebehlo pomocou takzvaného učiteľa (supervised learning), nakoľko boli trénovacie dáta, ktoré sa kontrolovaly na testovacích. Celý model obsahoval 72 tried, čiže gést a proces trénovania prebehol v počte 10 000 krokov. Tabuľka na ďalšej strane ukazuje postupné učenie sa modelu v danom kroku, aký čas trval jeden krok a aká bola výsledna strata na danom kroku:

Kroky	Čas vykonania jedného kroku	strata (Loss)
100	1.984s	1.856
500	1.930s	1.181
1000	2.344s	0.815
1500	2.024s	0.797
2000	2.085s	0.960
2500	2.027s	0.684
3000	2.052s	0.823
3500	2.063s	0.605
4000	1.954s	0.766
4500	2.004s	0.530
5000	2.027s	0.439
5500	2.586s	0.437
6000	2.338s	0.472
6500	2.211s	0.373
7000	2.157s	0.501
7500	1.992s	0.534
8000	2.063s	0.409
8500	2.499s	0.316
9000	1.960s	0.358
9500	2.074s	0.469
10 000	2.020s	0.328

Tabuľka 4 Metrika modelu počas trénovania

- Strata poukazuje na číslo, aká bola predikcia modelu v danom kroku na jednej testovacej zložke. Čím nižšie číslo, tým lepšia predikcia.

-Z tabuľky vyplýva, že modelu sa darilo formovať jeho predikciu nakoľko strata klesala. Je možné tvrdiť, že pokiaľ by sa model dotrénoval o niekoľko krokov, strata by bola ešte nižšia.

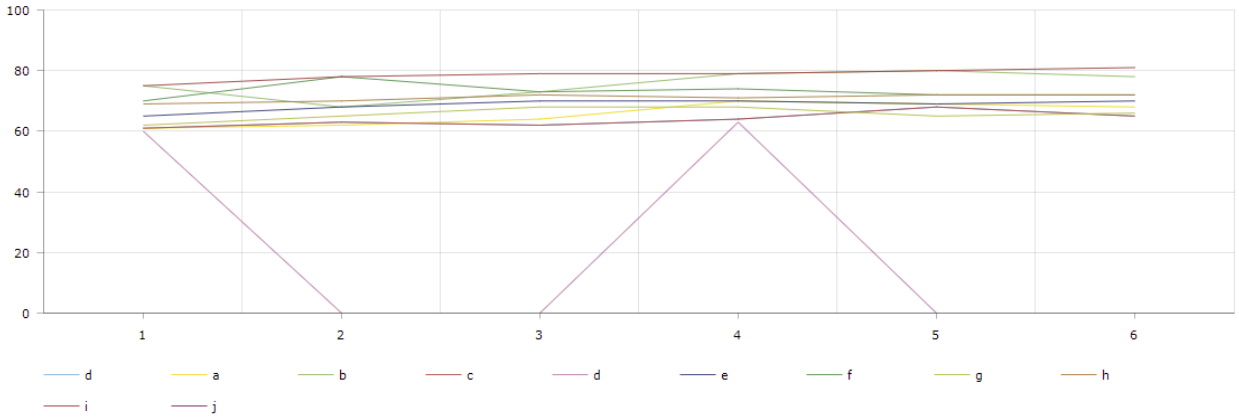
- Tabuľka obsahuje krokovanie po tisíckach, avšak reálny výpis bol dostupný v kokoch po stovkách. Zo všetkých krokov sa zrobil priemerný čas trvania jedného kroku, čo dalo hodnotu 2.24, čiže výsledkom je, že celý proces trénovania tohto modelu pomocou CPU trval (((2.24x10000) / 60) / 60) = 6,22 hodín.

Výsledná strata sa vzhľadom na počet tried dá považovať za nedostatočnú a preto sa odporúča model dotrénovať.

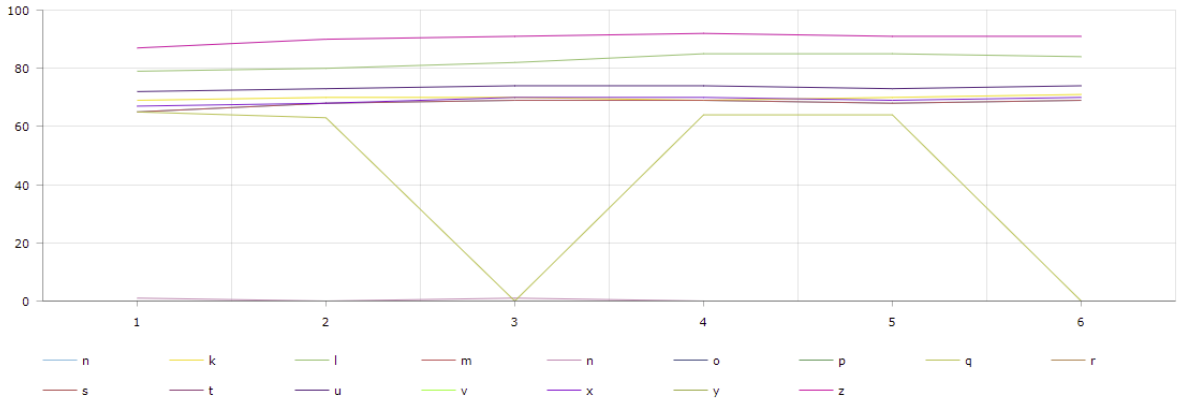
7.3. Výsledky rozpoznávania gesta

Na základe tabuľky 3 v kapitole 5.2. sa vykonal test ukázania každého gesta a vybralo sa 6 náhodných iterácií počas ukazovania. Program bol nastavený tak, aby nezaznamenalo predikciu, pokiaľ je pod 60%. Nasledovné grafy zobrazujú percentuálnu úspešnosť predikcie:

Abeceda:



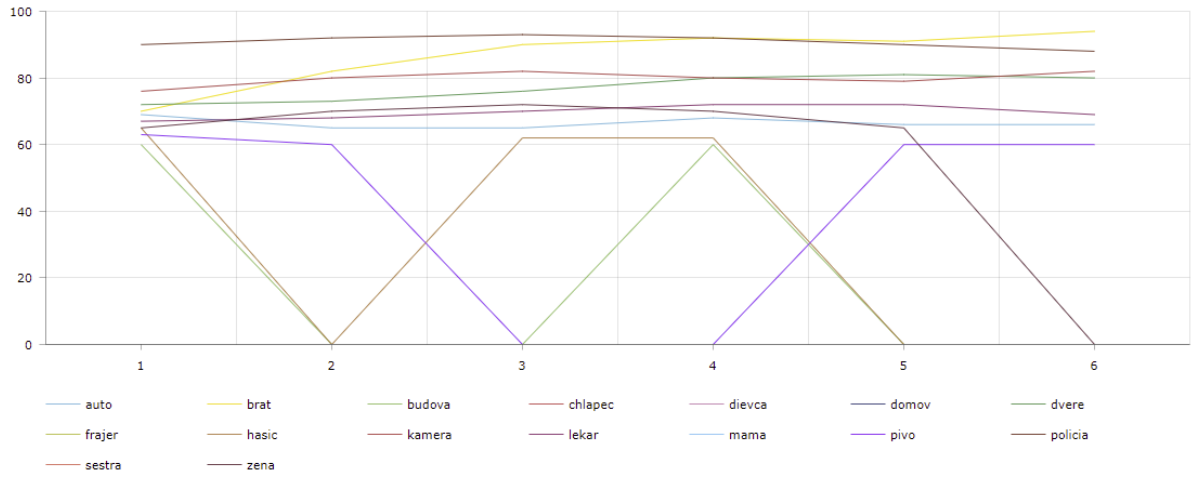
Obrázok 41 Graf úspešnosti predikcie pre písmena a-j



Obrázok 42 Graf úspešnosti predikcie pre písmena k-z

Ako si je možné všimnúť z grafov, niektoré písmena majú vysokú úspešnosť, väčšinu však je možné kategorizovať do predikcie okolo 70%. Žiaľ veľa písmen sa nepodarilo počas testu ani zaznamenať. A to: j,o,p,r,s,t,v, a y. Niektoré sa podarilo zaznamenať ale mali predikciu okolo 60%, preto ak predikcia klesla pod 60, hodnota sa nastavila na 0. Na grafe sú viditeľné skoky.

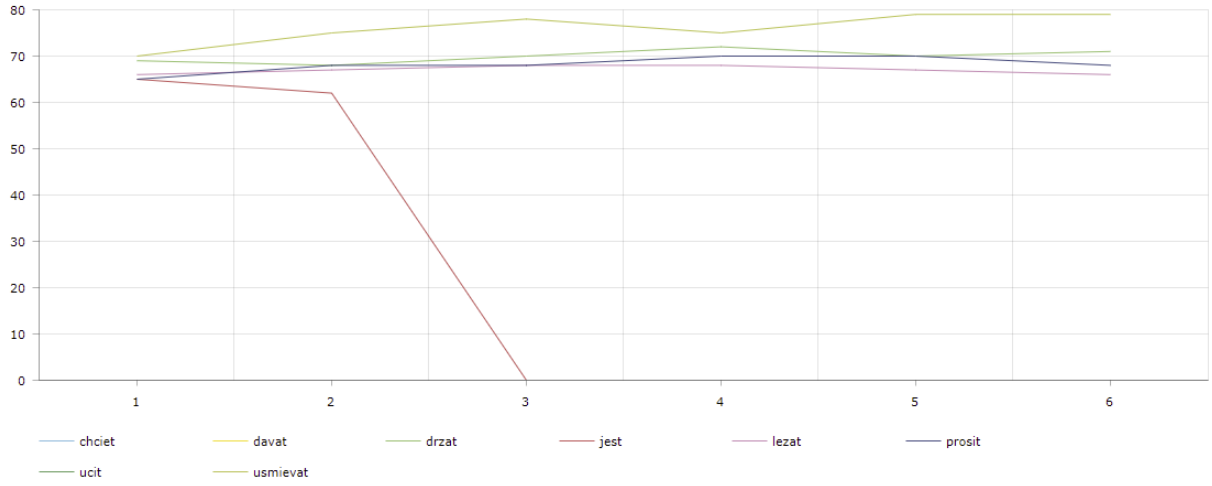
Prídavné mená:



Obrázok 43 Graf úspešnosti predikcie pre podstatné mená

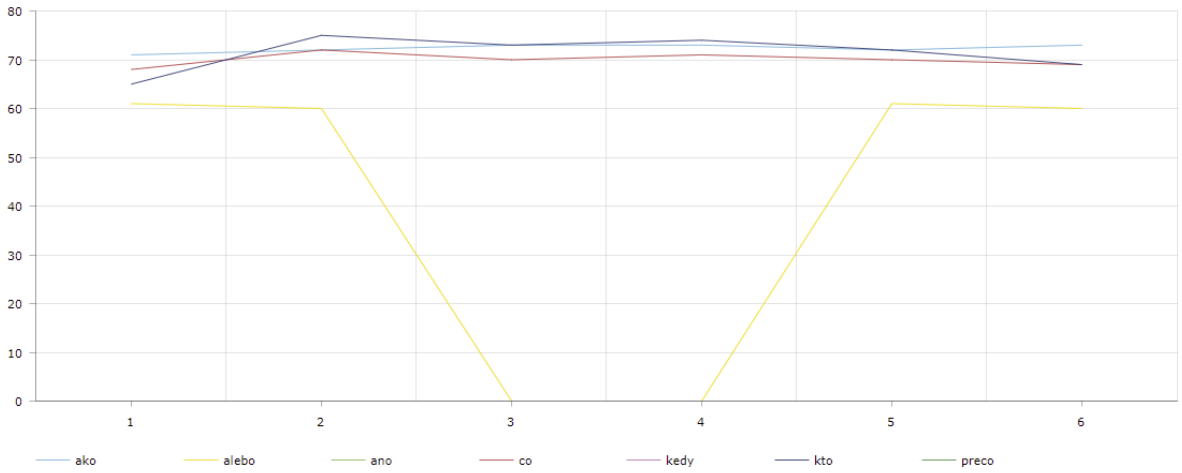
Rovnako ako pri písmenkách, niektoré slová mali silné predikcie, ako napríklad brat, avšak priemer ostatných sa znova pohybuje okolo 70%. V tomto prípade sa nepodarilo počas testovania zaznamenať slová: chlapec, dievča, domov, frajer, mama, sestra.

Slovesá:



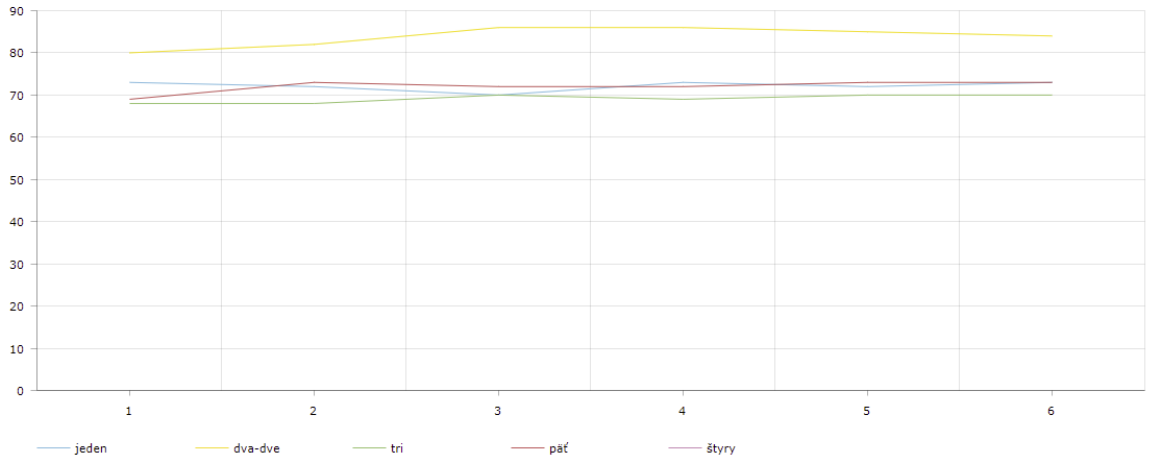
Obrázok 44 Graf úspešnosti predikcie pre slovesá

Nezaradené:



Obrázok 45 Graf úspešnosti predikcie pre nezaradené slová

Čísla:



Obrázok 46 Graf úspešnosti predikcie pre nezaradené slová

V grafe pre čísla sú zapísané jedine čísla po 5, nakoľko čísla, na ktoré treba obe ruky, nevedelo rozoznať vôbec. Číslo 4 sa nepodarilo rozoznať tiež.

7.4. Zhrnutie a záver overenia

Fotografie boli fotené pomocou webovej kamery bez zmeny prostredia jedinou osobou. Tým pádom fotografie a ich označenia môžu podliehať zmenám úspešnosti rozoznania v iných svetelných a priestorových podmienkach. Pre čo najlepšie naučený model sa preto odporúča vytvoriť a označiť fotografie z iných prostredí.

Proces trénovania bol implementovaný a vykonaný na základe existujúcich API, avšak je možné tvrdiť, že pokiaľ sa chce dosiahnuť menšia strata, čiže loss, je potrebný nie len pestrejší dataset, ale aj prípadne viac krokov učenia.

Na základe otestovania celého systému sa prišlo na to, že dochádza k veľkej missklasifikácií pri čom 19 slov počas testovania nerozoznalo vôbec, pri 6 slovách bolo potrebné mať presnú špecifickú polohu a zvyšné slová boli percentuálne okolo 70%. Len pri 7 slovách sa dá tvrdiť, že ich rozoznáva dobre. Nakoľko použité API na trénovanie modelu pozostávajú z toho, že model sa učí na základe charakteristických komponentov, ktoré objekt obsahuje. Použil sa aj krabicový štýl označovania pri gestách kde je potrebné zapojiť obe ruky. Model sa však učí aj vzdialenosť rúk medzi sebou, rotáciou rúk a podobne. Nie každý gestikuluje v presnej vzdialenosti alebo na presnej pozícií od tela. Pokiaľ sa skúšal tento model na datasete s 5 gestami, výsledky boli skvelé, avšak pri narastajúcom počte gest predikcia klesala a začala sa mýliť s inými gestami. Preto je vhodné sa zamyslieť nad riešením vytvorenia kostry ruky, pliec, prípadne tváre, ktorá vie mapovať človeka, a učiť takúto kostru predikcii namiesto predpokladaného tvaru na obrazovke.

Tento testovný set teda vyvrátil vhodnosť použitia Object Detection API s modelom SDD MobileNet V2 FPNLite 320x320 ako nástroj na rozoznanie posunkov z obrazu, ak by cieľom bolo implementovať aspoň základné slová, abecedu a čísla v posunkovej reči.

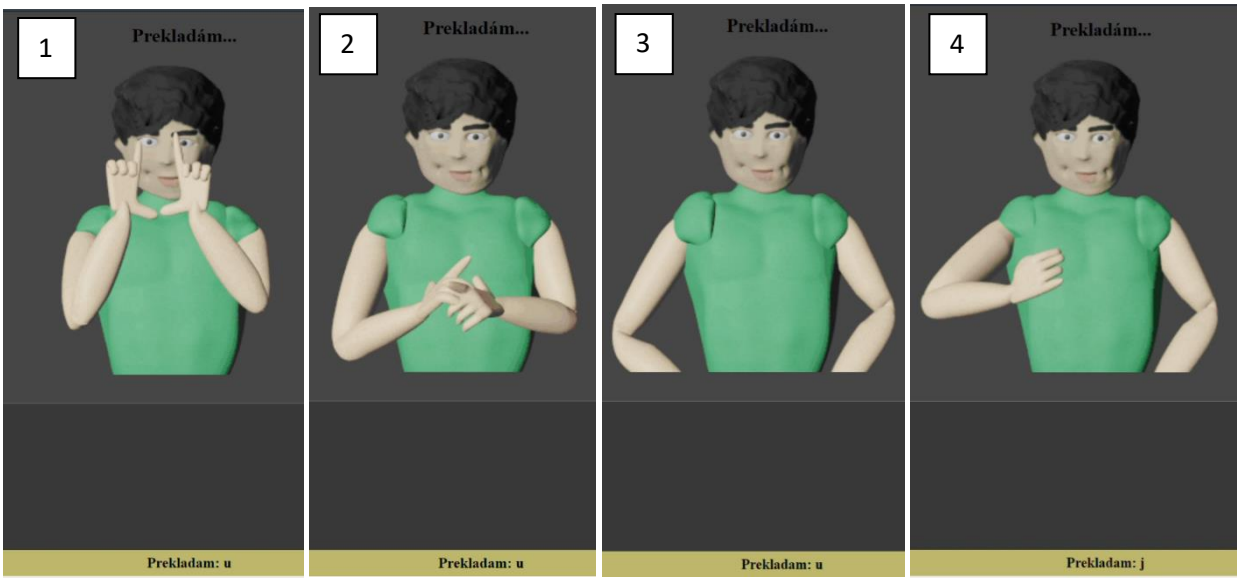
8. Overenie systému prekladu do posunkovej reči

Tento systém funguje na princípe vygenerovania polí možných verzií slov, ktoré prislúchajú zanimovanej animácií gesta, vloženia vstupu od používateľa, spracovanie textu a preloženie sémantického ekvivalentu pomocou vygenerovaných polí animácií na postupnosť animácií a nakoniec zobrazovanie animácie sporadicky podľa vstupu. Táto kapitola zosumarizuje vytvorenie týchto slov ako aj výsledok stohovania, určí približne prípadne duplicity, otestujú sa zvláštne vstupy ako aj nezmyselne veľa slov a špeciálnych znakov a nakoniec sa zrobí záťažový test pre veľké množstvo slov.

8.1. Model a animácie

Samotný model je jeden, opísaní v implementácií v kapitole 6.2. Tento model sa uložil aj ako blender projekt s rovnakým umiestnením kamery, počiatočnej pozície rúk a uloženými textúrami, ak by sa plánovalo rozšíriť počet animácií. Dokopy bolo zanimovaných 190 animácií, ktoré boli konvertované do gif formátu a komprimované aby mali čo najmenšiu veľkosť bez výrazného zníženia kvality. Všetky tieto animácie boli uložené aj na cloudových službách v pôvodnej kvalite ako aj komprimovanej. Projekt využíva komprimované verzie animácií. Bližšie informácie o uskladnení gifových formátov ako aj uložených projektov sú v systémovej príručke.

Veľká výhoda spracovania avatara namiesto reálnej postavy bol plynuli prechod medzi animáciami. Vďaka implementovanému modelu sa tento cieľ naplnil. Nasledovná postupnosť obrázkov znázorňuje príkladný prechod medzi dvoma písmenami :



Obrázok 47 Znázornenie plynulého prechodu medzi animáciami

Ako je možné vidieť na obrázku 47, avatar na prvej obrazovke ukazuje písmeno ‘u’, potom skladá ruky do pôvodnej počiatočnej polohy na obrazovke 2. Na obrazovke 3 má svoju konečnú polohu animácie. Keďže písmeno ‘j’ ma začiatočnú animáciu v polohe ako má ‘u’ konečnú animáciu, vie plynulo začať ukazovať písmeno. Na tomto princípe boli vytvorené všetky animácie ktoré sa začínajú a končia stále v rovnakej polohe.

Dĺžka animácií:

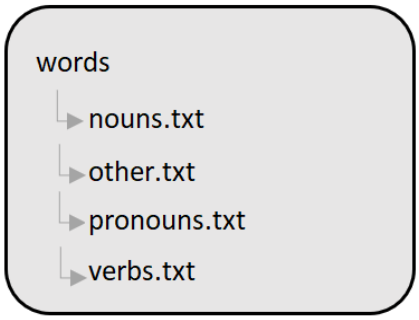
Keďže sa jedná o animácie vo formáte gifu, metrika pre určenie dĺžky animácie nie je v časovej jednotke ale v počte obsahujúcich fotografií (frame-ov). V plnej kvalite bol stále statický počet frame-ov. Avšak po komprimovaní sa tento počet znížil. Pre písmená a čísla sa pohybuje počet frame-ov okolo 21 a pri slovách okolo 40. Rýchlosť zobrazovania teda závisí od grafickej jednotky koľko stroj, na ktorom je systém implementovaný, číta frame-ov za sekundu. Keďže sa jedná o webové riešenie, táto jednotka bude ovplyvňovaná rýchlosťou internetu používateľa.

8.2. Stohované slová

Stohovanie slov prebehlo na základe syntaxe a algoritmoch implementovaných v kapitole 6.3. Pri stohovaciom automate pre podstatné mená sa spomína, že v niektorých vzoroch v slovenčine, sa forma slova opakuje, napríklad vo vzore dub, je forma slova dub v nominatíve singuláru ale aj v akuzatíve singuláru. Nevýhoda tejto duplicity spočíva pri väčšom počte slov, pri hľadaní slova, tým pádom algoritmus kontroluje aj duplicity. Avšak duplicity nebudú zo systému odstránené, nakoľko takto nastohované podstatné mena dávajú výhodu zistenia pádu podstatného mena, čo môže byť výhoda pri riešení problematiky napríklad s predložkami.

Implementovali sa teda 4 nové textové dokumenty:

1. Nouns.txt bol sprocesovaný pomocou algoritmu a syntaxe pre podstatné mená. Kde výsledný počet vygenerovaných slov a uložených v premennej programu pre 87 animácií je 1349.



2. other.txt bol - pomocou algoritmu rádového stohovacieho automatu, kde výsledný počet nastohovaných slov a uložených v premennej programu othersArray[][] na 13 animácii je 49.

3. pronouns.txt bol - pomocou algoritmu rádového stohovacieho automatu, kde výsledný počet nastohovaných slov a uložených v premennej programu pronouns[][] na 6 animácií je 50

4. verbs.txt bol - pomocou algoritmu rádového stohovacieho automatu, kde výsledný počet nastohovaných slov a uložených v premennej programu verbsArray[][] na 48 animácií je 342

Výsledný počet všetkých nastohovaných slov tvorí: 1790

8.3. Záťažové testy

Pri generovaní slov, hľadaní animácií a zobrazovaní, môže vzniknúť otázka, ako dlho by trval proces s implementovanými algoritmami, ak by slov bolo omnoho viac? Preto sa v tejto kapitole nasimuluje veľký počet slov na generovanie podstatných mien a potom sa vynúti proces nájdenia slova na konci vygenerovaného poľa aby našlo animáciu. Je dôležité spomenúť, že tieto testy sa vykonávajú na počítači s charakteristikou opísanou v kapitole 5.1. na obrázku 17, kde sú napísané základné informácie počítača, na ktorých sa systém implementoval.

Stohovanie veľkého množstva podstatných mien:

Pre simuláciu sa nakopírovali existujúce slová niekoľko tisíc krát. Podľa obrázka napravo sa nachádza v dokumente 81227 podstatných mien. Skloňovanie prebieha pri načítaní stránky.

```
81218  [policia] policajtka #Z (policajtk)# ,
81219  [lekar] lekar #CH (lekar)# ,
81220  [lekar] lekarka #Z (lekark)# ,
81221  [lekar] zdravotnik #CH# ,
81222  [lekar] zdravotnicka #CH (zdravotnick)# ,
81223  [hasic] hasic #CH (hasic)# ,
81224  [hasic] poziarnik #CH# ,
81225  [sanitka] sanitka #Z (sanitk)# ,
81226  [pivo] pivo #M (piv) S# ,
81227  [zrkadlo] test #Z (test)# ,
```

Na obrázku 48 je možné pozorovať odozvu výpočtu skloňovania takéhoto počtu slov:

Name	Status	Type	Initiator	Size	Time	Waterfall
verbs.txt	304	xhr	zone.js:3272	297 B	3 ms	
pronouns.txt	304	xhr	zone.js:3272	297 B	3 ms	
other.txt	304	xhr	zone.js:3272	297 B	4 ms	
nouns.txt	304	xhr	zone.js:3272	300 B	25 ms	

Obrázok 48 Znázornenie plynulého prechodu medzi animáciami

Dokopy sa vygenerovalo 982439 slov. Čas trvania je však nepovšimnuteľný a preto sa predpokladá, že ani pri veľkých počtoch by nebolo potrebné implementovať nejaké rozhranie, ktoré by informovalo používateľa, že prebieha výpočet.

Prechádzanie poľom veľkého počtu slov:

Súbor ostáva nezmenený, je možné si všimnúť, že na poslednom riadku sa sémanticky priradila náhodná existujúca animácia a slovo, ktoré sa nachádza ako jediné v poli na konci. Tým pádom, ak program bude hľadať, či podstatné meno má animáciu, bude nútene prejsť celým poľom.

```
app.component.ts:26
(982439) ["mama", "mama", "mamy", "mame", "mamu", "mame", "mamou", "mamy", "mam", "mama
m", "mamy", "mamach", "mamami", "mama", "mamka", "mamky", "mamke", "mamku", "mamke", "ma
mkou", "mamky", "mamiek", "mamkam", "mamky", "mamkach", "mamkami", "mama", "mamuska", "m
amusky", "mamuske", "mamusku", "mamuske", "mamuskou", "mamusky", "mamusiek", "mamuskam",
"mamusky", "mamuskach", "mamuskami", "otec", "otec", "otca", "otcovi", "otca", "otcovi",
"otcom", "otcovia", "otcov", "otcom", "otcov", "otcami", "sestra", "sestra", "sestry",
"sestre", "sestru", "sestre", "sestrou", "sestry", "sestier", "sestram", "sestry", "sest
rach", "sestrami", "sestra", "segra", "segry", "segre", "segru", "segre", "segrou", "seg
ry", "segier", "segram", "segry", "segrach", "segrami", "brat", "brat", "brata", "bratov
i", "brata", "bratovi", "bratom", "brati", "bratov", "bratom", "bratov", "bratmi", "rodi
na", "rodina", "rodiny", "rodine", "rodinu", "rodine", "rodinou", "rodiny", "rodin", "ro
dinam", "rodiny", …]
```

Obrázok 49 Konzolový výpis poľa zduplikovaných podstatných mien

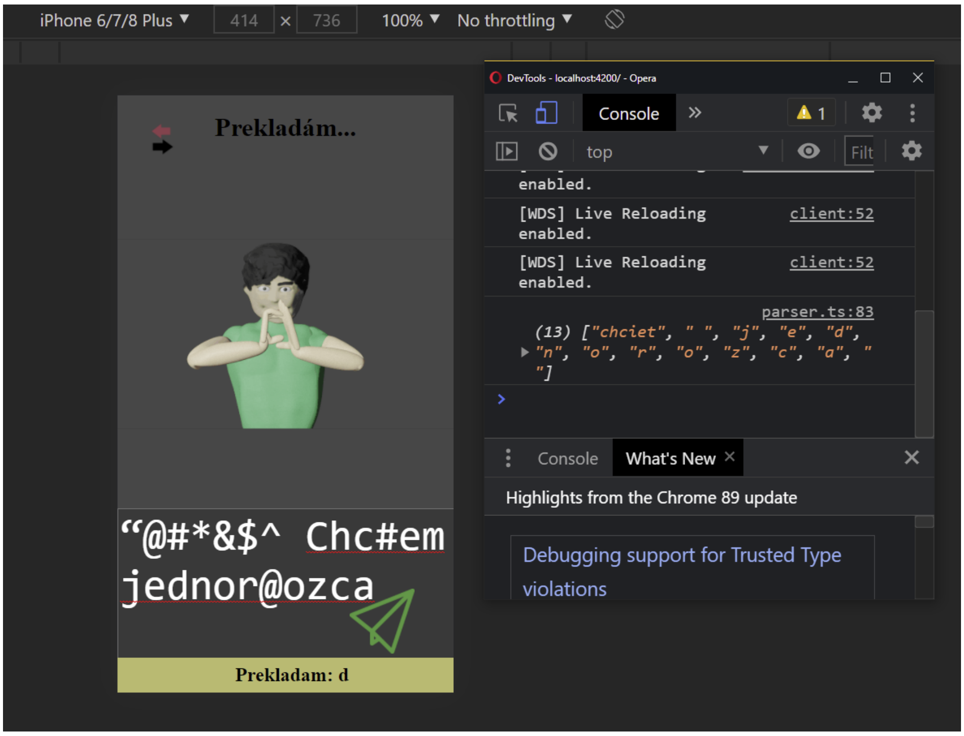
Vyskloňované slovo test vzorom žena, sa nachádza na posledných pozíciách veľkého poľa. Aby sa systém zaťažil, vloží sa nasledovný text do vstupu a otestuje sa jeho odpovedný čas: “*test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test test*”

Takto bude program nútený 72x prejsť poľom po 982427 pozíciu, čo tvorí 72x982427 krokov. Aj po takto nezmyselne dlhom vstupe je odpovedný čas programu menej ako pol sekundy. Preto sa predpokladá, že nie je potrebné implementovať žiadne načítavacie rozhranie. Vďaka konzolovému výpisu je možné aj potvrdiť, že animácie sa priradili správne podľa syntaxe:

Obrázok 50 Konzolový výpis správne priradenej animácie pri záťažovom teste

Test špeciálnych vstupov:

Do vstupu sa dá nasledujúci text: “@#*&$^ Chc#em jednor@ozca“



Obrázok 51 Test špeciálnych vstupov

Na základe obrázka 51, je možné tvrdiť, že systém sa vie vysporiadať zo špeciálnymi znakmi, dokonca ak sa aj nachádzajú v slove, ktoré je program schopný rozoznať.

8.4. Zhrnutie a záver overenia

Systém obsahuje animácie vo formate gifu, čo je priateľskejšie pre vykresľovanie a výpočtovú pamäť, avšak týmto spôsobom sa stráca možnosť 3D zobrazenia. Veľkou výhodou je plynulý prechod animácií, no niektorí ľudia môžu preferovať zobrazenie gifu reálnou postavou pre lepšiu asociáciu gesta.

Systém preukázal aj plne funkčné stohovacie automaty, z čoho jeden automat je špeciálne stavaný pre slovenské podstatné mená. Tento automat sa preukázal aj vďaka svojej syntaxe ako dobrý urýchľovač pri zapisovaní podstatných mien a ich vyskloňovaných verzií ako aj priradeľovač priamych animácií. Tým pádom stačí systém už len rozširovať o animácie v podobe gifov a zapísať ich do textového súboru, na to aby systém vedel rozoznávať nové slová. Nevýhoda stohovacieho automatu pre podstatné mená je tá, že syntax je veľmi striktná a treba sa ju naučiť.

Pri záťažových testoch sa preukázalo, že systém vie zniesť a vyskloňovať potencionálne všetky podstatné mená v slovenskom jazyku. Tak isto sa preukázalo, že opakovaný vstup s najväčším počtom iterácií v programe vypočítal vo veľmi priaznivom čase. Avšak je dôležité podotknúť, že pokiaľ by bol systém implementovaný na počítači, ktorý by mal výrazne slabšie kalkulačne jadro, výsledky sa môžu odlišovať. Záťažový test preukázal aj schopnosť systému vysporiadať sa so špeciálnymi znakmi a tým pádom sa vytvorila čiastočná korekcia textu.

Najväčšou nevýhodou systému je, že si nedokáže asociovať viacslovné výrazy a preto vety ako “Mam ťa rád”, dokáže preložiť doslovne, avšak v posunkovom jazyku to nie je sémanticky ekvivalent výrazu, ale sémanticky ekvivalent každého slova. Je to možné vnímať ako doslovný preklad pre anglický výraz “peace of cake” = kus koláča. Avšak spoločne ako výraz to znamená, že niečo bolo príliš ľahké.

Nakoľko sa zanimovali všetky slovenské písmená a ošetrila diakritika, je možné tvrdiť, že neexistuje slovenské slovo, ktoré by systém nevedel preložiť do posunku. Výhodou je, že systém obsahuje aj niekoľko tisíc základných slov, ktoré vie rozoznať tým pádom je preklad rýchlejší.

Záver

Bolo potrebné analyzovať spracovanie posunkovej reči obojsmerne, ako z pohľadu rozpoznávania, tak z pohľadu interpretácie posunkovej reči. Preto je v tejto práci vypracovaná analýza možných riešení spracovania posunkov pomocou rôznych technológií. Nakoniec sa vybrala technológia najviac srdcu blízka pre bežných ľudí a to mobilné a desktopové riešenia. V tejto práci je tak tiež návrh systému spracovania obrazu pomocou vytvorenia vlastného dátového setu, naučenia systému na gestá z dátového setu pomocou existujúcich rozoznávacích modelov a konečné rozoznávanie z obrazu. Na základe toho to návrhu sa implementoval systém pre overenie vhodnosti použitia rozoznávacieho modelu. V overení systému sa však prišlo na to, že tento model vie rozoznávať gestá, avšak pri väčšom množstve posunkov často dochádzalo k zlej interpretácií posunkov, čiže sa systém mýlil alebo posunok vôbec nerozoznal. Preto záverom pre tento systém na základe výsledkov v kapitole 7.3., bolo vyvrátenie vhodnosti použitia implementovaného modelu a odporúča sa použiť skôr taký model, ktorý mapuje človeka pomocou nejakej kostry a systém sa učí na základe kostry a nie na základe komponentov na obrazovke. Táto diplomová práca obsahuje taktiež návrh systému pre rozpoznávanie textu do posunkovej reči. Bolo potrebné navrhnúť a vytvoriť model tvorby a riadenia kinematickej štruktúry vrátane príslušnej sústavy transformácií za účelom schopnosti interpretovať posunkovú reč. Preto táto práca obsahuje návrh humanoidnej postavy s kostrou, ktorá vie byť animovaná, komplexný stohovací automat s vlastnou syntaxou pre podstatné mená a všeobecný stohovací automat. Tak tiež obsahuje návrh pre algoritmy, ktoré vedia prerozdeliť text na slová a priradiť a zobraziť animáciu na základe nastohovaných slov podľa syntaxe. Na základe návrhu bolo potrebné systém implementovať, vrátane vizualizácie modelu s prezentáciou posunkovej reči a jej riadenia pomocou definovaných skriptov. Preto práca obsahuje implementáciu riadiacich skriptov pre spracovanie vstupu a vizualizáciu v kapitolách 6.3 a 6.5., podľa návrhu. Nakoľko bolo potrebné tiež experimentálne overiť a zhodnotiť príslušné programové vybavenie, sa v kapitole 8.3 vykonali záťažové testy pre tento systém a overenie riešenia sa preukázalo ako vhodné riešenie pre rozoznávanie textu do posunkovej reči, vďaka kladným výsledkom zosumarizovaných v kapitole 8.4. Vylepšenia pre tento systém by mohlo byť doplnenie možnosti výberu animácií na základe reálnej postavy. Tak tiež by bolo možné rozšíriť systém o audio vstup, ktorý by bol prekladaný do textu a z textu už podľa implementovaného systému. Ďalšie, celkom zrejme rozšírenie, by mohlo byť zanimovanie čo najviac slovenských slov, aby sa docielilo čo najmenšia potreba rozdeľovať slová na písmená a tým pádom by trvanie zobrazeného prekladu bolo oveľa kratšie.

Zoznam použitej literatúry

[1]. posunková reč. In:*, et al.* Encyklopédia jazykovedy. 1. vyd. Bratislava *:* Obzor*, 1993. 513 s.* ISBN 80-215-0250-9*. S. 331*

[2]. Snepeda, Roman Vojtechovský, Posunky.sk "Vysvetlívky", [online] [2018-2021], Dostupné na: < https://www.posunky.sk/vysvetlivky/ >

[3]. Facebook oculus.com, "Oculus Documentation", [online], Dostupné na: < https://developer.oculus.com/documentation/>

[4]. Facebook Technologies oculus.com, "Hand Tracking in Unity", [online], Dostupné na: < https://developer.oculus.com/documentation/unity/unity-handtracking/>

[5]. Microsoft microsoft.com, "Microsoft HoloLens", [online], Dostupné na: < https://docs.microsoft.com/en-us/hololens/ >

[6]. Microsoft microsoft.com, "HoloLens 2 gestures for authoring and navigationg in Dynamic 365 Guides", [online], Dostupné na: < https://docs.microsoft.com/en-us/dynamics365/mixed-reality/guides/authoring-gestures-hl2 >

[7]. Myo myo.com, "Myo Scripting Basics Part One Setup", [online], Dostupné na: < https://developerblog.myo.com/getting-started-myo-scripts-part-1/ >

[8]. Paolo Visconti, "Obrázok", [online], Dostupné na: https://www.researchgate.net/profile/Paolo-Visconti-2/publication/324889539/figure/fig33/AS:665950072340482@1535786278127/EMG-insertion-sensors-result-invasive-for-the-patient-A-while-MYO-armband-worn-on-the.png>

[9]. Unity Technologies unity3d.com, "Getting started with VR development in Unity", [online] [2021-04-05], Dostupné na: < https://docs.unity3d.com/Manual/VROverview.html >

[10]. Facebook Technologies oculus.com, "Import Oculus Integration Package", [online] , Dostupné na: < https://developer.oculus.com/documentation/unity/unity-import/ >

[11]. Samuel Sam, tutorialspoint.com, "C# Language advantages and applications", [online] [24-jul-2018 15:37:53], Dostupné na: < https://www.tutorialspoint.com/Chash-Language-advantages-and-applications >

[12]. Lua lua.org, "Documentation", [online], Dostupné na: < https://www.lua.org/docs.html >

[13]. Myo myo.com, "Myo Scripting Basics Part One Setup", [online], Dostupné na: < https://developerblog.myo.com/getting-started-myo-scripts-part-1/ >

[14]. Myo npmjs.com, "myo.js", [online], Dostupné na: < https://www.npmjs.com/package/myo?fbclid=IwAR3tC8hbokVF0WrFPKLRgvCAVTYHscThHXXePI z7SeYSn2DCRlW94plvVso >

[15]. Mahesh Chand, c-charpcorner.com, “Best Programming Language for IOS App Development”, [online] [Feb-21-2019], Dostupné na: < https://www.c-sharpcorner.com/article/best-programming-language-for-ios-app-development/ >

[16]. Microsoft typescriptlang.org, “TypeScript Documentation”, [online], Dostupné na: < https://www.typescriptlang.org/docs/ >

[17]. Arduino arduino.cc, “Language Reference”, [online], Dostupné na: < https://www.arduino.cc/reference/en/>

[18]. RespberryPi raspberrypi.org, “Python”, [online], Dostupné na: < https://www.raspberrypi.org/documentation/usage/python/ >

[19]. Google Inc. Modelviewer.dev, “<model-viewer> Easily display interactive 3d models on the web & in AR”, [online], Dostupné na: < https://modelviewer.dev >

[20]. Jonathan Huang, Vivek rathod, Vighnesh Birodkar, Austin Mayers, Zhichao Lu, Ronny Votel, Yu-hui Chen, Derek Chow, TF Object Detection Team, [online], Dostupné na: < https://github.com/tensorflow/models/tree/master/research/object_detection>

[21]. Jonathan Huang, Vivek rathod, Vighnesh Birodkar, Austin Mayers, Zhichao Lu, Ronny Votel, Yu-hui Chen, Derek Chow, TF Object Detection Team, “Configuring the Object Detection Training Pipeline” [online], Dostupné na: < https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/configuring_jobs.md >

[22]. Jonathan Huang, Vivek rathod, Vighnesh Birodkar, Austin Mayers, Zhichao Lu, Ronny Votel, Yu-hui Chen, Derek Chow, TF Object Detection Team, “TensorFlow 2 Detection Model Zoo” [online], Dostupné na: < https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md >

[23]. Blender blender.org, “Blender Documentation”, [online], Dostupné na: < https://docs.blender.org >

[24]. Anaconda Inc. Anaconda.com, [online] Dostupné na: < https://www.anaconda.com/products/individual >

[25]. Tyutalin, LabelImg, Git code (2015) [online] Dostupné na : < https://github.com/tzutalin/labelImg >

[26]. Nicholas Renotte, RealTimeObjectDetection, [online], Dostupné na: < https://github.com/nicknochnack/RealTimeObjectDetection/tree/main/Tensorflow/scripts >

[27]. Jupyter jupyter.org [online], Dostupné na: < https://jupyter.org/install >

[28]. Daniel Oakey, [online], Dostupné na : < https://github.com/doakey3/Bligify >

Prílohy

Príloha A: Systémová príručka

Príloha B: Používateľská príručka

Príloha C: DVD médium